CS 144 Notes
W 1 M Lec 1-4-16

- Web Applications
  - Dealing with web apps - learning PHP, JavaScript, Java, etc.
  - HTML, CSS, JavaScript
    - Go over some of the standards and some of the rules of those
    - Client side code
  - When you build any application, especially web applications, there is also a server side
    - This does a lot of intelligent things in the backend
  - When you build a web application, what are the basic protocols and standards that enable connecting all these components together.
  - HTTP protocols
    - Build correct and well functioning applications
  - XML standard
    - A way of organizing and representing data
    - Before XML, there were a lot of ways to represent data, but nowadays, XML is the de facto method of data representation.
    - Reasonably straightforward
  - Keyword based search
    - Learned all the things about SQL and storing and retrieving data using relational databases.
    - How to support keyword based searches
      - Very fundamental part to give access to the underlying data
      - Go over some of the theory and how keyword based search is supported
      - You have to know how to use relational database systems and explain what those are.
    - SQL
      - Express data using SQL and if you want to grab something from a particular field name
      - If you really think about what kinds of conditions i.e. field content, values
        - Doesn't support searching on text based fields.
        - It doesn't easily say that this particular field should have this kind of keyword.
        - We will see how to support those kinds of queries.
    - Doing anything over the Internet is insecure, so we have HTTP protocols for that.

        - Significant part of the class goes over the backend of web applications
        - Know relational databases really well
            - Go over how to use the relational databases
            - Know the basics of networking and how information can be exchanged over nodes on the Internet
    - This class has a lot of projects and you have a sequence of class projects
        - Done in Java
    - Confident Java programmer
    - Grading - 5 Projects (every two weeks) - 60%
      Final - 40%
    - 30% A's, 40% B's, 30% C, D, F
    - CS 143 grading scale is similar - all graduate students are removed and compute the curve based on the undergraduate students
        - Undergraduate students are always concerned about the graduate students
        - For Ph.D, you are concerned more about research and not about getting good grades

Project 1

- On the class website, we made a Virtual Machine image with pre-installed software packages
    - MySQL
    - Lucene
    - Tomcat - Java servlet
        - Java
- First project is to get familiar with the environment and sharpen up on Java and MySQL
    - First project is very easy if you know about these

HTTP Request Example:
GET /cs144/examples/show_request/ HTTP/1.1

Host: oak.cs.ucla.edu
Connection: keep-alive
Accept: text/html, application/xhtml+xml, application/xml; q=0.9, image/webp, */*; q =0.8

- In many cases, we go on a webpage and click on a particular link
    - The Referrer field generates a request

    - Server is able to figure out the sequence of the links the user clicked on to arrive at a particular field

There are a number of different methods allowed

- Two most popular ones are:

1. GET
2. POST

- Any request is encoded in the URL
    - Some requests are inserted inside the body and are in HTML forms
    - Being able to tell the difference between GET and POST is very important
- If website changes its server a lot, it will happen that some of these web search engines using the GET method will not leave any side effect on the server side.
    - In reality, if the website changes its content a lot, things might be permanently changed and damaged
- Delete method should be implemented using POST
    - If we use GET, there are potentially bad side effects
        - The web search engine doesn't know the meaning of the Delete button, so it blindly requests this
        - The assumption is that any GET request has side effects and this is okay, and all the files will be deleted.
        - Implement things correctly to avoid any side effects and avoid standard requests.

W 1 W Lec 1-6-16

- HTML
- CSS

All online communication is to use Piazza - people post anonymously and the instructors can see who wrote it

- At the end of the quarter, who helped most in terms of answering students' questions and the top contributor will get lunch
- Use virtual machine to ensure everyone is on the same environment
    - Whenever you do a project, test it in the virtual machine
    - Ensure that the code works well on the virtual machine - you could get as low as 0 points on your work
- Pick up Java as soon as possible
- Last lecture
    - HTTP: What is going on behind the scenes for that interaction to happen
    - TCP/DNS: Allows the resolution or mapping into a physical IP
    - TCP/IP Protocol: Allows communication between client machine and server machine

- HTTP Protocols
- All interactions between client and server are based on request/response paradigm
    - There is no way for the server to send something back to the client if they don't get a request, and vice versa.
    - Fundamentally, it is request/response protocol
        - What an HTTP request looks like
        - GET /index.html HTTP/1.1
        - HTTP/1.1 200 OK
    - Response and request protocols
        - By NOT requesting or allowing the server to proactively contact the client, this allows ->
            - The server needs to know the list of the clients that exist in the world and the server has to know a lot about the Internet
            - In the request/response protocol
                - Server does NOT have to know anything about the client
                - Simplifies the overall complexity of the server block -> based on the request.
- HTTP is also a stateless protocol
    - It does NOT require for the server to maintain any state for the past request it gets from the client
    - That response is purely based on that and you can ignore past requests.
    - Just based on the request/procedure, send back to the response to the client
    - Simplifies the complexities of the server
- When is request/response not ideally suited
    - Warning system - the server wants a list of people to know when something is happening i.e. a tsunami is coming
    - In order for the server to send anything to the client, it should be initiated by the client.
    - We need to communicate only when the client knows or needs something.
    - The outside event initiates this communication and it starts from the server.
    - Stock market example
        - Whenever the stock price changes, you want to send out the information to the client

        - Not necessarily suitable for HTTP protocols
        - To implement the real time/push technology, there have been a # of hacks to show that it seems like the server can push something out to the client
        - long-polling
            - streaming
            - standardized how to push this technology
            - Web Socket -> standardized techniques to show the real time pushing technique
    - stateless
        - If it is simply downloading preexisting resources from a source, that particular indicates what the page is and we can discuss what we want to generate
        - generally good enough in the past
        - Applications used today (Amazon, Facebook, Twitter) -> the actual content is generated from who I am
            - If I send a particular request to the server, the server should know that it came from the same user and change the response from who the user is.
            - HTTP doesn't maintain anything from the user or client
                - Ensure a sequence of the requests is really coming from the same client
                - The simple way to make sure that a sequence of requests comes from the same client is from cookies

- HTML
    - Kleinrock is the Father of the Internet at UCLA
        - There should be a way for users to provide this information to a webpage
        - Easy typing and easy provision of user-provided information
    - HTML Forms
        - Go to particular webpage and we want to download that form
        - Form is downloaded from CS 144 website and then we press the Submit button
            - Not from the oak server, but rather the Google server
            - Google sends back a response, and the browser displays that response
            - Some area in the webpage in which the user can provide information

- Source code
  - Starts with <!DOCTYPE html>
    - The meat occurs in the form
    - Two child elements
      - <input name="q" type="text>
      - <input type="submit">
    - Necessary ingredients for a Submit form
    - The second input element creates the Submit button you just saw
  - These are all types of input elements and we can create check boxes, etc.
  - The user can process a Submit button for information
  - We need some way to indicate where that information should be translated to.
  - User provided information is in the form element
    - action field indicates the destination
    - Those will be transmitted to the Google server
    - By default, the method is "GET"
  - The name of the input field, there can be multiple input entries: names, emails, etc.
  - Name provided is "q"
  - How is the information encoded?
    - In order to see how values are included, use the URL as part of the server
  - That is the destination, and the actual content is behind the question mark
    - URL of the action field
      - URL?q=ucla&name=John
    - Using the GET method is good enough
      - The assumption about the GET method is that we aren't concerned with safety
      - It should not leave any significant side effect on the server
      - Many of the applications have cases where the information provided actually changes on the server
  - Facebook
    - Putting up a Facebook status
      - Use a POST method

- CSS
  - Specifies how the browser should display or render

        - All the children should inherit parent properties unless it has different child properties specified for it

W 1 Dis 1-8-16
Logistics

- TR 10:30 - 11:30
    - BH 2432
- Individual/private questions?
    - amoghparam@gmail.com -> preferred email
    - aparam@cs.ucla.edu
- Game developer at Zynga
    - Machine Learning Researcher
- Worked on Draw Something
    - Involved a lot of web development so he was working a lot on Node.js, Ruby, and Java
    - He can help out a lot of connections
- Make sure Piazza questions are as clear as possible
    - Be as specific as possible and give context so they can help you
    - Do not post any code on the forums
    - Don't send code to a TA or professor
        - Go to the office hours
- Projects and Exam
    - One Final exam - 40%
    - Projects - 60%
        - Projects are graded by someone else
        - Follow the directions on the specs
        - Contact grader if you have issues with your grade
        - If problem is not solved, contact one of the TAs.
- Project Submissions
    - Test submission on VM
    - If the zip project archive and scripts don't run, you get 0 points
- 4 day grace period - no more than 2 days per project
    - If you need more days, contact Cho or your TA
    - Any additional day costs 20% of grade
    - If you submit less than 24 hours after submission deadline, you lose 20%
- You may work in a 2 person team
    - If your team dissolves, you cannot team up again
    - You and your partner submit independently, the final grade will be the minimum of both (minus another penalty of 10%)
    - You need a team.txt file

- For every project submission, include a team.txt file -> keep it the same from the first project to the end
- Make sure it is the same and follow team rules
- Include a README.txt if you want to add extra information about the project
- Make sure you cite information if you get it online or somewhere else
    - Include an acknowledgment and source where you got the solution
- Try to submit as early as possible
    - There are a lot of people submitting and people might hang up things
    - Don't lose your points!
- Web Development course
    - Working with web technology at some time in your life
    - Do not end up in a crappy web developer role
- Review
    - How does a browser show a webpage?
        - HTTP request is created by a client - browser
        - This is sent off to some server and it is handled by the server
        - The client knows what it wants but it doesn't know what the actual content is
        - Send this request off and the network knows that it is going to this specific server
        - We don't know where the actual IP address is so we will look that up.
        - When the request comes to the server, it sends it back to the HTTP response
            - The actual bytes are sent using TCP, IP, or UDP
        - HTTP (Hypertext Transfer Protocol)
            - Text that signifies something that is more than just letters or words
                - Signifies some kind of action or links to something else with associated meaning
                - Links to URL's, some buttons that you see, etc.
                    - Forms a unit of websites and is the most basic component of the worldwide web.
                    - Links to different information and websites
                - Always between clients and servers
                - Usually you have a browser that makes request and the server provides a resource
                - You can have an API that talks to Youtube and get information about a certain video
                    - Two computers talking to each other

            - Stateless protocol - anything that the client needs is represented by the request and sent through the HTTP request
                - Knows everything about the resource that is requested
        - HTTP/1.0 | HTTP / 1.1 | HTTP/2
            - HTTP/1.1 is the most common
                - wget and curl use HTTP/1.1
- HTTP
    - Communication between the client and server
        - Send back to response to get the request in the first place
        - Basically what the header looks like -> first line is the actual request followed by the header
        - Host is the name of the web server/domain name
        - User-agent specifies what kind of client it was
        - Referrer is the webpage that links to the next page
        - Accept describes the type of content that the client will take
        - Connection signifies whether you can keep a connection open or not
            - This specifies whether the connection should stay on for a longer time or not.
            - There will be overhead where it goes, connects, etc.
            - The connection will be persistent
- Why is there a host?
    - This is part of the HTTP request - it still comes to the server and the server can still see the host
        - This is redundant, no?
        - This is for a confirmation and the server can be serving multiple websites.
            - All the requests need to be served by the right website.

curl example:

- Get this resource using the HTTP 1.1 protocol
- Accept all kinds of information that the server sends back
- Tries to connect to the IP address
- Go to network and monitor the activity
    - You can see all the response headers and see the request headers, timing, etc.
    - Understand everything under the hood and fix bugs more easily

HTTP response

- Server sends a response to the client
    - Make the same request but change the option
        - curl is used to make an HTTP request to different servers
            - Useful for debugging
            - Make a request from your browser that might not be explicit
            - Send an external JSON response and different resources might be achieved
        - STATUS is 200 OK
        - Says it is all good and there is no problem with the user request
- Apache server that runs CS 144 website and Ubuntu
- PHP on the backend
- It knows how it should render the content vs text/xml or app/JSON, etc.

Status lines

- 2xx: Success - Tells you the request was successful, understood, and accepted
- 3xx: Redirection - needs something at the client level
- 4xx: Client Error - resource not found on the server i.e. 404 Not Found error
    - Generic error handler on the backend and you can have different kinds of 404 Not Founds
- 5xx: Server Error - The server failed to fulfill an apparently valid request
- ETag - how can you improve load time (caching) -> ID for the cache
    - If there is a server that is NOT on the data, it will preserve the resource that is already cached.
    - If there is a static page that is NOT changing too much, it can serve the resource much faster
- When you keep making multiple requests, it doesn't have to go to the entire server if you know the header
    - If you make 1 edit, that would change the ETag
    - When Chrome makes a request to the server, it will be different from what is cached
        - Go and fetch the actual new resource
        - This is a hash value and it is something similar to that
- Content-Length
    - Type of resource that is coming back from the server

HTTP Verbs/Methods (different HTTP requests)

- GET
    - Gets some resource back and gets a specific resource

    - Should not make any changes on your server besides giving your resource back
        - In that scenario, you should use a POST request
    - Retrieve a resource that the client requests
    - You can encode information in the URL
    - Example:
        - Google - search query that searches for information and sends it back to you
        - Check information for this query and find a string that is encoded within the URL
            - Sends it as a get request with “q=<search>”
            - Done in the address part of chrome with encoding information along with the query
- POST
    - Log in for Facebook
    - You don’t want to show the outside world your password
        - This is when you want to use a POST request
    - Performing an action on the server
    - You can use actions for PUT and DELETE just using a POST request
    - Change anything at the backend and send information to the backend
    - Even if you send an email, it is sent as part of the POST data
- PUT
    - Uploading photos to Facebook
- DELETE
    - If you want to delete from Facebook photos, they will mark that information
    - Snapchat hides information!
- HEAD
- TRACE
    - Used for mostly debugging purposes
- Used for applications on top each other

HTML

- Versions
    - HTML 5 - Current Version
    - HTML 4.01 - Most Popular
- Basic HTML
    - Markup languages - HTML, XML
        - Something that gives structure to hypertext
        - Most people think of it as an actual programming language

        - Just used to give structure to some context
        - Gives structure to the content that is requested for
- Extremely useful to developers
    - Provides a lot of features as well and provides cool features which we can go into a bit later
    - Tags + Text
    - Render content in a specific way
        - Tags are enclosed within <> brackets
- HTML Tags
    - Represent structure not style
        - The tags themselves don't really represent much besides giving attributes certain values
    - If you start a tag, you need to close it generally
        - Some tags like <img src do not need matching tags
            - You only need it if it has some textual content within that tag
            - All the information is within source
    - Attributes
        - class signifies CSS styling and id signifies unique name for the element
            - id is always unique to every element
            - class names can be similar to many elements so we can stylize them in the same way
            - src is where the image would come from
    - Comments
        - <!— This is a comment in HTML —>
        - You need to section out your HTML in comments
        - Good practice to use comments

HTML

- Example HTML

```
<!DOCTYPE html>
<html>
    <head>…</head> <!— Houses metadata about the document itself, links to JS and CSS. You can also have internal styling —>
    <body>…</body> <!— Has all the elements that make up the content of your HTML page —>
</html>
```

HTML 5

- <audio …>

- <video …>
- Provides a really good JavaScript API you can plug and play with
    - All the cool stuff would use HTML5
- Drag and drop support comes out of HTML5
    - Pretty simple with HTML5

Forms

- You have a form tag and any information you have inside of it
    - Input tags come in many different shapes and sizes
    - We have many different things for input
- Have a specific name for these things
    - First name, last name, ID's
    - If you get information in the backend, you need a specific way to identify what is the first name and what is the last name part of the data
- You will then have submit which sends out the request
- Action
    - Endpoint where you will be sending the request to
    - This is the script on the backend that handles this
    - When this request is sent, all this information is sent to submit_data_handler.php in this example

CSS (Cascading Style Sheets)

- Set of rules
- To create style for your content
- Generalize it to a lot of different elements
    - Create different rules for specific elements
    - Have common rules for a lot of different elements
    - Have content be styling requirements
- Different levels of specification
    - External - declaration rules of how it is rendered (CSS files - Document Level)
    - Internal - put rules in the style tag (Head Level)
    - Inline - have it as an element inside your HTML (Tag Level)
- GPA example - you would want a specific inline tag so you want what the header should be
    - Put specific styling information
- Most specific rule wins
    - There are multiple rules the browser can pick from and the more specific rule wins
    - If you have many general rules, but you have an exception case

        - Inline styling when you put the styling information in the tag -> this is the most specific
            - Goes in order of External -> Internal -> Inline in terms of rules
    - Try to avoid using inline tags because your content might not end up how you want it to look

Box Model

- Every HTML element can be represented as a box
- If you take a link tag or a paragraph tag
    - All the information is inside the content
    - Above that you have padding - which is the space between border and margin
    - If you have a blue color around this and you expand it, it expands until the content
    - Border color can be specified, and you have a box right around your content.
    - If you specify, you get different sizes of your border
    - Margin is space between other elements surrounding it.
- CSS layout
    - display: block
        - div, p, ul
    - display: inline
        - span, a
- This thing that creates one rule of your content is a block, and if it flows through a block, it is inline i.e. span, a

CSS (Positioning)

- static (default)
    - If you don't specify the rule, this is picked
    - It won't change if you specify "top", "bottom", "left", "right"
- relative
    - Signifies that it is positioned relative to its normal position
    - Considers the "top", "bottom", "left", "right" properties
- absolute
    - Positioned relative to its nearest positioned ancestor
- fixed
    - Positioned relative to the window viewport
    - Websites were originally built to accommodate to a website
        - Viewport is used for your mobile devices

    - You use positioning depending on four other rules -> “top”, “bottom”, “left”, “role”

CSS

- Float property: “wrap” text around the box
- Overflow property: dealing with overflow text
- Overlapping elements and z-index
    - z-index tells you which property should be shown above another
    - You want to see which one comes in front of the other one

Most styling will be done with CSS

- Inline is the most specific
- The more specific values are picked up
- Right after that, it will be picked up
- Use * to get all elements
- Have a border for every single element
    - This helps you figure out where your positioning is going on
    - border: 1px solid blue;
    - text-align: center <!— This centers everything —>
    - Use . for class names
    - Use # for id names
- Use margins to set things -> give it one specific value -> Used 100px for top, right, button, left
    - You don’t want to
    - Use auto to give enough space to figure out where the center is
    - Hacky way to center things
        - Put it in a div and text-align: center
        - You could potentially put the button in the middle with JavaScript
- HTML5 and CSS3
    - Cool features and you don’t have to write a whole lot of JavaScript to do all this stuff
- onHover or use jQuery and have document.getElementById();
    - With CSS3, it becomes a lot more simple

Prezi is written in Flash!

- Transition to different sites -> go down and add GIFs
- JavaScript is very important for your future since you can use it in the backend
    - Does NOT have classes
    - Very widely used so it is very popular, really cool course
    - Should be well-prepared to get a job and/or start your own company

    - This is a very useful and important one that you would use in real life
- Data structures and algorithms
    - Teach you how to think but not often used in real world applications.

W 2 M Lec 1-11-16
Today's Topic

- CSS Layout
- XML
    - namespace
    - DTD
- Project 2
    - Parse data and load it into XML
- Last Lecture
    - Basics of HTML
    - Attributes
        - The formatting now needs to be separated out into CSS
            - CSS is a standard for specifying how the content of the HTML document needs to be formatted
            - The listing and the specification of the tag and the actual set of rules is the declaration block
            - Use this to specify the formatting or rendering of the HTML document
        - Middle of learning bout how to use CSS specifications and provide the layout of an HTML document
- The document should have a header, menu bar, and the rest is the body (content of the document)

CSS Box Model example:

- .inline-box: the width and height specification are affected in inline elements
    - Able to specify the exact positions of these boxes to make sure they are in the right places
    - The boxes are created on the left and right side
- Use CSS position property
    - top, left, bottom, right
    - Three different ways to specify the position
        - 1. relative: from default location move it around position
        - 2. absolute: relative to parent element
        - 3. fixed: relative to current window
- The rest of actual content area should be flexible as the window size changes

        - Create one div element corresponding to the header and another div element corresponding to the content area
        - The width, height, and relative locations
            - The width should occupy as much space horizontally
                - You can use either default width or width: 100%
                - Height could be 80px if that's what we want to specify
                - Menu bar should be smaller, we have to use the calc function!
                    - calc(100% - 80px)

XML (Extensible Markup Language)

- After HTML became popular, people wanted a similar method of exchanging data
    - HTML is a way of formatting documents
    - The actual description of the content is oriented around documents
    - Many HTML documents contain interesting structured data
    - Because they are formatted as part of the document, if there is any computer program that wants to understand the type of data in there i.e. office hours, we need to write a scaper/parser that grabs meaningful data from HTML document
- It might be more useful if we come up with a standard that encodes any kinds of data
- All these structured data that we manage/store/exchange
    - Manage this kind of data in our relational database system
    - Bidding items and the descriptions of them in our relational database system
    - Exchange data with a standard way of encoding/representing the data so that people can easily recognize the type of data they are getting
    - Parse this into their own database system.
- When people created XML, they wanted to make their own de facto language with data that they need
    - Success of HTML and make that standard for XML as close to HTML as possible
    - Structured tags in XML
- XML DOM (Document Object Model)
    - A conceptual representation of the data captured by the XML document
        - Tree-based, hierarchical model
        - Create a tree with relationships between these
    - DOM: Any XML tag corresponds to a node in the XML DOM

        - Chrome, IE, Firefox will follow the standard of including the whitespace
    - It doesn't really tell us what is the overall structure that the data follows
        - Sure, you can make predictions, but you cannot make a guaranteed conclusion that the remaining data is structured the same way
    - This particular data instance doesn't describe what is the fundamental structure/schema of the information you are transmitting
        - This particular set of data may not follow the same particular structure
        - The things we mention before from this XML document can vary
            - If we want to transform a piece of XML data, how can we transform it from one format to another format
- XML namespace
    - Used to declare namespace to avoid naming conflicts
    - com.sun.Schoolcourse -> this is essentially like a package id
        - Any URL can be declared as his or her own namespace and preface all their data using that namespace

- xmlns = "http://amazon.com" >
  "http://xml.com/shopping" >
- Any element that does NOT follow the namespace could be prefixed now

DTD - Use this example and specify the schema if this data
- In case of XML, there are three kinds of things to be aware of
    - 1. tags
    - 2. elements
    - 3. attributes
- To specify the schema, you can use <!ELEMENT> to specify the type of an element

PCData - Parsed C data
- &amps is treated differently from "&amps"
- We usually don't parse attributes in this way

W 2 W Lec 1-13-16
Today's Topic
- XPath
- XML schema
- XML to RDB

- Project is done in Java
    - Hearing about new languages: Ruby on Rail, PHP, Python, node.js, etc.
    - Why do we have to learn Java for programming web applications
    - Legitimate question
        - Considering to Python or keeping it as Java
        - The reason we did Java was almost all companies look for people who have skills in C++ AND Java
            - Web development company or non-web development company
                - The two languages that companies like you to know are C++ and Java
                - The only way to learn a language is to do a lot of programming with it
                - 143 - a big part of the project is done in C++ (more training in C++)
                - 144 - more training in Java
            - Look up references in Java (a lot of clunkiness and core logic)
    - The popular questions are PHP, Python, and Ruby
        - The reason why he will switch is that you focus on core logic rather than technicalities of details
        - You will likely have to deal with Java eventually
            - Everywhere (mobile dev, web dev) is going to use Java, so you might as well get familiar with it.
            - Many companies still use Java (very fundamental skill to learn for software engineering)
    - Python is much easier to learn if you know Java
        - Easier learning curve (school is the time you are supposed to learn something more fundamental)

XML

- Represent data on the web with an extensible path
    - There is potential problems of name conflicts so we learned about XML namespace
    - How to specify a particular domain
    - XML - the user can come up with any structure and it is good to define a schema
        - We learned about DTD specifications
        - Continue learning about XML, especially regarding XML schema definitions

        - More extensible way of defining XML schema
        - Xpath
            - Lets you select a particular part of the XML document or XML data
- If a book has multiple authors, we can represent them as multiple authors using multiple children
    - CanNOT use attributes in this case because they only correspond to one element

GOOD
<book>
    <author>JC</author>
    <author>JJ</author>
</book>

BAD
<book author="JC" author="JC></book>

Q. What is a possible implication of ID and IDREFS on data model?
A. XML DOM is a tree (hierarchical) model. Most people assume this, but once you start introducing IDs and IDREFs, it becomes more a graph model because IDs can be cross referenced. Introducing ID and IDREFs means that it becomes a graph model. XML is no longer a tree-based model, but rather becomes a graph-based model.

- People hate graphs (usually)
    - The complexity of the algorithm to traverse graphs is very different and can be potentially more expensive than the tree based model
    - We still do NOT think they are real edges

Xpath

- Get rid of all whitespace and clean up the XML document

XML Schema

- Very generic markup language
- Wanted to be similar to HTML, so XML is a subset
- DTD was described to be a specification of XGML standard
    - DTD is the only standard language used to describe a document
    - DTD is getting used for describing XML
- The schema description of XML should be written in XML
- XML is able to create quite sophisticated data
    - Not really able to describe these constraints of data
    - <schema xmlns="http://www.w3.org/2001/XMLSchema">

</schema>

- Only elements belong to the default namespace
- We have to be prefix it

```
<element name="Book">
    <complexType>
        <element name="Title" type="xs:string">
        <element name="Author type="xs:string" minOccurs="1"
maxOccurs="unbounded">
        <element name="Remark" type="xs:string" minOccurs="0" max="1">
    </complexType>
</element>
```

W 2 Dis 1-15-16

XML

- Designed to store and transport data.
- Very conceptual way of organizing data
    - Schema is a method of organizing this data
    - Data that fits this schema is called an instance of that data
    - Give it one instance where you have data and it conforms to that structure.
- XML was created because it is a way to transfer data from the backend to the frontend
    - HTML represents the actual formatting of this data
- XML is used to transport data from the backend to the frontend -> pushes it to the client
    - Show the data in some form of HTML
    - Consists of tagged elements | Attributes on elements | Text
        - Give it a specific format: Machines can read it and it is very simple to read
            - Three elements
                - artist
                - album
                - sound
            - Spotify has a huge database, we are requesting for the band Metallica
                - Grabs information about Metallica
                - XML parses this and sends details for this XML
        - Your client has an XML parser that parses through this body of text and extracts information from this

            - This is going to have one album, five songs in that album, and each song has a name
                - Shows this information accordingly.
                - Builds a tree from this structure -> creates a hierarchical representation of this document.
- XML Tree
    - These tags become nodes in the tree and even the attributes become nodes in the tree
        - It has its own text node
    - We will be using xmllint: small XML client
        - This should be available on Terminal
        - Give an XML document and validate if the XML is good or bad
        - Used to conform to specifications
    - xmllint —debug music.xml
        - —debug is the option to show the XML tree
        - If you minify the XML document for you to read, this is a padding (empty content)
    - We have the album with the child of the artist
        - ATTRIBUTE title
        - Has songs within the album and some content
            - Text content
- You want to see what kind of data comes into your frontend and see how it goes.
    - Shown in a nicer way
    - All these links will be displayed later today
    - Code Beautify is a good tool and this XML is extracted from that
        - Nicer way of seeing how the XML looks like
        - A lot of people use XML and JSON
            - Depends on application you are doing

Q. When would you use
<artist>
    Metallica
</artist>

    vs

<artist name="Metallica">
</artist>

A. When you have multiple titles with the same name in an element, you would pick the 1st option

- Elements v/s Attributes
    - Can't have two attributes with the same name in an element (except?)

<musician name="Kirk Lee Hammett" instrument="guitar" stageguitars="ESP KH2">
</musican>

- If you use namespace, you can have the same names for that value
    - The entire name of the attribute includes the namespace, so we can have the same attribute name in that sense
    - Does NOT see them as having the same value

XML Namespaces (xmlns)

- Well-formed XML has a single root element
    - Has matching tags
    - Name tag can have nested child elements
- These could be empty elements that have open and close tags
    - If it doesn't contain text or another element, that is when it is empty

When would you use namespaces?
Family name analogy

- Name a first child very unique: "Baby A"
- Name a second child very unique: "Baby B"
- Name a third child very unique: "Baby C"
    - Unique within your OWN family but not within the world
    - The kid will have a unique identifier based on where he/she is from
- This is why you would use namespaces
    - Interact with multiple XML applications
    - If you have one thing that interacts with multiple XML applications, they know exactly how to interact with multiple client applications
- Child elements inherit the namespace of the parent

Namespace Example:

- Building an application for Bruin Online (undergrads). Deal with student info like id, name, language, rating
    - All the data that you get will be of this kind
        - It will be straightforward to work with this data
    - If all the data is exactly like this, it won't be hard for your application to make mistakes
- Say you are building something for grad school

    - The tags could be the same but the info contained in the tags will be different
- If you are managing data for grad students and undergrad students, you WILL need a namespace to prevent ambiguity issues
- They will NOT know which is which unless you specify the namespace
    - The namespace will uniquely identify each tag
- All of these children will inherit from their parents

The cleaner method of doing this is putting everything in an overarching tag i.e. students and then you prefix each tag with the name space that it is from

- You have to include the prefix from the children node
- If you have a huge set of data, you don't want to repeat the data that many times
    - It is much simpler to aliases in CS 143, you can have prefixes that are much shorter and easier to link
    - Default namespace
        - If we don't have the other prefixes, it will have a default namespace

Why do we use URL's for namespaces?
A. We want it to be some space where unique names could be there. You want some space that has unique names within it. In this case, URL's are ideally posted at a specific IP address.

- If 5 of us build different websites but they are hosted on the same server
- It has to be completely unique; they canNOT have the same name.
    - Most likely there will be no collisions when you are building your app
- You won't have two URL's that point to two different resources

If you have two tags that point to the same URL, do they point to the same namespace?
A. YES!

DTD (Document Type Definition)

- Client will figure out whether it is formatted in the right way or wrong way
    - DTD is used to check if XML conforms to a set of rules
    - Data has to conform to a set of rules that makes sense
        - These rules will be validated and make sure it is correctly formatted
- Age field
    - 999 years (you need a way of checking for garbage values)

        - Format things and change documents
            - This is called XML validation
- One major problem of DTD is that it can't do data type related validation
    - You cannot specify if the number should be 0 to 100
    - You could specify many things but you canNOT put restrictions on your data
        - XML schemas are an improvement because they allow you to put restrictions on data
- Internal
    - Done in the XML document
    - You have a DOCTYPE tag and the DTD definition here
- External
    - You will have data for the definition of your document
    - Specify the path for the DTD data structure
- To define these things, you say if it is an ELEMENT or an ATTLIST
    - ELEMENT contains element name
    - ATTLIST is more like an array
- Make sure all the data is escaped if you have special characters i.e. \<, \>
- Primitive Types
    - PCDATA (Parsed Character Data)
        - Parsed by XML parser, which will validate with the DTD
        - Anything inside PCDATA that has more meaning than text will be expanded
    - CDATA (Character Data)
        - CDATA will NOT be parsed by a parser
        - Used with attributes
- PCDATA | CDATA
    - Can have mixed elements as innerText

This example is NOT valid!

- You have a definition for first and last
- The XML does NOT conform to this standard!
- Unless you get to the specific occurrence modifier, it should fix a few of the errors i.e. if you add the heading tag
- If you have #REQUIRED in an attribute list, all elements have to be included within it in the actual XML file

You can have either of them or none of them

- This is why you add a * at the end, and this is an occurrence modifier that will fix the error.

- When you work with #PCDATA, make sure to have *, it gives you the option to use either just text or also child elements
    - You might want 0 or more occurrences of any kind of data
    - If you ONLY have #PCDATA, you don't need the *
- You can put occurrence operators inside this
- Order really matters
- (body header) != (header body)

Occurrence restrictions on primitive types:
- ?: zero or one occurrence
- *: zero or more occurences
- +: one or more occurrences

It says you cannot start the name with a number, you want to start with a non-digit i.e. _ or a letter!
- Same rule as in C/C++
- The ID is a primitive type and we say that if it is an ID, it has to be unique

More Primitive Types
- ID: Value of a unique ID
- IDREF(S): The value is the end of another element or lists of elements

DOCTYPE definition: Tells you what kind of file it is.
- We can see there is a DOCTYPE tag and tell you where the dtd is.
- SYSTEM can be the type of system: this feature is optional
    - We can send info without the DOCTYPE but you generally want info if you want to see where it fails
- IDRefs are unique so the ID's cannot go to the same person

DTD Pros vs Cons
- Pros:
    - Compact structure
    - Can be defined inline
    - Wide support among parsers
- Cons:
    - Are not write-in XML
    - Don't support Namespaces
    - Don't have data-typing
        - Cannot say if a specific variable is a string, int, char, Boolean, etc.

ISBM Parser

- Doesn't care if it is actually an ISBM
    - The parser just needs a way of associating this information

#IMPLIED: optional
#REQUIRED: value has to be there

XML Schema

- Written for the same reason as DTD: XML validation
    - Used to specify what is right and what is wrong
    - Improvement on DTD
        - Support for namespaces
        - Support for different data types
    - Written in XML (lazy CS people don't have to learn a new language)
- Forms the legal building blocks of XML document
- Defines # of child elements and order of child elements

Sample Schema

- Validation: XML schema needs to have its own namespace
    - XML schema namespace and based on the XML schema: we have an XML document that will be validated against these rules
        - This namespace of the actual document is the targetNamespace
        - We would have an element tag and we look at the types of data we have
        - complexType
            - Text with attributes
        - sequence
            - All of these have to be inside of the book
    - Occurrence modifiers can specify a range
        - maxOccurs
        - minOccurs
    - Attributes come within the complex type
        - Anything more than text
        - It does not have attributes
        - Going back to what we are talking about, this defines the structure of the XML document
    - use="required" is for the attributes
        - Similar to maxOccurs="1" and minOccurs="0"
        - This implies this is a key (unique)
        - You cannot have multiple ISBN's!

        - Defined in the XML schema so parsers know what this means
    - Cho posted why you would not use a namespace for this case
        - In this case, this is part of the XML schema definition.
        - Datatypes you defined can be in your own namespace.
        - We will get into some examples of these
- Does XML schema require order to be preserved?
    - Yes!

XML Schema: Validation in XML

- Types of elements
    - Simple type
        - Text + attributes
    - Complex type
        - Elements + Attributes
        - Attributes only
        - Text + Elements + Attributes
- Putting restrictions means you would create your own datatype
    - You would create this and create a simpleType
    - This restricts what this guy can hold
        - All of these are defined in the XML schema namespace
        - There is a list of all the possible names or tags that you could use
        - This is limited to 256 characters
    - There is some inheritance in this case because the primary datatype is based on string
- You can use a regex pattern to add restrictions
    - If some field had emails, you would have a restriction that matches emails and conforms to the XML type
    - Passwords have to have a minimum of {8}
        - Don't send passwords unencrypted!
    - There is a bunch of things you can go through: Link in the discussion

Complex Type

<yearBuilt era="BC">100</yearBuilt>

- This is a lot of rules for just one line
- Have simple content and you would have implied inheritance
- Take a regular expression pattern matching

Complex Type - Complex Content

- Element of type employee
- Data type is ""fullpersoninfo"

    - Sequence of first name and last name
- fullpersoninfo extends personinfo so fullpersoninfo is a child of the parent personinfo
    - Be able to look at an XML schema and write conforming XML schema for the things you read

XPath

- A way to navigate through XML and select specific elements that you want
- Navigate through nodes and grab element values
    - Contains library functions
    - Make your job easier
- Simple “path expression” to navigate across an XML tree
- This is a neat tool where you can pass in XPath expressions
    - / means start with the root element
    - /AAA
        - Gives 35 because that is the only defined value there
    - /AAA/BBB
        - Gives blank, 35, 20 in separate rows
    - // says we don’t care what path you take, take something that satisfies the following rules
        - Grabs all the possible elements
    - Recursively goes through the entire document
    - //*
        - Grabs element like a Breadth-first search
    - /AAA/BBB/*
        - Grabs all the children of BBB :)
    - /AAA/BBB/@aaa
        - Selects all BBB elements with attributes of aaa
        - Try to keep Xpath expressions as simple as possible
    - /AAA/BBB[last()]
        - Grabs the last thing that satisfies this condition
    - /AAA/BBB[1]
        - Grabs the first thing that satisfies this condition
    - /AAA/BBB[@aaa=“666”]
        - Grabs the specific element that satisfies the condition
    - You want nodes with values more than 20
        - Dot operator is for the current node, and this determines if it satisfies
        - //*[. > 20]

        - It interprets this as an AND operation because 20 did not pass this condition!

W 3 W Lec 1-20-16
Today's topic

- XML to RDB
- Content encoding
    - MIME
    - Unicode
- Project 2: CSS and visual layout of the webpage
    - Another significant part: a particular snapshot of all the auction items on eBay and formatted in XML
        - Provided XML data with human description of the data
        - Go over the details and see how the data is structured
            - Write an XML data extractor that extracts real data from XML formatted data and load it into a relational database system.
            - Very high-level overview of what we need to do for Project 2
        - Convert XML into a relational format
- XML
    - Basics of what it looks like as well as how to organize
    - How to specify a schema of an XML database
    - You can use DTD or XML Schema
        - DTD is more popular because it is easier to use
        - XML Schema is more expressive and powerful
    - Xpath expression is based on a unique path expression and you follow this path and arrive at a particular HTML element
    - Namespace
        - Specify the namespace to prevent naming conflict
        - Figure out the schema to interpret the meaning of that particular XML data
        - There are a lot of standards to understand
    - Load the data into the system so we can add more data, use it, etc.
    - How do we convert XML data into a relational database system?
        - A number of challenges
            - Mismatch of the data model
                - Two ways to look at XML data
                    - Look at XML as a textual document that we get

                - The XML can be parsed using any method i.e. XML DOM tree
                    - Hierarchical structure (tree based model)
            - Most existing data models are not based on relational model or tree-based model
                - When we get either documents or trees, we have to transform that tree into a set of tables.
                - Is there a good way to convert his information?
                    - All these theories have to design BCNF, 4NF, etc. for Databases
                - We need an algorithm that converts one set of table design to something else
            - There is no program that automatically converts, so you have to parse it manually!
    - Going over the pros and cons of these options
- XML to Relation(1)
    - Top element has Book element
        - All of them appear only once
        - Book has two required attributes: (1) ISBN (2) Price
- How should we store this data?
    - We can create a table called XMLBook
        - It has a single column i.e. doc
        - It is very easy to implement and doesn't require particular intelligence
            - The problem with this approach is that it is prone to inconsistency
            - When there is an update, you have to retrieve the data from a giant table
                - This is an enormous pain and a nuisance!
                - We have to be careful about this problem
            - Impossible to query the data
                - Normally, it should be very easy to query and extract the data
                - In CS 143, we learned SQL to grab information from tables and databases
                    - Look at the data and retrieve a particular set of student information
                    - If we store everything in there, how can we specify particular data

        - Storing XML this way gives up all the nice functionalities that a relational model provides
            - If we have to implement something in this way, this is undesirable
            - We can think of this as storing the data as a document and preserving it in that way
        - Convert this tree-based data into a table-based data
- Using relational database system, you can express much more complicated conditions
    - If you want to find all the ancestors of a particular node, then you look at pid-cid relationship, then get the join relationship for those structured queries using SQL
    - This is the solution we should use then! There is a potential problem though.
        - Everything is separated by so many tables, so you may need to join a lot of tables
            - Too verbose and inefficient
            - If you want to find the books whose title is “AA”, how can we express it?
                - It requires many joins, and this entails a lot of overhead for a simple query
                - This is something that will be issued many times, so we don’t want to make this so convoluted because it kills time efficiency
            - We preserve a lot of information, but we will have to convert something else to try to get what we need.
                - Tree structure carries the actual content of the data
                - Because we understand the semantics of the name, we care about build information and the ISBN’s.
                    - The Book table will have information that will be of great use to us that we can readily access.
                    - Write queries to select certain books might be very easy.
                    - From the book table, title is equal to something and return an individual tuple.
                - As human beings, we express content meaningfully for our data.
                - There are a few tricky situations that we have to worry about.

        - Change Author to 1 or more, change Remark to 0 or more, and Price to #IMPLIED
            - This makes Author required and Price is implied
    - We want to ask queries on particular things including using SQL
        - For a single book, can we create multiple tuples?
            - Since author can appear multiple times in a single book, it is not possible to have a variable number of columns for a particular attribute.
            - BookAuthor
                - Contains information about BookAuthor
    - How do we enforce certain constraints?
        - Foreign key constraint? NO, we cannot use this
        - Trigger or enforce it within the application itself
- Learn AJAX and JavaScript later
    - The rest of the today's lecture is about the standard that we learned.
- Content encoding
    - Exchange of data
    - Uses Content-Type header:
        - Either server or browser can be added and this indicates what kind of content (HTML, doc, image, video, audio, PDF, etc.)
        - Both browser and server can provide this information
    - Many very popular formats to store data in various forms
        - There are so many different content types, how would we make them standard compliant?
            - What type would they have to support?
            - Even if there is a browser that does NOT have a standard, it is up to a browser's implementation to decide the file formats it supports.
        - Almost all popular browsers will support popular formats.
    - The patent for GIF was done in advance and expired in 2003. Even though some companies did pay for this feature, not many companies were willing to purchase at this rate.
    - PNG was created because GIF patent holders were charging people for that usage
- If everyone is using H.264 to exchange video, what if they demand $1 million for every one minute of the video?
    - MPEG LA was very vague about their encoding, so Mozilla did NOT implement it
    - These guys are charging an enormous amount of money so you don't know the amount of money you have to pay.

        - Most smartphones are using less efficient encoding schemes, but they don't care because they don't want to worry about the browser's problems
        - If you are in the position to decide which encoding scheme to use, be mindful of this dilemma.

Multipart Form Data Example

- Single request that you send has multiple parts
- Inside the single body, you have to have multiple things and this is where multipart content is used
    - In this particular example, the particular type we use is multipart/form-data
- In any computer system, we don't have text, it deals with binary!
    - How do we represent alphabetic characters
- Similarly, we can feed it into the ASCII based program and it is unlikely that it will take off.
    - Come up with a standard that is cross-compatible
    - For UNICODE, it should work in the legacy program

W 3 Dis 1-22-16

Lesson Plan

- Character set/Content Encoding
    - Joel Spolsky started Stack Overflow and a bunch of other companies
        - "If you are a programmer working in 2003 and you don't know the basics of characters, character sets, encodings, and Unicode, and I catch you, I'm going to punish you by making you peel onions for 6 months in a submarine. I swear I will" - Joel Spolsky
    - Be aware of content encoding types and character sets available

Content Encoding

- Mapping between numeric numbers and alphabetic characters
    - EBCDIC (Extended Binary Coding Decimal Interchange Code)
    - ASCII (American Standard Code for Information Interchange)
    - ISO-8869-1 (= Latin-1)
        - DBCS (Double Byte Code Character Set)

ASCII

- 7-bit representation
    - 0-31: Unprintable control characters

- Used to represent characters in the English language i.e. non-accented alphabetic characters, digits, punctuation, and special characters
- You can represent 128 characters using 7-bit representation
    - You can represent 128 more characters with 8-bit bytes
- People thought 128 characters was enough so we wanted to know what to do with extra 128 characters
    - First 128 characters were consistent throughout the world
    - 128-255 range was very debated over!
        - Around the 1960s-70s, people wanted to use the remaining bits for extra characters to represent
            - Greece vs US: completely different character sets!
- A lot of people around the world had the same idea that you had a range of 128-255 to represent characters apart from ASCII encodings

ANSI

- ANSI Codification
    - IBM - Additional accented European characters
        - Defined an encoding scheme for 128-255
        - Added a few additional European characters
    - Many people followed IBM and included their own encodings here.
    - Created **code pages**
        - They gave each a unique number and based off the code page number, you had a bunch of character set encodings.
        - Microsoft release of DOS in Israel - 862
        - Greek - 737
    - Assuming you send an email to a person in the same region, it would be fine since everyone uses the same code pages.
    - If you send an email to a different region, it would be translated differently so it wouldn't really make sense if you send an email from one region to another.
    - It was still possible to translate in a different region if you knew the other region's translation, it would be fine.
    - In the world, there are so many different languages, especially Asian characters, that it was tough to represent this.
        - You cannot represent these characters in just 8-bits

DBCS

- Messy system because some characters were represented with 1 byte and others were represented with 2 bytes

    - If you did something like s++ and s was pointing to a string, you wouldn't have an idea of what the ASCII character was
    - Parsing backwards was a problem as well
- Developers used ANSI mixed and ANSI previous and this was painstaking

Unicode

- Character Set (Not an encoding scheme)
    - Represent all the possible characters in the world with some system
- All these characters are NOT represented by 16 bits only
- Each character represents an idea or concept and this is represented by a code point
    - Represented in your data using some encoding scheme

Example:
A -> U + 0041

- There is no limit to how many characters you can represent
    - You are representing each letter with some unique number, and you can have billions of these unique numbers
    - You only care about what code point it is since it is consistent throughout the world
- Aimed to represent all the characters in a consistent format
- Characters are just theoretical concepts
- Characters map to code points
- Unicode group tried to map almost all possible characters to the magic numbers/code points
- Macro view: 65536 characters were enough to represent most of the characters
    - Lead people to think Unicode meant 16 bits (but not true)
    - You will see some discrepancy.
    - It is unlimited and you can have an unlimited number of characters and we could potentially include that as well.

Encoding

- Actual representation of an idea (code point) into raw data (bits)
    - ASCII Encoding:
        - Support systems that were ASCII-based
        - Include ASCII characters as a subset of UNICODE
    - Make things backwards compatible with all the ASCII code
    - First 128 bits of UNICODE are identical to ASCII
        - Able to render it correctly and give you the right characters
        - Anything after >= U+0080 (first 128) will be dropped since they are NOT compatible

    - If you are using a modern text editor, it will pop out a warning saying you are in UTF-8 encoding and if you switch to ASCII, some characters might be dropped.
- UCS 2 / UTF-16
    - 2 bytes (16 bits) to store code points (maximum of 65536 chars)
    - Minor differences that we don't really have to worry about.
        - Create an encoding scheme and represent the character with two bytes
        - This is also part of the confusion because of UTF-16
            - You can actually have more than 16 bits but it is convention to just have 16 bits
    - Different architectures for big endian and little endian

UTF-8

- Problems with UTF-8
    - Wastes a lot of bytes
        - The first byte for almost all ASCII characters is 00 (unused!)
    - Each ASCII characters are represented by a single byte
    - Remaining characters could be represented by 2,3,4,5,6 bytes for other characters
        - Sequence of bytes will tell you what the next character is for the specific character
- You have the first byte as a "count" byte and you can find out what the byte represents
- These bytes start with 11..0:
    - If you have a sequence of bytes and binary digit
- The # of leading 1's indicates the # of bytes

Example:
ASCII - 0110xxxx
Unicode (not ASCII) -

2 bytes
110xxxxx 1011xxxx

3 bytes
1110xxxx 1011xxxx 1011xxxx

- Anything greater than 7F will have a count byte
- Observations about UTF-8

    - No null bytes. All ASCII characters (0-127) are the same.
    - Unicode characters start with “1” as the high bit, and can be ignored if compiled in ASCII format

<html>
<head>
<meta http-equiv=“Content-Type” content=“text/html;
charset=utf-8”>

- Encoding you use will affect the representation dramatically
    - Set charset=utf-8

MIME (Multipurpose Internet Mail Extensions)

- Needed some format to understand what content is being sent in the email
- Used for other web services i.e. interacting with the client and used almost everywhere now.
    - The way each part of the message is specified
- Default is US-ASCII
- Text is the main type and the plain type represents the subtype.
    - vnd means vendor specific
        - If you are using another client that isn’t a browser, you would use something like this.
    - x means that it is NOT part of the actual specification
- File signatures: first few characters of the document will tell you the kind of file it is
    - This figures out it is a PDF and we can determine if this is true or not.
    - It is usually the first part when you open the file in terminal or Sublime and see the file signature in first two bytes (or more bytes!)

W 4 M Lec 1-25-16

- Project 2 is due this Friday
    - Get XML data and convert it into Relational Database System
    - Get it to match specifications based on CSS
- Review:
    - How to interpret bits as character data
    - Translating XML to databases
- In computer systems, we come up with some ways to organize programs
- Based on fundamental ideas that is important is to remember the names of each idea that we learn

    - UNICODE: How we encode or translate our characters into a sequence of bits

        - Before UNICODE, different countries had different standards, which obviously got very disorganized
        - One-to-one mapping with UNICODE, so having a 2-byte representation for each character was sufficient
    - MIME: Transmit a sequence of bits with an image encoded in JPEG standard
        - These are the two important standards being used on the Internet
        - There are certain terms you have to remember in order to succeed

- Java came after UNICODE, so we don't have to worry about that
- C/C++ are older than UNICODE, so we need certain functions to ensure that the data we are dealing with is UNICODE.
    - char_t
    - wcslen: Measures the UNICODE UTF-16 data length
- Many programs nowadays use UTF-8

Web Server Architecture

- What should happen with a standard request to a web server?
    - Interact with the class website
        - Type in a URL, send a request, get a response in return
    - The class website has to send back information from somewhere on the server
    - Sends the content back to the browser
        - Maps that URL to a location of that particular file and sends the content of a file back to the browser
        - This is called a **static website**
            - Just has a mapping function between file and URL and sends it back
            - Many web servers are good at waiting for HTTP requests and sending information back.
    - Apache maps the request and sends back a particular file
        - It has to know how the particular URL maps to a location of the file.
        - Any websites by default need a root directory location in which the files are located
        - Interpret this as relative to the ~/document location
            - If it is within this path, it can deal with static web content

- Specify mapping is a very easy job for servers to do
  - DocumentRoot/var/www/html: This is the general format
- Other websites are dynamic websites
  - Amazon, Youtube, Facebook
  - There isn't a particular static page that you keep downloading
  - Content changes almost constantly
    - These websites aren't about mapping the URL to a particular location
    - Run program to generate something on the fly
    - There should be some process that can invoke a particular thread so we know what kind of program to run to generate that content.
  - It depends on the URL and depending on where we visit.
    - There are a bunch of processes that delivers the user requests
    - Knows which process has a resource, so the system can readily access it.
  - Makes it easy to retrieve bits and pieces, and there is a program that understands the logic to get a request
- Inherent weaknesses of the Internet
  - Everything is transmitted in the clear, so people can intercept things and we want to avoid this.
    - Both parties have to encrypt their data so no one can actually look at it.
    - Many websites use encrypted HTTP protocols to safeguard user privacy
    - Both requests will have user-related information
      - Status update: Need to transmit it through HTTP
      - We have to do parsing and if some framework does that conversion automatically, we don't have to worry about as much.
      - Invoke different processes per user request
        - Just worry about the programming logic
        - Combine pieces of information and generate a certain output.
        - Server architects will be the ones who have to worry about handling all this crap related to logistics

Keyword Based Search

- When we look for information, what do we do?
    - If we heard about “optical flow”, and we want to learn about it, what do we search for?
    - Hopefully, we will acquire some interesting information based on that keyword.
    - We are so used to this keyword based interface that we expect a search bar wherever we want to retrieve some information
- This is our primary expectation
    - Because we are so used to it, when you build a web application, you have to have a search box so people can find whatever page they want.
        - This is our expectation.
    - What are the things going on behind on the scenes on the website to support a search interface?
        - The implementation is reasonably straightforward, but there is a lot of investigations behind the scenes and this is our high-level goal.
    - Before we talk about keyword-based search interface, we want to use our computers for number crunching.
        - Very good at crunching numbers
        - Networking users utilize computers as a communication device
        - People in graphics think it is a visualization tool using a library
            - Virtual systems inside a computer that people enjoy
        - Information management
            - Computer is an agent that helps us to deal with information
            - From the birth of humanity, we have had a need for processing information
                - Hunting animals in those days and gathering vegetables
            - Before humans developed language, their knowledge was limited to their experience.
        - We were thinking about how to use the computer to store information.
            - How can we store information and retrieve it later?
                - File system: managed in a hierarchical tree structure and we can always retrieve it back later
                    - Digital: more reliable, store more information, we don’t have to move around
                    - Whatever we retrieve later is exactly what we stored. Use computers in more ways, how else can we store more information
                - Relational databases

        - Tables
            - Columns and rows
            - Very well-defined schema of a table, which is nothing more than a collection of tuples
            - This is useful and popular because it gave way to SQL, which allows us to specify what kind of data that I want.
        - Because computers have clearly defined semantics, the computer goes over the table in brute force ways to identify tuples
            - In the past, in order to retrieve the kinds of info that we need, humans had to go over this data
                - Computers are good at this, so this is much more efficient than a human parsing through all the data
    - RDB
        - Stores the data using the schema
- Artificial Intelligence
    - Expert Systems
        - Well-defined indication of people's needs and brute force searching.
        - In case of Expert Systems, they wanted us to do significantly more (using predicate logic)
            - LiveWith(John, Christine)
            - LiveWith(John, James)
            - LiveWith(A, B) ^ LiveWith(B, C) -> LiveWith(A, C)
        - Language semantics combine two knowledge representation bases together
        - Granularity may be completely different depending on who developed it.
            - This is why Expert Systems did not become very popular.
- Information Retrieval (IR) System
    - Successful approach but this one has its own limitations for a # of reasons
    - What fraction of information is stored in RDB?
        - A very small fraction!

        - Most data today is represented in plain text formats
    - Mapping our conceptual need into SQL is not necessarily a straightforward test
        - CS 143 exams :(
    - Our data representation is whatever that is already there
        - All plain text documents that already exists in the world
        - Given the difficulty people have learning SQL, we don't need any of this formal logic.
            - We just need plain language that we use today
            - System will magically try to understand these individual documents and understand the common English language query and identify what language captures that query well.
    - If IR System works well, we tap into a bunch of sources and humanity will benefit a lot.
    - The problem is that as human beings, we don't have a good theory of how to understand the document and what individual documents mean
- The simplest possible thing is to interpret any document that contains those keywords!
    - As long as the document contains those keywords, there is a valid match
        - This is considered a Boolean model
            - "bag of words": if we find a match, we don't care about the order, just as long as they appear inside the document
                - If they don't appear, we consider this to be a mismatch.
        - This worked pretty well in the early days of information retrieval research
        - As a computer scientist/software engineer, we have to implement this model on our system to return a value in a reasonable amount of time
        - The system can actually look at every single document and return it to the user.
            - If we actually implement the Boolean model in a naive way, it will take several days to return relevant results.
    - We need a more efficient way to come up with an efficient algorithm.
- Precision/recall
    - What is identified as matching documents is likely to be different from true matching documents.
    - precision (p) = |R intersect D| / |D|

        - Out of truly relevant documents, what fraction are returned by the system
            - How much of the truly relevant documents does the system recall
                - recall (r) = |R intersect D| / |R|

Inverted Index

- Allows quick lookup of document ids with a particular word
- Every word is indexed with a pointer
    - Contains a particular keyword that is pointed by
    - These are the document ids that contain certain keywords
    - Talk about how we can construct this data later on
- How can the user process this and return all the document ids here.

W 4 W Lec 1-27-16

Today's topic

- Information retrieval
    - Model of the data is represented as textual documents (numbered sequence of words)
        - Interface uses everyday language
        - Try to exploit the existing collection of documents and use the most familiar interface.
        - Puts all the challenges of identifying particular documents to the system.
        - Computers are dumb and don't know how to interpret the language.
        - Make certain assumptions of how to interpret the query.
    - Look for bag of words
        - How did we interpret the intention
        - Check if that document contains that keyword or not
        - It does not truly capture the user's intention, we have to do one thing
            - This particular model is called Boolean model
            - This simplification does NOT capture what is truly relevant!
                - It may or may not be the same as what the user wants
                - To measure how well the user performs, we came up with two metrics to indicate if they are doing well.
    - Precision/recall
        - Precision(p) = |R intersects D| / |D|
            - How many documents are truly relevant that are returned
        - Recall(r) = |R intersects D| / |R|

        - Out of the documents that you are supposed to return, how many of them are relevant
    - Given a user query, go to every document and look for a particular keyword.
        - See if it there or not, and decide to return it if possible.
    - If they have to do this task for every single query, it will take a very long time to retrieve a subquery.
        - Come up with some simple/ingenious ways of performing a test very quickly.
    - Inverted Index
        - These allow for quick lookup of document ids with a particular word
        - From document, you want to identify the keywords that were inputted in it.
        - Can we go from document -> keyword, or keyword -> document? The natural mapping is inverted and this is made efficient by the inverted index.
            - You can take the intersection between two sets and return the intersection that contains both key words.
            - Check if word exists in the DIC, and create a postings in DIC
            - Keep going until you run out of documents in your collection
    - 400 GB is definitely much larger than 34 GB
        - All these things, at some point, will accumulate
        - What can we do in strange scenarios
        - Put the rest of the postings list in the disk, an when you need it, you put it in main memory

Size of Inverted Index

- Look for posting ID's that contain they keywords by looking for the intersection of the postings.
    - Look up posting entry with PID and return this to the user
    - Helps to avoid scanning the entire table.

Project 3

- Support searches efficiently
- Create an inverted index and perform the searches efficiently using multiple conditions together
- A lot of work because you have to implement an inverted index

    - Created a B+ tree and it is painful to implement and combine them together
    - We have to create an inverted index and we use an excellent library called Lucene that creates most of the inverted index for you
        - Feeds some keywords and you have to look up some matching entries
        - Learn how to use this well-known library to do this.
        - Any application in the real world requires an inverted index
    - Performance will be so horrible that users will now use it.
- Lucene is a very good option in which you can just grab the open source code and learn how to use it.

Boolean Model

- Apply it and identify all documents that contain UCLA?
    - There could be billions of documents that contain UCLA
    - Does Boolean model tell us which is more or less relevant?
        - No it does NOT!
        - Every object is either a match or not a match
    - With a million matching documents, we have a problem. It is not a good enough solution in that case.
    - What is more relevant that requires more of your attention and take a look later!
    - Rank the documents based on how relevant they are!

Vector Model

- This is what we do for the document under the Boolean model
    - Not enough for our goal right now.
    - We want to be able to rank the documents based on how relevant they are to a particular keyword.
    - It either has zeros or ones, so we cannot rank them based on that representation!
- How do we show the relevance of a document to be able to incorporate relevance.
    - Instead of using zeros and ones, we can use a continuous # to represent how relevant it is to a particular keyword!
- Generalizes a binary vector into real valued vectors (each has a particular weighting from 0 to 1)
- If the query is UCLA, what do we need to do?
    - Sort the documents based on the column value.

    - Once you incorporate the relevancy value, we can compute the relevance since it is already there.
    - Construct the matrix and then you can rank them.
- The big question we are asking is how do we get the relevance?
    - The real question is how do we obtain the relevant scores for every document and keyword?
        - How do we get the #? People can assign these arbitrary values.
        - Human minds are easy to fool so the arbitrary #'s cannot be fully trusted.
    - Somehow, we need some way to automatically generate those relevancy scores.
- How do we identify the relevant scores?
- The broad concept seems to be more relevant to that thing.
    - Guideline to keep in mind. We want to penalize the broader concept so it doesn't skew our search query.
    - We need more specific guidelines for a particular concept.
    - Some mathematical definition to capture the essence of this idea.
    - What hint in the data set can we use?
        - If something is a broader concept, what kind of observation can we see.
        - The list of the document related to the keyword is longer.

Inverse Document Frequency (IDF)

- Given a particular term, how many documents contain that term (Document Frequency)
    - If document frequency is high, it is a broader concept.
    - We want to penalize broader concepts, so if DF is large, we want to make the weight small
    - Algorithm:
        - log (1/|DF|)
- In general, if a document is genuinely written and use the words whenever it is appropriate, more times than not, it will be more relevant!
    - Incorporate that idea into relevance weighting scheme, this is known as term frequency (TF)
    - How many times a term appears (TF)
- Mainly about how specific a term is.
- Term frequency: How relevant a document is for a specific concept
- Multiply TF * log (1/|DF|) and use this as our weighting mechanism for vector model
- Document frequency is specific to a term

    - Term frequency is specific to a document
- Our next task is how to rank a document
    - If we see “UCLA”, how do we rank these categories?
        - What if my category is a two keyword category i.e. “Princeton University” and “Harvard University”
        - Rank based on a combined category (worry about which is more general and which is more specific)
            - Give some more weight and when we look at specificities, take into account the waiting scheme
            - There is no reason to think one is more important than another, so just sum them up and sort them based on the sum.

Cosine Similarity

- We have document vectors and we have a query
  Q = Princeton University
- Q = (….., 1, ….1)
- Look at handwritten notes
- We may want to keep the size of the document
    - If there is a document that is really long and contains every existing word in the world, that document will have very big scores all-around
    - If we rank the document based on this, then that document will essentially dominate everyone else because it contains so many words many times.
- The ones that have many words need to be penalized, otherwise people will create longer and longer documents just to be highly ranked
- Dividing that penalizes the longer words, so there is no unfair advantage for longer documents.
- Divide this by the size of the document, and that is **cosine similarity**

W 4 Dis 1-29-16

- A seller’s location may or may not be the same as an item’s location.
- A seller is based on a particular city, state, and he/she may use a warehouse located somewhere else. So if a seller’s location is not specified in our data, please assume that we simply do not know his or her location.
- Similarly, we do not know ht location information of most (If not all) bidders.
- Make sure to submit the best submission by 12 o'clock

Lesson Plan

- Basics of databases, normal forms, things that we are familiar with right now.

    - More in-depth discussion
    - Library provided by Apache that does information retrieval
- Next week, the first 20 minutes
    - Generating code examples for next week and watch out for that.
    - Normalization of functional dependencies
- RDBMS: Functional dependencies
    - A functional dependency (FD) on a relation R is a statement of the form “If two tuples of R agree on all of the attributes A1, A2, …, An, then they must also agree on all of (another list of) attributes B1, B2, … Bm.
    - If there is a functional dependency, you will have repeated values for multiple sets
- Star Wars -> there isn’t a functional dependency on three different actors

Keys

- Keys of Relations
    - Set of one or more attributes that uniquely identify data in your rows
        - There is no proper subset of A1..An that functionally determines all other attributes of R {it is minimal}.
        - Uniquely determines which actor you are talking about.
- Primary Key
    - Attribute or set of attributes that uniquely identifies a record
        - Example: ItemID is a primary key
    - If it was a composition of two or more attribute, that would be a composite key
        - Example: Movie, title, actorName
    - Candidate Key or Alternate Key
        - Attribute or set of attributes that defines a row
            - Not a primary key, but it has potential to be a primary key.
            - You can have multiple combinations of things that uniquely identify rows.
            - Any of these combinations that makes sense could be considered a primary key
                - The rest become alternate keys
- Superkey
    - If you had a key appended with other attributes, it would still be a super key.
        - Examples:
            - title, year, starName
            - title, year, starName, length
- Foreign Key

    - A Primary Key in another table that is in a field that the table is considered

Functional Dependencies

- Types
    - Trivial - In the trivial case, you are saying that if you have an FD X in Y, X is a subset of Y
- Armstrong's Axioms:
    - If you have a set of attributes, then the latter set of attributes determines the former set of attributes
        - A_1 … A_n determines B_1 … B_m
    - Augmentation
        - A_1…A_n -> B_1….B_m
        - You can append additional attributes in the same order and a functional dependency will still hold.
    - Transitivity:
        - A_1 …A_n -> B_1….B_m, C_1 -> C_k
        - Transitive property in math
- Closure of a Functional Dependency
    - You can find out candidate keys using closures more systematically
    - Closure of A, denoted as A+, is set of all regular FDs that can be derived from it
- Review 143 notes

Candidate Keys

- Other sets of attributes that could potentially be primary keys
    - Let F be a set of FDs, and R a relation
- AD would be a good primary key because you can grab everything from Amogh's example

Need for Normalization

- You want to avoid redundancy because it is tougher to scale
    - Robust enough to prevent any anomalies.
    - Update anomalies
        - Scattered throughout the world having multiple tables
        - If you have certain things that reference each other, it might NOT update other things
    - Deletion anomalies
        - Parts of it were left undeleted because they were saved somewhere else.
    - Insert anomalies

        - Insert data but that record doesn't exist, so there is some inconsistency with that database.
        - Break down their table into something that is manageable.

Normalization

- Boyce Codd Normal Form (BCNF)
    - The only possible functional dependencies in BCNF are those where the left-hand side is a super key
        - If it is not a super key, it is NOT in BCNF
    - Our previous relation is NOT in BCNF, since for FD:
        - title, year -> length, genre, studioName

BCNF Decomposition Algorithm

- Input: Relation and set of FDs
- Output: Decomposition of R_0 into a collection of relations
- Start off with R = R_0 and S = S_0
- Look at slides for algorithm

Information Retrieval (IR)

- Finding material (usually documents) of an unstructured nature, satisfies an information need from within large collections
    - Google search
    - Querying data
- Use it for some structured format (schema)
    - Unstructured data is just a bunch of text that doesn't really mean anything to a system that is looking at it.
    - The computer just sees text documents as metadata and a sequence of characters
    - Information need
        - Learn how to cook pasta, a bunch of documents that talk about a bunch of different things
        - How do you filter the specific set of documents that are good for you.
    - Cannot store in a database, but you can use intelligent ways to compare documents and queries on some level.
        - Figure out which document is most important to whichever query.
        - Large collection is dozens and dozens of text data.
        - Not limited to web search, use information retrieval for things like shopping (eBay, Amazon, etc.)
        - Natural language queries that forms the most relevant query

            - Natural query
                - Similar to natural language or Boolean queries
    - Mac OS X - El Capitan
    - Use Spotlight Search
        - Natural Language questions
            - Email I sent to Frank four days ago
            - Can find the email you actually sent
        - Professor Cho's main area of research
            - Probability theory, linear algebra, some calculus
            - Very fun area where it has some machine learning involved
                - A lot of natural language processing involved

- Huge collection of documents
    - You assume you have a ton of documents to run your models/algorithms on.
    - The goal is to retrieve information that is relevant to the need of the user
- IR - Classic Search Model
    - Look at Slides
    - Data it has on its collection
        - Runs its algorithm and tries to find its most important results
        - Sometimes, there is a requirement to reformulate the query in some way to reformat the documents.
    - Format
        - User task
        - Info need
        - Query
    - A few things that could possibly go wrong
        - Misconception about the information given
            - The computer doesn't know what alive or killing it means (it is just looking at documents and queries)
            - There is a need to create robust systems
            - See how to evaluate a specific retrieval system without getting into the model
- Precision and Recall
    - Precision
        - Fraction of the documents that are actually relevant
        - If my information retrieval system returns X documents, how many are actually relevant?
            - What is the % that is actually relevant to my need
    - Recall

        - Fraction of the relevant documents returned
        - Out of all the possible relevant documents returned, how many are returned by my query?
            - Assuming that your information system returns 10 documents and there are only 10 documents on a topic in the world. Your recall is 100%!
        - Based on all the information it returns, you will probably use a combination of these methods and use two things.
    - Depending on the actual application, you will want to group these things differently.
    - Use something like a harmonic mean called the f-score

IR Problem Algorithms

- Manually read through and find
- Grepping/String matching
    - Run a grep command on Brutus, Caesar, and Calpurnia
    - For this body of text, it would be a reasonable solution without too many words.
    - This system would NOT really scale if you had a ton of words and documents.
- Need to build things that scale in a good way.
    - Not flexible enough for the NOT operators
    - You could obviously NOT use grepping and string matching
    - What does “regular expression” count as
- Other requirements:
    - Which result is more valuable than other results
    - Need a ranked retrieval
        - Find which answer is more relevant than the others
        - Automated system to do this on a big scope
        - Find the best string matching algorithm and handle a lot of documents
        - You won’t have ranked retrieval and you won’t be able to differentiate which is more important to you.
- Build an Index!
    - Binary Model
        - Boolean retrieval model
            - Look at an entire big body of text
            - Create a ginormous table and run a page rank count through this.
            - Binary because it comes through one or zero in the term

        - Have all the words and all the possible documents and build something like this
    - This is NOT the best model
        - You have so many different documents that it would just explode in the size of the matrix
        - You can formally pose any query which is in the form of a Boolean expression of terms
            - Previous need was that you had a bunch of queries and you had to run that against a system of documents. Not a super good solution but we will see how that is done in this case.
- Call each a vector
    - Look at the vector, take the complement and do a bit-wise AND on each.
- Space Calculation
    - Uses up a large amount of space
        - If you have 1 million documents, let's add a constraint saying each of these documents cannot have more than 1,000 words.
        - You will have a lot of textual data
            - Much bigger documents would involve more data
            - If you had distinct terms like 500K distinct terms, this would be 500K * 1 M => 0.5 trillion 0s and 1s
    - This uses up way too much space, the matrix is so sparse
        - Doesn't give much information and it is TOO big.
        - Takes up a lot of space.
    - Let's assume we need a better solution, let's try to figure out the same thing to be a little more efficient
- Inverted Index
    - Key idea for information retrieval and forms the basis of almost all the things that come after that
    - Key data structure that underlies modern IR
    - Exploits the sparsity of the term document matrix
    - Supports efficient retrieval
        - Give each term a document ID
        - Then, do the same thing we did in the binary model (create distinct terms)
            - Find where each occurs instead of using a big matrix
            - Get a list of document IDs

        - A bunch of ways to actually store the document IDs with fixed size arrays
    - You don't want to see fixed size arrays
        - You want to use Postings Lists!
            - These are either linked lists or variable size arrays
            - Dictionary is stored in memory, while the postings are stored in the disk
                - Every document would take up a lot of space
    - Organization
        - Index Construction
        - Documents to be indexed
        - Token Stream
            - Get all the possible terms with one list of unique words
            - You would have linguistic modules that would run text processing on top of this
        - Modified Tokens
            - Linguistic Modules
                - Break it down into the root of the word
                - Apply text processing and use those in the indexer
        - Inverted Index
    - Text Preprocessing Steps
        - Tokenization
            - Character streams are tokenized into word tokens.
        - Normalization
            - United States of America -> U.S.A -> U.S -> US -> USA
        - Stemming
            - Authorization -> authorize
            - Friends -> friend
        - Stop Words
            - a, an, the, to, be, and, but, of
    - This entire thing would come many, many times more.
        - These common phases come a lot more times and don't want to influence retrieval using these stop words.
        - Let's say you are searching and building indices
            - You can potentially reinclude Stop Words for basic natural language processing
- Steps for Indexing

    - Tokenize them (run a bunch of text processing modules and take them down to a unique list of terms)
        - List of DocID that they occur in
            - Caesar
            - Once you have this entire list, sort the terms and try to reduce the # of occurrences
            - Split the dictionary and postings
        - Look at document frequency and count
    - Tokenize everything and get a bunch of words, sort things
    - Mergesort -> the way merge works is that it merges based on them being sorted
- If you want to merge, you can start off with two pointers, and then from there, you can put the 2 down there and move it forward.
    - Keep searching and find the index
    - Change the merging algorithm
- Four TA's
    - Next week is the last week teaching you guys, but Amogh will have his last week next week.
- First 20 minutes will be for 4NF.

W 5 M Lec 2-1-16

- Project 3 will be posted soon
    - Provide support for interesting queries in this project
    - Another class of query is keyword based query
        - People provide a set of keywords and you identify documents
        - There can somehow contain those kinds of keywords
    - Spacial queries
        - These days, now that we are so used to mobile devices, a lot of data we obtain is related to geographic location
        - If you want to locate items somewhere for faster shipping, you would use location/spacial queries
            - Identify things close to a particular location
    - For each, you would use a # of different things
        - B+ tree or whatever index
        - Keyword index
        - Spacial index
    - Once you create the indices for supporting these different queries, you want to expose or export the functionality so that the clients are able to access it.
    - Use standard web service protocols.

    - Types of approach
        - REST
        - **SOAP**: We will be using this standard instead of a REST API
            - There are very well-known packages/libraries you can use
            - Put Java class into that library and it is easy to incorporate
        - This shows how convenient things are when supporting new libraries
- We learn about queries and in the next lecture, we will go over SOAP
    - Last lecture, we learned a more advanced version of Boolean model
        - Vector space model
        - If there is a lot of matching documents, the user would have to go over all the documents in a Boolean model.
            - Rather than looking for an individual module, we look at the documents as a vector of real #'s that correspond to how relevant the documents are to the search query.
        - The most popular scheme we learned in the last lecture was inverse document frequency (IDF)
            - Relevance uses term frequency (TF): Used to measure the relevance of the documents by the keywords.
    - Most systems use some variation of TF-IDF waiting scheme
        - When we create this vector for every 2-D matrix, we have to rank the documents.
        - For a single keyword, it is very simple, but other keywords, it is much more difficult.
    - Use dot product between query vector and document vector to find the cosine.
    - If a document has a lot of different words, we have to rank the document manually.

- Q = (Princeton, Univ)
- D_1 = (car, racing)
- D_2
- D_3
- IDF(Prin) = 1
- IDF(car) = IDF(racing) = IDF(univ) = 0
    - I f you have a billion documents, can you pre-process?
    - Yes, of course!
    - There are billions of combinations and you have to save it all up and you get
- How many of documents are likely to be relevant?

    - Only a handful because there is simply too many objects
- Given a particular query, can you quickly identify queries where the output is close ?
    - Query and document show have >= one word
- You can filter out the vast majority of documents that immediately tells us if a document is a keyword or not?
    - Inverted index!
- Instead of scanning the entire document collection, what can we do?
    - We need to know these entries in the worst case
        - Know the TF values and their IDF values
        - How should we update our inverted index?
            - Do we need to know all of the entries?
                - No! Some of the queries are NOT relevant.
                - The rest are zeroed out.
            - These entries corresponding to non-zero terms will survive.
            - The only entries we need to know are TF and IDF values.
    - TF, IDF values
        - Know the mathematics (32A) behind this!
- If you store information this way, is the value exactly the same as when you go over the entire table?
    - Yes, you have that guarantee.
    - Still satisfies the actual values that are supposed to be printed out.
- It is a matter of cleanly categorizing the elements that are returned.
    - Some things are scored as 0, other things are scored as 1
    - In this case, a returned or matching document is NOT really clear.
    - So how do we actually measure precision and recall?
        - We could use a metric seeing how many are truly relevant at a range i.e. the relevant Top 10, 20. etc.
    - TF, IDF becomes essential when the document gets really BIG!
- In terms of whether you missed something, you don't care!
    - You only care about the upper-tier of a certain value.
- We need the computer system to figure things out automatically, and we need some ways of getting hints to find keywords associated with the document.
    - Whoever comes in can get paid, and they will associate that keyword with that specific URL.
    - Whoever pays more will able to associate that particular keyword.
    - Scheme that we still use today, not independent search engine.
        - In order to use this as a Google or a Bing, it has to do a good jump for regular searches, NOT just niche searches.
- How do we figure out the expected page to be returned?

    - One thing that turned out to work very well is to exploit the link structure.
    - If you go to your personal home page, you can highlight it and point it to another detailed option (hyperlink)
    - Highlighted anchor text seems to be a very good summary of what that linked to page is about
        - That is the page that we expect to see for that particular keyword.
        - We want to see how relevant it is for a particular page.
- Linked-based
    - PageRank
    - We may want to compare TF and IDF to get very good representations of what things are about.
    - When you rank the document, for every page, count how many other pages are pointing to it.
        - Those with more page references will have a higher weight.
        - The problem is that NOT every link is the same.
        - If people figure out how search engines are based on that, they will create bogus links!
    - We want to give higher (non-zero) weight coming only from trustable, high-quality pages.
- A page is important if a lot of other pages are pointing to it.
    - The importance of a page is defined to be a sum of PageRanks pointing to you
- PageRank
    - If there are a lot of important pages pointing to you, then you want your PageRank to also be high.
    - A PageRank of p is defined as follows:
        - PR(p) = PR(p_1) / c_1 + … + PR(p_k) / C_k
            - p_i: page pointing to p,
            - c_i: # of links in p_i
- Example: Web of 1842
    - Netscape, Microsoft and Amazon
    - Look at the slides
- Multiply with the matrix, the product of these two matrices is the same as the input.
    - When we get the input, we arrived at the solutions.
    - If those are real values, and given the solution, then that value is the solution.
        - Multiply initial random value with matrix and we get some value as output.
        - Multiply the new value with that matrix and get a new value.

        - Plug it into the equation again and get the value.
    - We repeat this process until it converges (the difference is negligible)
- This is called iterative calculation of PageRank

Damping Factors

- For every page, instead of assuming that each page points to only the page it points to, we guess with some probability that PageRank will propagate to everyone else
    - Original PageRank equation ignoring this part is web graph matrix multiplied by PageRank.
    - With remaining PageRank, we are distributing everything to other pages.
    - This # of 0.8 is called a damping factor.
- We compute it based on 80% of the structure, and distribute the remaining 20% throughout the pages

A page is important if a lot of other important pages are pointing to it.

- Matrix equation that we can determine through iterative computation
- We can use the random web surfer model as an alternative solution.

PageRank: Random Surfer Model

- Assuming a completely random web surfer, PageRank is determined by the probability that the random web surfer will arrive at that particular page.
- A random web surfer just goes to any random webpage as the initial starting point
    - The person then looks at existing links on the page, and the person clicks any of those links with equal probability.
    - Keep repeating this process and clicks one of these links with equal probability.
- Assuming this kind of random surfer, it is a simplified model of how to browse the web.
    - For this kind of user, PageRank is equal to the probability of landing at that page.

W 5 Dis 2-5-16

- Most of the discussion will be about ranked retrieval
- Step through a Lucene example and see how we can build indices from our database.
- Try to see how we create indices from the data

Normal Form: Fourth Normal Form (4NF) and Multivalued Dependencies

- Consequence of normalizing into 4NF
- First time you look at the database is to make sure it doesn't have duplicate values for a tuple.
- {Actors, Cars, Houses}
    - You cannot have a value for Lamborghinis, Porsches, etc.
    - Break multi-valued set attribute into multiple atomic values
    - Have multiple tuples for multiple values
- Actor1, Car1, Actor2, Car2, etc.
- If multiple values can have multiple attributes, that is where the problem creeps in.
- When you want to represent this, you repeat each for attribute values as well.
- Repeat all of the values of one multivalued attribute with another.
    - These are independent attributes.
    - It is both independent and you have to repeat them even though they aren't stable.
- Exists when there are two or more multivalued dependency

EMP

- He can work on multiple projects and have multiple dependencies
- The projects and the dependencies are not related
    - Non-trivial multivalued dependency: So much redundancy here
    - Still in BCNF and there are NO unneeded functions
    - This is still redundant even though it is in a fairly normalized form.
- Make sure there aren't so many redundancies.
- t1[X] = t2[X], this implies that
    - t3[X] = t4[X] = t1[X] = t2[X]
    - t3[Y] = t1[Y] and t2[Y] = t4[Y]
    - t4[Z] = t1[Z} and t2[Z] = t3[Z]
- Dname is outside of Pname and Ename
    - You have two multivalued dependencies

Multivalued Dependency

- If X ->> Y
- Possible conditions
    - Y is a subset of X (Y $\subseteq$ X)
    - X U Y = R
- Trivial MVDs:
    - Where 1 or 2 holds
- Nontrivial MVDs:
    - Creeps in later on

Trivial Multivalued Dependency
EMP_Projects

- Have it for departments as well
- This is essentially what the nontrivial multivalued dependency looks like
    - X U Y != whole relation, NOT a trivial multivalued dependency

Break it down into something more normalized

Information Retrieval - Ranked Retrieval

- Boolean Model
    - Huge term-instance matrix
        - Have individual terms and documents and a 1-0 matrix where a specific term occurred or not.
        - Consider row vectors, perform Boolean operations.
    - They either match or they don't match
        - AND query, you only have documents that match both
        - You might NOT get any term
    - Get all of Brutus or all of Caesar
    - You would have to be really good at writing this Boolean queries
        - Not good for majority of the users
        - They wouldn't be able to write the Boolean queries well.
    - Pretty complex in terms of writing queries and it isn't robust enough in terms of result.

Solution: Ranked Retrieval -> query it with the search engine and returns documents with a score and ideas of what is relevant to your search query.

- You want a specific kind of basketball, you get a description match adhering to your search query.
- Solution to answer all these problems.

Ranked Retrieval

- System returning an ordering of documents
    - A bunch of results that you can moderate
    - Support free text queries/natural language queries
        - User can write in whatever language he or she wants
    - Moderate the amount users returned so they aren't overwhelmed
- Basic premise is the scoring: You need an algorithm that scores and gives the set of documents back that are most relevant to your query
    - Score each of these documents.
- Approach

    - Score each document between 0-1 based on how well the document matches the query
- Assumptions:
    - An instance: One term search query: “Cookies”
        - No occurrence, document gets 0
        - Higher the # of occurrences, higher the document score

- Jaccard Coefficient
    - Find the overlap of two sets - A and B
        - Jaccard(A, B) = |A intersect B| / |A union B|
            - Get all the terms in A, get all the terms in B, compare in some form
            - Divide it by union of both A and B
        - Jaccard(A, A) = 1
        - Jaccard(A, B) = 0
            - If A intersect B = {}
    - Score is between 0 - 1
    - Look at the highest Jacquard Coefficient
- Takes occurrence in terms of if it appears, not necessarily the term frequency.
    - This is BAD!
    - Jaccard Coefficient does NOT consider **term frequency** or how many times a term occurs
        - A few terms don’t occur as many times.
        - Information retrieval systems use stop words
            - “to”, “and”, “an” have no additional meaning other than giving structure
- If you have a document about the president, there are probably a million articles about all the articles that the president has had.
    - When you look at the statistics of each document, the name occurs very few times, but these pronouns occur many times.
    - These words don’t have much information about a specific name.
    - The specific name or term occurs less frequently in comparison to words that occur many times.
    - Certain terms should weighted more than other terms than words that occur frequently in the system.

Term Frequency:

- Binary Document Incidence Matrix:
    - In information retrieval, count and frequency are interchangeable. Frequency is NOT a rate!

    - Any time we refer to frequency, it is just a count.
    - Here we have the context of Shakespeare's documents and stories.
        - What you would do is get row vector of Brutus, row vector of Caesar, etc.
        - This didn't scale well, not robust enough.
        - You want some improvements.
    - Change to term frequencies and what it looks like.
        - Caesar occurs 232 times in Antony and Cleopatra
        - Very insignificant character in those books.
    - We would consider all the row vectors as well as the column vectors -> size of vector would be # of words and terms in your vocabulary.
- N^|V| space where you break this down into this format.

Bag of Words

- Bag is an extension of Sets (different from the mathematical definition of Bag)
    - It leads to a problem because it doesn't consider order.
    - The biggest problem is NOT getting any documents.
- Term Frequency (TF):
    - # of times a specific term occurs in a specific document.
    - Raw term frequencies are NOT exactly required.
        - # of occurrences refers to how certain it is relevant.
        - Occurs 100 times vs 10 times, is the 1st document actually 10x more important? No!
    - Scale the term frequencies in some way that it's less than linear

Term Frequency Weighting

- Take log of term frequency:
    - if(tf_(t,d) > 0):
        - W_(t,d) = 1 + log(tf_(t,d))
    - else:
        - w_(t, d) = 0
- Calculate log of this term frequency for any of these documents.
    - Some document came in with a query that comes into the search engine and only has Caesar (NOT Julius Caesar)
        - We would calculate a score for each column and it would be the sum of all these values
    - You do this for all the possible documents that you had. Rank them up according to scores.

(Inverted) Document Frequency

- Rare terms in document frequency are weighted more than frequent terms
    - You probably want all the documents that mention Obama to be weighted more
- Look at # of documents that has that specific term. You would see how many documents in which the specific term comes in.
    - Doesn't really matter how many times it occurs, but if RARE terms **APPEAR**
- We want to rate these rare terms more than frequent terms.
- Frequent Terms - High Document Frequency
- Rare Terms - Low Document Frequency

Take the log in order to dampen the ratio and you want to look at some measure of getting a feel for the ratio of terms and documents.

- idf_t = log_10(N/df)

TF-IDF Weighting

- TF-IDF weight for a term:
    - w_(t, d) = w_tf
- TF-IDF
    - Boolean -> Bag of words -> Weight matrix

Vector Space Model

- |V| - dimensional space
    - Consider the same matrix
    - Instead of looking at terms that occur in both queries and documents, we represent the query itself as a vector.
- Look at vector space model with specific terms.
    - Consider many languages, all the words in a specific language
        - It would be pretty big (understatement!)
- Most words in would be pretty sparse in documents if we consider EVERY word in the English dictionary.
- You would represent the query itself as one of these vectors and we would do the same calculation for the query.
    - Consider the query itself for the document.
    - Perform vector algebra/vector arithmetic to see which document is more relevant than the others.
- Use proximity/similarity of vectors
    - An even better way is to use Angles between the vectors
- The smaller the angle, the higher the cosine
- Cosine is a monotonically decreasing
    - The higher the angle, the lower the cosine value

- Length normalization:
    - Vector's length is normalized by dividing each component by its length (L2 Norm) -> unit vectors
- Cosine Similarity
    - A * B = ||A|| ||B|| cos(theta)
    - Look at the formula on the slides
- These are pretty fundamental ideas to understand.
    - Most of the models that we have build off of this.
- After length normalization:
    - This would be the unit vector
    - Get q and d vectors
- Summary
    - Represent Query at TF-IDF weighted vectors
    - Once you have the cosine of all query-document pairs, rank them based on decreasing cosine scores
    - This is how the vector space model will occur

Information Retrieval - Apache Lucene

- Build an inverted index on Hotels information
- We want to build an inverted index
    - Our index has to support keyword search for id, name, city, or the full content (union of all the fields)
    - Search engine should display id, name, city
    - We'd like to make even more advanced searches involving a Relational Database
- Lucene
- Query Processing (Example)
    - Brutus AND Calpurnia
        - Brutus - Postings List
        - Calpurnia - Postings List
    - Creates indices and forms a mere to indicate how important these documents are
        - This is the general idea of how it would work.
- Apache Lucene
    - Atomic detail that we always considered is that we have collections of documents.
    - In reality, all the data is in some database.
        - Transform all the data into a Lucene document.
    - Document doc = new Document()
        - Create a document of this length and add fields to that.

        - Things that probably won't have many words on them.
    - Text fields
        - Longer descriptions
    - Create a document and add new string fields
- Get "ID" and make a database call. You wouldn't want to return all the searches! That would be insane
    - What if you get 10 documents and you only have a small snippet of each of these documents
    - It just links to each of these documents
    - Get information of the actual page.
    - Store it here in the index.

Inverted Indices (review)

- Normalize this and stem it.
- Stemming
    - Authorization -> authorize
    - Friends -> friend
- Stop Words
    - a, an, the, to, be, and, but, of
- Tokenizations
    - Character streams are tokenized into word tokens -> breaks down many of the long blocks of text
        - Converts into lower case and all of that
        - Should serve the purposes better
- Say you have an IndexWriter, you would create a new Object of that class, provide a directory, and provide a configuration
    - Like the previous slide, we would add each of the fields as a Lucene document that you would put into one of the methods
    - Do this for all the documents you are interested in.
    - Add them as documents as descriptions of the items.
        - These would be added as documents to the writer
    - Similar to thinking of it like a dictionary
- Search Index
    - Search for actual query and have something called an IndexSearcher using the indices.
        - Using the writer's index, you will create a new Object of that searcher and you can get a list of documents.
- Query Parsing
    - You want to break this query (search term) using a StandardAnalyzer()

        - Both indices and queries have to be in the same format (consistency!)
            - Default field so make sure it is the same as what you use in the config.
            - You are essentially parsing the search term.
            - Tokenize and normalize, if you were removing stop words.
- You can represent which default field you want for your search query

Lucene testing

- Eclipse - Testing Only
    - Add all the required JARs
    - Go to Build Properties, and add JAR files
    - Make sure when you try to compile through command line that you have all this information in your JAR files

Indexer has a property called IndexWriter where you provide information about where things are created.

- In the next step, it gets IndexWriter and the value is null.

Hotels that are returned from the hotel class

- For each hotel we will add that into an index
    - We could look at each hotel and find an IndexWriter
    - Find a new document for each and add them into the fields
- Create this doc that has all of these fields
    - We index all of these and once we create the index, we perform the actual search
    - The searcher and the parser would be set to the default field
- We return all the documents in topDocs and run performSearch
    - topDocs and scoreDocs -> return a bunch
    - Score of each of these things
- Project 3
    - Connect Java to DB
    - Create Inverted Spatial Discussion

W 6 M Lec 2-8-16

- Today's topic
    - Distributed transaction
    - MVC
    - Sessions
- Location-sensitive queries and how to process them efficiently

- Traditional B+ tree is a one dimensional index
    - When our query depends on multiple attributes, we run into some trouble
    - People come up with a # of index instructions
        - Allows us to get a lot more tuples than a single index
        - We talked about three spatial indices
- 1. Grid File
    - Creates a lot of inflexibility because it affects too many things when we try to move it around.
    - We want to localize things when we split instead of putting up a huge boundary all-around.
- 2. Quad Tree
    - Instantiation of the idea was to think of it as four quadrants
- 3. R-tree (Region Tree)
    - Here, every node overflows and splits it into multiple regions
    - The region's children's may overlap, so it is okay to have overlapping regions.
    - Any region and any of its children will have a chance that there may be data there.
- Ideally, we want to make all three of these spatial indices as small as possible.
- If individual subtrees are smaller, there is a lower chance of overlap in this scenario.,,
    - Try to minimize the subregions of the trees (high-level idea)
    - The basic idea is that by considering multiple coordinates together, we can query a lot more things than if we have a one-dimensional index.
- Quad Trees and R-trees are very popular data structures
- Web service data standards
    - SOAP
        - Based on very specific, well-defined standard protocols
        - Easy to use, which is known as the SOAP standard
        - Encapsulate function call responses instead of XML elements
    - REST
        - Whenever we have a function call, we encode it as an HTTP request where all the request parameters are encoded in the URL
- One more standard
    - Web Service Definition language (WSD): Used for formally describing how to access my data
        - The web service publisher can specify at what URL the service is available.
        - We will wrap up our discussion about web services by talking about distributed transactions.

- We want to integrate multiple functionalities described by different entities available.
    - Have a travel website that integrates various websites and have a single destination where all the information is available.
    - Whenever the user comes in and tries to conduct a transaction, things get quite complicated and there are certain things you want to ensure.
    - The user comes in and the user tries to buy it
        - They have an availability that fluctuates all the time
        - When the user comes in and tries to buy the particular package, what happens is that they make a purchase arrangement on the website.
    - Make some commitments that involve many different entities
    - User wants to be provided a single commitment point so they can either get everything or get nothing.
- Either you get all of these three in the commit tree, or none of them do (atomicity of commits)
    - This is formally called distributed transaction
    - Appeared in the context of the distributed database systems.
    - There should be no partially committed transaction.

2 phase commit

- Before you start buying things out in separate entities, once the user indicates his or her willingness to buy something, let me first check with individual sources to ask their intention on whether to go with individual transactions or not.
    - This is the basic idea behind 2 phase commit.
        - There is a first phase where you inquire whether everyone is really ready to participate.
            - Solicit participation and proceed only if everyone is willing and able to.
        - After confirming the first phase, we go through the second phase.
    - Check if you have votes of yes from everyone, otherwise you don't have permission to go through the transaction!
    - We have to go through a growth stage and a shrinking stage in distributed transactions
- Potentially bad cases when something goes wrong?
    - There are problems if any of the parties fails to respond!
        - This guy is waiting for everyone to say either Yes or No. They put their resources aside and wait there forever!

- To minimize this kind of impact, you have to ensure that you are in the wait stage.
    - Make sure there is no ambiguity in order to add a track everything.
- There is likely something that can go wrong, and after that, I can go ahead and proceed!
    - We are very risk-averse and this is why we want to hear yes from everyone first.
    - The real world is different because we don't always have this ideal situation where everyone will respond to you.
        - Lots of possibilities to this operation in the real world
        - The chance of this happening is really low!

Asynchronous transaction

- Things can go wrong but your phase commit can be reverted
- Every transaction assumes that everyone else will be fine and you can roll back if necessary
    - Compensating transaction
        - In case of something that goes wrong i.e. ordering coffee or buying food and not picking up your order

Under what scenarios would a asynchronous transaction be better than a 2-phase commit

- Look at the transactions that don't that have any overlap issues

Programmatic Approach (Servlet)

- MVC frameworks
    - Model
    - View
    - Controller
- This is a paradigm that is great to follow
- Java Servlet provides all these frameworks to forward these requests into a Java function
    - In order to generate content corresponding to a request, you have to a write a program to generate the HTML document.
- Working call -> better than nothing
- This is UGLY code!
    - Look at all these tags that are intermixed!
    - Not very scalable and it is FUGLY!
    - Recompiling is a headache, this whole code mess is a pain in the ass.
    - Working approach, but not the optimal solution!

- Template Approach (JSP)
    - Supports JSP pages and in order to generate dynamic content, we simply write the HTML tag inside that page and whatever portion in which we need to update or generate content dynamically can be embedded.
    - Only part at which you have to dynamically update will require the request parameter.
    - This is how we somehow change earlier code to our current code.
- Potential problems with these two issues
    - We are generating a user's first name, but what if we have to collect a lot of ofter information (do joins, apply filters, embed things, etc.)
        - How are we going to process all this stuff?
        - Within this portion, we have just a line, but elsewhere, we have to write very complicated code.
    - Once the logic of the content generation becomes complicated, this is a very complicated mixture of HTML code and Java code.
    - This template approach does NOT solve all the problems
- Actual rendering of content could be very important.
- CS 144 webpage is ugly and Cho hates it!
    - He is NOT a designer and he wants someone to present things beautifully
    - He can write efficient code but he doesn't know how to make it pretty
- People who can make things work are NOT necessarily those who can make it pretty
    - You have web developers who make things pretty
    - You have software engineers who make things work
- In those cases, if everything is mixed together in a single thing, the designer and engineer to need to work hand-and-hand with each other.
    - For the ownership of the code, it is a very good idea to separate out how things are presented.
    - Even further, all these things and generated content result from the underlying data.
    - Some new data may be added and existing data can be updated, but the business logic may change much more often than the underlying data.
    - We want to separate out the underlying data from the business logic and creates a final presentation based on the output.
- MVC framework tries to separate these things out
    - Model layer: Underlying data model that has all the data defined.
    - View layer: Presentation layer to make things look pretty
        - Template Approach is well-suited to the View layer.
    - Controller layer: Business or application logic. What is the data you need to get, what operations do you need to do?

        - Programming Approach (Servlet) is well-suited for the Controller layer
- One way of implementing MVC framework is to write all the Java code that does computation of all the final output that is presented within that program.
- Once you do all this computation, forward the request to the JSP page, which does the rendering (display) of the content.
- You want to avoid doing complicated computations and just display the data in the Java servlet.
    - Sometimes, you need some construct that is similar to a program depending on how many people.
    - Iteration over a loop to list out all the entries.
    - Use for loops and while loops in this case
- In other standards, you have JSTL
    - Standard libraries that let you support all these iterations
- Inside the JS page, the recommendation is to use whatever sample iteration rather than computing the Java code, and simply display the data and avoid having all the computation done as the Java servlet.
- Project 4 -> use Java servlets in order to organize the code extremely formally!

Cookies and Sessions

- HTTP requests are based on the request and response framework
    - Client has to request first, then the server responds
    - There is another kind of important assumption (stateless protocol)
        - Any response is based on that particular request that was just sent to the server
        - It is just for that particular response and we don't have to worry about it.
        - This is something that makes simple HTTP server efficient
        - It is becoming more important (in some cases essential) to customize future responses based on earlier requests
    - Amazon.com
        - To encourage you to buy things, it shows suggested content you are potentially interested
        - Amazon will NOT remember things about you, but Amazon recognizes that this set of requests comes from the same user (Cho)
        - We try to identify what you are interested in and customize the page for that particular user

        - At the basic stateless protocol, there should be some way to identify that this list of sequence of requests are really coming from the same person.
    - How can we do that? What are the mechanisms that identify the set of requests from the same user?
        - We can use cookies!
        - If we look at the HTTP request, it has cookies!
        - Any TCP/IP packet (from 118) has information
        - One possible hint that may reveal something about the user is the source IP.
            - Roughly, we will be able to understand where the source is located (this is one heuristic to determine the location of the user)
            - Not a perfect solution because that particular machine may be used by a # of different users.
                - Even if it has a static IP, people can use it for different things.
                - If it is a dynamic IP, there is no guarantee that the request will come from the same user.
    - How does Amazon ensure that the requests come from the same user?
        - The HTTP protocol layer does NOT support any correlation, so at a higher level, how can we identify if it comes from the same user?
            - Make sure things are identifiable!
            - Whenever a request comes from the same user, they should have the same uniquely identifiable information
                - Use cookies to record the session!
            - HTTP protocol layer doesn't support it, so we need a high-level mechanism to embed key-value pairs
                - This is defined as a **cookie**
                    - A **cookie** is information (string) that the server can request a web browser to include for all future requests.
                    - It allows the server to ask the client to embed certain information in all future requests
        - We can do this using the set-cookie header!
            - The essence is that we have a name-value pair (name = value)

- Example
    - id = 12567
    - Set cookie to id = 12567

    - The client in his request needs a header field called Cookie that has the same name-value pair
        - All future requests will now have id = 12567
    - The server will somehow remember that the user is named John
    - The same browser can now keep embedding this HTTP field and include it as part of the request.
    - When the server gets a particular request and now we know the user is John.
    - This is how the server keeps track of requests from the same user.
- Note, this is NOT the only use for cookies
    - We don't recommend this, but we can set cookies for a user's credit card information.
    - Whatever information that we want the server to remember, we can have the server ask that and retain this information for future instances.
- In general, Set-Cookie will send information, but in some cases, server can get a cookie only within a particular portion of their website.
    - We can limit our path as follows:
    - path=/cs144;
    - We can specify the domain as well to be something like domain=cs.ucla.edu
    - Another field we can specify is called "expires"
        - If no expiration data, the cookie is deleted when the browser closes
        - This is called a **session cookie**
            - Valid only within that current browsing session
        - expires is used to make a cookie into a **persistent cookie**
    - Using this, we can identify the set of requests and there will be some kinds of issues related to security
- Security concerns
    - You don't want to leak certain information to other browsers!
    - In order to prevent leakage from one domain to another, all the standard browsers need to adhere to the **same-origin policy**
        - Nobody else can have access to this cookie!
        - If the cookie is saved to oak.cs.ucla.edu and we go to a USC website, the browser will never reveal that information because the cookie can only go back to the original website or the original domain
        - This is for security reasons and is an excellent feature to have
- Later, if there is a browser 12567, can we trust that it is the same user?

        - Can I trust that it is something I asked the browser to actually remember?
        - These kinds of attacks are known as **cookie poisoning**
            - Server relies on the fact of seeing if the browser is trustable, and they will only send out authentic information and we don't have to worry about it.
            - Any browser can embed any cookie that they desire.
            - If the user has malicious intentions, the user can poison his or her own cookie to whatever value they want it to remember.
                - The data has been manipulated by the user
- Completely random #'s are likely to be NOT very useful
    - We can set the value to something that is really meaningful or to a user's current session ID
        - We can intercept another person's response and look at his cookie ID
        - Take his cookie ID and embed it into our request.
            - This is what is called **cookie theft**
    - Stealing cookies can be used for malicious purposes as well! We need to be aware of this when we are browsing the Internet
- If you are using public Wi-Fi and going on Facebook or Twitter, another user can host anything on your Facebook page through these kinds of mechanisms
    - On public Wi-Fi, all messages are broadcasted without any encryption. Smart hackers can look at responses and requests and look at people's cookies.
    - The session ID is embedded as part of the cookie, and we can identify the session ID and from that point on, we can embed that session ID and send all the requests to Facebook.
    - Facebook will think that this session ID belongs to the thief and it was a very exploitable hack.
- All these web companies soon moved to HTTPS protocols so that it is harder to hack an encrypted service.
- Have you experienced looking at some products on Amazon and you look at a news website?
    - In the advertisements on the news website, you see all the advertisements from Amazon being shown.
    - The cookie should only go back to the company you visited! There shouldn't be an interlink between New York Times and Amazon, should there?
        - This is NOT necessarily the case!
        - Look at notebook

- Track user's activities who are able to track which pages you visit via Google Adsense
    - These are called **third-party cookies**
    - These third-parties can identify users visiting all these different webpages.
    - We have the option of disabling third-party cookies on our browser
        - Cookies will be set only on the pages that the user is currently looking at.
- When you buy things from Amazon, you have to authenticate yourself.
    - When you purchase things via the push of the button, if you forget about the case where we have HTTP only, the server doesn't know if this sequence of requests is coming from the same user.
    - If we don't know that, the server might have to conduct important transactions
    - When the user authenticates himself, we don't want the user to authenticate themselves again and again.
        - How do we correlate all these requests from the user once they have authenticated?
        - The server assigns a unique session ID to that particular browsing session.
        - We need to remember that particular session ID and embed all the future requests.
- Whenever the user authenticates, the server stores this information and can fetch this information for future use cases.
- Sessions do NOT need to re-authenticate in this case, but a potential pitfall is if the ID is stolen from someone else.
    - If it turns out to be a valid session ID, the user will do whatever is valid and we have to make sure that this ID is NOT easy to steal.
    - It should be close to impossible to randomly guess a session ID.
    - It should be much smaller than the overall space of the session ID.

W 6 W Lec 2-10-16
Today's topic

- AJAX
    - Event-driven programming
    - JavaScript
    - HTML DOM
    - XMLHttpRequest
- MVC: a very high-level approach to all forms of programming
- Stateless request: the results are all coming from the same user

    - Where are all the requests coming from?
        - The server can use a cookie
        - The server can ask the client to set the cookie
        - One way of identifying the requests are coming from the same user is if you embed a unique ID.
        - We can see that it is coming from a client.
        - Use cookies to identify a sequence of requests coming from the same client.
        - Other people may steal the cookies or the browser can write a client that somehow manipulates the cookies.
            - There is no guarantee that what the client sends back to the server is what the server asked for!
    - How can we use cookie to establish a user session?
        - We want to authenticate the user and we want them to be who they claimed to be.
        - Multiple requests in order to conduct a transaction
            - Browse through Amazon's product pages and put them into shopping carts
            - Put it into credit card information, etc.
            - We want these actions to ensure that it is the same user but we don't want to ask for the userID-password pair request each time
                - Each user should have already authenticated himself or herself
        - By using a cookie, we can create a session (assign a unique session ID and return that session ID for all future requests)
        - The server has to remember that the server is from a particular user
        - This still has a problem of someone being able to steal the session ID or poisoning the session ID.
- Cross-domain authentication
    - Google owns many web domains: Youtube, Gmail, Nest, etc.
        - In order to avoid different websites, cookies follow the same-origin policy normally.
        - Any standard browser ensures that the copy being sent back came from that particular origin.
            - We will never cross the domain boundary in this scenario.
            - This, however, makes it complicated for companies that start up many smaller sub-companies

    - When I go to Gmail, I authenticate myself and get a session ID from that particular session from specific session (Gmail).
    - If you got to Youtube, you do NOT have to authenticate again because the server remembers you.
- Don't worry about re-authenticating users and we can trust that end of the user.
    - B can establish a new copy and return it on the the center of the cookie!

AJAX - Web 2.0

- Traditional web interaction model
    - Look at webpage and depending on what you want, look at some of them and decide which new URL you want to navigate to.
    - Wait and wait until it is reloaded and then it will return a result finally.
- Google Maps is one of the first versions of Web 2.0 (AJAX)
    - People previously had no notion of the drag and drop concept
    - You are able to click and move like you would on a desktop application
- While the server is waiting for the request, the computer outputs a white, blank screen.
- User tends to have a lot of down time waiting for other requests.
    - While the server is waiting for a client response, we used AJAX to avoid having large amounts of downtime where the client waits and waits and waits.
    - Program flow immediately goes back to the browser so that client does NOT get stuck!
        - The AJAX action can handle the information that the client already put in.
        - As long as the browser has cached the information in the area, it will be able to grab the info
- Sometimes, the update is only a small part of the page
    - Depending on what is needed, we will use dynamic, in-place absorbance
    - When we type in a keyword, the AJAX intercepts it and it will suggest recommended search terms to query for.
    - AJAX engine wants to have in-place updates to whatever is necessary
        - Not stuck waiting for all the interactions of the other server
- **Asynchronous** programming: Idea that the code does NOT necessarily have higher-level protections over himself.
- Dynamic in-place update of the page
- Languages to describe interactions via XML or other networking formats.
    - JavaScript is a language used is this case
- XMLHttpRequest: Handles events on the HTML Dom
    - Event-driven callback function

The Fall of Microsoft

- AJAX eventually led to the demise of Microsoft
- The XML, GUI, and various other features preceded Microsoft.
- The programmer himself or herself does NOT control what happens next.
    - What happens next is driven by what the user does and what the server says
    - Completely driven by the events generated by the user and the protocols.
- **Event-driven programming:** This programming paradigm where control is driven by the event tells you which action to take under a certain secnarioo
    - We write what are called event handlers
    - Collection of function calls designed to deal with a particular event that occurs.

Background-color change

- Monitor the clicking of an event
- It changes the background-color of the document and what Cho described in this event is to monitor for the click event handler
    - When that event happens, you wanted to perform the action
    - The action in this particular case is changing to a different background color!
- DOM: May be associated with particular events that the user can take.
- Javascript: Writes down when events happen and lets you know what you want to do.
    - How can we embed Javascript within a webpage?
        - Use the script tag!

<script type="text/javascript>

</script>

- The syntax is virtually the same as C++ and Java

```
if(a > 2) {
        x = y;
} else {
        y = 8;
}
```

Comparison operators

- == does automatic type conversion
- === checks for both type and value

Differences from Javascript from C and Java

- Very loosely typed language
- Type associated with a particular function changes with what the variables are doing at a particular time.

Defining variables:

```
var a;
a = 1;  //a has number type value
a = “John”;   //a has string type value
typeof(a);      //Returns data type of a
```

Defining functions:

```
function ftn(x, y) {

}
//Return value does NOT strictly adhere to a single type, very loosely typed
```

Three most important primitive types of JavaScript

- boolean (true or false)
- number (we can represent integers, but all numbers are really floating points)
- string

Forcing numeric conversions

- Boolean(“ “)
- Number(“ “)
- parseInt(“ “)

Important complex types

- Arrays

```
var a = [1, 2, 3];
var a = new Array(2, 3);

var complexA = [1, “a”, [2, 3]]
complexA[2][0] => 2
```

When we manipulate these, there are a wide variety of unique function calls

- We can sort them based on the search function, but we have to pay attention on whether they are mutators or accessors

Mutators vs Accessors

- mutator: mutate the content of the Array
- accessors: don't change the content of the Array, but rather look at the content at a specific index

Object types

```
var o = new Object();          //Allows for OOP
o.x = 30; === o["x"] = 30;
o.y = 20; === o["y"] = 20;
```

- We do NOT need to pre-specify the attributes for each object (we don't need to declare x or y in advance)

```
var o = {x: 30, y: 20};          //Another way of representing this as an object
u = o;  //Assignment by reference, NOT copy!

var o = {x: 1, y: [1, 2], z: {a: 1, b: 2} }
o.x -> 1
o.y[0] -> 1
o["z"]["a"] -> 1
```

HTML DOM (Document Object Model)

- If you think of an input box, the user may type something into the input box and we can generate things based on the actions that the user takes.
- There are properties that are associated with it.
- The main body is associated with a particular body and there are properties associated with it.
    - Once we construct the HTML DOM tree, there are properties associated with each particular node in the tree, and each node has an associated method that we can call.
- If we want to clear an input field, we can call the reset function and the browser will clear out whatever it has.

Mouse click

- In order to intercept events and take certain actions, you have to override the default event handler with your own function

```
document.body.onClick = ChangeColor
```

Dynamic page updates

- Create elements individually and add it as a child of your body

```
document.body.innerHTML = "<p>a</p>";
```

- The code below is equivalent:

```
var p = document.createElement("p");
```

var t = document.createTextNode(“a”);

document.body.appendChild(p);

p.appendChild(t);

W 6 Dis 2-12-16
Web Services, Servlets, and JSPs

- SOAP
  - Service that needs to be provided by the web
    - Maps, hotel booking, flights, etc.
    - Need to be provided and use the web as an application
    - People thought of usefulness in the web and we invited a kind of format that can standardize this web application
  - Simple object access protocol (SOAP)
    - We can see the request and response are in the XML format
    - Write a very formal XML file
      - Specify namespace and enclose the tags with “soap” (without quotes)
    - Inside the soap body, indicate the kind of message you will receive
- WSDL
  - Doesn’t specify the exact format of what we will be containing in the body tag.
  - Consists of service, binding, portType, message, types
    - Service
      - Location
    - Binding
      - Protocol and encoding scheme

WSDL Example

- We need a very formal definition regarding namespace and all these attributes
  - Specify the things you need in your message body
  - Types specify the type of input/output parameters
  - We need to know the path that our message is in.
- We also need to know the operations of our services, input message and output message
- Consists of a lot of redundant information about bindings and many XML tags that are not recognizable by human beings
  - Not used quite commonly

REST

- Commonly used for API's like in hackathons
- Requests like "GET", "POST"
- Less complicated:
    - Elements and values are directly incorporated into the URL
- Return and response can be highly diverse and can be defined arbitrarily by the web service.
- This kind of redundant definition can be saved and it is more practical to develop a more simple, but practical web service.
- What are the problems of the REST interface?
    - The standard means we have a very detailed specification of this tool
    - If two web services that we do not know have an internal implementation, we will know what the service will be like and the structures of messages you are passing in.
    - Since anyone can implement their own REST interface, you have no idea what attributes will be used, and you don't necessarily know the type of values for each parameter
        - This can cause a lot of confusion
- Google Maps provides a REST interface
    - Provide Google Maps interface, refer to the document and see what kind of arguments it can take in
        - HTML or JSON response
        - Widely used because it is very lightweight and has human readable results
        - Gets a very wide application and is very popular.

Java Servlets

- If some developers are more familiar with Java, they can use Java Servlets to implement things in PHP
    - Java is faster than PHP
    - This is the benefit of servlets
- The general framework of Java servlets
    - Server-side application like Apache Tomcat
    - When the browser sends an HTTP Request, the browser will pass a request into a servlet, which manages this request.
        - When the server receives the request, it will issue some different queries or computations to work with the backend.
        - The servlet will then generate an HTTP response to the web server, which then sends something back to the client-side

Servlet functions

- void init(ServletConfig config)
    - Executed at the beginning in the servlet lifetime
- void service(ServletRequest request, ServletResponse response)
    - More appropriate to use doGet and doPost methods
- void destroy()
    - Analogous to a destructor

HTTP Servlet

- Helps you implement a more detailed implementation
    - You can write corresponding actions or statements in different methods
- doGet and doPost can be used instead of **service**
- To access parameters in doGet, we can assign a return value from request.getParameter("<parameter name>");

Servlet Example

- Look at slides
- Content type is "text/html"
- Get this PrintWriter from the object
- Write the HTML file into the response
    - Since we are writing Java code, we need to embed the HTML tags into the style
- A lot of boiler plate code, we generally only care about the HTML style that we write out
- Close the read/write stream at the end
- Send the info to the web server at the end.

Servlet-Invoker Example

- Need to write the client side code and remember that in the first discussion, we talked about using a URL that ends with a PHP extension
- PHP take charge of handling this request
- Servlet will be in charge of this request instead of a PHP file
    - Let the client know which URL he can send the request to.
- Server automatically finds "welcome1" servlet and the servlet can do the computations (Look in the slides to see what "welcome1" is)

Deployment Descriptor web.xml

- We need to write another file which is keeping this deployment description as our mapping
- Description and then we have already written a servlet called WelcomeServlet

    - Every Java file will be compiled into a servlet-class
- Mappings will map things into a servlet-name
    - Move welcome1 into servlet-name

Steps to Deploy a Servlet

- You have libraries imported into your code for standard libraries

Q. Difference between servlet-name inside servlet tag and servlet name inside servlet-mapping?
A. servlet tag defines the different names, but the servlet-name is just a name for the specific servlet. Outside the scope, you can refer to it by the servlet-name.

Java Server Pages (JSPs)

- JSPs are **Servlets** in disguise
- Look to the programmer more like HTML in the program
- We need to write boilerplate code that isn't meaningful but necessary to compile the program
- It is useful because it reduces the amount of code since you get rid of boiler plate code
- You can write Java code inside the HTML file in this method
    - Put this Java code into special constructs
        - Directives
            - <%@_%>
        - Actions
            - <jsp:action />
        - Scriptlets
            - <% block; %>, <%= expression %>, <%! declaration; %>
            - We can put a single expression and elevate it to a variable

forward1.jsp Example

- Start with the directive at the top
- Imports two Java libraries (notice how it looks like an HTML format)
    - There are Java statements embedded into the HTML (look at the scriptlet)
    - It will decide the parameter of first name to decide if the value is null or not
    - Generate a format that represents the date format of the user's current location
    - Pass a param with a name called "date" that is completed by this Java expression
- Include a page with called "form.jsp"

- Since Java code is embedded into HTML, anything in the curly braces should be treated inside the scope of the statement
    - The jsp forward structure will be incorporated into the code if the “if” statement is true
    - Otherwise, it will go into the “else” statement and invoke the jsp:include structure
    - Important features to understanding JSP

form.jsp
- Tells you to type something into a form

forward2.jsp
- Does the processing request from the forward window
    - Directly forwards to the JSP file
    - Get parameters from the first name and print out a statement and get parameters from there.

forward1.jsp
- This is inside the web server
- The servlet for .jsp is the same thing but provides two different ways to write server side scripts, whether you can use HTML or embed in a Java file.
- You can also use Java statements to embed it into an HTML file

JavaScript
- Most popular browser-side language
- Node.js can be used on the server-side as well
- Becoming more and more popular everywhere
- Popular for the following reasons:
    - Dynamic and flexible expressions
        - It is quite different from mainstream languages like Java or C++
        - A very strong impression is the scripting part
        - Loosely typed but we can change the type at any time.

```
var simple = 2;         //Simple variable that is a number
simple = “string”;      //We can change simple’s value into a string
```

- We can test this in a browser’s console

OOP in JavaScript
- “this” keyword refers to the object, so we include the attributes of the object into the arguments.
- Follow an object chain in this statement.

- There needs to be a method we can reuse and save you a lot of time in Java and cleanliness.
- Use a “prototype” object for the Range object to work
    - Include methods inside prototypes
    - All the range statements will be inherited from this object.
    - Create things to be written as prototype objects
- toString is another method that is returned in a human readable format.
- Every object has a secret link when you use the “new” keyword

Prototypal inheritance: Objects **inherit directly** from other objects

- JavaScript uses Prototypal inheritance
- All of the business about classes goes away.
- Links objects together in a hierarchy

Variable Scope (JavaScript)

- Global and local
    - A variable that is declared outside a function definition is a global variable, and its value is accessible and modifiable throughout your program.
    - A variable that is declared inside a function definition is local. It is created and destroyed within the scope of the function
- If the variable is declared with that value, it will have a value of undefined until it is reassigned to a new value.
- Good JavaScript writers will define the variable sat the beginning of the function

JavaScript’s Event-Driven Environment

- Compared to a PHP script
    - Put together the HTML and the script exits and you finish
    - /tweets/page/2
    - Runs, loads all of its stuff, and then it dies
- JS loads up your page, stays in your memory, and waits
    - Just jQuery
    - Way of associating things with the JavaScript
    - “$” is a shortcut for selecting an element from an HTML page
        - Makes it equal to window or the whole HTML document
    - JavaScript is doing nothing as it runs this
    - We are basically giving the $(document).ready function one argument
        - When you write this script and then you import it into your HTML page, the script will run until the time it runs its imports
        - There actually isn’t any useful thing inside this function

        - Whenever the button is clicked, it fires its function as argument #2
        - It is event driven, listening for a click!
    - Stuff inside the function does NOT belong to anything and represents a binding and anything inside will NOT be called when it is NOT written to.
- Runs the little piece that has to run each time it stays in memory.
- It doesn't rerun the application, it just reruns the part in $(document).ready
    - JavaScript as a language is NOT really fast, but it has the ability to stay in memory and run immediately, which is a huge advantage of node.js

W 7 W Lec 2-17-16

Today's topic

- AJAX
- HTML5

Project 4

- We already established the backend, so now we actually want to build the website
- Integrate the Google Maps services and build a nice web interface
- AJAX programming as well as servlet coding in Java
- Mandatory:
    - Experience of separating models, views, and controllers (MVC principle)
    - Due next Friday (2/25)

AJAX

- We will somehow be able to allow website interaction on the webpages.
- We need to be able to update things in place as users trying to interact with anything available on the webpage.
- Previously, users would click on a button and the entire webpage goes blank until we get a new page from the server
    - We don't want this kind of delay
    - We want the user to be able to interact with whatever is available while it is loading
        - This necessitates **in-place dynamic updates**
- 3 core technologies
    - 1. Pinpoint a portion of the webpage and say this is the aspect we want to be updated
    - 2. Monitor user interaction and be able to specify that this is the sequence of actions I want to take on the webpage.
    - 3. Even though I send a request to the server and I am waiting for the response for the server, I want to be able to interact with my program
        - Return control flow back to my program and send a response back

- These are the fundamental building blocks for improved user interaction
- How do we pinpoint what we want to modify?
    - Using the HTML DOM tree!
        - Using our knowledge of how to traverse the DOM tree, we can pinpoint the part of the tree and specify which event we are referring to or looking for.
- JavaScript can be embedded on any webpage and we can specify what we want to do.
    - Change background color, change font size, etc.
    - Implements core programming language logic to specify things we want to do in the code.

Interacting with the server asynchronously

- We don't want to wait; we want to be able to return to our browser and start doing things without much down time!
- If the primary mode of operation is interacting with the user, the user determines what they want to do next.
    - The kinds of programs we write do NOT control the overall flow.
- This kind of programming approach
    - Event-driven programming: A sequence/list of events and appropriate functions to handle them
- We also learned about different events by overwriting event handlers
    - Certain callback functions that can override other functions as well as interacting with each other

Cho Google-suggest example

- Provides suggestions for the completion of the query
- In order to implement it, we need space to output the returned values of the queries as well as events you have to remember
- What kind of user action am I interested in?
    - Typing out the event name in this case.
    - The event handler is supposed to handle the typing function with automatic updates
    - We need to send this information to the Google suggestion server and ask if the search query is now complete or not.
- How can I deal with this?
    - Send something and it immediately comes back
    - How do I specify that these are the things that I want to do.
    - Dump it into that area and specify this.

        - We need a way to interact with the Google Suggest server in an asynchronous manner
            - How do we do that?
            - The event is NOT coming from the user; it is really the event of receiving something from the Google Server
- Particular JavaScript object called **XMLHttpRequest** object that allows for JavaScript asynchronous interaction
    - Specify the event handler when you get a message back from the server
    - Send a request back to the server and call whatever the return value is.
- When this event handler is called, we need to know what kind of response we got from the server
    - This obtained response is stored in **responseText/responseXML**
- We can check the **readyState**
    - Using the readyState, we can figure out which state to exclude except the ones immediately adjacent to your level

```
var xml htttp = new XMLHttpRequest();
xml http.open("GET", "url");
xml http.onreadystatechange = ftn;
xml http.send('null');
```

Two events you have to deal with

- Dealing with the key-up event (1st)
- Whenever you receive something back from the Google server, you will want to check it at a higher-level (2nd)

Where do we set the InterruptHandler?

- "this" points to the particular element that has been created in the scope and also to the one that updated this.

Override values to show suggestions

- Setting up EventHandler and later sending requests to the server

Check whether the readyState == 4

- Escape special characters so we know they are recognized
- At least 1 function for each event I want to be able to handle.
    - One of them handles at least these two events.

Q. Why do we send onready requests each time you have a state change request

A. Cho is NOT SURE, he never thought about doing it other ways. Maybe the JavaScript can reset its onreadystate handler

- One important thing to keep in mind is that onreadystatechange should always happen **before** the send
    - This helps prevent a race condition since NOT everything has been initialized or set.

Q. What is encode-url?
A. encode-url is used to escape certain characters like " ". When it sees special characters that needs to be escaped, it does it's job.

- Ambiguity to which server you are sending to. You aren't really talking to the Google server
    - This is a result of the **same origin policy** that we learned before.
    - The same thing from the HTTP/HTML request occurs. You have methods of overriding it.
    - All the browsers all enforce **same-origin policy**
    - Can make a connection only back to the same server but not to someone else
        - This, however, limits our interaction
        - Obtain the result from the server and send it back to the guest.

- We need to access the response XML and traverse the XML DOM tree
    - Very similar overall in terms of code structure and code body.
        - Only the response handling header is the major difference
- From the responseXML, get the complete URL and make the path more specific

Contemporary Apps

- Many modern apps are developed in JavaScript
- Is XML still good to develop in today's error?
    - XML has a lot of redundancies and overlaps (some people don't like it)
    - Some people asked why we had to use XML when there is a lot more representation of data using JavaScript.
- Instead of formatting in XML, many apps nowadays use JSON (JavaScript Object Notation)
    - Inside response.txt, you can simply call an eval() function on the text
        - Evaluates it as a JavaScript command and recreates an array of three elements
- Look at handwritten notes for JSON syntax

Animation

- There are many interesting animation effects that can be implemented into web apps.
- How can we implement this into our web apps?
    - In order for the text to go around like this, we need a particular text element, and in order for it to look like it is moving, we need to modify the CSS
    - Modify the text periodically so it gradually moves from right to left
        - We need some callback function that is periodically called that does whatever animation effect
        - It gradually moves everything to the left
- Inside the tick, change the location of the tag and slide it to the left.
- Look at Cho's webpage:
    - The function that deals with the sliding is tickerSlide
    - The ticker element within the webpage is the one sliding and flowing to the right
        - Get the current location and increase the number and set it to the new location of that particular slide
        - If it goes to far to the right, we want to slide it back to 0
        - Here, we are using parseInt because it is a string, so we want to parse it back into an integer
        - Get the parameters and add the two values up.
    - tickerStart is called in the event handler load event in the body of the HTML
        - After the entire HTML document is loaded, it calls the onLoad function
        - We implemented this using tickerStart as a load function and we then get the ticker element and set its content to be “Hello there…”
        - We then set the timeout function
    - Why did we wait for the entire page to load?
        - The tickerSlide function assumes there is a ticket element already there
        - The ticker element is NOT created until the entire page is loaded
            - As the browser reads that JavaScript code, there is a possibility that the content is coming very slow from the server
            - In that case, the callback function will try to access the ticker, which is NOT available
                - This generates an error

    - Wait for the entire page to load so we don't access variables that have not been generated yet.
        - In general, we want to wait for the entire page to have been loaded and then call a function using the event handler **onLoad**
    - Gradually move from one place to another using CSS rules
        - How do we specify these CSS rules?
        - Use what we call **keyframe rules**
            - @keyframe: Specifies the keyframe for animation effects
            - Start with web background color and then we make it a yellow color later on

```
@keyframes  backgroundani {
        0%    { background: red; }
        50% { background: blue; }
        100% { background: yellow; }
}

div {
        animation: backgroundani 5s;
}
```

- Cho's examples on his website are very thorough for animations: explore those and play around with it.

Q. Do CSS animations change gradually?
A. Yes, we specify keyframes and we can use it for text motion as well.

- We have to arbitrarily assign values in terms of %

Before browsers, there was no easy way to store permanent data for an application

- For an email client, they have to download from the server and display it from the page
- We are kind of wasting a lot of bandwidth and they are storing it permanently so that it persists throughout the lifecycle of multiple applications
- We can do it in cookies but that is not the proper way
- When you download a particular message, you can use it in the local storage and check if it is in a local storage
    - It is also possible to make things available offline even if we cannot download anything.
    - Use it so that the user can play with it

```
localStorage[“msg1”] = “ “;
```

Graphics in a game

- You need a display area to show beautiful characters that we want to show
- In HTML5, we have a tag called “canvas”
    - Has a particular width and height
    - Once we create on a “canvas” element, we can draw a lot of things within the canvas
    - Create things like “rect” or “rectField”
- canvas tags and graphics you can build on top of it
- Drag and drop kind of interface
    - Originally implemented in Chrome
    - Gmail: attach things using drag and drop
    - That particular element was called using the onDrop and we can do whatever we need to do
- You want to have this big text area and the user can type in whatever they want there.
- We want to actually update the HTML page itself
    - In HTML5, we can actually accept something called design modes

W 7 Dis 2-19-16

AJAX

- Asynchronous Javascript And XML
    - You can reduce the web communication
    - For the AJAX, the core function is called XMLHttpRequest object
        - When we are building a static website
        - Use this when we are expecting some response
- What does this function do? Is it missing anything?
    - Initiate an HTTP Request
    - Specify if we want to use a “POST” method and which file we want to log it to.
    - Send the message, in plain-text, as the request body.
- Not complete but we expect that we are sending a web request with either a GET method or a POST method
- What can we expect?
    - We can expect a response.
    - We usually have some function where we can handle the response
- Retrieving the server’s response (from GET request)
    - What we are doing is initializing the GET request object and we want to specify the URL
    - Very important function called onreadystatechange

        - We didn't specify here what happened
- Asynchronous: Our browser will continue working and it doesn't mean we are going to be waiting for the web server to come back.
- request.onreadystatechange = function()
    - The server sends a message back and we will check the state change and handle the response form the server
    - When we handle the response from the server, we will check if we really have the response and we are going to check the message status
    - We have many status codes to represent the state of the server
        - After we make sure this message is okay, we will try to process the response
        - Get the response header and it can become text or JSON (a few possible types)
        - We can just use callback functions to pass back to the callback functions
    - When we request some content to be changed, AJAX will send something back
        - How are we going to change this part of the webpage?
        - Callback function will send it to a new area.
- "GET" and "POST" will pass parameters to the server
    - For "GET", we just include it in the URL
    - For "POST", we need to encode something in the URL body
- Retrieving the server's response (from POST request)
    - We can check it in the callback function
    - What is the body we are actually using?
        - Set the header and set it using the "Content-Type"
        - Use an application defined encoding
    - In order to comply with the declaration, we are going to have some encoding functions that encode in our data into this format
    - Imagine there is some corresponding decoding function that will help us

JSON

- JavaScript Object Notation
- You can frequently see that JSON is used in the HTTP body.
- Used as a key-value pairing
    - If you want to send this over the Internet, you need to convert it to a String
    - Call the JSON method called .stringify()
        - Converts the object to a String and this could be put in the body of an HTTP request

    - p = JSON.parse(s); //p is a deep copy of o.
- How do we use eval() instead of parse()?
    - Here, we are using a package called JSON
    - JSON is NOT embedded in JavaScript, so not ALL JavaScript will have JSON
        - In this case, if you get a String in JSON format, you can use eval()
        - This will give you an object that is the JavaScript key-value pairing

HTML5

- Introduce the new <canvas> tag
    - To declare it as a canvas, you can use JavaScript to draw different shapes and interact with different users
    - This is quite powerful since we usually used Flash in the past
        - Flash was primarily used for animation and used a plugin
- HTML5 provides a lot of functionality that is almost identical or even more powerful than Flash

Useful links

- http://www,html5canvastutorials.com
- http://www.effectgames.com/demos/canvasvalues

Cookies

- Strings containing
    - name = value;
    - path = path;
    - domain = domain;
    - max-age = seconds
    - secure;
- If the website contains some information on the browse that we want to deleter, we just directly set max-age with to be 0

In a Java Servlet

- Cookies are stored in browsers on the local machine
- Server can send a request to a Cookie
    - It seems that the cookie is stored on a local machine
- We can use JavaScript to modify the content as well.
    - Some notes about Cookies

Project 4: Building a Web Site

- Follow up of Project 3
    - Built some basic web services i.e. providing eBay data

- In Project 4, we are asking you to build a website
    - You should be able to search for some items and search for Item keyword or Item ID
    - We will ask them to add more interactive features to the website.
    - Queries should be able to do auto-complete and query suggestions
    - Adding Google Maps
- This illustrator shows the relationship between Project 3 and Project 4
- We provided some web services that can send a request
    - Web service can show search result, index, spatial search, etc.
    - Build a web server and build some servlet program
    - Initially, we will have some webpage and we want to conduct some web search
    - We can send a request to our backend program and get the result
        - After we get the result and generate our webpage, the user can see this webpage.
        - This is the general workflow of Project 4
        - We also want more interesting features on the website
    - We want to provide all the auto-complete/auto-suggest functions
        - We give you some tutorials so you can try to implement an auto-complete module
        - When you give some results, the user can actually navigate each item.
    - We want to be able to display a map there and use Google Maps service.

Project 4: part A

- http://localhost:1448/eBay/search?q=xxx&numResultsToSkip=yyy&numResultsToReturn=zzz
- Implementing the Search Interface
    - The query should call the web API
- Implement SearchServlet
    - You can type the services and send the request
    - Make sure you process it and extract useful information
        - Do NOT dump XML all on the webpage
            - User-UNfriendly and hard to read
- Adding Navigational Links
- Why?
    - (1) For keyword-search result page, create **clickable link** for each item.
    - (2) For keyword-search result page, create **Next** and **Previous** links
    - (3) For keyword-search result page, create an **input box** at the top or new query.

Project 4: part B

- Adding Maps and Query Suggestion
  - Detailed tutorial on the webpage
  - Just read through it and it shouldn't be difficult to implement these two functions
  - When using Google Maps, we need to be able to translate latitude and longitude to some geolocations
- Query Suggestion
  - Use a Google service with a suggested server
  - Gives a bunch of information we can use to display
  - Build a suggestion proxy

Same-Origin Policy (SOP)

- They are only allowed to respond to the same server rather than a different server

W 8 M Lec 2-22-16
Today's topic

- Security

Final Topics

- HTTP, cookie, session: Request and response, stateless protocol
  - Problems when you want to identify the set of requests from the same client
  - Ask the client to embed a unique code in the client (cookies)
- HTML, CSS, DOM, JavaScript -> JSON (AJAX)
  - Basic standard that lets us create a webpage with a bunch of interact-able components
  - We separate out the structure of the document with the presentation of the document.
  - We should be able to dynamically change contents using JavaScript and DOM
- XML, XMLSchema, DTDm XPath, XML namespace
  - Able to avoid name conflicts and search the XML using Xpath
- SOAP/WDL vs REST
  - Related to web services
  - There are browsers that do understand all these, but w need to have a particular layer.
- Look at notes for diagram

- Often times, depending on how we want to access the data, we have different kinds of algorithms and paradigms to take.
    - Two important things happening over the last ten years is to have a keyword based search interface.
- MVC
    - Model View Controller
- IR
    - Information retrieval
- Indices
    - Inverted Index
    - Spatial Index

Security

- We want to ensure that everything we do on the web is secure.
    - We have assumed that the users are good and don't have malicious intents
    - Nowadays, we have many layers devoted to keeping our product safe and secure
- Looking at common vulnerabilities in our software and try to patch them up.

Scalability

- In the case that the app becomes really possible, how many users can our app support?

These are the last two topics covered in the class!

Security breaches/problems

- There are a lot of malicious users trying to gain control of your account
- Goal:
    - Fame
    - Rich
        - Steal information from the server to obtain money.
    - There is a lot of information on the Internet and you potentially impact the lives of the original phone user in a bad way.
        - It can either make the system work less reliably or it can CRASH the system!
        - This means you can blackmail the original system owner
            - Send another email and starting from tomorrow, we will take down the system for one week.
    - A lot of companies do give in to hackers all the time.

        - By identifying vulnerabilities you can blackmail people.
- How do they do it?
    - Denial of Service (DoS) attack
        - Flood networks with bogus message and this can overrun the service
        - Control more resources on the Internet and get a hold of a lot more resources available on the Internet
            - Get a hold of more resources to get more power.

A few year's ago, the White House was hacked

- By infiltrating a famous person's website, you can grab a lot of attention, which is why terrorists wanted to aim of this target to have the biggest impact

(2) How do the do it?

- DDOS
- Phishing: attracted users by giving them bait.
    - Website will look like the authenticate originally.
    - If people mistype it, they will believe it is the Bank of America site and if they submit information, this is called phishing.
- SQL injection/command injection: Put a SQL command in the input box that doesn't do anything and potentially, this parameter can be embedded in such a way to grab sensitive user information
    - You can simply insert a command like "DROP DATABASE" and now all the information there is completely wiped out
    - SQL injection is very powerful and scary!
- Cryptowall: encrypt all the files on your computer and you have to send them a ransom
    - When you download or accept some program, they embed the code into your system and it encrypts everything in your file system with a secret key unknown to you.
    - The next day, it says file is NOT acceptable because it has been encrypted.
    - All the work you have done has been encrypted, so how do you access it?
        - You need to pay a ransom to decrypt your own file
- Man in the middle attack
    - Inherent vulnerability on the Internet
    - Someone is sitting in the middle between you and the other party with whom you want to communicate

    - You believe you are communicating with just the bank, but in reality, the person in the middle can intercept the message! (CS 118)
- Pharming (DNS poisoning): When I try to resolve a particular URL/host name, they send back an incorrect IP
    - Careless users will think the IP belongs to the actual site. In reality, it is the IP of the hacker's address.
    - When you actually use your laptop, all you see are these pages and there is no way to tell where the pages are coming from.
        - You got the packet from somewhere, but how can we verify it came from Google or Amazon or wherever we wanted?
            - We canNOT verify this unfortunately :(

There are a lot of malicious things that users can do to try to take control of your browser/account

- When you use your browser, you cannot trust it in some cases.
- Can you trust the Internet to do things like accessing my bank account?
    - We do it all the time, so there is something these servers do to guarantee nothing bad is happening.

Guarantees

- There is NO guarantee that the other party will receive the message.
- Banking transaction example
    - In order to send secret information like bank account information, what do we want to be guaranteed?
    - We want to send messages and make sure that the message will NOT be seen by anyone else.
        - While it is transmitted over the Internet, we want to ensure they cannot intercept and look at it!
            - Very important property (**confidentiality**)
            - Often times, I want confidentiality so that if the info does get intercepted, the hacker cannot read it!
    - When you have received a message, you want to make sure it has NOT been tampered by anyone else
        - We want to be able to detect tampering and modification
            - Ensure the integrity of the messages we are receiving (try to prevent man in the middle attacks)
        - **Message/data integrity**
        - Nobody in the middle should be able to tamper with the information!
    - The message should come from the right source

        - We want to make sure it really came from that source i.e. Amazon, the bank, etc.
        - We don't want it fabricated, so we want to **authenticate** the user
            - Ensure the **authenticity** of the other party
            - Make sure the message is really going to Amazon, NOT someone else!
    - **Authorization**
        - When I let someone do certain things on my website, I want to ensure that the other party has proper authorization.
            - They need to have the proper right to modify the schemas but we shouldn't grant permissions to modify our files.
- These four properties are vital for ensuring security on the web
    - Some applications do NOT need all 4 of these, but most care about at least 1 of these 4 properties

How can we guarantee these kinds of properties?

- What are the tools that people develop to ensure these properties?

Ensuring confidentiality

- How do we ensure that people cannot see our message
    - **Steganography**: "embed" true message with a harmless-looking message
        - Osama bin laden: A lot of people believed that what was communicated was NOT what he said, but rather where he looked at, his body language, etc.
        - Inside a plain looking message, embed something that has an actual meaning
            - This usually implies some secret "code" between the two parties
        - Kathy is laughing loudly
            - Forget everything except the first letter of each word
                - K.I.L.L
        - Popular technique between the two communicating parties but we shouldn't use that because it is security by obscurity
            - Depends on the protocol, so once this is gone, it loses effectiveness.
            - We need to hold on to a message that is truly secret, so we do NOT want anyone else to figure it out.
    - **Encryption**: We randomly shuffle the bits based on our secret key, so that other people who are looking at it cannot figure out the original message.

        - If they do NOT know the secret key, they cannot figure out the message.
        - We can send it over an insecure network, but this is okay because unassociated parties should NOT know the encryption and decryption methods
        - F, F^-1 : cipher
    - Common methods of encryption
    - NIST
        - AES (128 bit): Developed as a result of the evolution of modern hacking, so we needed more bits to shuffle in order to prevent a brute force attack from being feasible
        - DES (64 bit): Previously considered secure, but modern technologies allowed hackers to brute force just 64 bits
    - AES Example:
        - Input and keys are based on 128 bits
        - Cipher key is the secret key that we use to encrypt the message
        - The data goes through a # of stages and that algorithm will convert the cipher key to form the round key
            - This shuffles them together and repeats the process 10 times (in this case)
            - The cipher key goes through the key schedule and transforms the original key into something else.
            - These keys are derived from the original cycle.
        - Message goes through all the operations and produces the Cipher Key
            - Four different operations and these are very straightforward operations.
                - 1. SubBytes
                    - For every byte, it looks up a pre-determined table to replace it with something else.
                    - It does this for every byte within the 128 bits
                - 2. ShiftRows
                    - Shift the rows over by an arbitrary amount
                - 3. MixColumns
                    - Take columns and multiply by the matrix to mix up the bits to match it to a different one
                - 4. AddRoundKey
                    - These transformations are repeatedly executed

        - From cipher text, we want to ensure the hacker will never be able to decipher the text
            - What would be the formal definition of the privacy here?
            - It isn't easy to understand or formalize!

Information Theory

- Shannon (Crossover from CS 118)
    - How much can we info can we take and what is the limit for it?
    - Shannon proposed what is called **"perfect secrecy"**
        - If a particular encryption function is "perfectly secret", it should satisfy a certain constraint
            - P[m | c] = p[m]
            - Concept of **one time password (OTP)**
        - Important property that we want to guarantee
            - When we talk about some of these other encryption functions, we will briefly talk about the perfect secrecy of the original message

If I want to send my credit card information to Amazon secretly, what can I do?

- I can encrypt my credit card information with a secret key.
- Amazon can decrypt it and receive my information

Is there a missing hole here?

- The problem is the **key-establishment problem**
    - How does Amazon actually know which secret key I actually use?
    - What can we do?
        - Use **out-of-band communication!**
            - If there is another channel where we can send information, we can send it there.
            - Once we establish a shared key, we can go back to the main channel and share information once we establish the shared key.
- Let's say there are 3 people in the room, but we want to share secret information.
    - Look at the notes.
    - Problem is that it isn't very scalable!
        - Uses n choose 2 keys where n is the # of parties
        - No simple solution except this new scheme
- Currently, the algorithm we picked was <u>symmetric key</u>

Asymmetric Key Cryptography

- Encryption progress where message m is encrypted by function f
    - Use Encryption key e and Decryption key d
    - Look at diagram.
- Say you want to send your credit card information to Amazon
    - How do I send my message secretly to you?
    - To encrypt the message, you have to tell the other party the encryption key.
        - Use the key to encrypt it.
        - The receiver gets the message (any other parties won't be able to decrypt the message because they do NOT have the decryption key!)
            - You never shared the information with anyone, so no one else can find out what it is.
    - You have to protect the decryption key with the strongest password possible
        - Any encrypted message shouldn't be able to be decrypted by anyone else EXCEPT me
- **Encryption key == Public key**
- **Decryption key == Private key**
- Now, you can establish secret connection with anyone and only I can decrypt this.

Key Pairs

- Under this scheme, it should never be possible to guess the decryption key from the encryption key one
- Consequently, the keys need to be together for the security to be effective.

Next topic: RSA

W 8 W Lec 2-24-16
Today's topic

- Security

In order to connect secure transactions on the web with piece of mind, we need three or four different kinds of properties.

1. Confidentiality: When I transmit information over the Internet, even though someone may be able to intercept it, we don't want them to understand

2. Message/data integrity: If someone tries to tamper with the data, we want to ensure that this never happens. We want the integrity of the data to be preserved!
3. Authenticity: When I talk to Amazon, all I see at the end of the day is a whole bunch of packets that I received from somewhere. We want to ensure that when I see something, I want to make sure it really came from Amazon. We want to make sure we are really talking to the real party we are expecting.
4. Authorization:

- Some applications may only care about one of these, or all of these. After we discuss these three important properties, how can we ensure that our data integrity is preserved?
    - Make sure that they don't get the actual message
    - Encryption: we shuffle the bits randomly based on some secret key. Using the secret key, they can reshuffle the data to get the message back
- We need to come up with a particular secret key as long as I am the other party.
    - We need some way to mutually agree on the shared key.
    - On the Internet, the main problem is the shared key establishment problem
- People actually came up with a small trick to this overall scheme.
    - If I want to get any secret message, send the encryption key and get encrypted cipher text.
    - This is really my private key and I will never share it with anyone and someone can make use of that encryption key.
- Advanced scheme that is used and is the cornerstone of Internet security.
    - This **encryption key** is shared with the public (**public key)**
    - The **decryption key** is private (**private key**)
- Public and private keys are used for asymmetric key cryptography

RSA

- How are they chosen?
- In RSA, in order to generate the encryption and decryption key pairs, we first have to select two large, random prime #'s
    - p, q
    - **These have to be random and large!**
    - Pick a value e
        - It doesn't have to be random!
        - A popular choice is e = 65537
- In order to generate the private and public key pairs, we solve this equation
    - de % (p - 1) (q - 1) = 1
- Secret key becomes d, n

    - n = pq
    - Throw away p and q after (NEVER keep the trace of p and q)
    - (d, n) becomes the decryption/private key
- Multiplication of these two random prime #'s and e can become the public key
    - How do I use these key pairs to encrypt data?
    - How do I use these key pairs to decrypt data?
- F(m, (e, n)) = m^e % n
    - This will become the new encryption function
- F^-1(c, (d, n)) = c^d % n
    - This will become the new decryption function
- (m^e)^d % n = m
    - This theorem shows that this is the case for taking m and raising it to the power of e
    - We will get back the original message from here
        - This guarantees that you can recover the original message from the decryption/encryption process
- Two original important properties to ensure the security of our particular cipher
    - We should NOT be able to guess the original message m from c
    - We should NOT be able to guess the original decryption key d from the encryption key e
- The hacker has access to our information
    - What does the hacker know?
        - m -> **c**
        - **(e, n**)
        - (d, n)
        - Hackers have access to the any bolded values
            - What are the original relationships of these compared to m
    - Cipher text is the result of the original message raised to the power of e, mod n
        - c = m^e % n
    - Hacker knows c, e, and n!
        - The only one unknown is m!
            - This is BAD! They can grab our original message **m**
- All the smart hackers have NOT been able to solve the equation of **m**
    - This is the cornerstone of solving the RSA scheme (**RSA problem)**
        - There is no mathematical problem, but people haven't been able so solve this problem.

Secret key = (d, m)

- As long as they know m, they can find d

    - How did we solve this?
        - Pick two large, random prime #'s and obtain the d value
- If the hacker is somehow able to solve this problem from d, they can solve for the encryption key based on the decryption key

In order to solve the equation, we are missing (p - 1) (q - 1)

- n = pq
    - Everyone who knows RSA knows this information
    - If the hacker knows n, and they can factorize this n into p and q, they know n
        - We should never been able to factorize into p or q
    - That impossibility is dependent on the inability to factorize large play
- If someone solves the two factors, stay away from the Internet!
- Brute force attack is NOT feasible and we cannot use this info
    - How big is m?
        - ~128 bits

AES

- Shuffles bits all the time
- Expensive but NOT as expensive as RSA
    - RSA raises values to a huge exponential value
        - Several orders of magnitude that symmetric key

Q. If RSA raises it to such a large number, how can the OS store it? Isn't there a limit to integers and doubles?

How can we ensure the properties that we want to store?

- Go to Amazon, give it the public key, encode it using the asymmetric key cipher
    - This whole process can be very slow
    - We worry about not only correctness, but also efficiency
- How can we make it faster?
    - Message needs to be separate KB
    - The key is maybe 128 bits or 256 bits
- The key is several bytes
- They want to avoid using asymmetric key cipher, but we can use the asymmetric key cipher to establish a shared key
- Using the actual message, almost all encryption schemes will use a variation of this idea
    - Instead of encoding my real message using this public key, we can pick up some random private key.

- Using the private key, we can send it to Amazon and it is using a symmetric key cipher
    - Decrypt it using the symmetric key cipher
    - That private key can send it to Amazon, and Amazon will decrypt it using the key I just sent.
        - Don't pay all the overhead.
- How we can get the benefit of the asymmetric key cipher without paying all that cost.
- Authentication
    - Ask the user a question only they would know
        - Username and password!
    - Talk to Amazon and ask them for their private key?
        - Can we do that? NO! Private key is something we will never share with anyone.
        - We don't have to know the private key, but we want to ensure that Amazon knows their own private key!

Challenge/response

- I generated a completely random #
    - Encrypt the random # using Amazon's public key and send it to Amazon.
    - If you are Amazon, I dare you to decrypt the message and send it back to me.
    - If Amazon really does hold the private key, they can decrypt the private key and send back the message.
    - If the original sender confirms that the sent back message is the same as the expected value, we have authenticated the receiver!

Integrity

- We want to see if someone tampered with the message and see if it has NOT been tampered with.
- How can we ensure this?
    - Think about real-world scenarios
    - Get a letter from grad school or the Dean, how do we know if it authentic or not?
        - Make sure that the letter I wrote, what do people do?
        - They add their signature at the end, and supposedly they should be the only one's who can add their signature.
    - By embedding something only they can generate, we check the integrity of this document.

- Somehow, we want to attach some additional information and by ensuring that it is theirs, we can say whether or not the message has been tampered with.

Compute the hash value and attach it to the end

- The hash value will NOT match the expected hash value and we can tell that it has been tampered
- Hackers try to thwart this by overwriting the original hash with their own modified hash
- We want to make sure this hash value is the same as that attached by the original author.
    - It cannot be generated by someone else.
- How do we know this part has NOT been tampered with.
- Think about the encryption scheme and the decryption scheme

Whichever function (encryption and decryption functions) is applied first doesn't matter!

- The key idea of this scheme is to make sure the message integrity is working.
- Compute the hash function and encrypt that hash function using my private key.
    - If I receive this message, what I do is check the message and decrypt this message using the public key encryption key
        - This hash value is the same and that hash value will be the original hash value
        - If a hacker wants to tamper with it, the hacker should recompute the hash value based on the tampered content and encrypt it using the secret key.
    - The hacker doesn't have that key and we want to decrypt it using the decryption key
        - That is how it works and only the person who knows the secret key can generate this!
- Only the person who knows the secret key can generate this!
    - There are a lot more complications, and the hash value is based on the content and it is a result of the public key.
    - This is how we ensure message integrity.

Q. In order to send secret information to Amazon, I want to encrypt the public key, but how do I know if I am really talking to Amazon?

A. In order to check the message integrity, the signature has to be decrypted using the public key of Amazon. Where do I obtain this public key of Amazon?

- PGP scheme: What you can do is if you want to send a secret message on a secret email, you can send a secret/important message and use this to communicate

- Many cases, go to the real website and this is authentication for that person.
- Even then, this can be tampered with so how do I know if it is really an authenticate public key of who I want to communicate with.

Every browser should have every public key of every browser embedded in it.

- What are the problems with this?
    - There are so many websites (old and new) that are coming and going.
    - Read out the public key almost every single day and they don't really want these
- How can we ensure it?
    - Even though we talk about this problem in the context of Internet transactions, this is a problem that everyone has.
- I am John Cho, I want my money
    - They don't just give him money, they want to verify his identity.
    - In reality, there are some schemes that we implemented in reality to ensure that we check the identity of the person
- What does the bank do?
    - They ask for your **Identification**
    - They look at the ID and it seems to be John Cho
        - This single piece of crap (why do they trust it?)
        - They have a **hidden assumption!**
            - Only UCLA can generate this plastic
            - Driver's license or passport, only the government can generate these forms of identification
    - A lot of security schemes and these can make sure it is difficult for anyone else to generate.
    - They check and see if it is the authentic one.
- In Internet, they replicate the same things to check that the public key is really the public key that belongs to them.

PKI (Public Key Infrastructure)

- CA (certificate authority): In some sense, the certificate authority play the role of the government on the Internet.
    - They are able to authorize or authenticate a person and give this ID to them for individual companies.
- Approach one of the certificate authorities and get documentation IDs
    - They issue an ID called a **certificate:** This is Amazon, I want to verify this is really Amazon
        - Verify it and trust them, almost like our ID
    - Generates this certificate and this is their public key.

    - As long as I can trust this certificate authority, then as long as the browser can verify the CA, then we trust it.
- Let's say there is a single certificate authority. If Amazon wants to do anything else, we can send this certificate and ensure this has been generated by the certificate authority.

Q. When a browser receives a certificate, how can a browser know that it was generated by a certificate authority?
A. It has to make sure it was generated by a particular certificate authority. One possibility is to contact the certificate authority and they can confirm if it was generated. Almost every transaction would contact the certificate authority and that's a nuisance.

Q. How can the browser authenticate without contacting the certificate authority?
A. Add a signature to it and this is the message integrity problem. This message should be generated by that particular thing. Use their private key as part of it and the browser should be able tok now the public key. Possibly hundreds of certificate authorities and they should verify whether these are decryptable or not.

- CA plays the role of the government and certificate plays the role of the ID
- Use the public and private key-pairs and add a signature to them

Trusted Root Certification Authorities
- There should be certificate authorities
    - Symantec
    - Microsoft
- These are the certificates that the Windows OS trusts
- Maybe Apple doesn't trust Microsoft haha!

We have to get divine inspiration and use a random # generator to select a secret key
- We can never remember the secret key
- They are encrypted using the passphrase we have picked
- Those passwords are NOT encryption keys
    - They will be decrypted using pass phrases and they can be used for actual transmission
- Even that scheme seems kind of insecure
    - We have to get a hold of encrypted random # and we have to know my pass phrase in order to decrypt it.
    - It is NOT an impossible task and we can decrypt it.

    - Just ensuring that things only they are supposed to know are supposed to be compromised.
- Many of the companies, especially the big ones become a target of many of the hackers
    - Checking only secret information that they know about
    - If you go to Google, add HTTPS
        - Apple account, you can enable 2-factor authentication
- If you work for companies that are very security conscious, they often give you this device called a **Smart Card**
    - Only their employees will have access to it.

OTP card

- In order to log in to your account, you have to type the # that is there at this particular point and time.
- People cannot provide this random # and the fact they can provide this random # means they are in possession of this device.
    - The device that you are in
    - Google authenticator can be used on your smartphone and by ensuring that you type in your credentials at that time, Google can verify it really is you.
- Interception of random #, 30 seconds later, the hacker can use the same pair of information and the random # can be completely different
- Double down on their security.

HTTPS

- Has a lock to show that the site is encrypted
- This ensures it has been generated by Google's public key
    - Generate a random # and send it to Google and we can use this shared secret key to share all the communicated messages from now on.
- Lock symbol with www.google.com
    - verify that this site is safe and if you click the Details, it will show how it verified it along with the Certificate it contained.
- You don't want anyone to see the message that you are exchanging, so out of all these complicated schemes, we don't need certificates issued by one of these trusted certificate authorities.
- What is unfortunate is that the certificate authority needs to be vigorous and careful.
    - Prove that you are the entity that owns that particular domain
    - You have to do a lot of legwork to make sure you are fine.
    - Typical kind of price is $100

- You sometimes want to communicate with oak securely.
- Chrome fails in authenticating the certificate, but at that point, if they intercept the message, they cannot interpret what is being communicated.
    - That could be fine depending on the situation

W 8 Dis 2-26-16

Desired Guarantees

- Confidentiality
    - Make sure that this conversation is only happening between two of us and no one else can get the content or info between us.
- Message/data integrity
    - When we send some message, we don't want anyone to change or modify the content of the messages.
- Authentication
    - We want to make sure the other party is who he/she claims to be.
- Authorization
    - Step after authentication
    - After we have authenticated the party, we want to control access to resources.

These are the 4 basic requirements for secure communications

- We need some framework via encryption

Encryption Algorithm

- In the insecure channel, we don't want to send plain text because other people can see or change the message, so we have some encryption functions and encryption key to encrypt this message.
- This produces a cipher text, not readable and not decipherable
    - The other party would be used for a decryption key and we would be able to get the message
    - Based on this framework, we can categorize the encryption algorithm into two broad categories.
- Encryption key and decryption key will be the same for symmetric encryption
- Encryption key and decryption key will be different for asymmetric encryption

When we get some cipher text, we want it to fulfill some properties

- Perfect secrecy: When we are doing a random guess about a message, we have 1 probability to get it right
    - When we actually see the cipher text, it should NOT change the probability of guessing the actual message
    - Every encryption algorithm tries to achieve this communication

    - When we try to achieve this communication, naturally, some guy came up with a method
- Send some password and if other people could decrypt the text or guess the password, we will keep it as a secret
    - One-Time Pad (OTP)
        - Change the password immediately after we do the encryption
        - Guarantees perfect secrecy and we will never use the same key when we do encryption
    - One potential problem: sending a single character or a single word

Symmetric-Key Cryptography

- We use the same key to do decryption and encryption
    - Use XOR to do encryption and decryption
    - We originally had some message and we designed some encryption key
        - We are going to do the XOR operation between them to obtain the cipher text and just send this message to the other side and use the same key to perform another XOR operation.
    - The property of the XOR allows us to do this!
- Common problems for symmetric-key cryptography
    - Uses the same key on both sides
        - You have to synchronize Part A and Part B!
        - This is the only way to allow for communications
        - You need to find some way to make an agreement on your key
    - We can find another secure channel so we can pass in the key and communicate in that way
- So why do we still need encryption?
    - It is hard to agree on **k**, it is difficult to pass keys between the parties to ensure security.
        - How can we make sure each property of secure communication is satisfied?
            - Authentication
                - We found ways to pass the party around
                - We know that if we have a key at A, we would also have a key at B
                    - We can send a challenge message (i.e. username and password).
                    - If the receiver can read it just fine, then they are authenticated
                - Send a challenge like a message (a random #)
                - For N parties, our probability is n choose 2

        - Authorization
            - If A is trying to send a message to B, A should have a message and also use K_3 to encrypt this message and send all of this stuff to B, and B can send all of this stuff to C
                - C can verify this info and this allows us to verify at each step if the sender has permissions

Asymmetric-Key Cryptography

- These requirements look quite trivial
    - When we encrypt something, we use an encryption key with the cipher text as well as the decryption key
    - Get the message spit out again, and this is the whole encryption/ decryption process
    - When we get the cipher text, you cannot get the original message from the cipher text.
        - If this is not satisfied, it is not cryptography
    - The 3rd one is about the key
        - They are NOT trivial because they are a mathematical constraint
        - When we want to satisfy such properties, we can use these as an implementation of the asymmetric-key cryptography
        - This is the theory that we have such an encryption key, decryption key
            - We need actual #'s to help us finish this process
    - RSA is a method for this
        - One implementation i.e. Number series result and theorems can be used to construct private key and public key
            - These encryption/decryption conditions
            - We can do this to do one implementation of this asymmetric key cryptography

Asymmetric-Key Cryptography Applications

- Confidentiality?
    - After there is an agreement, we can communicate with the symmetric key
    - After we use the symmetric scheme, it will be quite efficient
- Authentication?
    - As long as the other party can decipher the message and send it through back to me, then the other side will say it is authenticate it.
- Integrity?

    - We can always compute the checksum of some message and send the encrypted checksum to another party.
    - We can always compute a checksum and encrypt this checksum in this scenario
    - Send the file and checksum to another party.
    - You can encrypt the file or not depending on the requirements
    - Only authorities can prove your identity via driver's license or passport (official government document)
        - Use another party (**central authority**) that we can trust i.e. Symantec
        - When someone claims to be party A, we need the central authority to endorse this person
            - Each time this person sends info to me, he sends a public key that is also endorsed by the central authority
                - In this way, we can trust this party to be the authentic party that we want to talk to.
                - See the whole message as well as the endorsement together (**certificate**)

Common Vulnerabilities and scalability

- SQL injections
- Buffer overflow
- Among them, which is the most common vulnerability in the computer
    - Phishing
        - Very common vulnerability on the Internet because of fake websites
    - Before 1990 (World Wide Web), we had common vulnerabilities
        - Buffer overflow
        - #1 vulnerability in the computer

Buffer overflow

- When we write a C program and we have local functions, if we declare some local variable and corresponding memory space to be allocated for them, when we finish the execution of the local function, we are going to return.
    - When we return, we load some return address and we go back to the function call.
    - All this information is stored in the stack
        - Stack overflows come from this issue
- User can type in some password and the potential problems are if the user forgets the password

        - If the user types a much longer password than the limit, what will happen is that there will be an overflow that will overwrite the return address
        - In this case, if there is a careless user who types a much longer password, the software sucks!
        - However, if it is NOT a careless user but rather a careless hacker, they will deliberately set a return address and have malicious code to point to a position to where they put the malicious code.
            - This is a very common vulnerability.
            - Quite easy to implement
                - Try to use buffer overflow and this is the #1 vulnerability
- How can we avoid buffer overflow?
    - When we finish the function execution, we will check if the number is changed and if it has not been changed, we will return.
    - If the number has changed, we will show an error message indicating an overflow problem
    - Activate a “stack guard” with a canary!
- Client-State Manipulation
    - After you check a value and send it to a server, you need to finish a transaction and there is a big vulnerability in such a design.
    - If we have a form, send it to a server, and this form contains the price, it will tell the server what the price is of the product.
        - We can manipulate the price to something much cheaper!
    - Never store sensitive information in the client

<Get notes from someone later>

W 9 M Lec 2-29-16

Today’s topics

- Common application vulnerabilities
    - buffer overeflow
    - client-state manipulation
    - command injection
    - XSS
    - XSRF

Allow users to input the sensitive information like credit card information

Project 5 Info:

- Use https for safe security reason
- Interface should be mobile friendly!

Almost done with the discussion of security. Remaining will deal with scalability and issues of adding users

- RSA cipher deals with two random prime #'s and multiplying them together gives you an RSA value
    - We assume are #'s are impossible to crack
    - c = m^e % n
        - If you know c, e, n; you cannot solve for m!

1. Confidentiality: Encrypt everything (asymmetric key cipher is difficult to encrypt, the real encryption uses established keys)
2. Authentication: The basic idea is that we want to make sure that the other party only knows the information it should know (use the private key only). To ensure that the other party has the info, use **challenge and response** like username and password.
3. Message integrity: Message has NOT been tampered by any other person
    1. Compute a checksum of the sender and attach the checksum against the checksum of the receiver
    2. Encrypt it using the private key; no one else can encrypt the private key in the same way as you.

Internet has an open nature

- Inherently insecure
- https ensures browser authentication, confidentiality, and message integrity.

Fundamental limitations of the Internet

- Another kind of problem!
    - When programming the website, we will go over 5 common mistakes when we implement things as a web-application.

buffer Overflow Attak

- All these things are done in main memory and you even have next=register flag

attack String

- Your programs and modern company try to avoid using newer languages i.e. Java and C++
- There are still a lot of C programs lying around but we don't have the resources to upgrade them.
    - How do we modify these while maintaining the original files and structure?

StackGuard

- Send canary in first to test if the the scheme is working.
- Without stack guards, we would have to write in the new address and that would become the new stack.
- At the end of the function call, check to see if a local variable modifies all the files.
    - Stack guards must have different latencies
- All of the canary should always be random!
    - If the attacker knows the side, can he still overture
    - User cannot guess a random #, so this will be different from the canary then it will be for us

Client State Manipulation

- Whatever the user has to remember, I just push it tot he user
    - include an input type i.e. “hidden” with a corresponding “item”, “price” and “value”
    - Look at particular the of sending request
        - The user trusts the server and there is an implied agreement between sender and receiver

- How to avoid this problem?
    - Never trust user input!
        - Basic mode of operation that any code should have.
    - If you don’t trust users, what can you do for dealing with this kind of thing?
        - You can store the value on the client!
- All sensitive info always stays in the server
    - Even for this scheme, the user still has a transaction and a section ID associated to the server.
        - The user might then be able to do something that the user won’t want to be able to do.
- How can we ensure that user changes their session_id to something else?
    - We want to pick a large random # so that it is unlikely to be called a valid session.

Let’s say today is Black Friday

- Users buy a bunch of things at a cheap price
- We need to ensure an expiration date that we can continue until we no longer send data.

- Client State Manipulation is **VERY** important

    - Easy to make mistakes in, which is obviously very bad.

SQL/Command Injection

- We want to let the users look up and retrieve product info based on Product ID
    - What is the problem with this example?
        - The user can possibly put malicious input and perform SQL injection that can wreak havoc on your program.
- The problem is we are too trusting.
    - NEVER trust the user! This is the bottom line because they can perform many malicious things.

Web speaking languages have a property called **taint propagation**

- Python
- PHP
- taint propagation: Each variable has a label called tainted
    - By default, the variable is NOT tainted
    - The output of the function call is flat with tainted info

```
a = gets();     //a is tainted here
c  = a + b
```

- If I put the tainted variable as part of the argument, you will get a warning about this!

In order to avoid taint progression

- The programmer can explicitly untaint a variable

```
untaint(c)
```

- Doesn't really protect it, but it helps the application program

Common kinds of injection attacks

- SQL
- exec

Many modern mechanisms provide mechanisms that prevent injection attacks from happening

- Used a **PreparedStatement**
    - Set particular values using "?"
        - **whitelisting** user input
        - Only take it if it's a valid sequence of characters
- Always use PreparedStatements with this sequence of variables.

Project 5 Notes

- Rather than putting constraints or restrictions on sensitive data, you might want to just encrypt those information.

Cross Site Scripting (XSS)

- This is how profile page's are implemented!
- What is the problem is this?
    - You can have a cross site scripting attack, but what exactly does this mean?
    - We should NEVER trust users!

<script> hack(), </script>

- Any user can go to this particular user's profile page and this particular script will be executed.
    - A naive user who has a browser can visit this profile and get hacked.
    - Be wary of whatever the JavaScript function call does
- As a FB user, what do you assume?
    - Anything coming from FB is trustable.
    - As we execute with the privileges of the FB call, we want to be aware that information can come from another party.
    - This script does NOT come from this side but rather from another domain.
    - This is what is called XSS (cross-side script) attack
- How do we protect against this kind of attack?
    - When we take in user input, if it is okay, we should never let users type in HTML tags
        - In many cases, user's cannot embed any script so we would be reasonably safe in that case.
    - In some cases, it might be necessary to allow HTML tags, so we have to use multiple mechanisms
        - Allow minimum possible safe HTML tags to be used.
        - FB actually uses this particular scheme -> develop their own formatting language similar to HTML but not the complete same
            - A lot more advanced situation that has standard HTML and create your own language.
- Sometimes, the script can be embedded in places that we never expect!
    - Well-known example:

<img src = "javascript: _____">: User can embed malicious code in places we didn't expect like this one.

    - See possible standards that are allowed there and be mindful that nothing bad can happen.

- Given these problems, all security people recommend doing is whitelisting: Explicitly specifying the sequence of known strings that are safe
    - Everything else is rejected
- This is the opposite of blacklisting: an explicit list of unsafe strings
    - Everything else is accepted (okay!)
- Security people always prefer whitelisting users

Cross Site Request Forgery (XSRF)

- Whenever the user sends the authentication, this allows the user to do whatever is necessary
    - Victim.com will set the cookies and the user's don't have to worry about additional embedded information
- Until the user logs out, victim.com will be authenticated
- Cookies enforce **same-origin policy**!
    - Only be sent back to the same domain
- Whatever JavaScript is there will never be able to see or access the cookie for victim.com
- It will never be able to see the cookie, which is nice.
- Action field changes the URL from victim.com to evil site.com
    - In that request, it will be submitted to victim.com
    - This request that is crafted by evil.com will be executed!
    - It does include! Any request going there is required to include all of the open sessions.
- The valid session ID will be attached to that particular request so we have to check if the user has authenticated or not.
    - This can be used to do malicious things like transferring $1 million from one back account to another
- Another website forges this genuine looking request and sends it back to the victim website
    - They can still use it to attach a session ID to the request.
    - There are a lot of things that you are NOT supposed to do in XSRF

One lesson to learn from all this:

- For sensitive things, it is important to close that session
    - Log out
    - Close the browser
- This is especially information if the website pulls a lot of content.
    - As a user, this is something we have to do from the user's perspective.
- From the website programmer's perspective, how do we understand that this is detected and somehow rejected?

    - Any time you send a form to the website, you would have to also include a secret key.
    - The 3rd party should NEVER be able to send a forged request.
    - If they send a forged request, we should be able to detect it.
- It can essentially use the security mechanisms that we learned i.e. **action key**
    - That action key is encrypted using the secret key
    - When a user downloads a page that has a user request, that particular page embeds that action key and that is a random # that changes in each of these sessions
- Action key can be included in many different places and you can put it in a hidden input field.
    - Any request will have that action key attached to it.
    - The action key can be used to include victim.com
    - This is how we enforce the origin of a particular request.

W 9 W Lec 3-2-16
Today's topic
- Scalability

Common issues
- Some of the scripts may be embedded by malicious hackers
- Given that the user can embed some bad code, if we don't want to let any bad code run, we should look at all the user inputs and make sure it doesn't do anything bad.

Cross Site Request Forgery (XSRF)
- When you don't close a particular session, the 3rd party app cannot see your cookies but they can generate a request that is valid for your particular website.
- In order to protect against it, the application developer can process a request and make sure that the request is authentic.
- What we could do is insert an action.
- Something only that action can generate.
- Make it originate from the website.

SQL injection
- The user may be able to craft a particular input so that the system can execute the command that is injected by the user.
- This means the user input shouldn't contain the original code.
    - Commonly exploited attack in SQL with ways to protect someone.
- Basic operating mode in which the user is NEVER trustable

    - User input should be carefully validated by a particular string i.e. whitelisting
        - These are the things allowed and if it is NOT allowed, reject it!
- We design or write apps in order to protect against these common vulnerabilities.

Scalability

- As your website becomes popular, you have to worry about using particular hardware that you start your application with.
    - How do we handle this as demand for our application increases over time?
        - First topic that we want to talk about is capacity planning.
        - When I buy hardware (cloud server), we need to figure out roughly how many users are able to deal with it.
            - How many requests can I serve per second?
        - Based on this estimate, we can get the expected loads from the users
    - Estimate how many users we will be able to deal with.

Reading from disk

- The speed ranges from 100 MB - 500 MB: this is the rough range at which we can read data from the disk
    - The typical disk that the server uses to store the data is about the fastest rate we can store data
    - Mechanical device to locate the disk head (read or write head on the write track)
        - When we say we can read 100 MB/s, we don't assume physical movements and we keep reading it from the disk.
        - If the data is NOT already there, it takes time to move things around.
- What is the typical time?
    - The distance that we have to travel on average is 10 ms
    - You have to wait for it to come below the head and ignore the rotational delay.
- How many random I/O's can you perform?
    - This # is about 10K for an SSD
- For 10 ms, how many random I/O's can we do?
    - 100 random I/O's
- SSD is so much faster because it has a much better seek time
    - Cho recommends to use SSD and he bought an SSD for $80

    - Best bang for his buck in his whole life
        - This machine is 8 years old and slow as balls, but spending $80 allowed him to keep using the laptop for 5 more years (damn Cho!)
- We don't know exactly how things are in our application and we don't want to worry about CPU cost.
    - Read the data for the disk and translate over the network.
    - The majority of Cho's website is static (HTML files and images)
- How many requests can we deal with?
    - Have a rough estimate of how individual responses might be.
    - One page on average can be 10 KB/page
- Let's try to see the transmission speed to figure out all these #'s
    - Assuming we don't have to worry about the random seek, how many pages can we translate from the disk into the RAM.
    - We can translate 100 MB/10 KB = 10,000 pages/s
- The maximum we can transmit is 10,000 pages/s using a magnetic disk
    - Assuming we have a random seek, the # of pages we can send is the random seek time.
    - We can have one random I/O that costs 10 ms.
- 1000 m / 10 ms = 100 pages
    - In terms of the network, we can deal with 10,000 pages/s and we can deliver between 100 pages and 10,000 pages/s in theory.
    - Can the real software actually achieve this rate?
- Very well-optimized servers make it easy to serve this theoretical amount.
    - Important assumption is that we don't need to do 1 random seek for every request.
    - If it is residing on SSD, we can do much better.
    - In order to achieve this, what do we need to do?
        - We need a large enough main memory to cache the content.
        - The random seek can guarantee it and achieve things to serve 5,000 pages/s

node.js

- Server framework (implement a web server using JavaScript): We can implement a backend using an HTTP server
- Before node.js, if we actually think about a lot of big companies, they are using the Java servlet framework
    - As you have implemented so far, in order to deliver the content, we run a few disk accesses or SQL queries and once we have this resource, we deliver the content to the user.

- If things are implemented this way, most of the time is spent retrieving data and we wait, and we wait, and we wait some more.
    - This is where we spend most of our time. As we can see, this can get slow as balls.
    - We need to invoke one thread for every request, and if we deal with simultaneous request, they will get stuck at the database query.
        - Once the # becomes reasonably large, the server gets bogged down.
        - The reason node.js can be much more scalable is that they realize that data access makes things more scalable.
    - Event-driven programming (asynchronous) data access
        - Requests data that we need and we can have the callback function and in that way, instead of creating tens of thousands of threads, one single request can deal with this whenever data is available.
- Different parts of the page are created dynamically and if we think about it, there is only a very small portion of a lot of requests to the server.
    - Access a particular portion of the data and the data access can be much simpler than without the server.
    - Each individual requests can be involved using data access.
    - Server environment is more suitable and this is why many big companies have migrated to node.js due to scalability from asynchronous data access.
        - Cost of using node.js is that we have to worry about node.s's asynchronous nature
            - As long as the server is simply accessing the data, then node.js is a very scalable service.

Simple case where we don't do complicated calculations

- The fact that we are doing dynamic things is that we are doing a lot of complicated actions.
- How long would that take?
    - Depends on the complexity of the problem i.e. video encoding is hard, arithmetic is easy.
- What is the rule of thumb I can sue to plan for this capacity?
    - How many requests can I serve per second?
    - Over time, people looked into it and this is an important question to ask.
        - Per core, we can usually deal with 10 requests per second.
- Is there some magic on how many pages we can serve per website?
    - Why this #?
        - This is a good # to assume because developers are lazy.

        - You develop something and we don't worry about optimization too much.
            - Implement whatever works and then check if it is fast enough.
    - Loading 1 page in 100 ms is almost instant.
        - We are generally happy because we have achieved our goal.
    - Include all these complicated functions and try to optimize.
- Take away some features and do all these until the response time is in sub-seconds.

Dynamic webpages
- We can serve 10 requests/s in this case.

Terminal
top: shows all the processes running on your server and shows the process that uses the CPU resources.
ps -ax: shows everything including all he background processes running on a particular server
pstree: on Unix, whenever we boot the particular computer, it represents the init process
- Shows the sequence or tree of the processes

iostat
- Shows the average speed we are reading or writing
- Used when you are worried about pressure on the disk

netstat -i
- Shows how many packets are sent per second and how many add-ons there were.
- These are the #'s of packets that I have been sent, and how many received packets were in error state.
    - At least the network seems to be fine here in this case.
- Buffers and cached shows the caching in the disk content
    - Monitor the state of the Unix server.

Increasing scalability
- Here, you can buy either bigger or larger servers, or you can buy more of the same kind

Various approaches
- Scale up: buy a larger, more expensive machine
    - Advantages

        - You don't have to worry about rewriting or changing your software.
    - Disadvantages
        - The # of requests we can deal with vs the amount of money we would spend
        - Given a current technology, there is an upper limit to what we can do.
- Scale out: add more machines
    - Advantages
        - No upper limit; more requests, just keep buying more computers
        - Given the cost advantage, and the enormous traffic that some of these web companies deal with, scale out is the approach most companies use nowadays.

How do we employ multiple machines to deal with growing user traffic?

- Encryption layer is completely parallelizable and they can be done completely independently.
    - Easy to scale out, run the encryption engine and spread it out
- HTTP layer
    - What does it do? It gets the result from the application and sends it to certain endpoints.
    - The capacity runs out not because one single request is so expensive, but rather there are so many of them.
    - Independence of different requests could entail using different machines and process them on different machines.
    - Use the scale out approach
- Application layer
    - In order to see if we can use scale out for that layer, we have to see why the capacity runs out and see if the different requests are independent of each other or not.
    - They can be highly correlated and if we go to Gmail, the fact that I am accessing my Gmail account is independent of another use who is using Gmail.
        - They are completely independent of each other
    - Facebook
        - The fact I am looking at my Facebook timeline does NOT affect someone else's Facebook timeline.
        - Some relationship but mostly those are independent.
            - They will share the same data and as long as these multiple app servers are accessing the same database, then we

know there are inter-dependencies if they share the same data.
        - In many applications, there is no interdependencies between these types of requests.
    - As long as these App servers access the same server, we will be fine.
- Highly scalable using the scale out approach

Database layer

- Most difficult layer to scale out
    - If you run a bank, people are willing to see no error in your data.
    - You go to Facebook and a user posts a status update. Is it essential that we never lose the status update?
        - We buy something on the Oracle server and we don't want to pay that much money to guarantee no problems for the database server.
- We will go over some simple examples of some characteristics for certain kinds of applications to scale out our database.

W 10 M Lec 3-7-16

- Scaling out for various layers
    - Application layer
    - Persistence layer (database)
- Encryption is pretty easy to scale out
- HTTP layer and application layer is also fairly easy to scale out
    - The persistence layer is harder
        - Dependencies
        - This layer needs more attention so we try to think about 3 different examples whose data access characteristics are slightly different from each there.

Page 3 of notes

- Global read-only
    - Google Maps
    - When I access Google Maps, am I writing into Google Maps?
    - Read all the data that is there and there are a lot of similar applications i.e. Yellowpages
- The user only reads the data and does NOT write into the data
    - We have to do 3 disk IOs and this is the assumption for the application.
    - Every machine can support 30 IO/sec and on average, we can support 3 IO/request

- We can support 30 IO/sec and one request requires 3 IOs.
    - We can support up to 10 request/sec
    - Maybe 10 users can use this simultaneously per second.
    - As the load we are getting is > 10 request/sec, in that case, how do we scale out?
- Google Maps: how big will the data be?
    - Several terabytes of data (huge!)
    - What was the storage capacity of the devices?
        - The typical kind of user requests is essentially vectorizing the map and those can fit the size of that data.
            - That data could be around **20 GB**.
- How can we use multiple machines to handle all these requests?
    - We can copy or replicate all of the data on the same machines
- Main assumption we make is that the data is NOT that big.
    - All the data is purely reading and there is no writing from the request.

Global <u>reading</u> => replication

- If it is only reading, then it is easy to scale out by simply replicating the data.

- What if we also need to write into the data as well?
    - Gmail, if you have an email application. You will have a lot of users and the emails have to be maintained.
        - Which part of the data are you interested in looking at?
            - Only my account data!
- It is reading and writing in this case.
    - If I think about this, it is both reading and writing and the range of the data that we need to look at is very well-defined.
        - I only care about my own data but I can both read and write to it.
        - All you care about is your own bank account and you should NOT be able to see anyone else's account.
- What can we do to scale out?
    - To think about this, let's assume some #'s
- Can we use the same replication strategy to scale out and support more user requests? Is there a better strategy to deal with these kinds of request strategies?
- One problem is the following: whenever data is replicated, we want to deal with read requests and you want to ensure that if anything is written into the data, then that write should go to both of these replicated data.
    - If your request does include writes, then one single write is NOT a single write into the machine anymore!

    - It can incur 2 disk writes into the machine and this will be written into three separate machines.
    - In terms of write, when you replicate the data, it doesn't really increase our capacity and this gets translated to one write for each of these machines.
- When we read from the data, the read can come from any of the replicated data.
- In terms of writing, we don't really gain anything from this.
    - Any writes get copied to all of these machines.
- If you replicate this, what are all the requests we can deal with per second? How much traffic can we deal with?
- If there are 10,000 users, we will be ~3,300 users there and there will be no replication and partition the data into each of these machines.
    - If we do it this way, then we first need to figure out how to allocate to the user.

1 user requesting 2 reads, 1 write will be incur 1 total request. This can be done a total of 10 requests/second and this is if we have an even distribution.

- Deal with these things 30 times/second.

If every request accesses only a small portion of the data, we change how we partition the data and we distribute it equally over multiple machines.

- Local read and **write** => partitioning (sharding?)

Online auction -> you want to see what everyone else posted as well.

- Both reading and writing can be global (not just the data that I care about)
- Worry about all the data for reading and writing
    - In this case, we cannot partition all the data.
- Facebook example
    - Users in China are interested in who are using Facebook in China
    - Users in the U.S. only generally care Facebook in the U.S. unless they have family abroad

In order to exploit partitioning, we need to figure out what is the set of things being affected

- We are NOT gaining anything int terms of the write request, but we are gaining something for the read request.

What is the maximum # of requests w can support if we have read-only and write-only requests?

- Cannot support tor > 30 IO/sec

Helps with the following:

- Scalable
- Strategies that are relevant to a particular application

Distributed file system

- When we use a server, and it opens. Can we create this?
- Google File system
- (HDFS) Hadoop filesystem.

Distributed DB.
There is a single master server that provides a namespace of the Google File System to handle all this stuff for you.

(2) Distributed DB

- NoSQL database
    - Help to protect the application developer for future data consistency issues.
    - When storing the data using the query, just write a single SQL statement.
        - All this convenience comes with a cost….
            - In order to use complicated SQL, you need to use complications that
    - There are a # of SQL databases optimized for distributed systems
        - More of a nice
- Teradata: Built an extremely parallel, extremely distributed database
    - Because the database wants to develop for big companies, they charge millions of dollars for their software.
    - Tradeoff between buying expensive software and buying expensive machines
    - Without their relational database system, it has to deal with access characteristics
        - When they have to implement things, think of ways to commonly access data.
        - At the end of the day, if they can implement an entire database, that will be good.
            - Building this relational database system on a single node took 2 decades!
    - Out of everything that they can do, they can figure out what is the first implementation that you want to have.
        - Can be supported using the 1st generation.

    - Because of that reason, we cannot afford to build it, so we want a much less expressive API that can use the common ways of accessing data.

ACID

- Is there an easy way to avoid using the ACID principle for transactions? What are the API's why need to support?

A: Atomicity (all or nothing): When you perform a transaction, either everything is executed or nothing is executed. Nothing is partially executed
C: Consistency: When we are in a consistent state, we want the data to remain in a consistent state.
I: Isolation: Even though multiple things are executed together, the end result should be the same as if it occurred in isolation.
D: Durability: When we write something and change something, we want to ensure that what is changed stays there.

- When we try to provide this ACID guarantee, things get very difficult
    - To illustrate difficulty, let's use this example of Byzantine generals.
        - How can the two generals coordinate?
            - No cellular devices haha
            - Send a messenger and the general says that we will attack on Sunday 9 AM.
            - The other general can get the message and our enemy will still be very strong
        - The messenger is NOT able to get to the other side.
    - What is the protocol for General A to communicate to General B?
        - They decide on a particular time to attack each other
        - What's the problem with this idea?
            - There is 20% probability that B won't get a message.
            - Any improvement to this protocol to make things better?
                - Wait until you get an acknowledgment from General B (CS 118!)
                - Once General B receives the messenger, General B sends another messenger to General A, and they agree to attack at the same time.
    - If you are General A, will you follow the protocol?
        - This is NOT perfect still because you don't get the SYN-ACK back.

        - General B wants more guarantees and if he gets the acknowledgment, send an acknowledgement back to me and continue this process.
- 0.2 * 0.2 = 0.04 probability that both are captured
    - 4% probability is too high and we can keep sending messengers (if General A is really conservative)
    - The problem is this is that there is no one else left to attack, and this is the main problem of a distributed system
    - Synchronize that information on both sides, and there should be a shared state that both share and this isn't easy.
        - No easy solution to this problem.
        - The cost of maintaining it outweighs any advantage from this coordinated environment.
        - Very difficult problem as we can clearly see.

CAP theorem (by Eric Brewer)
- Consistency
- Availability
- Partition Tolerance

Guaranteeing all 3 of these is impossible!
- You can guarantee at most 2

Q. How can we relax these consistency guarantees to obtain what we need?
A. We provide a BASE guarantee!

Basically
Available
Soft-state
Eventual consistency

- It is very difficult (close to impossible) to guarantee the ACID guarantee, so we use a slightly different model to describe our consistency guarantees.

Different consistency data models to find what we guarantee.
- Read Your Own Writes (RYOW) Consistency: As a user, if I write something, I should be able to see what I wrote. I don't care what other people wrote, but the things I wrote in the same transaction can be checked by myself.
    - Facebook status analogy: If I post something, I should be able to see it. I don't necessarily see everyone else because they can have custom privacy settings.

- Session Consistency: What I have written, I should be able to see. Similar to Read Your Own Writes (RYOW) Consistency
- Monotonic-Read Consistency: If any user has seen the writes up to that point, any reads can see any other reads up to that point.
    - From that point on, what the reads see will be everything up to a certain point.

W 10 O.H.

- To find the nearest point, do we just find the nearest calculation like a SQL query.
    - First check it, compute the distance to the nearest point, and if that distance is small enough so there are no shorter distances, then we have to look at other points.
- How can you be sure that there are no other points?
    - Compare distances to nearby cells.
- Understand the high level syntax

Project 5 question

- Using HTTPS, normally when logging in, the user wants to see the HTTPS and think it is safe.
    - Entering credit card info, you do NOT need it to be HTTPS

Inverted indexes

- Finding the # of words in a dictionary, don't trust the approximation there and use that.

PageRank

- We can have a dangling page and a crawler trap
    - Assume a random jump

Cosine similarity

- Q * D

XML HTTP request

- You can only connect with the server
- Wherever you get the info, you cannot leak it out to anyone else
    - Security reasons
    - For this particular object, you can create a connection only to that server.
- Why aren't we connecting directly to the Google server?

    - It is about the page that I downloaded and when you do your project, from what page was the JavaScript downloaded?

Final

- Will we have questions describing CSS and the main elements in that would be required?

Spatial indices have special queries

Client-state has two solutions

- 1. session id
    - Much more difficult to send info but the server has to create a session to memorize this.
- 2. signed-state sent to client
    - Server doesn't have to remember, but w have to be careful when someone leaves us
    - When I remember, there is NO way to change and remember the value.
    - We could do signature verification but unless we are very careful,t he other party can exploit a security hole.

If you don't add an expiration date, people can exploit this and keep paying the discount rate (this obviously isn't what we want)

XPath questions

- Does the order of the children in DTD matter?
    - Yes it does. We define it as a structure of a list of other elements
- Bookstore -> list of books <title, author, price>

DTD
<!ELEMENT Book (Title, Author) > //Order matters in this DTD definition
<!ELEMENT Title> //Order doesn't matter for these
<!Element Author>

OR

<!ELEMENT Book (Title, Author) > //Order matters in this DTD definition
<!Element Author> //Order doesn't matter for these
<!ELEMENT Title>

W 10 W Lec 3-9-16

Partitioning

- Each user partitions the data so that every user goes to different machines
- Positives
    - Scalability in terms of the read request and the write request has to go everywhere.
    - Guarantees whether a particular message will be guaranteed or not.
- In a distributed environment, the challenge of keeping a consistent state over time is a bit harder.
    - Sacrifice a consistency guarantee
    - In many newly developed systems, instead of providing ACID guarantee, we provide a bit more relaxed definition of consistency.
        - We don't need it over time and we may relax it over time.
        - All this expressive power of SQL can be stored as a table and this allows users to write queries using an expressive SQL query.

NoSQL: Has three interfaces

- (key, value)-store
- column-oriented
- document

These became very popular for a while and it is NOT good enough for other applications

- If we can support it, it will be good, but this isn't the minimum functionality we want.
    - Think about how we will access user data when we let the user log in.
- Amazon
    - User comes in and we want to authenticate the user.
    - User comes in and types in the user ID and password
        - Given the user ID and password, go to the database system and we need to access the user's records.
        - How do we actually retrieve the user's record?
            - Based on the user ID of the record.
            - Given the user ID, we want to retrieve the record related to the user and all other relevant information.
                - From that record, we want to compare the password field of the record and greet the user.
    - We can implement it using SQL, but if we don't allow general expressibility, what functionality should this database system provide?
        - Use the record based on userID.

        - The underlying system should return that user record given that user ID **very quickly**
        - This key value and the user can retrieve that value or record related to that particular key.
            - A lot of database accesses are in this mode and people started to say if we want the minimum possible interface to be employed on multiple machine, this is the core essence of any data access.

Key-value store

- Store key, value pair into the system
    - Very simple interface
        - Put key, value pair into the system
        - Get value from the key
- In order to access the data, retrieve everything and return it to the application
    - A lot of overhead because you have to retrieve ALL items together even if you only want a small subset of data
    - Another problem is that here, in order to update any part of your values, we have to completely replace the entire value because we don't know any internal structure of the value.
    - That creates quite a bit of overhead in terms of the network traversal.
        - Any updates will require a lot of change since we have to read it first and then make subsequent changes.
- Another problem is the following:
    - Sometimes, this field is about user password, names, addresses, etc.
        - When we look up some value, instead of providing the key, we can see from that user, we want to retrieve the password.
        - Instead of having this completely unknown, unstructured byte stream, we will have a structure known as **column-oriented** store
            - This is more similar to traditional MySQL databases from 143.

Column-oriented

- For that user, data is a set of rows associated with a # of column families that has a # of columns
- Give the user ID to retrieve a particular record.
    - If you search for Amazon's products, we won't know if we want products i.e. monitors that are 27 inches

Document-oriented

- Multiple fields associated with a document
    - The system allows us to search over each of these fields.
    - One of these can be a key over a unique ID
        - We can search over IDs or over a unique attribute.
- Quite popular i.e. MongoDB, CouchDB, and Amazon DynamoDB
- Once you get to this level, it becomes similar to relational databases, and we can imagine that there are consistency problems.
    - Less likely to be as scalable as key-value stores.
- MongoDB -> any need to deal with a distributed database system.

Hashing is inflexible

- We may need to move a lot of these data to other nodes and reshuffle the data
    - Quite inflexible in this dynamic environment and it depends on the situation

Consistent hashing

- Using any hash function, data will be assigned to a point in the circle
- For every node, (say you have 3 nodes): n1, n2, n3; each node is assigned to a point in the circle using some random hash function

Adding nodes

- Do we have to relocate things if we add something to the circle?
    - It will only be in this case where we add all the data and assign it somewhere else
    - Reassign it somewhere else, and each other node will remain in the same location
- Solves periodic reorganization problem
    - Small portions of the data will be reassigned to a different node
- All data should be attached uniformly around the circle
    - The # of nodes you assign is typically much smaller than the # of data points in the circle
    - For three nodes, even if it is a good hash function, it is unlikely that the nodes are uniformly distributed.
        - With a consistent hashing scheme, some nodes i.e. n1 might be responsible for more data points than n2
        - Data may NOT be assigned roughly uniformly and this could be a significant problem
            - Consistent hashing may NOT be split and this won't cause issues.
    - Differences in nodes can be quite significant in the downside of the hashing.

- Virtual node
    - Adds one more layer of indirection
    - 20 different virtual nodes into 20 different slots
        - Create a lot of different virtual nodes
    - Even though we may have some outliers, we know that due to the nature of large #'s, the average node will be averaged out to evenly store things in the circle.
        - This helps solve the imbalanced node problem for consistency.

Q. Why do we assign the nodes using a hash function? Why can't we do it manually?
A. You can do it that way as well but it is essentially the same as putting it in a hash function. If you don't use this one level of indirection, avoiding randomness can be very useful.

- In fact, it is very important to NOT just avoid the randomness problem. If we split it very equally, how will we assign the nodes? Evenly split the location and some of these nodes would be overflowing.
- To assign the 3rd node, either it can be there or on the other side.
    - **NOT** a well-balanced assignment
- If individual nodes are split into hundreds of virtual nodes,, the virtual nodes will do a good job of averaging out the data to approach the mean

Q. How do we retrieve the data according to the key value? Given a particular key value, how can we retrieve the data? How do we know the machine?
A. Hash the key value and place the key value into the ring. If it is there, it will be a part of the node and will be associated with the virtual node and we can map from that.

Q. Do we have to keep the ring?
A. We have to compute the node on the fly and we can do that in principle but we don't have to do this over time.

Using caching schemes

- Cache whenever something is slow
- Use a similar idea to support more traffic
    - When we say caching, what can we cache to improve scalability?
        - At almost every layer, we can implement a cache!
        - Between that database and the OS, we can apply more RAM to use it as a file buffer
    - Adding a cache between application and data
        - We can cache results for database queries
        - Get a set of rows from the databases

            - Rows typically correspond to an object and a particular row is likely to correspond to a particular user
            - Doesn't necessarily incur a disk I/O and it can be cached into the main memory
                - Deal with it for the cache without dealing with the underlying databases
- More traffic can be dealt with because of this caching algorithm
    - memcached
        - Provides a key, value interface
        - Store a particular database entry as a (key, value) pair
        - Main-memory based cache
        - Retrieve it based on the key, value
        - Distributed
            - Can be run on multiple machines
        - Contact and retrieve it so by having a # of machines with reasonably large memory
            - Application before it issues the query according to the database, it contacts memcached and tries to grab the value
            - If it is NOT there, we go back to the database there and retrieve it from that.
        - Very well-known cache system
- Between HTTP and App
    - The output from there should be a dynamic webpage.
        - Does it make sense to cache these dynamically generated webpages
        - What should happen is we get the request and send the database objects and create these HTML pages
        - Return it to the user later on.
    - What we can avoid is retrieving the underlying data and avoid it by caching it.
    - CPU computations of generating this actual content
        - Get quite significant benefits for this.
        - How common is it to reuse the same dynamic content again and again.
    - If we can somehow create this dynamically generated content that will be repeatedly used, we will see significant benefits
    - If it is only used once, we don't see much benefit, so don't need to do it like that.
    - Figure out if the contents are reusable like that

        - Even if the contents are reasonably static, we can still probably reuse it.

Wordpress

- Uses PHP and MySQL in the underlying database
    - If you look at what is returned and what data changes, most websites don't have changing data
- Pretty static content using Wordpress so that is why we use it.
    - For these kinds of websites, if we cache the dynamic pages, we don't have to go through the underlying data.
- Very well-known and just a plug-in to install and if you install the caching plug in, you get 2 orders of magnitude improvement in scalability.
    - Even if the content is NOT necessarily static, you can customize the info to a particular user type.
    - If only my name changes and Amazon changes the first page, it will only customize the first name for me.
        - Due to this nature, we can craft the overall template and then create an iFrame to change any part that needs to be modified to fit the needs of the user.
        - Separate out the frequently changed part to the reasonably static part.

Q. HTTP layer, can it cache? If you cache there, what do we gain?
A. Content distribution networks: In principle, whenever the user wants to retrieve a particular webpage, the user request has to come through a particular server and the content has to come from the server.

- What if we cache all these webpages closer to the user?

TCPI doesn't know the actual content.

- Q. How do we ensure retrieval is actually done?
- A. When the user wants to try to retrieve a page, they will type in a URL. Use the browser to get info from these individual caches and how can this be done?
    - When the user types in a particular URL, the DNS server can resolve that address name so that the closest location to the cache is returned.
    - Most of the webpage contents for big companies like Amazon is dynamically generated and changes all the time.
- We can get significant improvement from multimedia i.e. images, videos, sounds, etc.
    - Any of these cacheable content can use a particular URL or host name and this is managed by the content distribution network.

    - Maintain a DNS server to map it to the closest IP location
- In terms of the web developer's perspective, we may have to change the URL to refer to the content distributed network.

How big companies use all these clusters to serve their users

- The tech giant has 12 data centers around the world and these contain sensitive information about you.
- Top secret facility in North Carolina
- Google data center says no user can enter without a non-disclosure agreement
    - Inside, security prevents people from entering.
    - If more than one person enters through at one, alarm will sound.
- 55,200 servers on the floor
- Employees must wear earplugs to protect their ears
    - Repairing a server requires you to reconnect certain things
- The company is very protective of its green image
    - Google reduced a lot of its energy consumption
    - A lot of power still comes from non-renewable energy.
- Use tens of thousands of machines to serve the users

What kind of things do we need to worry about for hundreds of thousands of machines?

- HDD (hard drives)
    - Usually last about 3 years
    - Work station -> 3 years sounds reasonable, and every year, we will have to replace hard drives.
- Power supply
- Network
    - If someone hits the cable or disconnects it, it can disable the components of the machine

These are designed as such because these are the 3 major components that can break, so we need to be able to replace them easily and effectively.

- 99.9%
- 9 hours per year of downtime

If you have 10,000 machines in one data center, it starts ago add up a lot!

- 0.1 * 10,000 = 10
- When we build the overall software infrastructure, we have to make sure the overall structure is durable enough to endure that it can handle a few node failures.

Don't have time to go over Hadoop

- Something to be aware of when dealing with these overall systems.

Say you want to start a company called CS144:

- How can I get this website up and running?
  - Buy the domain for CS144.com
  - There are a # of domain registration services
    - GoDaddy - $10 per year for a domain
  - We need a place to host our website!
    - Either buy a web hosting service or buy a physical machine and connect it to the network.
    - PC: $800 - $1000 and more than enough for hosting all the students at the same time
      - $100 for the internet connection from the Internet Service Provider (ISP)
        - They do NOT necessarily do it this way because maintaining the server is a hassle
        - Adding more PC's is a pain in the ass!
          - Many startup companies are buying hardware equipment from many cloud computing facilities!

Major cloud services you can buy:

- Amazon Web Service: biggest player
- Google Cloud Platform:
- Windows Azure: Microsoft's version of cloud computing

Sell virtual machines and lets you upload your machine images

Infrastructure as a Service (IaaS)

- Overall infrastructure
- Physical machines that are virtualized as a virtual machine
- Your job to install the OS, applications, etc.
- Make sure all the software stacks are kept up to date.

Platform as a Service (PaaS)

- Why do we need (as an entrepreneur or application developer) have to worry about underlying OS layers and software stack layer
  - Thus, I can just implement my own application on it and make sure my application is safe.

    - Provided by service provider and implement the actual application at the top layer

Software as a Service (SaaS)

- Microsoft Office - web version
- Google Business Web
    - Google Apps for Work
    - Gmail service, Calendar service, Google Drive, Google Docs, etc.
        - This is Google's version!
    - Whenever a company wants their own collaboration framework, then the company can buy that package (Gmail + Google Drive + Calendar, etc.) as a service
    - Each user pays $50 per year and all the URL's and all the emails will be customized for that particular company.

EC2

S3: Distributed file system

- Storage space from Amazon S3 and you need storage available on any web applications
- Buy EC2 and create the virtual machine in Amazon S3

Glacier

- Try to make sure the data stored in S3 is quickly accessible
- We do have to remember the data based on the customer record just in case something happens in the future.
    - Archive a kind of level of the storage.
    - Whenever we need it, it is free to access.

Amazon has DynamoDB and RDS

- Amazon's versions of NoSQL and relational database systems, respectively.

ElastiCache

- In-memory object caching

CloudFront

- You can buy almost anything you need for cloud version systems that we are talking about.