Web Advanced: Javascript

PGTE 5505. Section B. CRN 5264.
Fall 2016. Umi Syam

Class 1 - 8/31/2016

Today's menu:

Get to know me.

Get to know you.

Let's talk about syllabus.

Github class repo.

Refresh your memory.

Let's start coding.

Get to know me.

http://umisyam.com

Get to know you.

Are you registered or are you on the waiting list?

Name? 1st/2nd year?

On a scale of 1-10, how good are you at web?

What do you hope to learn in this class?

Name one of your go-to lunch places near here. (I won't judge - or maybe I do)

Syllabus

Canvas > Syllabus >
SYLLABUS_WebAdvJavaScript-SectionB-Fall2016.pdf

Or check our Github class repo
https://github.com/umisyam/WebAdvJS_Fall2016

Welcome to the world of web!

How well do you think you know web?

WEB DEVELOPMENT IN 201x

BACK-END DEVELOPMENT

3 STEPS TO START:

1. PICK YOUR LANGUAGE.
   - Top 3 most popular right now:
     - Ruby (Ruby on Rails, Sinatra)
     - PHP (with Laravel, CodeIgniter, etc)
     - NodeJS (with Express.JS / Hapi.js / Koa.js / sails.js, etc)
   - Other lang:
     - C# (.net)
     - Python (django)
     - Java (spring, grails, play)
2. LEARN THESE THINGS USING YOUR LANG OF CHOICE
   - APIs / RESTful services
   - Security
   - Authorization / Authentication (OAUTH2, JSON Web Token)
   - SOA (Service Oriented Architecture) / microservices.
3. DATABASES & CACHING
   - MySQL
   - MongoDB
   - Redis
   - PostGre SQL
   - Parse (dying..)
   - Nginx (server)
   - Apache (server)
   - Database (redis)
   - Mem-cach (in-memory caching)

DEV-OPS: Bridging the operations world (server-administration) vs the Developer world

- creates better server-deployment workflow
- automatically spin up servers (1)
- automatically provision servers (get it configured!) (2)

- WEB PLATFORMS (3)
  - Digital Ocean, Rackspace, AWS, Heroku, Azure, Engine Yard, Google App Engine, Node Jitsu
- CM / CONFIGURATION MANAGEMENT / SERVER MGMT. (2)
  - Salt, Puppet, Chef, Ansible, Linux, Docker
- CONTINUOUS INTEGRATION
  - Github Hook Deployment, Travis CI / Jenkins
- DEPLOYMENT
  - Vagrant (local-deployment), Flightplan (node-based)

DEV-OPS WORKFLOW

THE BASIC KNOWLEDGE

- HTML
- CSS
- JavaScript
- jQuery

BASIC THINGS TO LEARN FIRSTHAND:

- FTP & Web Hosting Setup
- Basic Terminal Usage
- Basic SSH
- Basics of Github
- RESTful web services / GET-POST-PUT-DELETE requests.

FRONT-END DEVELOPMENT

- CSS TOOLS
  - Responsive Web Design
  - Precompilers: SASS / LESS (enables you to define CSS variables "@")
  - CSS Frameworks: Bootstrap / Foundation
- BUILD-SYSTEM TOOLS
  - * What it does: (1) Repetitive Task → concatenate multiple JS files, pre-fix CSS, compile SASS
    (2) Utilities → uglify / minify your JS; JS Lint, JS Hint: a Javascript code quality tool, show error in your JS files
    (3) Local Server
    (4) Live Reload
  - * Why it's needed: PAGE SPEED & DEV. WORKFLOW
  - TASK RUNNERS → Grunt, Gulp, Brunch
  - DEPENDENCY MANAGEMENT → Browserify, Webpack, AMD / Require.JS (this is only efor JS-useful if you have TONS of JS lines & you need to break it down
  - PACKAGE MANAGEMENT → Bower (like NPM)
  - GENERATING PROJECT FOLDER STRUCTURE → Yeoman.io
- MV* JAVASCRIPT FRAMEWORKS - call it "Templating Engines"
  - REACT.JS / Flux: Backed by Facebook. Flux = the methodology to code React. They're getting super popular!
  - ANGULAR.JS: Backed by Google. The most popular now.
  - OTHERS: Handlebars, Backbone.js, Ember.js, Mithril, Ractive
  - Tools to do Unit Testing on those Frameworks: Mocha, Jasmine, Karma (test-runner)
- JS UTILITY LIBRARIES → the swiss-army knife! → makes every-day JS tasks easier! → provide utility functions for common programming tasks
  - Underscore.js, Lodash → (most-popular!), Wu.js, Sugar, Boiler.js, Sloth.js, Lazy.js, etc.
- JS VISUALIZATION LIBRARIES
  - Toolkit for Visual Programming for Artists: P5.JS, Three.js, Gibber, gistsandbox (if you wanna play with shaders), VVV.JS
  - For Audio engines: Tone.js, Lissajous, Wavepot, gibberish, overtone.
  - Maps & Data Visualization: D3.JS, Leaflet / mapbox, cartoDB, chart.js, google charts, Vega, Ephi, Dygraphs, etc.

Nicolas
@necolas

Follow

Are you new to front-end web development? Here's a secret: no one else really knows what they're doing either.

RETWEETS	LIKES
462	222

1:40 PM - 17 Jan 2013

462 222

WEB DEVELOPMENT IN 201x

BACK-END DEVELOPMENT

3 STEPS TO START:

1. PICK YOUR LANGUAGE.
   - Top 3 most popular right now:
     - Ruby (Ruby on Rails, Sinatra)
     - PHP (with Laravel, CodeIgniter, etc)
     - NodeJS (with Express.Js / Hapi.js / Koa.js / sails.js, etc)
   - Other lang:
     - C# (.net)
     - Python (django)
     - Java (spring, grails, play)
2. LEARN THESE THINGS USING YOUR LANG OF CHOICE
   - APIs / RESTful services
   - Security
   - Authorization / Authentication (OAUTH2, JSON Web Token)
   - SOA (Service Oriented Architecture) / microservices.
3. DATABASES & CACHING
   - MySQL
   - MongoDB
   - Redis
   - PostGre SQL
   - Parse (dying..)
   - Nginx (server)
   - Apache (server)
   - Database (redis)
   - Mem-cach (in-memory caching)

THE BASIC KNOWLEDGE

- HTML
- CSS
- JavaScript
- jQuery

BASIC THINGS TO LEARN FIRSTHAND:

- FTP & Web Hosting Setup
- Basic Terminal Usage
- Basic SSH
- Basics of Github
- RESTful web services / GET-POST-PUT-DELETE requests.

DEV-OPS: Bridging the operations world (server-administration) vs the Developer world

- creates better server-deployment workflow
- automatically spin up servers (1)
- automatically provision servers (get it configured!) (2)

- WEB PLATFORMS (3)
  - Digital Ocean, Rackspace, AWS, Heroku, Azure, Engine Yard, Google App Engine, Node Jitsu
- CM / CONFIGURATION MANAGEMENT / SERVER MGMT (2)
  - Salt, Puppet, Chef, Ansible, Linux, Docker
- CONTINUOUS INTEGRATION
  - Github Hook Deployment, Travis CI / Jenkins
- DEPLOYMENT
  - Vagrant (local-deployment), Flightplan (node-based)

DEV-OPS WORKFLOW

JS VISUALIZATION LIBRARIES

- Toolkit for Visual Programming for Artists: P5.js, Three.js, Gibber, gistsandbox (if you wanna play with shaders), VVV.js
- For Audio engines: Tone.js, Lissajous, Wavepot, gibberish, overtone.
- Maps & Data Visualization:
  - D3.js, Leaflet / mapbox, cartoDB, chart.js, google charts, Vega, Sephi, Dygraphs, etc.

FRONT-END DEVELOPMENT

- CSS TOOLS
  - Responsive Web Design
  - Precompilers: SASS / LESS (enables you to define CSS variables "@")
  - CSS Frameworks: Bootstrap / Foundation
- BUILD-SYSTEM TOOLS
  - * What it does: (1) Repetitive Task → concatenate multiple JS files, pre-fix CSS, compile SASS; (2) Utilities → uglify / minify your JS; JS Lint, JS Hint: a Javascript code quality tool, show error in your JS files; (3) Local Server; (4) Live Reload
  - * Why it's needed: PAGE SPEED & DEV. WORKFLOW
  - TASK RUNNERS → Grunt, Gulp, Brunch
  - DEPENDENCY MANAGEMENT → Browserify, Webpack, AMD / Require.JS (this is only efor JS-useful if you have TONS of JS lines & you need to break it down
  - PACKAGE MANAGEMENT → Bower (Like NPM)
  - GENERATING PROJECT FOLDER STRUCTURE → Yeoman.io
- MV* JAVASCRIPT FRAMEWORKS - call it "Templating Engines"
  - REACT.JS / Flux: Backed by Facebook. Flux = the methodology to code React. They're getting super popular!
  - ANGULAR.JS: Backed by Google. The most popular now.
  - OTHERS: Handlebars, Backbone.js, Ember.js, Mithril, Ractive
  - Tools to do Unit Testing on those Frameworks: Mocha, Jasmine, Karma (test-runner)
- JS UTILITY LIBRARIES → the swiss-army knife! → makes every-day JS tasks easier! → provide utility functions for common programming tasks
  - Underscore.js, Lodash → (most-popular!), Wu.js, Sugar, Boiler.js, Sloth.js, Lazy.js, etc.

Now, let’s refresh our memory a little bit.

Bootcamp wasn’t that long time ago - or was it?

HTML

- Semantic HTML tags
- Newer HTML5 tags: <section> <header> <nav> <footer> <article>, etc
- Including CSS
- Including JS
- Importing custom fonts and other assets

CSS

- Inline, Internal, External Styles
- Basic CSS Selectors
- Values & units (px, %, em, rem)
- Basic web color principles (RGB, RGBA, hexadecimal color, HSL)
- CSS Layout (The Box Model)
- CSS Positioning
- Floats and clearfixes
- New CSS Features (box-shadow, text-shadow, etc)
- Responsive CSS with media queries

Javascript: Back to Basics

- JS definitions
- Variable and Data types
- Arrays vs Objects
- Iteration
- Conditionals
- Functions: Declarations and Expressions
- Scopes and Closures
- Timing functions

Javascript: A Little More Advanced (next week!)

- Module pattern
- Namespacing your app
- Programming patterns and paradigms
- Public? Private?
- Immediately-Invoked Function Expression
- What is “this”?
- Anonymous functions
- Callback functions

Naming Things - anything other than:

abstract - boolean - break - byte - case - catch - char - class - const - continue - debugger - default - delete - do - double - else - enum - export - extends - false - final - finally - float - for - function - goto - if - implements - import - in - instanceof - int - interface - long - native - new - null - package - private - protected - public - return - short - static - super - switch - synchronized - this - throw - throws - transient - true - try - typeof - var - volatile - void - while - with

Naming conventions

For HTML & CSS - Class or IDs:

```
<!-- on HTML: -->
<button id="btn-submit"></button>
/* on CSS: */
#btn-submit { background-color: yellow };
```

For Javascript - variables or functions:

```
// on Javascript: better use camelCasing
var btnSubmit = document.getElementById("btn-submit");
var submitClick = function() { };
```

DO THIS NOW:

Setup your Github repo for this class

Folder name MUST be in this format:
<FirstnameLastname>_<username>_WebAdvJS_Fall16

Example:
UmiSyam_syamu557_WebAdvJS_Fall16

Then, email it to umi@newschool.edu

Clone the repo

Let’s set up our coding environment.

LET'S CODE.

Homework (1)

Read everything inside the folder ‘Week 1’ on our Github class repo, and run them, if necessary -- to make sure you understand the basics.

Homework (2)

Choose one theme below to implement:

1. Calculator - can be anything from simple numeric calculator, a tip calculator, to the quirky ones like Love calculator, Friendship calculator, etc.
2. A Trivia/Quiz App with at least 5 questions (inputs can be multiple-choice, sliders, text-box, anything you want). Think: Buzzfeed-style, or make it even better!

Requirement: must have a visual interface, not just console.log() on Terminal
Submission: upload to your own Github repo and send the link of your repo to umi@newschool.edu before the next class starts.