CSCI E-50 WEEK 10

TERESA LEE
[teresa@cs50.net]

April 2, 2018

IMPORTANT

Next week April 9th 2018 will be the final section

Next section will be Review

Please forward your questions to teresa@cs50.net

Agenda

- JavaScript
- DOM
- jQuery
- AJAX
- Final Pset

- **JavaScript**, like Python, is much newer than C (1995 vs. 1972), but is also very heavily inspired by it.

- To start writing JavaScript, open a file with the .js file extension.

- Unlike Python which runs *server-side*, most JavaScript applications run *client-side*, on your own machine. (Modifications to JavaScript, such as the popular Node.js, is server-side.

- JavaScript, HTML, and CSS together basically comprise the backbone of the internet.

- Much like with CSS and <style> tags, you can directly write JS between <script> tags, but you can also link external JavaScript files (which is probably the preferred approach!) by way of the src attribute of <script> tags.

- Variables in JavaScript do not require a type specifier, and do not need to be declared in advance. But there is a special keyword for introducing them.

- Variables in JavaScript do not require a type specifier, and do not need to be declared in advance. But there is a special keyword for introducing them.

```
let x = 44;
```

- Variables in JavaScript do not require a type specifier, and do not need to be declared in advance. But there is a special keyword for introducing them.

```
let x = 44;
```

- Variables in JavaScript do not require a type specifier, and do not need to be declared in advance. But there is a special keyword for introducing them.

```
let x = 44;
```

- Conditionals are the same as C, and curly braces are used to delimit the blocks again.

- Conditionals are the same as C, and curly braces are used to delimit the blocks again.

```
if
else if
else
switch
?:
```

- Loops are the same as C, and curly braces are used to delimit the blocks again.

- Loops are the same as C, and curly braces are used to delimit the blocks again.

```
while
do-while
for
```

- Functions are introduced with the function keyword (basically equivalent to Python’s def).

- JavaScript functions can be *anonymous*--you don’t have to give them a name!
  - We’ll revisit this idea shortly.
  - By the way, Python technically has this ability too!

- Declaring arrays (again called arrays in JavaScript) looks really similar to a Python list, and can contain mixed types as before.

- Declaring arrays (again called arrays in JavaScript) looks really similar to a Python list.

```
let nums = [1, 2, 3, 4, 5];
```

- As was the case with Python, JavaScript has the ability to behave as an object-oriented programming language, with properties contained within the object, and methods that apply only to objects that define those methods.

- JavaScript objects look a lot like Python dictionaries:

- As was the case with Python, JavaScript has the ability to behave as an object-oriented programming language, with properties contained within the object, and methods that apply only to objects that define those methods.

- JavaScript objects look a lot like Python dictionaries:

```
let herbie = { year: 1963, model: "Beetle"};
```

- Loops are the same as C, and curly braces are used to delimit the blocks again.

while

do-while

for

for ... in

- How do we iterate across all of the keys of an object?

```
let herbie = {  year: 1963,
                model: "Beetle",
                sound: "honk.mp3"
              };
```

Let’s Look at an Example

`practice/templates/objects.html`

- How do we iterate across all of the keys of an object?

```
let herbie = {  year: 1963,
                model: "Beetle",
                sound: "honk.mp3"
              };
```

```
for (let prop in herbie)
{
   console.log(herbie[prop]);
}
```

- How do we iterate across all of the keys of an object?

```
let herbie = {  year: 1963,
                model: "Beetle",
                sound: "honk.mp3"
             };
```

```
for (let prop in herbie)
{
   console.log(herbie[prop]);
}
```

```
1963
Beetle
honk.mp3
```

- How do we iterate across all of the keys of an object?

```
   let herbie = {  year: 1963,
                   model: "Beetle",
                   sound: "honk.mp3"
                 };
```

```
for (let prop in herbie)
{
    console.log(prop);
}
```

- How do we iterate across all of the keys of an object?

```
let herbie = {  year: 1963,
                model: "Beetle",
                sound: "honk.mp3"
              };
```

```
for (let prop in herbie)
{
   console.log(prop);
}
```

```
year
model
sound
```

- Loops are the same as C, and curly braces are used to delimit the blocks again.

while

do-while

for

for ... in

for ... of

- How do we iterate across all of the elements of an array?)

```
let wkArray = ["Mon", "Tue", "Wed"];
```

- How do we iterate across all of the elements of an array?)

```
    let wkArray = ["Mon", "Tue", "Wed"];
for (let day of wkArray)
{
   console.log(day);
}
```

- How do we iterate across all of the elements of an array?)

```
    let wkArray = ["Mon", "Tue", "Wed"];
for (let day of wkArray)
{
   console.log(day);
}
```

```
Mon
Tue
Wed
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

```
console.log(wkArray[day] + " is day number "
            + (day + 1) + " of the week!");
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

```
console.log(wkArray[day] + " is day number "
            + (day + 1) + " of the week!");
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

```
console.log(wkArray[day] + " is day number "
            + (parseInt(day) + 1) +
            " of the week!");
```

- Strings can be concatenated in JavaScript using the + operator… but be careful mixing types!

- As with Python, there are still underlying data types, we just don’t often have to worry about them. But here’s the tradeoff of losing the precise control we had in C!

- Arrays are a special case of an object (actually, everything in JavaScript in a special case of an object!). Many methods can be applied to them natively.

```
array.size()
array.pop()
array.push(x)
array.shift()
array.map()
```

- Arrays are a special case of an object (actually, everything in JavaScript in a special case of an object!). Many methods can be applied to them natively.

```
array.size()
array.pop()
array.push(x)
array.shift()
array.map()
```

- Arrays are a special case of an object (actually, everything in JavaScript in a special case of an object!). Many methods can be applied to them natively.

```
array.size()
array.pop()
array.push(x)
array.shift()
array.map()
```

- This one will give us a good way to introduce anonymous functions.

- map() accepts as its parameter a function to be applied to every element of the array. We could define the function in advance and pass it… or we could just define the function in our call to map()!

- map() accepts as its parameter a function to be applied to every element of the array. We could define the function in advance and pass it… or we could just define the function in our call to map()!

```
let nums = [1, 2, 3, 4, 5];
```

- map() accepts as its parameter a function to be applied to every element of the array. We could define the function in advance and pass it… or we could just define the function in our call to map()!

```
let nums = [1, 2, 3, 4, 5];


nums = nums.map()
```

- map() accepts as its parameter a function to be applied to every element of the array. We could define the function in advance and pass it… or we could just define the function in our call to map()!

```
let nums = [1, 2, 3, 4, 5];

nums = nums.map(function(num) {
   return num * 2;
});
```

- map() accepts as its parameter a function to be applied to every element of the array. We could define the function in advance and pass it… or we could just define the function in our call to map()!

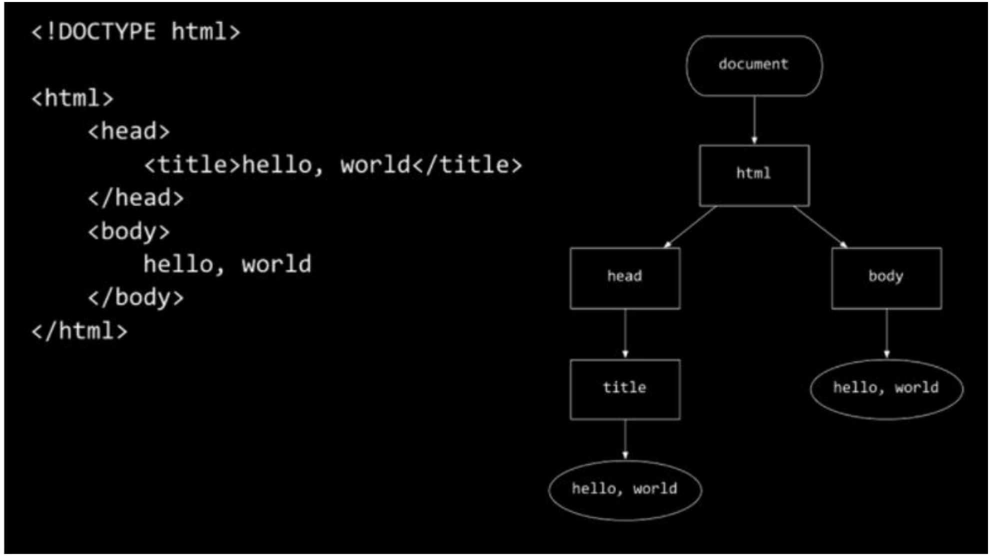
```
let nums = [2, 4, 6, 8, 10];

nums = nums.map(function(num) {
   return num * 2;
});
```

- An **event** in HTML/JavaScript is a response to user interaction with the web page. (e.g. user clicked a button, a page has finished loading…)

- JavaScript supports **event handlers**, which are functions that respond to HTML events.

```
<html>
   <head>
      <title>Event Handlers</title>
   </head>
   <body>
      <button onclick="">Button 1</button>
      <button onclick="">Button 2</button>
   </body>
</html>
```

DOM



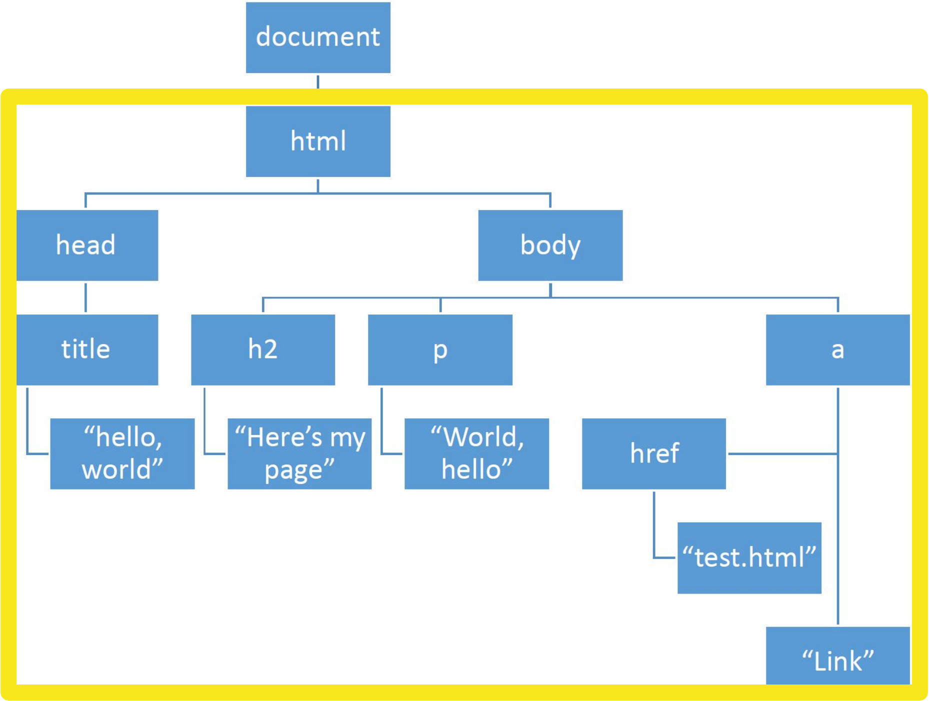
```
<!DOCTYPE html>

<html>
    <head>
        <title>hello, world</title>
    </head>
    <body>
        hello, world
    </body>
</html>
```

A structure of a simple HTML file can be represented as a tree.

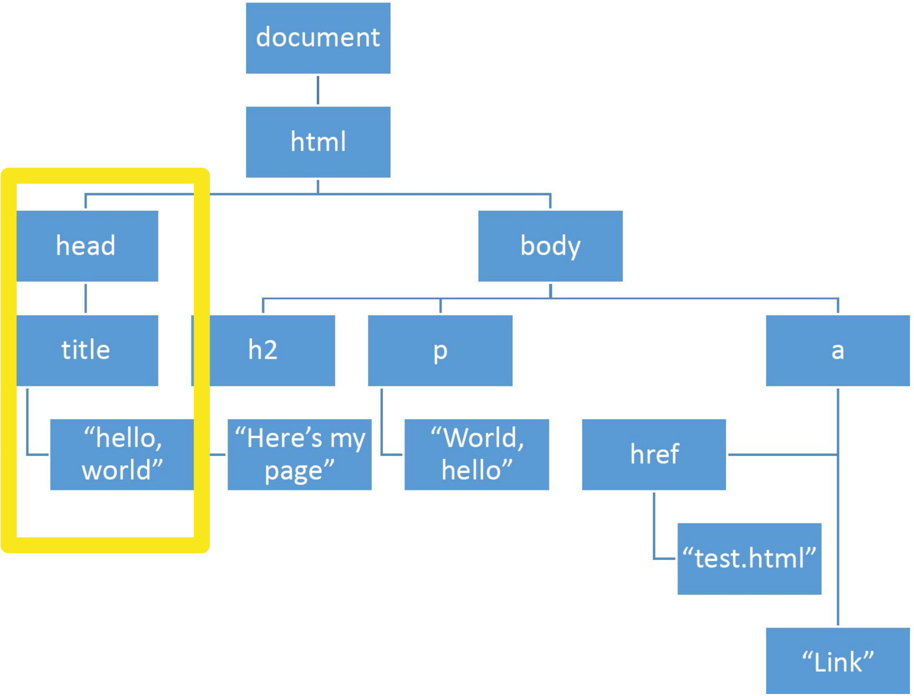
With JavaScript, we can write code to change this tree after the browser has downloaded the HTML file and displayed it to the user.

DOM

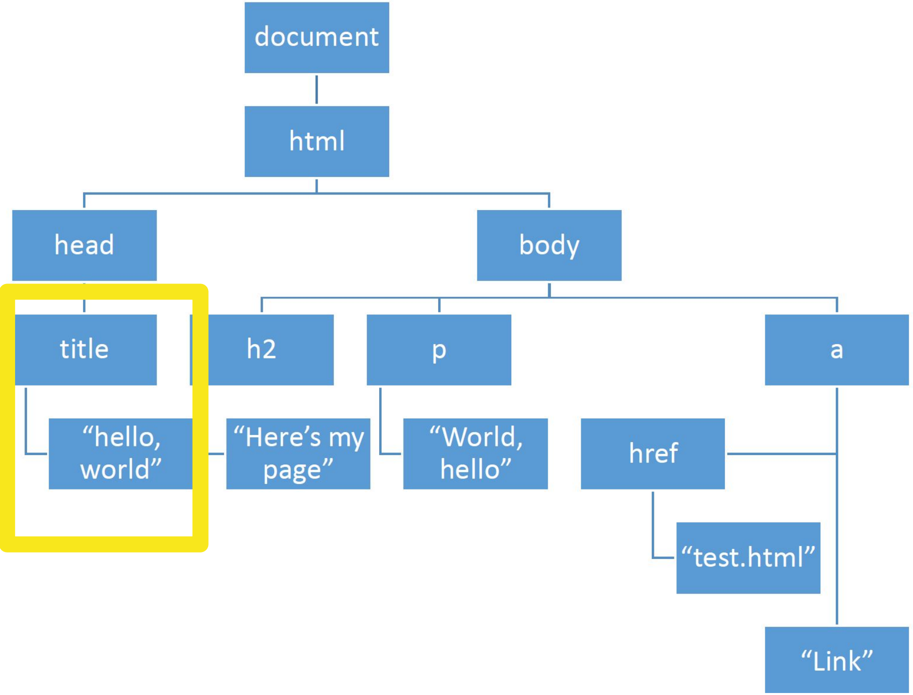
- JavaScript has a special object called the **document object**, which is effectively a single object that represents an entire web page.
- Because objects can have numerous fields, can because those fields can be of any type, including being themselves objects, this lends itself to a nice hierarchical organization.
- By organizing an entire page into a JavaScript object, we can manipulate its elements programmatically.



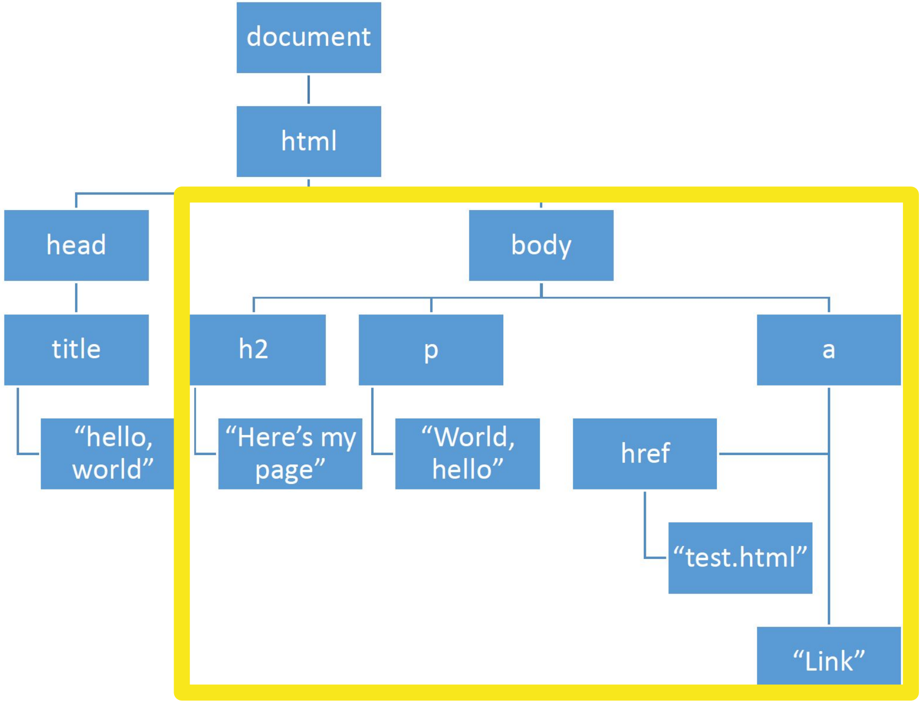
```
<html>
     <head>
          <title>Hello, world</title>
     </head>
     <body>
          <h2>Here's my page</h2>
          <p>World, hello</p>
          <a href="test.html">Link</a>
     </body>
</html>
```



```
<html>
    <head>
        <title>Hello, world</title>
    </head>
    <body>
        <h2>Here's my page</h2>
        <p>World, hello</p>
        <a href="test.html">Link</a>
    </body>
</html>
```



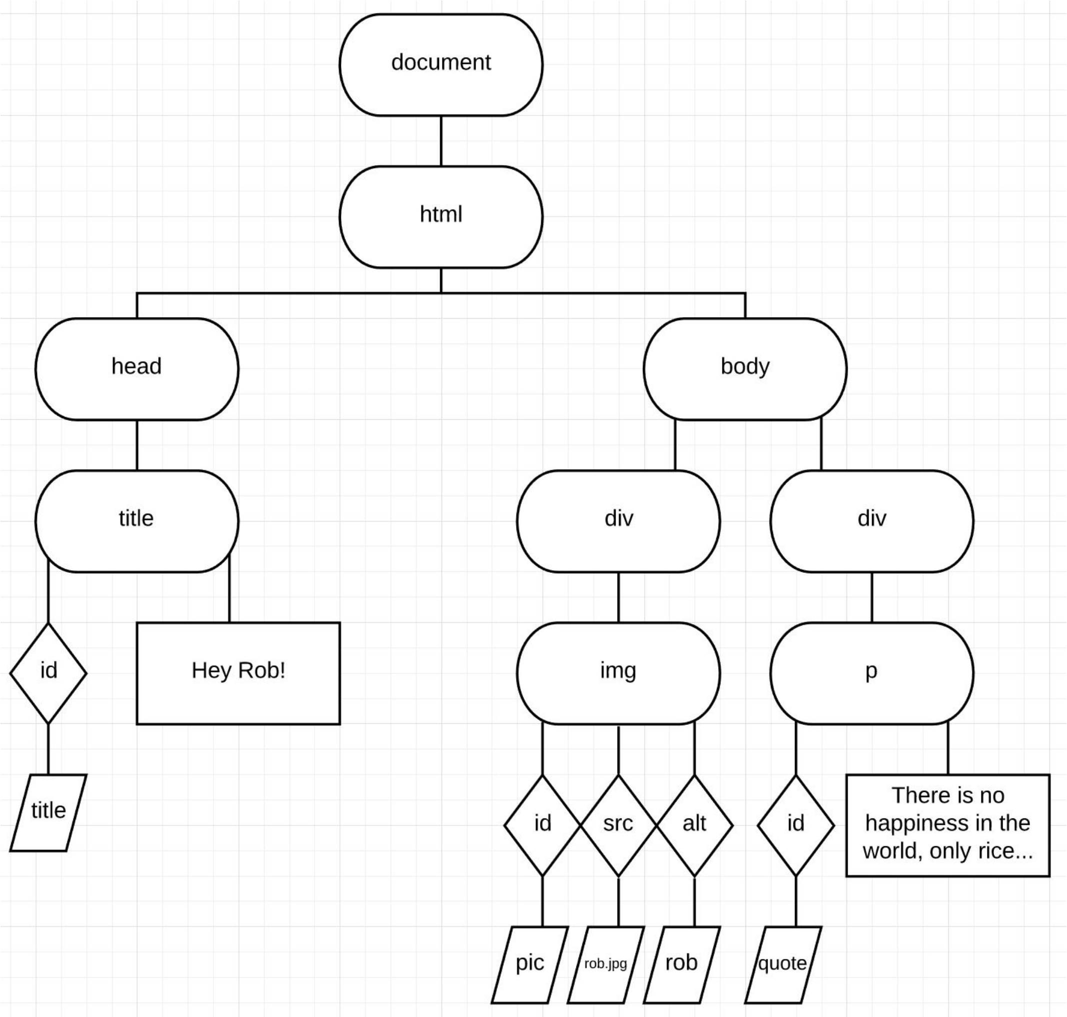
```
<html>
    <head>
        <title>Hello, world</title>
    </head>
    <body>
        <h2>Here's my page</h2>
        <p>World, hello</p>
        <a href="test.html">Link</a>
    </body>
</html>
```



```
<html>
    <head>
        <title>Hello, world</title>
    </head>
    <body>
        <h2>Here's my page</h2>
        <p>World, hello</p>
        <a href="test.html">Link</a>
    </body>
</html>
```

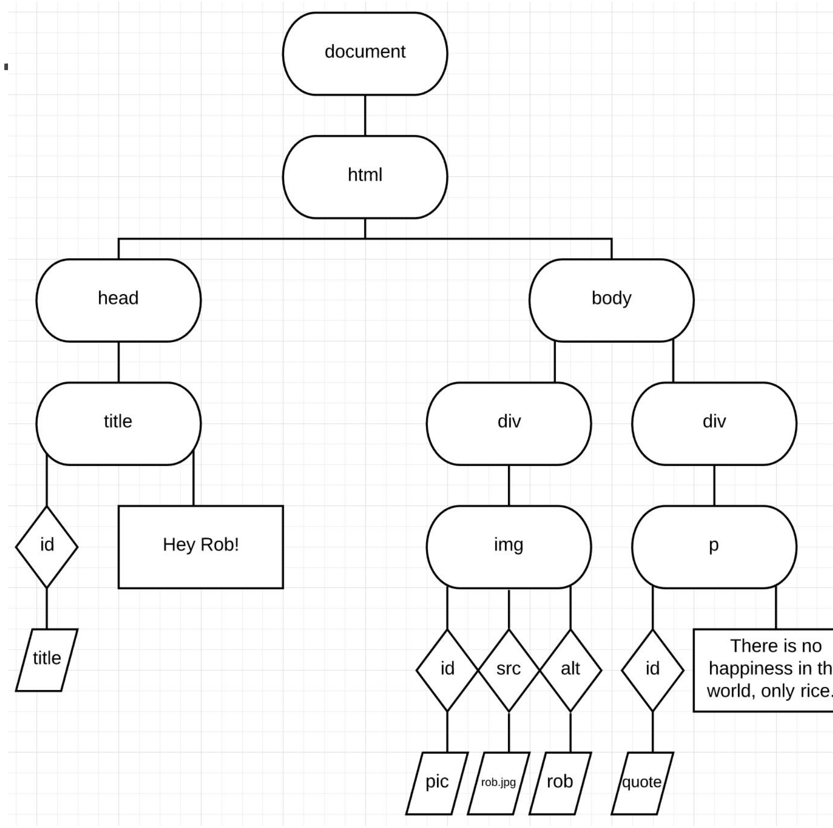
Turn this into DOM

```html
<!DOCTYPE html>
    <head>
        <title id="title">Hey Rob!</title>
    </head>
    <body>
        <div>
            <img id="pic" src="rob.jpg" alt="rob"/>
            <img id="pic2" src="rob.jpg" alt="rob"/>
        </div>
        <div>
            <p id="quote">There is no happiness in the world, only rice...</p>
        </div>
    </body>
</html>
```


document
html
head
body
title
div
div
id
Hey Rob!
img
p
title
id
src
alt
id
There is no happiness in the world, only rice...
pic
rob.jpg
rob
quote

Change the DOM dynamically with

```
<script>
    var title = document.getElementById("title");
    title.innerHTML = "David";

    var pic = document.getElementById("pic");
    pic.src = "david.jpg";

    var alt = document.getElementById("pic");
    alt.alt = "David";

    document.getElementById("quote").innerHTML = "allllright";
</script>
```



DOM: Properties

PROPERTY	DESCRIPTION
`innerHTML`	Holds the HTML inside a set of HTML tags.
`nodeName`	The name of an HTML element or element’s attribute.
`id`	The "`id`" attribute of an HTML element
`parentNode`	A reference to the node one level up in the DOM.
`childNodes`	An array of references to the nodes one level down in the DOM.
`attributes`	An array of attributes of an HTML element.
`style`	An object encapsulating the CSS/HTML styling of an element.

DOM: Methods

METHOD	DESCRIPTION
`getElementById(id)`	Gets the element with a given ID below this point in the DOM.
`getElementsByTagName(tag)`	Gets all elements with the given tag below this point in the DOM.
`appendChild(node)`	Add the given node to the DOM below this point.
`removeChild(node)`	Remove the specified child node from the DOM.

JavaScript: Events

An event in HTML and JavaScript is a response to user interaction with the web page

JavaScript has support for event handlers, which are callback functions that respond to HTML events

Many HTML elements have support for events as an attribute

```
<html>
       <head>
              <title>Event Handlers</title>
       </head>
       <body>
              <button onclick=”alertName(event)”>Button1</button>
              <button onclick=”alertName(event)”>Button2</button>
       </body>
</html>

function alertName(event)
{
       let trigger = event.srcElement;
       alert(‘You clicked on ‘ + trigger.innerHTML);
}
```

JavaScript: Functions

Declare with `function` keyword

Anonymous functions don’t need names. Particularly for those bound to HTML elements

Dom0

```
<html>
    <head>
        <script>
            function greet()
            {
                alert('hello, ' + document.getElementById('name').value + '!');
            }
        </script>
        <title>dom0</title>
    </head>
    <body>
        <form id="demo" onsubmit="greet(); return false;">
            <input id="name" placeholder="Name" type="text"/>
            <input type="submit"/>
        </form>
    </body>
</html>
```

Let’s Look at an Example

```
practice/templates/button
```

```
practice/templates/abstract_index
```

jQuery

A popular open-source library that is designed to simplify client-side scripting such as DOM manipulations

```
// what is this doing?

    document.getElementById(‘colorDiv’).style.backgroundColor
    = ‘green’;

    vs

    $(‘#colorDiv’).css(‘background-color’, ‘green’);
```

jQuery

Check out: http://api.jquery.com/

But especially these:

- $(document).ready()
- Selecting Elements
- jQuery's Ajax-Related Methods

AJAX (Asynchronous JavaScript and XML)

- A technology that allows browsers to make requests dynamically
- Allows us to create a dynamic webpage, that can get new information, without getting an entire web page again from the server.
- jQuery makes AJAX much simpler with an easy-to-use API that works across a multitude of browsers.

JSON (JavaScript Object Notation)

A format for storing data in a hierarchy, and that looks like this:

```
{
    "name": "Netflix, Inc.",
    "price": 123.30,
    "symbol": "NFLX"
}
```

JSON: Examples

- http://mashup.cs50.net/articles?geo=02138
- http://mashup.cs50.net/articles?geo=06511
- http://mashup.cs50.net/articles?geo=90210
- http://mashup.cs50.net/search?q=02138
- http://mashup.cs50.net/search?q=Cambridge
- http://mashup.cs50.net/search?q=06511
- http://mashup.cs50.net/search?q=New%20Haven

Shorts

- JavaScript
- DOM
- AJAX

Other Resources (THANKS NICHOLAS!)

https://www.w3schools.com/jsref/

https://jsfiddle.net/

Pset 8

Final Pset!