Learning objectives

- Get to know your classmates, instructors, and staff members.
- Explain the structure of the course and tools that will be used.
- Discuss the benchmarks for assessments in terms of class participation, homework and unit projects.
- Install and configure Node.js, npm, Git and other command line tools.
- Identify common issues that might arise and solutions that will be used during the course.
- Practice programmatic thinking by writing pseudocode.


JAVASCRIPT
IS NOT FUNNY
memegenerator.net

Early History

- Invented in 10 days in 1996 by a guy at Netscape named Brendan Eich
- At its heart it was originally a functional programming language
- Java was all the rage, so for marketing reasons they changed the language to make it look more like Java and have superficial OOP Java-like features
- …and they changed the name from LiveScript to JavaScript
- More details: http://speakingjs.com/es5/ch04.html

Object-oriented programming vs. Functional Programming

- Fundamental paradigms used in JavaScript: OOP and functional

- Don’t worry about understanding them now, but keep them in mind throughout the course

OOP

- OOP is all about modeling data the way we think about the real world, as things that have properties and behaviors, or things you can do with them

- A “car” object has properties like color, style, and whether the gas tank is empty or full; behaviors can be “honk” or “fill gas tank”

- The idea is to make it easier for human programmers to visualize the program by tying the program to concrete things, which lets you group things logically

- Fundamental building blocks are the objects, their properties, and their behaviors (often called “methods”)

Functional

- Fundamental building block is the function — a thing which takes inputs and produces outputs
- Data is not grouped together with the function, like it is with OOP
- Functions do not alter their inputs — they just produce new outputs
- Functions can be chained together in a pipeline
- The inputs and outputs can even be functions themselves (don’t worry, we’ll be going over that a lot)

Structure and Benchmarks

- 20 classes
- Fundamentals of programming and JavaScript (datatypes, variables, loops, etc.)
- Object-oriented and functional programming
- Using JavaScript to manipulate websites and applications
- 3 projects

Unit breakdown

- Unit 1: Fundamentals of JavaScript
    - project: Slackbot, starting end of Lesson 3
- Unit 2: The Browser and APIs
    - project: Feedr, starting end of lesson 10
- Unit 3: Persisting Data and Advanced Topics
- Unit 4: Building and Deploying Your App
    - project: Final Project, starting end of lesson 13

Set up Slack

- We'll be using Slack to communicate in class. We will also use this as the primary means to take attendance. Follow the instructions below to sign up for slack.

- Visit https://slack.com/downloads to download the application.

- Sign up using your email and join the Slack team jsdev2 (we'll be using the #general channel).

- Upload a profile picture to Slack.

brew, git, Node, Sublime Text

- See instructions at https://gist.github.com/richardharrington/de14808f2e2cfa10935662d616340bdd
- If you get frustrated, watch this vine: https://vine.co/v/hPXTA6l9AqQ
- Then watch this video. It has nothing to do with anything but I love it: https://www.youtube.com/watch?v=u8qgehH3kEQ

Thinking like a programmer

- Writing effective programs requires a major shift in thinking.

- We need to try to understand how computers think and follow our instructions (code) in order to become great programmers.

- A very common first step is to write anything we want our computer to do in plain English — or pseudocode.

Pseudocode

- Pseudocode is an outline of a program, written in a form that can easily be converted into real programming statements.

- No formatting or syntax rules

- Often written as a series of steps, much like a recipe

- Benefit: lets you concentrate on the flow of the program without worrying about the syntactic details

- Helps you to figure out in your mind specifically what needs to happen, sometimes before you even decide what language you want to use

Pseudocode example

Say I want to get some milk. I prefer whole milk, or as close to it as possible.

```
go to the store
go to the shelf with the milk
if they have whole milk,
    then grab a carton of whole milk
if they don't have whole milk but they have 2%,
    then grab a carton of 2%
if you found any milk at all,
    then pay for the milk
go home
```

Pseudocode exercise

Write the pseudocode for a program for that controls a light that can change color based on user input. Imagine that we have 3 different colored buttons (red, blue and yellow) and a light, and if a user taps one of the colored buttons the light changes to the selected color. If the selected color is tapped again, the light turns off.

You do not have to write any actual JavaScript for this exercise.

Conclusion

- How was this installation process different than using a GUI (graphical user interface)? Why did we do it this way?

- What questions do you have about the course or the specific tools we installed today?