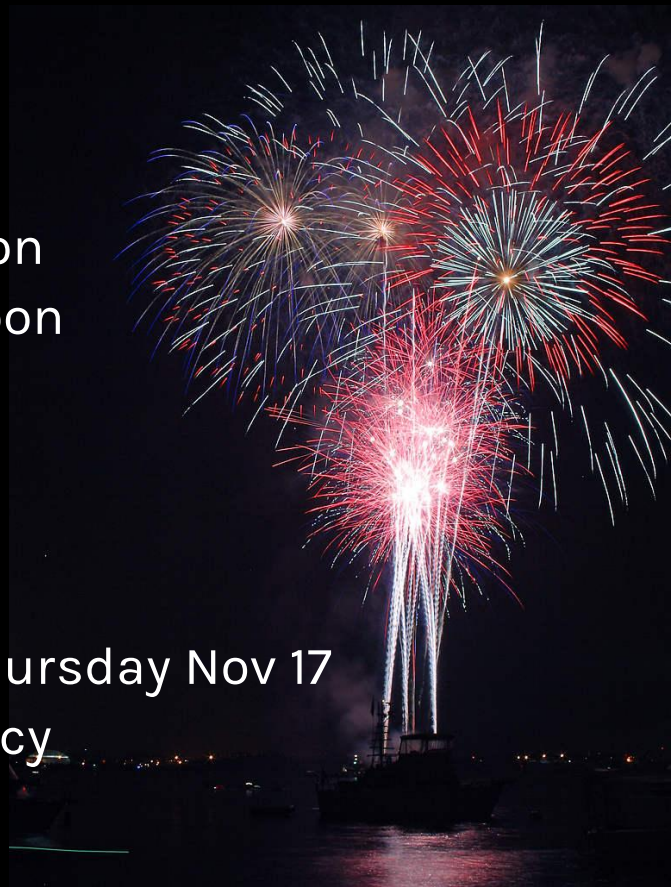
Welcome to CS50 section! This is Week 10 :(

- This is our last section!

- Final project dates
  - Official proposals: due this Friday at noon
  - Status report: due Monday, Nov 28 at noon
  - Hackathon: Thursday, Dec 1 at 7pm
  - Turn in: Thursday, Dec 8 at noon
  - Present: Friday, Dec 9

- Second midterm: Monday Nov 14 through Thursday Nov 17
  - Same “take home, no collaboration” policy



Chugging along

Course timeline:

~~Raw C code~~

~~Distribution C code~~

~~Raw Python code~~

~~Framework Python code (Flask)~~

~~HTML/CSS~~

JavaScript

Base frameworks: jQuery

Distribution JS code

Chugging along

Course timeline:

~~Raw C code~~

~~Distribution C code~~

~~Raw Python code~~

~~Framework Python code (Flask)~~

~~HTML/CSS~~

JavaScript

Base frameworks: jQuery

Distribution JS code

Application timeline:

~~On a computer~~

- ~~No input allowed~~
- ~~Input through cmd line~~
- ~~Input through files~~

~~Through a server~~

- ~~Input through API~~
- ~~Input through website~~

On a webpage

- Input through JS

Before starting pset 8

- JavaScript vs Python vs C
- JavaScript syntax, data structures

- The document object model
- How to interact with the DOM through JavaScript
  - jQuery selectors and interaction

- Theories of AJAX

- Debugging in JavaScript

JavaScript

- Syntactically similar to C
  - Belongs in the family of “C-based languages”
  - Semicolons recommended but not technically required

- Practices just-in-time compiling (JIT)
  - Browsers will run JavaScript slightly differently

- Very weakly typed, similar to Python except even worse

- Some people hate JS, others love it
  - Modern JS programmers will “compile down” to JS

JavaScript → Syntax

```
// while loop
while (true)
{
    // do something
}

// for loop
for (initialization; condition; update)
{
    // do something
}
```

JavaScript → Data structures

- Normal variables can be of any type
  - `var my_integer = 20`
  - `var my_float = 20.0`
  - `var my_string = “Brandon”`

- Arrays in JavaScript are similar to lists in Python:
  1D, mutable, contain anything, defined by square brackets
  - `var my_array = [20, 21, “apple”, [“another”, “list”]]`
    `my_array[0] // Returns 20`

- Tuples a la Python do not exist

JavaScript → Data structures

- Objects in JavaScript are similar to dictionaries in Python and structs in C: mutable, contain anything, keys are integers/strings
  - `var my_object = { name: “Brandon”, year: 2019 }`

- Functions in JavaScript can be both first-class and “anonymous”:
  - `function func1() { return true; }`
  - `var func2 = function() { return true; }`
  - `var func3 = function exec_name() { return true; }`

- JS objects are actually very complex:
  - They can contain functions, and thus objects can function similarly to classes (with constructors, etc)

JavaScript → Functions in objects

```
var school_app = {
  apple: 5,
  load: function() { // Do something },
  read: function() { // Do something }
}

// Now I can call those functions
school_app.load()
school_app.read()

// Should still use bracket syntax to index into objects
school_app[“apple”] // Returns 5
```

DOM

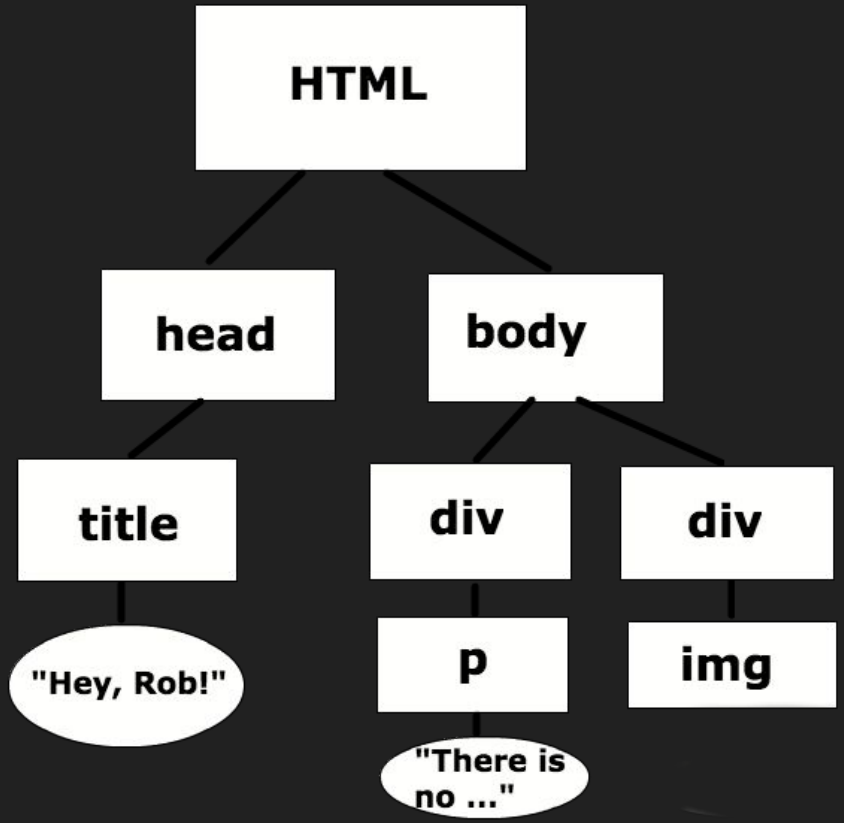
- We can think of all of our HTML as a giant tree. Things are nested beneath each other, etc.

- We call this tree the document object model (DOM)
  - Why? Each of the nodes is (sorta) like an “object” (synonym here for dictionary a la Python)
  - But it’s not exactly this way-- hence “model”

DOM

```
<!DOCTYPE html>
    <head>
        <title id="title">Hey Rob!</title>
    </head>
    <body>
        <div>
            <img id="pic" src="rob.jpg" alt="rob">
        </div>
        <div>
            <p id="quote">There is no happiness in the world, only rice...</p>
        </div>
    </body>
</html>
```

DOM



DOM and JavaScript

- The main reason software engineers created the DOM is to develop a good way of interacting with the page through JavaScript.

- Benefits?
  - Why not just through Python?

DOM and jQuery

- It’s actually a bit of a pain to do things with JavaScript directly, so libraries have been created to help us out.

- jQuery is a popular DOM manipulation library.
  - (It does other things too)

- Letting us turn:

```
document.getElementById("quote").innerHTML = "allllright";
```

Into:

```
$(“#quote”).innerHTML(“allllright”);
```

Selecting elements through jQuery

- Select elements through the jQuery selector:

```
$(“ELEMENT”)
```

```
<element id=”apple” class=”orange blueberry”
attribute=”blah” attribute2=”bleh”>
  Hello!
</element>
```

Selecting elements through jQuery

```
<element id=”apple” class=”orange blueberry”
attribute=”blah” attribute2=”bleh”>
  Hello!
</element>
```

Select element by ID:

```
var $element = $(“#apple”)
$element.attr(“attribute”) // returns “blah”
```

Selecting elements through jQuery

```
<element id=”apple” class=”orange blueberry”
attribute=”blah” attribute2=”bleh”>
  Hello!
</element>
```

Select element by class:

```
var $element = $(“.orange”)
$element.attr(“attribute”) // returns “blah”
```

Selecting elements through jQuery

```
<element id=”apple” class=”orange blueberry”
attribute=”blah” attribute2=”bleh”>
  Hello!
</element>
```

Select element by classes:

```
var $element = $(“.orange.blueberry”)
$element.attr(“attribute”) // returns “blah”
```

Selecting elements through jQuery

```
<element id=”apple” class=”orange blueberry”
attribute=”blah” attribute2=”bleh”>
  Hello!
</element>
```

Select element by element name:

```
var $element = $(“element”)
$element.attr(“attribute”) // returns “blah”
```

Selecting elements through jQuery

```
<bah>
  <element id=”apple” class=”orange blueberry”
  attribute=”blah” attribute2=”bleh”>
    Hello!
  </element>
</bah>
```

Select element by DOM hierarchy:

```
var $element = $(“bah element.orange”)
$element.attr(“attribute”) // returns “blah”
```

Selecting elements through jQuery

- Selector names are the same as in CSS

- You can do a lot of things through jQuery
  - Change values
  - Change CSS of an element, i.e. change its look and feel
  - Adding events

Adding events, you say?

JavaScript → Events

What's an event?

An event is an action that happens on a page. Anything from a click, to the page loading successfully, to your mouse moving around, to a keyboard action, and so on and forth.

In JavaScript, we can attach functions (called callbacks or event handlers, depending on the context) that do something when these events occur.

JavaScript → Events

What are some examples of events?

AJAX

- AJAX → “Asynchronous JavaScript and XML” (outdated definition!)

- A better definition: AJAX is an asynchronous method of communicating with a server, usually with JavaScript and JSON.

Let’s talk about:

- Synchronicity and async
- JSON (JavaScript Object Notation)
- Events in the context of async (callbacks, etc)

Debugging in JavaScript

A live demo...

Before starting pset 8

- JavaScript vs Python vs C
- JavaScript syntax, data structures

- The document object model
- How to interact with the DOM through JavaScript
  - jQuery selectors and interaction

- Theories of AJAX

- Debugging in JavaScript

That’s all for today (and the term!)