CS1320
Creating Modern Web and Mobile Applications

Lecture 6:

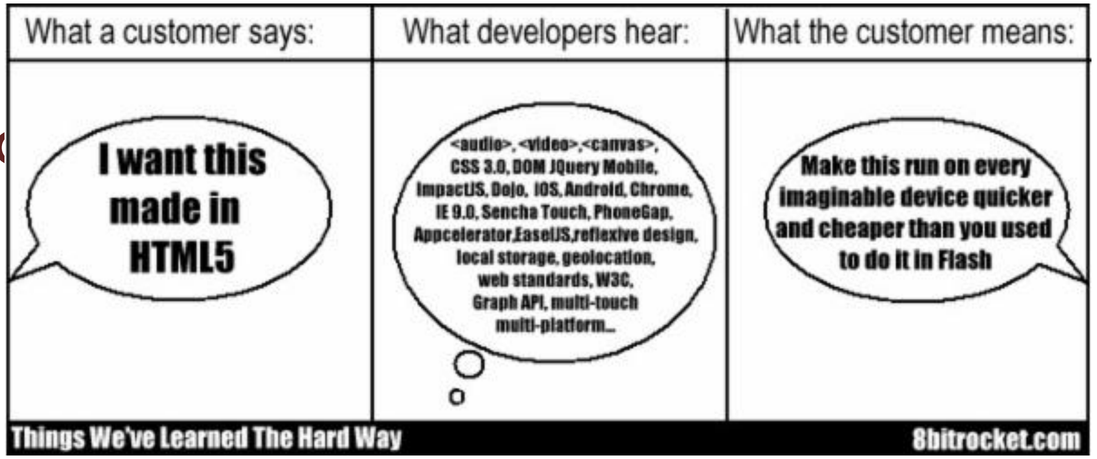
Dynamic Web Pages II

Final Projects

- Team and project assignments posting
    - If you have issues, mail the head TAs
- This week
    - Team should meet as a group
    - Decide responsibilities
    - Discuss project ideas and understanding
        - Make sure you are all on the same page
    - Contact sponsor – they are waiting to hear from you
        - Introduce yourselves
        - Set up a meeting for this weekend or early next week

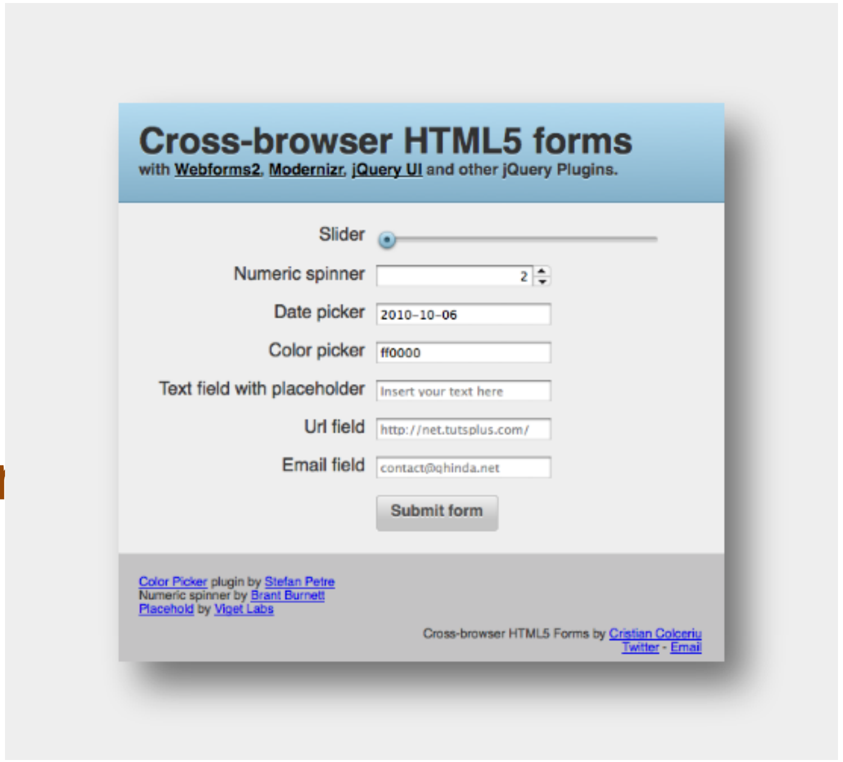
HTML = HTML5

- HTML5 is designed to support modern web apps
  - More interaction
  - More devices
- Multimedia and animations are more c
  - A large fraction of web sites are using them
  - They shouldn't require plugins to be usable
  - These should be standard in all browsers
- Other features have similar properties
  - Simple databases, cookie management, …
- Basic HTML doesn't provide enough context information
  - About the page (for search, readers, …)
  - About forms (numbers, dates, …)


What a customer says:
What developers hear:
What the customer means:
I want this made in HTML5
<audio>, <video>,<canvas>, CSS 3.0, DOM JQuery Mobile, ImpactJS, Dojo, IOS, Android, Chrome, IE 9.0, Sencha Touch, PhoneGap, Appcelerator,EaselJS,reflexive design, local storage, geolocation, web standards, W3C, Graph API, multi-touch multi-platform...
Make this run on every imaginable device quicker and cheaper than you used to do it in Flash
Things We've Learned The Hard Way
8bitrocket.com

HTML5 Forms

- Do forms work on your smart phone/tablet?
- Forms are the basis for much HTML interactio
  - But they are quite limiting
  - And not well-oriented to tablets / smart phones
  - And require JavaScript to validate
- HTML5 significantly expands the input types in forms
  - Text, password, submit, radio, checkbox, button
  - Color, date, datetime, email, month, number, range, search, tel, time, url, week
  - With built-in validation
  - Generic regular-expression based validation



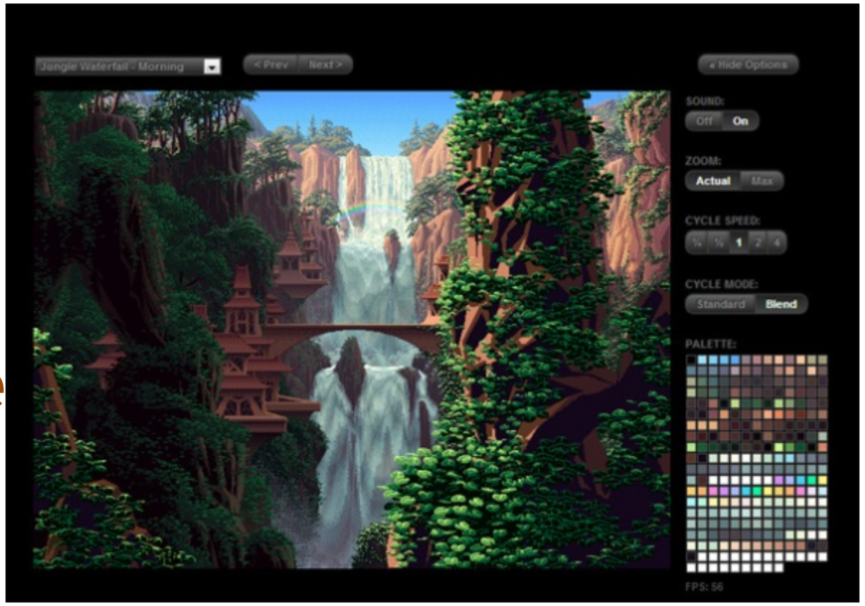
Cross-browser HTML5 forms
with Webforms2, Modernizr, jQuery UI and other jQuery Plugins.

Slider	
Numeric spinner	2
Date picker	2010-10-06
Color picker	ff0000
Text field with placeholder	Insert your text here
Url field	http://net.tutsplus.com/
Email field	contact@qhinda.net

Color Picker plugin by Stefan Petre
Numeric spinner by Brant Burnett
Placehold by Viget Labs

Cross-browser HTML5 Forms by Cristian Colceriu
Twitter - Email

HTML5 Canvas



- A canvas is a drawing area on the page
  - Use JavaScript to draw on that canvas
  - Drawing is similar to Java2D drawing
    - Similar primitives, transformations, coordinates, etc.
    - Rectangles, paths, arcs, text
    - Java Graphics2D maps to HTML5 Context
  - Can be used for static graphics and animations
- http://www.youtube.com/watch?v=xnAiEJEBLJg
- http://www.youtube.com/watch?v=oZInfZ0wecw

SVG Graphics

- Different approaches to graphics
  - Procedural calls to draw everything
  - Structure representing what should be drawn
- SVG takes the second approach
  - The structure is part of the DOM
    - Can be manipulated by JavaScript
  - Objects correspond to various primitives
  - Often easier than functional drawing
    - Refresh handled automatically
- http://www.youtube.com/watch?v=6SDKN-Amlyo

HTML5 Multimedia

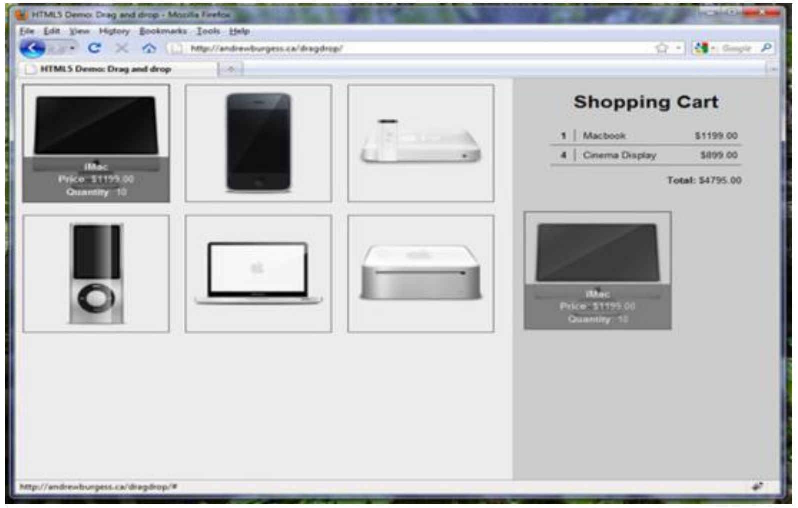
- <audio> and <video> tags
  - Controls
  - Multiple formats can (and have to) be provided
- Examples

```
<video width="320" height="240" controls="controls">
        <source src="movie.mp4" type="video/mp4" />
        <source src="movie.ogg" type="video/ogg" />
        Your browser does not support the video tag.
</video>
<audio controls="controls”>
       <source src="song.ogg" type="audio/ogg" />
       <source src="song.mp3" type="audio/mpeg" />
       Your browser does not support the audio element.
</audio>
```

HTML5 Drag and Drop

- Direct manipulation interfaces are sometimes based on drag and drop
  - That's what users have come to expect
- HTML5 lets any element be dragged
  - And any element can be a drop target
- HTML5 also provides JavaScript events to support
  - On drag start (set the content of the drag)
  - On drag over (allow/disallow drop)
  - On drop (use the contents)
- Much simpler to use than Java drag and drop

Drag and Drop Example


HTML5 Demo: Drag and drop - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://andrewburgess.ca/dragdrop/
HTML5 Demo: Drag and drop
iMac
Price: $1199.00
Quantity: 10
http://andrewburgess.ca/dragdrop/#

Shopping Cart

1	Macbook	$1199.00
4	Cinema Display	$899.00

Total: $4795.00

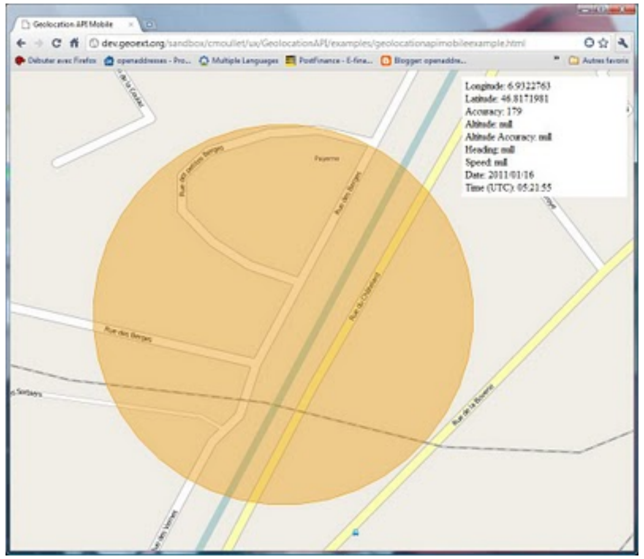
```
<!DOCTYPE HTML>
<html> <head> <script type="text/javascript">
function allowDrop(ev)      { ev.preventDefault(); }
function drag(ev) { ev.dataTransfer.setData("Text",ev.target.id); }
function drop(ev)
{
        var data=ev.dataTransfer.getData("Text");
        ev.target.appendChild(document.getElementById(data));
        ev.preventDefault();
}
</script> </head> <body>
<div id="div1" ondrop="drop(event)“ ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true“ ondragstart="drag(event)" width="336"
height="69" />
</body> </html>
```

HTML5 Web Storage

- Cookies are not efficient or secure
  - Have to be sent with each HTTP request
- HTML5 offers several new facilities
  - Local storage (name-value) of arbitrary data
    - Permanent, fixed time, or session-based
  - Generation of public-private keys
    - Offers secure communication
    - Rarely used – use HTTPS instead

HTML5 Geolocation

- HTML5 enables using the current location
  - Accurate from a device with GPS
  - Approximate from other computers
- Can use this with JavaScript
  - Locally (place on a map)
  - Globally (send to server)
- Can also get automatic updates
  - JavaScript code that is invoked as the position changes
  - There's an event for that


Longitude: 6.9322763
Latitude: 46.8171981
Accuracy: 179
Altitude: null
Altitude Accuracy: null
Heading: null
Speed: null
Date: 2011/01/16
Time (UTC): 05:21:55

Geolocation Example

```
<script>
var x=document.getElementByName("demo");
function getLocation()
{
        if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(showPosition);
        }
        else { x.html("Geolocation is not supported.");}
}
function showPosition(position)
{
        x.html("Latitude: " + position.coords.latitude +
                "<br />Longitude: " + position.coords.longitude);
}
</script>
```

728 morons within a radius of 400 feet
TWEET TWEET
phew...
27 bitches within a radius of 400 paws
Wow!
Fix

HTML5 Messaging

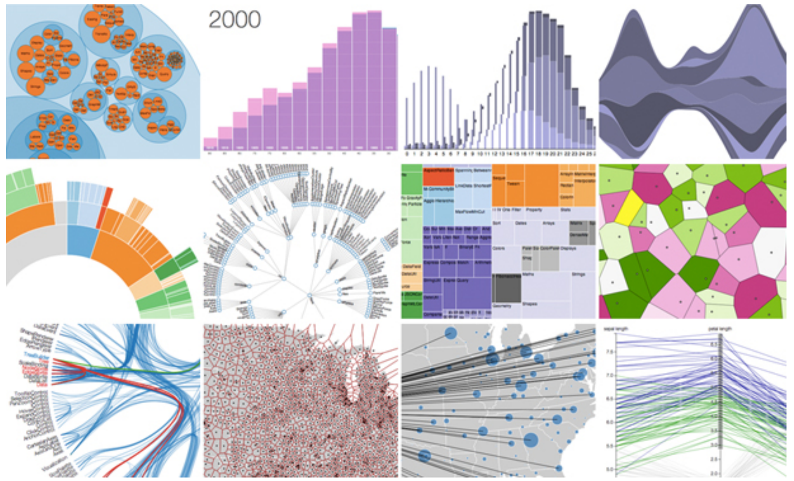
- Mashups
  - Web pages composed of information from multiple sources
  - Browsers limit where requests can be sent based on URLs
    - Make mash-ups difficult to implement
  - Messaging allows this to be bypassed in a selective manner
  - Usually embedded in libraries – not something you do directly
    - Maps: Google maps, leaflet.js
    - Payments: Stripe, Paypal
    - Other: address decoding, weather, …
- Web Sockets
  - Continuous communication with your server
  - Easy to set up and use (callback functions on both ends)

Animation on Web Pages

- Is animation a good idea in a web application?
- Something moving (changing) on the screen
- Properties
  - Can be one-time or continuous
  - Can be smooth or jerky
    - All animation is jerky, why does it appear smooth
    - Persistence of vision, frames per second
- Types of animation
  - Movies
  - Sound
  - Bitmap animation (canvas)
  - Vector animation (svg, flash)

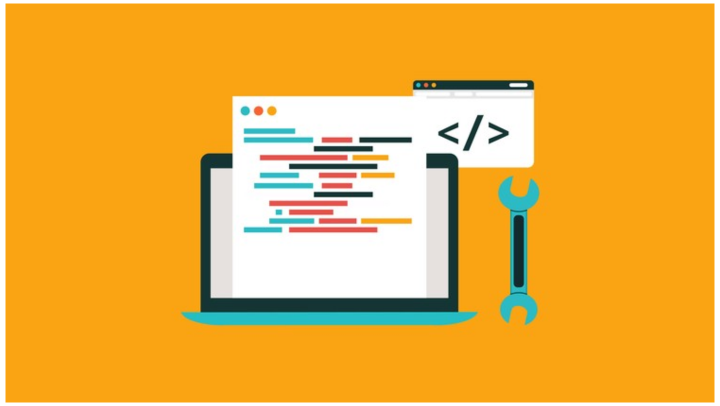
Data Visualization

- Canvas/SVG
- D3
  - http://www.youtube.com/watch?v=0oOC2FYNo1M



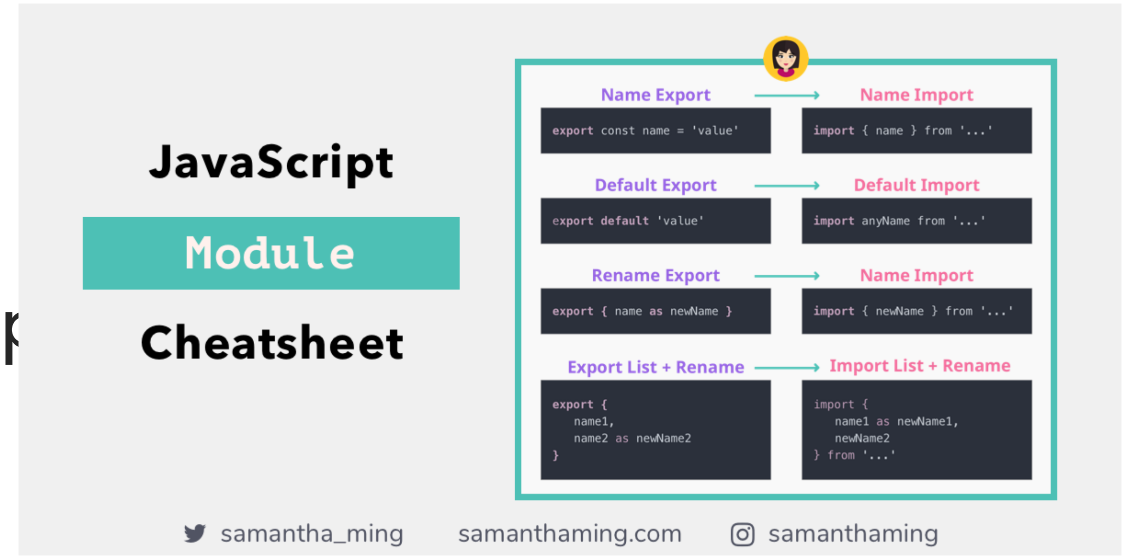
Other JavaScript Features

- Modules
  - Ability to write code in separate files without name conflicts
  - export names from a file to be used elsewhere (selective set of names)
  - import names from a module (and give them a local name)
  - This makes is possible to write more complex programs
- Multiple assignments
  - Multiple variables, array elements, object fields
- Promises



Modules

- Separate files with separate name sp
- File can export specific elements
  - export function x { … }
  - function x { …} ; export x;
- Other files can import a module of individual components
  - import 'module'
  - import { name, name, … } from 'module'
- Use script type module
  - <script type='module' src='name.mjs'></script>
  - You can also package all the modules and the main file into one file (for production)


JavaScript

Module

Cheatsheet

Name Export → Name Import

```
export const name = 'value'
```

```
import { name } from '...'
```

Default Export → Default Import

```
export default 'value'
```

```
import anyName from '...'
```

Rename Export → Name Import

```
export { name as newName }
```

```
import { newName } from '...'
```

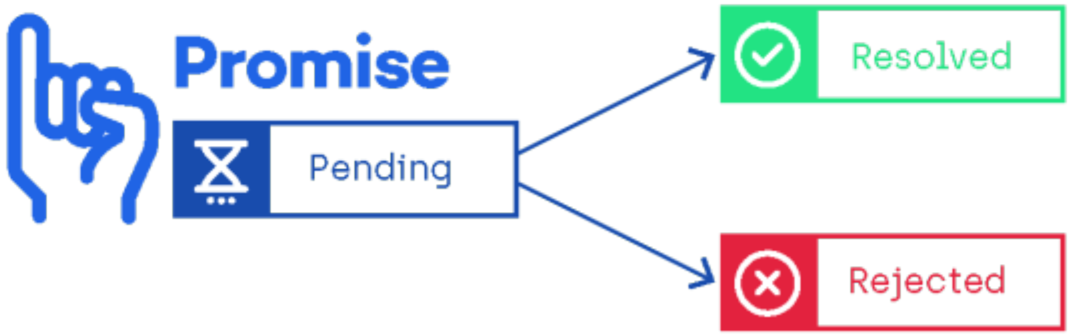
Export List + Rename → Import List + Rename

```
export {
   name1,
   name2 as newName2
}
```

```
import {
   name1 as newName1,
   newName2
} from '...'
```

samantha_ming   samanthaming.com   samanthaming

Promises



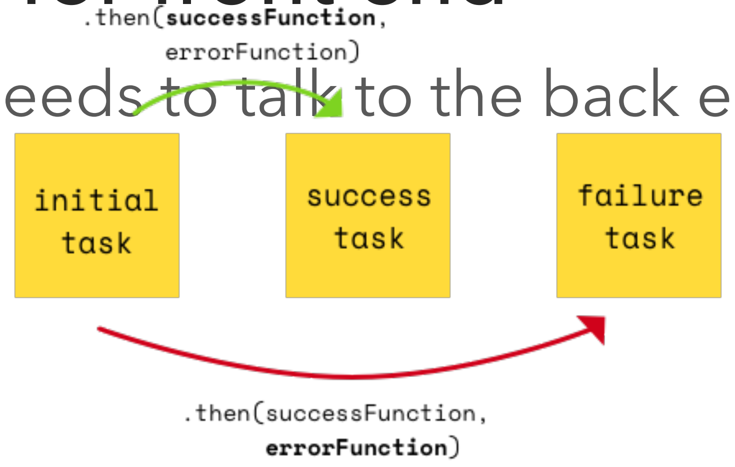
- Proxy for a value not necessarily known
  - Pending: initial state
  - Fulfilled: value known, execution successful
  - Rejected: operation failed
- let first = new Promise((resolve,reject) => { function body}
  - setTimeout(function() { resolve("Success!") },250); });
  - Useful when the internal function is asynchronous
  - Use instead of passing callbacks directly into function
- Allow chaining of callback functions
  - promise.then(), promise.catch()
  - let first = new Promise(
  - let second = first.then((msg) => { console.log("Show: " + msg) });
  - let third = second.then(…)
  - let x = new Promise(…).
    - .then(…).then(…).then(…)

Promises

- Traditional coding
  - function work1(callback) { …. callback(result,error); }
- Traditional coding
  - function work1(..) { … action(function(err,rslt) { work2(arg,err,rslt); }); }
  - function work2(..) { … action(function(rslt) { work3(arg,rslt); }) }
  - …
- Promise based coding
  - new Promise()(work1).then(work2).catch(err2).then(work3)…
  - function work1(resolve,reject) { … resolve(arg,rslt); else reject(err); }
  - function work2(arg,rslt) { … return { a: arg,r : result, r1: val } }
  - function work3(obj) { … }

Promises

- Not that useful for simple JavaScript
- However, will be very useful when coding the back end (Node.JS)
- And will be useful for front end
  - When the front end needs to talk to the back end and act when it gets a reply



Simplified CSS: less

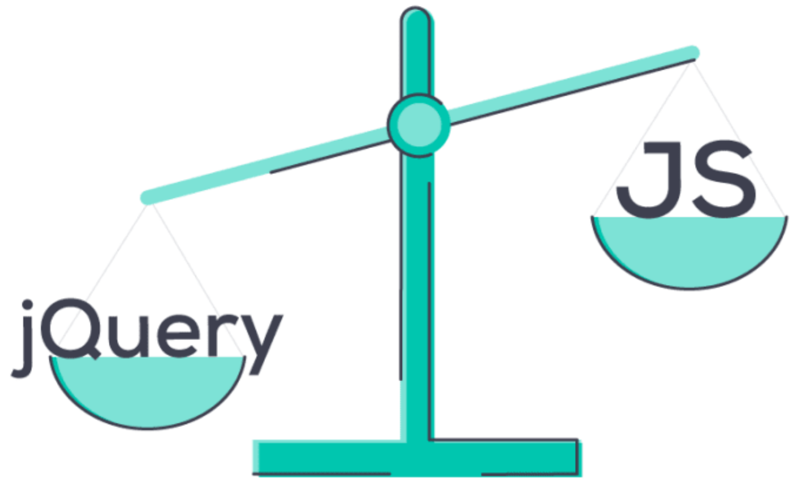
- Variables
  - @width: 10px
  - #header { width: @width; }
- Mixins
  - .bordered { … }
  - .post { .bordered(); … }
- Nesting
  - #header { …; .navigation { … } }
  - Used in place of #header .navigation { … }
- Expressions, maps, scoping, importing
- Requires running lessc to generate the actual css
  - Also does syntax checking to catch CSS errors

Simplified CSS: scss / sass

- Variables
  - $color : #ff00ff
- Mixins
  - @mixin name() { … }
  - .elt { @include name(); … }
- Nested Rules
  - As in less
  - &:xxx : qualified nesting
- Expressions, control flow, etc.
- Requires a preprocessor (scss)
  - Essentially the same capabilities as less, different syntax

jQuery : A DOM Manipulation Library

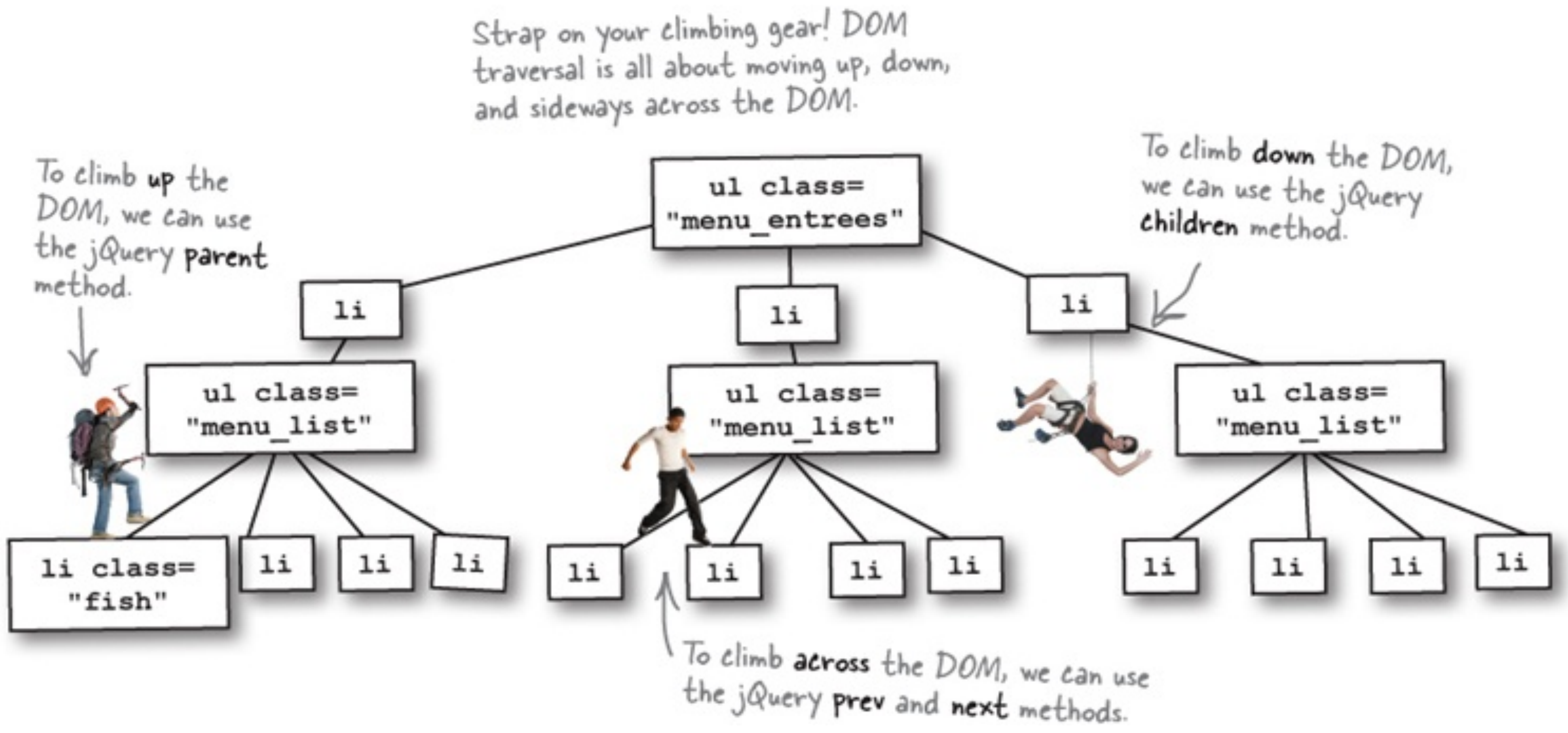
- Last time we saw how to manipulate the DOM using JavaScript
  - getElementById, querySelector, querySe
  - Setting classes, styles, text
    - For individual elements
  - Creating new HTML
- Not the easiest to use or the bes
- jQuery provides an alternative



jQuery DOM Access

- jQuery is a library to simplify DOM access/modification
  - Plus make it easier to do standard manipulations
- $(“selector”)
  - Selector is effectively a CSS selector
  - What follows applies to **ALL** matching elements
  - $(“.test”).hide(), $(“#Sum”).val(sum)
  - $(“#sample”).html(“<em>This is sample text</em>”);
  - $(“.error”).attr(“color”,”red”);
- Using jQuery is pretty standard
  - And easier than using pure JavaScript

jQuery DOM Traversal

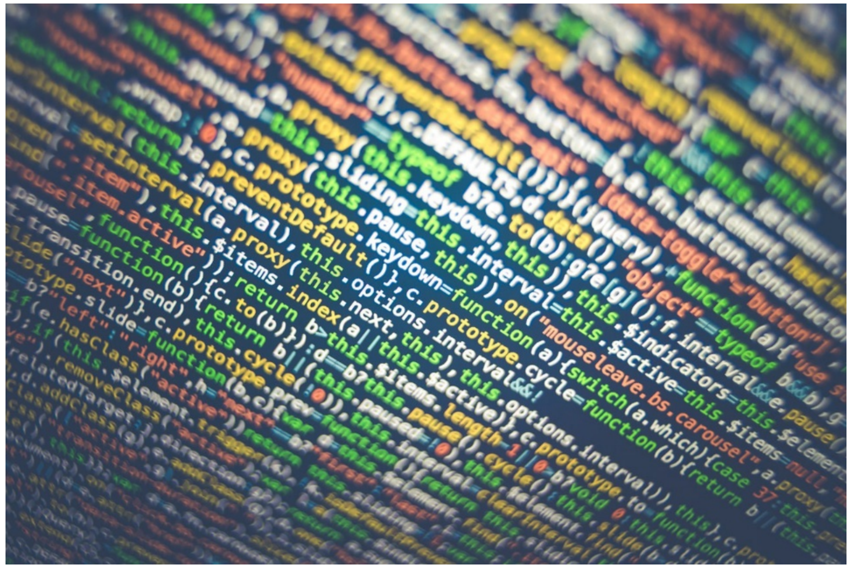

Strap on your climbing gear! DOM traversal is all about moving up, down, and sideways across the DOM.
To climb up the DOM, we can use the jQuery parent method.
To climb down the DOM, we can use the jQuery children method.
To climb across the DOM, we can use the jQuery prev and next methods.
ul class= "menu_entrees"
li
li
li
ul class= "menu_list"
ul class= "menu_list"
ul class= "menu_list"
li class= "fish"
li
li
li
li
li
li
li
li
li
li
li

Using jQuery

- $(…).onChange(function() { … }) [onXXX for all events]
- $(“<div>….</div>”) (returns the corresponding DOM)
- $(…).html(“<….>”),
- $(…).text(“ string”)
- $(…).show(), $(…).hide()
- $(function() { … })
- $(…).animate({height:300},”slow”)

- <script type=‘text/javascript’ src=‘https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js’></script>

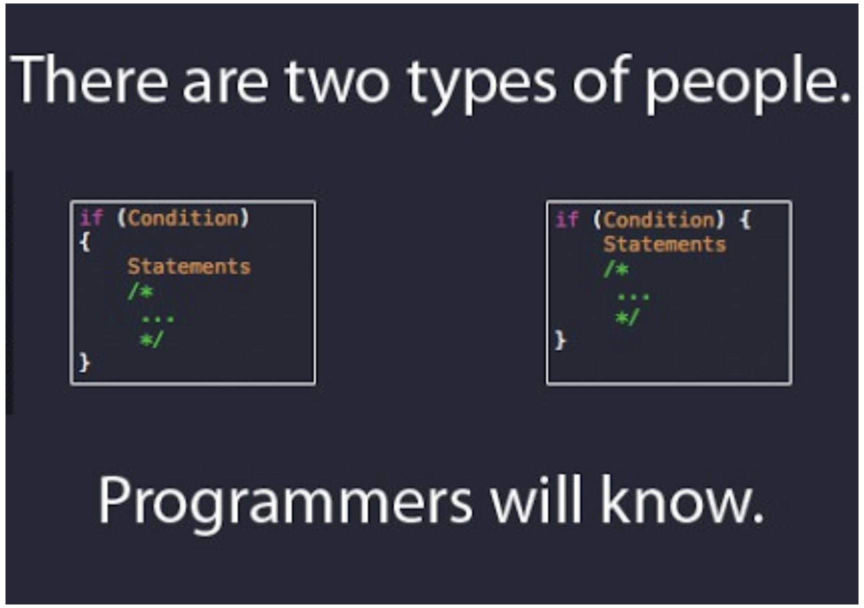
jQuery Pros and Cons



- Pros
    - Simpler to write (less typing)
    - Can create complex html from a string easily
    - Operations work on multiple elements by default
- Cons
    - Need to include the jQuery script file (more to download)
    - More difficult to debug
    - Not a framework

HTML/JavaScript Coding Style

- The browser is very forgiving
  - HTML is case insensitive
  - New lines are optional
  - Often don't need to close elements or quote attribute values
- JavaScript can be written in various ways
  - Variable names can be long or short
  - Functions can be inline, use => notation, nested
  - Objects can be declared in various ways
- But STYLE is important, especially in your final project



```
if (Condition)
{
    Statements
    /*
     ...
     */
}
```

```
if (Condition) {
    Statements
    /*
     ...
     */
}
```

HTML, CSS, and JavaScript Style

- Your HTML, CSS, and JavaScript are going to ch
  - The system will evolve
  - Bugs will be detected
  - New features will be added
- Write your code to be READ by a human
  - Not just to compile
  - Other than yourself – should be clear to whoever is reading it
  - Assume others in your final project will need to change your code
- Write your code with CHANGE in mind
  - Make it easy to change
  - Try to anticipate what might change
  - Assume things will get more complex, not simpler

Consistency and Complexity

- HTML, JavaScript, CSS should be consistent
    - Have a set of conventions and stick to it
    - Naming conventions
    - Formatting conventions
    - Coding conventions
- Consistency across the project
    - Teams should agree to and stick to a coding standard
- Avoid complexity
    - Complex code, complex HTML, …

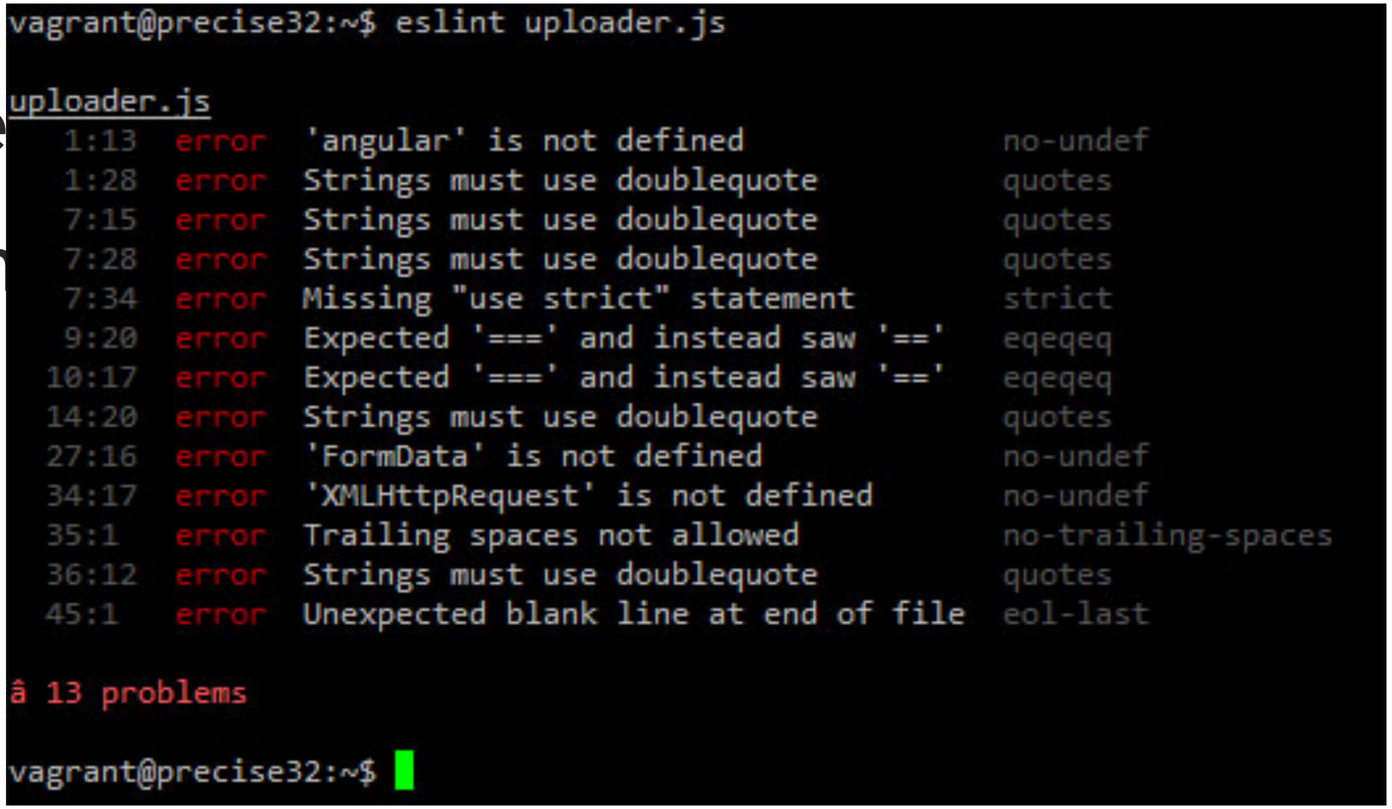
Checking Style: ESLINT

ESLint
Custom rules

- Tools exist for checking coding style
- For JavaScript, use eslint
- ESLINT
  - Can find (potential) problems with the JavaScript code
    - Common programming errors (e.g. undefined variables)
  - Can find violations of coding style
- ESLINT has a vast set of possible rules
  - Things that can be checked
  - A configuration file determines which you want checked

ESLINT Usage

- Example .eslintrc.js file
- Embedded in environn
- Example of running it



```
vagrant@precise32:~$ eslint uploader.js

uploader.js
   1:13  error  'angular' is not defined                    no-undef
   1:28  error  Strings must use doublequote                quotes
   7:15  error  Strings must use doublequote                quotes
   7:28  error  Strings must use doublequote                quotes
   7:34  error  Missing "use strict" statement              strict
   9:20  error  Expected '===' and instead saw '=='         eqeqeq
  10:17  error  Expected '===' and instead saw '=='         eqeqeq
  14:20  error  Strings must use doublequote                quotes
  27:16  error  'FormData' is not defined                   no-undef
  34:17  error  'XMLHttpRequest' is not defined             no-undef
  35:1   error  Trailing spaces not allowed                 no-trailing-spaces
  36:12  error  Strings must use doublequote                quotes
  45:1   error  Unexpected blank line at end of file        eol-last

â 13 problems

vagrant@precise32:~$
```

Next Time

- Requirements and Specifications
- Homework:
    - PreLab 2: to familiarize yourself with JavaScript