# Lecture 1/25/18

## Three Way Handshake Review

## Web Browsing Review

Request (HTTP) 🡪 Transport (TCP) 🡪 Addressed (IP) 🡪 DNS Server 🡪 Link Layer

## VPN

Virtual point to point connection

Uses the internet to create a private tunnel through a network

Connects to a server that then provides the internet to you

## HTTP

### Request

Application level protocol used every application

443 port is the secure port?

#### Structure:

Request

Headers

What you accept

Specify keep alive to keep it

### Response

Status line (has various status codes 200 [okay], 404 [means that whatever wasn’t available]

Header info from the server

Cookie info

Specify whether to use the cache =

## URIs

Not the same as a URL (uniform resource locator, where a thing is) or URN (uniform resource name – somehow a bit different???what a thing )

URLs and URNs are examples of URIs.

## Request Methods

### Get

Requests a resource from the server

No content is sent in the request

Whenever you type in a url or click on a link

### POST

Used to submit forms’ – can also use get to submit forms though.

## CODEC

Look up what this is… ☹

# Lecture 1/29/18

## Business Case Stuff

Check LMS or guidelines

## Networking Applications

### Technologies Used

SSH, TCP, HTTP(s), etc.

### Web Browsing Review

“Encapsulated in Ethernet protocol”

“Addressed, sequenced, put back together” on the transport level on the other end

### Value

Metcalfe’s Law: the value of a network is proportion to the square of the number of users on that network. The cost of the network in linearly proportional.

### Telecommunications

The transmissions of messages over long distances

Cell towers, fiberoptics, and wifi are all microwaves??? Thought it was radio.

### Formal Definition of Network

Collection of transmitting devices

### Channel

A physical medium that carries a signal

OR: In a fiber optic cable – there are 20 different channels -- each can function independently or carry more info.

### Routing

To make use of limited bandwidth, info is broken up, addressed, and sent out via different paths to their destination – time division multiplexing

### Crosstalk

In the ye olden times – you could sometimes hear other people on the line – not encryption and limited lines.

# Lecture 2/1/18

## Hypertext Markup Language

### Components

#### Hypertext

Text that links to more text

#### Markup Language

Language that we use in order to achieve formatting – a way to embed formatting information within a text file.

A way to describe data or information contained in a text file.

#### Tag

A way of identifying / applying metadata / format within a document

#### Metadata

Data about data -- tags used to describe the information within a file

### New Def of Hypertext Markup Language

A simple way or marking up documents so that they can be shared and understood over networks.

We use it because HTML is simple to understand, lightweight, an easy way to send information both for users to see, and computers to interpret at the same time.

### Why Learn It?

It’s the bread and butter of the web

### History

XHTML 1.0 – 2000 --

HTML 4.01 – 1999 – first just gave us style…it seems.

HTML 5 – October 2014 – (derivative of SGML – granddaddy of all markup languages)

It’s constantly changing.

### Tags

#### Doctype

DTD = datatype declaration – contains all HTML elements and attributes, but does not include presentational or depreciated elements (like font). Framesets are not allowed.

In HTML, you don’t need large doctypes – just write doctype.

#### HTML

The root of the tree

#### Head

Contains the title

Links to stylesheets, scripts

Meta data about the page

#### Body

Where you find the actual content

### Tag Display States

#### Block

Makes the everything one after another – breaks before and after.

#### Inline

Makes everything right to left

#### None

Makes the element is invisible.

Kind of obsolete

## XHTML Documents

### Definition

eXtensible HyperText Markup Language

Consists of: Doctype, head, title, body blah blah

New stuff is attributes: apply more information to a tag.

### Syntax

Elements must be closed

Attributes must be quoted

Elements and attribute names must be in lower case

Nesting must be correct

DOCTYPE, head, title, and body are mandatory

Extra white space is ignored

### Code Validators

Send in your file and it’ll look for formatting errors and other stuff.

A code linter is a style checker (annoying formatting conventional stuff)

## CSS

### Overview

Cascading Style Sheets

A CSS document styles the HTML (content)

Consists of selectors on HTML elements

### Declaring Styles

Where allowed to put them

1. External Style Sheet – dis is the best – use <href=”style.css” rel=stylesheet” type=”text/css”/>
2. Style attribute in head
3. Inline Styles

### Selectors

#### IDS

Specific – refers to a specific element

#### Classes

Grouping elements – not a functional reason to prefer one over the other – just for best practice / semantic.

### \*\*\*Precedence

HTML is read from the top down. (remember that CSS links are put at the top)

There’s also various precedence things

IDs trump classes

More specific classes / IDs trump less specific ones

IF all else fails – more recent (bottom) to top (less recent)

Probably worth looking this up

### Sizes

Following units are acceptable

1. Em – ems are the size of an M is that font and scale relative to your font.
2. Px – specific pixels for when you need precision
3. % -- percent relative to the block that precedes it / parent container.

### Colors

Represented in RGB

If both digits have same value, can abbreviate.

### Box Model

Outer to inner =

Margin 🡪 Border 🡪 Padding 🡪 Actual width

### Floating Elements

Float – ya know

Clear – forces stuff on that side to not appear there.

### Making your code pretty

Comment ya code

Indents should be 2-3 spaces NOT tabs ☹

P {

}

NOT

P

{

}

## Interpreted vs Compiled Languages

Compile – does all the compilation at once – catches all errors at that stage

Interpreted – does all the compilation line by line.

## Aside on Encoding

### BLOBs

Binary large object

### CLOBS

Character large object

# Lecture 2/5/18

## Term Project

Build an application (doesn’t necessarily have to be finished. Description of what the application is going to do, who is it going to serve – basically a formal proposal).

Graded across 5 areas.

1. HTML CSS layout
2. Javasript
3. Pull static data from a data file
4. Pull real data from a database
5. Something else.

Will do some prototyping – get a final design – then build a high fidelity version – beta test.

# Lecture 2/12/18

## Bookkeeping Stuff

Tuesday is a Monday – guest lecture. On flow,

Tony Hawk game was made at RPI – sold to activision.

February 22 – Case 1 in class. We have 1 week to do it.

March 1st – first quiz \. Anything covered in class – valid for quiz – slack discussions – guest lectures. Labs. 60% technical. 40% non technical (guest lectures and stuff) NO multiple choice. Open book and open internet though, so that’s good . 5-6 Questions not equally weighted though.

## Building a Web Science

### Structure

Static websites consist of a file an folder hierarchy just like those you find o your computers.

Each html page links to or includes other pages and resources.

Index is the homepage. If you want another page you have to explicitly ask for it. You can navigate to other pages in a website the same way you parse folders.

Common folder is the resource folder – where you put files that are common to the rest of the site.

Topmost folder of a website called the “document root.”

### Links

The value of a hypertext reference (or a src attribute) is a uniform resource identifier

URL = uniform resource locator.

### URLs

#### Inpage Anchor:

Links to a heading in the page. Use the pound sign. Href=”#ID”

#### Absolute:

Include the scheme and host in the URL. Href=<http://www.rpi.edu>

ALSO much better for big stuff – because you don’t want to pay for bandwidth.

#### Relative:

A reference relative to the calling document.

../ one directory up. ../../ goes up two. ./ current path.

#### Which is better?

The answer is both. Relative for your own stuff. Absolute for your external resources.

HAHA These PLEBES.

### Navigation and Menus

Breadcrumbs = trail of navigation. Can view the path you have traversed, can go back. Like windows explorers.

Persistent links in headers and footers = like a logo being a home page.

## Information Architecture

Includes:

Document site maps, navigating models, content models

Understanding target audiences needs, tasks, and goals and translating them into easily discoverable elements.

In the context of web architecture – person who looks a different inputs, understands various technologies, and plans everything out.

### Pointers

Treat power with respect, but don’t overvalue power’s opinion. Don’t trust to boss all the time if they’re an idiot.

Avoid kitchen sinking – Trying everything at once – keep it simple then add complexity as time goes on.

## Lab 3

## Misc – important randos

Img tags are self closing – but I guess validators will yell at you if you don’t. Have to have an alt text.

Href = hypertext reference.

IRI = international resource identifier – Allows other non western characters to be used for URLs.

IBM owns a certain shade of blue??? Lol. Trademarked.

# Lecture 2/26/18

Oops

# Lecture 3/1/18

## XML Review

The language behind XML is Stylized General Markup Language

XHMTL is XML

What is a DTD?

HTML is not XML

XML is an intermediate language for transmitting data

HTML5 is a language for displaying data

I guess you can make your own tags in XML?

XHTML is a form of XML?

## Camel Case

Format for writing variable names.

Lowercase first word and uppercase fire letter of other words

## Namespace

A way to segment information in code so things can have the same name

Nut as in the food and nut as in hardware.

## Microformats

RSS feeds and all

A type of XML that is specific to news documents and stuff

# Lecture 3/5/18

## Javascript

### What is JavaScript?

Rich programming language

ECMAScript - widely adopted standard

Standard maintained by ECMA International (at one time European Computer Manufacturers Association)

Primarily ECMA-262,

3rd ed. – 1999

5th ed. – 2009

6th ed. == ECMAScript 2015 – 2015

7th ed. == ECMAScript 2016 – 2016

8th ed. == ECMAScript 2017 – 2017

9th ed. == ECMAScript 2018 – Draft - Oct 13, 2017 – Adopted Jan 23-25 2018

10th ed == ECMAScript 2019 – Draft – Feb 21, 2018!

Commonly used to interact with browsers and build rich web user interfaces / applications

Think Google, Facebook, Yahoo, you name it...

Interpreted (typically, though it can be compiled)

* Object-oriented capabilities Rich programming language
  + ECMAScript - widely adopted standard
    - Standard maintained by ECMA International (at one time European Computer Manufacturers Association)
    - Primarily ECMA-262,
      * 3rd ed. – 1999
      * 5th ed. – 2009
      * 6th ed. == ECMAScript 2015 – 2015
      * 7th ed. == ECMAScript 2016 – 2016
      * 8th ed. == ECMAScript 2017 – 2017
      * 9th ed. == ECMAScript 2018 – Draft - Oct 13, 2017 – Adopted Jan 23-25 2018
      * 10th ed == [ECMAScript 2019](https://tc39.github.io/ecma262/) – Draft – Feb 21, 2018!
  + Commonly used to interact with browsers and build rich web user interfaces / applications
    - Think Google, Facebook, Yahoo, you name it...
  + Interpreted (typically, though it can be compiled)
  + Object-oriented capabilities

### Using Javascript

* Javascript can be embedded in the <head> or <body> section of an HTML document:  
    
  **<script type="text/javascript">  
   document.write(firstName);  
  </script>**
* You can link to an external file in <head>:  
    
  <script src="my.js" type="text/javascript"> </script>

### Some Basics

#### Data Types

* Primitives: numbers, strings, and Booleans
* Objects: Arrays, dates, and regular expressions

#### Examples

var currentCount = 3; // an integer  
var firstName = "Mary"; // a string  
var isAvailable = true; // a boolean  
var now = new Date(); // a date object  
  
/\* The above is how we declare variables. See that  
 multi-line comments can be written like this. \*/

### More Basics

Javascrit is case sensitive

Statements terminate at the ends of lines and at semi-colons. Use semicolons though

Var myPhrase = “This is long

So I’m breaking” – this won’t work, but if you addd an escape character the / -- then it will work.

### Strings

Datatype for representing text

Can be concatenated using the addition operator

### Events

* A browser *fires* events when a user interacts with it
  + When clicking an item (an *onclick* event)
  + When the cursor enters an input field (an *onfocus* event)
  + When hovering over an item (an *onmouseover* event)
  + When a document is fully loaded (an *onload* event)
  + When a form field is changed (an *onchange* event)
  + And so on ...
* Try this:  
  <button type="button" onclick="alert(firstName);">  
   Who's there?  
  </button>

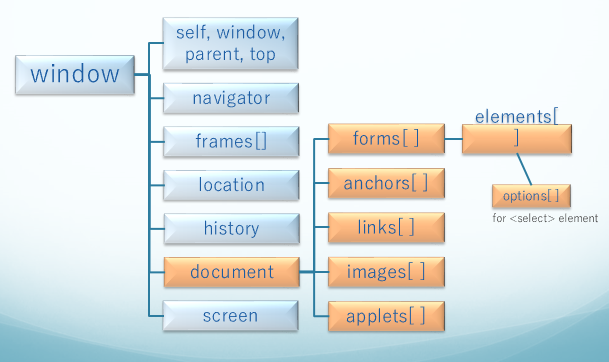
### Objects

* A collection of named values (called properties)
* When a function is a property of an object, we call it a *method*
* We access an object’s properties and methods using "dot notation":  
    
  **document.write("write() is a method of the document object");  
    
  var fnLength = firstName.length;**

### Specific Objects

* The Window object
  + The browser object that serves as the access point for all other objects
  + This object represent the web browser window itself (or a frame within that window)
  + Window.location (an important feature, I guess??)
* The Document object
  + A property of the window object that represents the HTML document within it

### Browser Object Hierarchy



## DOM

### What is DOM?

* The Document Object Model
  + It is the document subtree of the Window object
  + Level 0 of the DOM are those objects in the previous slide colored in orange
    - These are supported in all major browsers – and have become a de-facto standard
  + The W3C continues to work on DOM standardization:
    - "The Document Object Model is a platform - and language -neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents."
  + Ok – let’s look at the W3Schools DOM reference:  
    <http://www.w3schools.com/jsref/>

### Interacting with the browser

* You can call on the Window object’s property and methods using dot notation. The full path to some of the methods we’ve been using are actually:  
    
  **window.document.write("full path");  
  window.alert("I'm a method of window.");**

(But we often use shorthand when referencing the window object’s properties and methods.)

* Assign the following JavaScript to a button: **window.location = "http://www.rpi.edu";**

### Interacting with the DOM

* One of the most common activities in client-side scripting is manipulating pieces of the DOM
  + Making things appear and disappear (using css)
  + Inserting and removing elements (e.g. a news feed)
  + Manipulating form fields and values
  + Animating images
  + Changing element values

## Forms

### Overview

* Forms are used for capturing user input
* They are often used with the HTTP POST method
* Forms have a limited number of elements:
  + Text input fields
  + Text areas
  + Select (pull-down) boxes
  + Checkboxes
  + Radio buttons
  + Hidden fields
  + Submit buttons..
* Note that you can reference this form's elements in one of two ways\*:
  + by path: document.addForm.firstName;  
    document.forms[0].elements[0] will also work…
  + by ID: document.getElementById(firstName);
  + To get at the field’s value, you need to ask for that property: document.addForm.firstName.value;

### Example

**<form action="someserverpage.php" method="post">**

**<label for="fname">First name:</label>   
 <input type="text" name="fname" id="fname" />**

**<label for="lname">Last name:</label>**

**<input type="text" name="lname" id="lname"/>**

**<input type="submit" value="Submit" />**

**</form>**

### Passing Current Object

* The JavaScript keyword "**this**" refers to the current object
  + It is used as a shorthand to access the object that is currently being acted on, e.g. clicked  
      
    Add this onchange event to one of the input fields:  
    <input type="text" onchange="alert(this.value)"/>  
      
    What happens?

# Lecture 3/19/18

## JQuery

### What is it?

* A JavaScript library
* Lightweight (about 31KB for the minified version) – maybe a little bit more
* Simplifies HTML document traversing (DOM), event handling, animations, and more
  + "write less, do more"

### Minification

* Removal of all unnecessary characters in code
  + e.g. whitespace new line chars, and comments
* Reduces amount of data needed to be transferred
  + Smaller file size = quicker page loads, but less readability
* A lot of tools that compress the source code for you
  + jscompress.com is just one

### How to add jQuery

* Latest stable version is 3.3.1
* Download it and store locally
  + <script type="text/javascript" src="jquery.js"></script>
* Use the hosted jQuery library
  + <script type="text/javascript"[src="http://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">](http://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jque) </script>
  + <script type="text/javascript"src="http://ajax.microsoft.com/ajax/jquery/jquery-3.3.1.min.js"> </script>

### Syntax

* $(selector).action();
  + $ (typically) used to define jQuery
  + Selector - HTML element to "query" or find
  + Action - What to do jQuery action to perform

### Defining jQuery

* $ is shorthand for the standard function (full name is jQuery)
  + $(document).ready = jQuery(document).ready
  + Syntactically the same
* Problem: '$' is used as shorthand for other JavaScript library objects
* There's a way to get around this (but don’t worry too much about this now):
  + jQuery noConflict() method
  + Ex.

var **jq** = jQuery().noConflict();

**jq**(document).ready( function () {

**jq**("p").hide();

});

When there is a conflict – the second library wins because it overwrites everything else

### Event Handlers

* jQuery methods called when an event is "triggered" or "fired"
* It's common to put most jQuery functions within

$(document).ready(function)

* This waits until the entire page is loaded

$(document).ready(function() { -- function here is just an anonymous function

$("p.change").click(function() {

//do something here

});

});

Sometimes if javascript is supposed to be grabbing tags before the document loads, you can get errors

Use the onload function to resolve this problem.

### Element Selectors

* jQuery uses CSS to select HTML elements
* **$("h1")** - selects all <h1> elements
* **$("p.fname")** - selects all <p> with the class = "fname"
* **$("h2#lname")** - selects the <h2> with the id = "lname"
* **$("#contact")** - selects all elements with id = "contact"
  + (There should only be one!)

### Attribute Selectors

* jQuery uses XPath to select elements with the given attributes
  + **$("[href]")** - selects all elements with the href attribute
  + **$("[name = 'fname']")** - selects all elements where the name attribute is equal to "fname"
  + **$("[href !='#']")** - selects all elements with href attribute does NOT equal "#" – use this to select links that don’t reference current page?? -- # means current path I guess

### CSS Selectors

* Changes the CSS properties of the HTML elements
* $(**selector**).css("**css-property**", "**value**");
  + Can pass just the property to get the current value (the first matched element is returned)
  + Can pass multiple properties
* **$("h1").css("color", "green")** - changes the color of all h1 elements to green
* **$("h1").css({"background-color":"yellow","font- size":"200%"})** - changes all h1 elements to have a background color yellow and font size to 200%

### HTML Manipulation

* jQuery gives you some methods to manipulate the DOM
* .addClass() - lets you add one or more classes to the element
* .html() - sets or returns the content (innerHTML) of a element
* .before() - adds content before the given element
* .val() - sets or gets the value of a (form) element
* Ok cool, but what about this:
* You dynamically add a new element (via jQuery or some other method) and want to **bind** an event to it.
* You can use the **.on()** method

**$(parent-element-to-monitor)**.on("**event(s)"**, **"element- to-attach-event", eventHandler())**;

Ex.

$(document).on( "click", "p", function(event) {

alert("Cool text here!");

});

### jQuery Effects

* The "old" way to do hide, show, slide, toggle, fade, and animate
  + PSST: (A lot of this can be done with CSS3 now).
* **$("p#hideme").hide()** - hides the p element with the id=hideme
* **$("h1").fadeIn()** - does a fade in animation to all h1 elements
* The callback parameter
* Waits to execute a function until after the parent function is executed
* Useful since JavaScript is a interpreted language
  + Since JS executes line by line

$("p").hide(1000);

alert("The paragraph is now hidden");

^^ In this example hide happens as soon as page loads then alert happens, then is shown after 1000 seconds

$("p").hide(1000,function(){

alert("The paragraph is now hidden");

});

^^ In this example , alert doesn’t happen until after 1000 milli seconds

### Block vs Non blocking IO

I think this has something to do with synchronous stuff – but clarify this.

Non blocking io – fetching multiple files at the same time – I guess?

## JSON

### Intro

* *JavaScript Object Notation*
  + [www.json.org](http://www.json.org/)
* *Subset of object-literal notation*
  + All JSON is valid object-literal notation...
  + ...but not all object-literal notation is valid JSON
* *JSON is a data interchange format, functionally similar to XML*
* A way to represent JavaScript data
  + for data interchange
  + lightweight

human readable (i.e. text-based)

### Versus XML

#### XML

<?xml version="1.0" encoding="UTF-8“ ?>  
<book id="0375411275">  
 <author>Ben Rice</author>  
 <title>Pobby and Dingan</title>  
 <copyright>2000</copyright>  
 <publisher>Knopf</publisher>  
 <genre>General Fiction</genre>  
 <isbn>0375411275</isbn>  
</book>

#### JSON

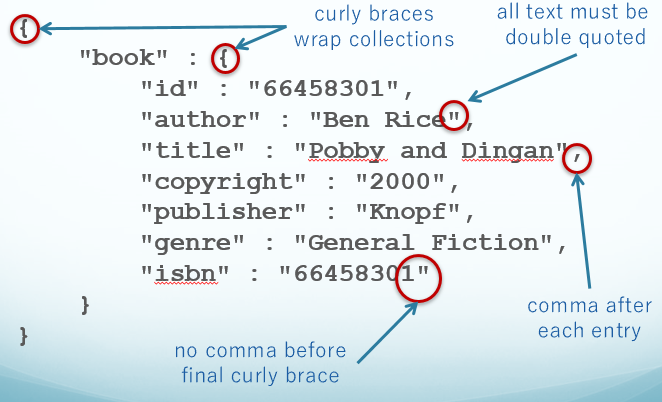
{  
 "book" : {  
 "id" : "66458301",  
 "author" : "Ben Rice",  
 "title" : "Pobby and Dingan",  
 "copyright" : "2000",  
 "publisher" : "Knopf",  
 "genre" : "General Fiction",  
 "isbn" : "66458301"   
 }  
}

### Even Simpler

Can be just a collection of "name" : "value" pairs

{  
 "givenNames” : "Jonathan",  
 "lastName” : "Swift",  
 "yearOfBirth” : "1667",  
 "yearOfDeath” : "1745"  
}

### More Detailed



### Data Structures

* Consists of two data structures:
  + Collections
    - A collection of "name" : "value" pairs (an object)
  + Arrays
    - a list of values or objects (including arrays)

### As a variable

* You can assign JSON to a variable in JavaScript **var name = {"first":"Peter","last":"Jackson"};**
* and access its values using dot notation  
  **alert("Name is " + name.first + " " + name.last);**

### Arrays

* Collections of indexed values or objects
  + A list of primitive values (e.g. [23,12,14,7,39])
  + A list of objects (e.g. an array of arrays)
* Example:
  + **var myArray = new Array("x","y","z");  
    alert(myArray[0] + " " + myArray[2]);  
    myArray[1] = "some new string value";**
* We can access arrays using square bracket notation:
  + ***array[index]***
  + index begins at zero
  + **if(myArray[0] == "x"){   
     alert("This will execute.");  
    };**

# Lecture 3/2/18

Oops

# Lecture 4/2/18