Creative Computing

Parsons The New School for Design
Spring 2014

See this README with a table of contents here. If you are a teacher or interested in the design of the course, see the meta document.

Course Info

- **Course:** PUCD 2035 B
- **Instructor:** Andy Dayton, email
- **Need help?**
  - Look through and create issues
  - Office Hours during Hacker Hours (see Meetup page for schedule)
  - Email for 1-on-1 help, or to set up a time to meet

Course Description

Learn best practices in JavaScript in this intensive, five-session course. Topics include data encapsulation, closures, binding, inheritance, and name spacing. Discover some of the lesser-known, yet useful, features of the language, such as how to debug JavaScript problems on different browsers and improve performance. Create interactive webpages using third-party JavaScript libraries.

Computers are provided in the lab, though you are encouraged to bring a laptop for in-class exercises.

Prerequisites

- INFO1-CE9755 - JavaScript (syllabus) or equivalent
- Understanding of variables, data types, control flow, and basic function usage in JavaScript - see Beginner Materials
- Strong intermediate knowledge of HTML, and at least basics of CSS
- Basic jQuery knowledge (DOM interaction) is a plus

These won't be enforced by the instructor, but you will be pretty lost without understanding those concepts.

Course Overview

We will dive into the nuances of JavaScript, how prototypal inheritance compares to classical inheritance, and how this can be used to build dynamic and complex web applications. Modern tools like jQuery and BackboneJS will be discussed, but students will learn the building blocks of these frameworks and after this course be able to understand what is happening under the hood. The focus will be on development for browsers, though most applies to other systems like Node.js, Phonegap, etc. Topics covered include:

- Encapsulation, closures and scope
- Classical vs. prototypal inheritance
- The event loop
- AJAX and JSONP
  - local
  - remote (e.g. Foursquare)
- Creating MVC-style models (a'la Backbone.js) from scratch
- Test- and Pseudocode-Driven Development

Topics will be demonstrated through live-code examples/slides, available at advanced-js.github.io/deck. Additional exercises will completed in-class.

See this interview for more background.

Homework/Projects

All assignments are listed within the Course Outline.

Workflow

1. Fork the repository for the exercise/project (found under github.com/advanced-js)
2. Clone the repository to your computer
3. Open the `index.html` file in a browser and open the Developer Tools
4. Modify the files to complete your solution
5. Refresh the `index.html` page to see the results, and repeat
6. Make sure all of your code is committed
7. Push/sync up to GitHub
8. Create a pull request on the original repository by the due time (generally the start of the following class)
9. You can continue to push fixes and improvements until the close date (listed in Classes) â€“ just add a comment in the pull request to let me know it's been updated.

When the pull request is created, you should see a message saying that "the Travis CI build is in progress" â€“ this means that your solution is being automatically checked for syntax errors. If this "build" ends up failing (which will show a red "X"), click through the "details" link and scroll to the bottom to see what the errors were. Per the requirements below, please fix the issues and push up the changes.

Feedback will be given in the pull request, so please respond with your thoughts and questions! You are welcome to open the pull request as the work is still in-progress if you are stuck and want to ask a question â€“ just mention `@afeld` with the question to make sure I know to look at it sooner.

Note that your solution will also be live at `http://USERNAME.github.io/EXERCISE`. For exercises with multiple levels/versions, leave a new comment in the pull request saying "Level X finished!" before pushing commits for the next level.

Requirements

These apply to real life, as well.

- Travis CI build should pass, which includes:
  - All HTML files should pass W3C Markup Validation
  - All written JS should pass JSHint
- Must apply "good programming style" learned in class
  - Functions should be "short" (see Sandi Metz's rules for developers)
  - Optimize for readability
  - For projects, use Object-Oriented Programming
- Bonus points for:
  - Automated tests
  - Creativity (as long as requirements are fulfilled)

Course Outline

Class 1

1. Introduction
    - Install GitHub for Mac or Windows
    - Sign up for GitHub
2. Student checklist:
    - Access NYU Classes page, where grades will be posted
        - Documentation
3. Explain how slides work
4. Get through `echo_exercise` slide
5. GitHub workflow
    - Walk through workflow
    - Create pull request on students repository
6. Get through "self_executing_functions" slide

Homework

- Read JavaScript Garden
- Finish up and submit echo and countdown exercises
- Complete blink exercise

Class 2

1. Look at various approaches for `countdown()`
    - Show recursive solution
2. Developer Tools walkthrough
    - Elements (HTML)
    - Console (JS)
    - Scripts (JS)
3. Pair program to build Memory v1 (see pairing tips)
4. Cover OOP, though "oop_inheritance" slide
    - Encapsulation example
    - Look at Backbone.js Events

Homework

- Read Mozilla's Introduction to Object-Oriented Javascript
- OOP exercise, through V2
- Memory v2 (individual)

Class 3

1. Code review Memory
2. Get through `oop_inheritance` slide
3. Cover automated testing
    - Build up a test framework from scratch
    - Examples in QUnit
        - Simple
        - Classes
    - Other frameworks
4. Cover AJAX/JSONP (files)
    - Network tab in Developer Tools

Homework

- Read Google JavaScript Style Guide
- OOP exercise V3

- Memory V3

Class 4

1. Finish slides
2. Getting Serious example
    - Quick intro to Backbone.js
        - Boilerplate
        - Click the Box example app
        - TDD?
3. Multiple async
    - Promises/jQuery.Deferred
    - Possibly show async library?

Homework

- Namespace exercise
- Mashup

Class 5

1. Present and code review Mashup projects
2. Possible topics (vote?):
    - Node.js
        - Server "Hello World" (from Node.js homepage)
            - HTTP module docs
        - HTTP requests
            - Status codes
            - Headers
        - CommonJS?
    - Regular Expressions
        - Convert live input, e.g. link Twitter handles from a textarea
    - Code Retreat â€“ possible "problems":
        - Game of Life
        - Tic Tac Toe

Pairing Tips

- Three people is possible, but two works best
- Agree on an editor and environment that you're both comfortable with
- The person who's less experienced/comfortable should have more keyboard time
- Switch who's "driving" regularly
- Make sure to save the code and send it to both people

Resources

Required Reading

- Google JavaScript Style Guide
- JavaScript Garden
- Mozilla's Introduction to Object-Oriented Javascript
- Whatâ€™s so great about JavaScript Promises?
- https://twitter.com/necolas/status/291978260433219584
- http://afeld.me/nerdery/1742468

Beginner Materials

This class assumes you are confident with this material, but in case you need a brush-up...

- Codecademy â€“ JavaScript and jQuery
- Eloquent JavaScript by Marijn Haverbeke, Chapters 1-5
- see also â€“ Other Lists

Recommended Reading

- Functional JavaScript by Michael Fogus
- Front-end Job Interview Questions by @darcyclarke (for testing yourself)
- JavaScript Best Practices
- JavaScript Patterns by @shichuan (thanks @iandrewfuchs)
- JavaScript Patterns by Stoyan Stephanov
- JavaScript Web Applications by Alex MacCaw
- JavaScript: The Good Parts by Douglas Crockford
- Learning Advanced JavaScript slides by John Resig
- Static Web Apps
- Test-Driven JavaScript Development by Christian Johansen
- The JavaScript Interpreter, Interpreted by Martha Girdler (video)

Specific Topics

- Classical Inheritance in JavaScript by Douglas Crockford
- Partial Application in JavaScript by Ben Alman (thanks @michaelBenin)
- HTML5 Rocks slides
- Learning JavaScript Design Patterns by Addy Osmani

Other Lists

- JS: The Right Way (an overview of the JS landscape)
- Code School
- Thoughtbot's Javascript Trail Map
- How To Learn JavaScript Properly
- Superhero.js
- Teach Yourself to Code

Tools

- code validation: JSLint / JSHint
- debugging: Chrome Developer Tools (tutorial) / Firebug
- sharing code snippets: gist.github.com
- asking questions: Stack Overflow

GitHub

- Git and GitHub
  - Official GitHub Help
  - Recommended resources
- GitHub Pages
  - Official site
  - Thinkful guide

HTML/CSS/JS Sandboxes

- JS Bin (recommended)
- bl.ocks.org
- Cloud9
- CodePen
- JSFiddle
- Plunker
- rawgithub.com

Frameworks

- Framework comparison: TodoMVC
- Testing

Reference

- Mozilla Developer Network and Learn JavaScript
- w3schools
- JavaScript: The Definitive Guide by David Flanagan

More Examples

- map/reduce (in Underscore)

Grading

- Class Participation â€“ 30%
- Homework â€“ 70%

Statements on Plagiarism

SCPS

> New York University takes plagiarism very seriously and regards it as a form of fraud. The definition of plagiarism that has been adopted by the School of Continuing and Professional Studies is as follows: "Plagiarism is presenting someone else's work as though it were one's own. More specifically, plagiarism is to present as one's own words quoted without quotation marks from another writer; a paraphrased passage from another writerâ€™s work; or facts or ideas gathered, organized, and reported by someone else, orally and/or in writing. Since plagiarism is a matter of fact, not of the student's intention, it is crucial that acknowledgement of the sources be accurate and complete. Even where there is not a conscious intention to deceive, the failure to make appropriate acknowledgement constitutes plagiarism. Penalties for plagiarism range from failure for a paper or course to dismissal from the University.

Instructor

Reuse and building upon ideas or code are major parts of modern software development. As a professional programmer you will never write anything from scratch. This class is structured such that all solutions are public. You are encouraged to learn from the work of your peers. I won't hunt down people who are simply copying-and-pasting solutions, because without challenging themselves, they are simply wasting their time and money taking this class.

Please respect the terms of use and/or license of any code you find, and if you reimplement or duplicate an algorithm or code from elsewhere, credit the original source with an inline comment.

License