9
Client side JavaScript

What is a browser? What does it do?

- Network communication using HTTP - fetch html documents (pages) from a url, post form content from a page to a server.
- HTML parsing - read the fetched html document and create a data structure representing the document contents - we call this data structure the Document Object Model (DOM).
- Render the document to the browser window. Requires a separate software component called a layout engine.
- Repeat the process all over again when the user clicks on a link.
- The modern web browser is composed of a number of separate software modules (http, layout engine, scripting engine, etc).

All about the DOM

- The root element of the DOM is the document object.
- The DOM can be modeled as a binary tree.
- We can “walk the DOM” by recursively visiting every node in the tree and do something relevant at each node.
- Our JavaScript code will interact with and manipulate parts of the DOM. This manipulation of the DOM is what makes web apps appear more dynamic.
- The DOM is a somewhat arcane beast that has inconsistent methods of access in various browsers. This is one of the motivations for the creation of libraries like jQuery to help programmers write code that works predictably in all the major browsers.

Walking the DOM

```
function walkTheDOM(node, func) {
    func(node);
    node = node.firstChild;
    while (node) {
        walkTheDOM(node, func);
        node = node.nextSibling;
    }
}
```

DOM Events

- Fetch event - resource was downloaded. Timer event - some timer reached zero. UI event - user clicked a button
- There is an event loop inside the browser that triggers these events.
- Events invoke their event handler functions.
- JavaScript code can receive notification when an event has occurred and respond to the event using an event handler function.
- Events can be registered to specific objects in the DOM - buttons, drop-downs, checkboxes, text fields, etc.
- Events are one of the primary ways in which we can make browser based applications interactive and responsive to the user.

HTML

- Not just JS, we have to code in HTML and CSS too!
- Hypertext Markup Language.
- HTML provides the structure to a web page. It can be thought of as the skeleton or house frame of the web page. It provides the rigid structure to the document and the JS and CSS “hang on it” much like tendons, ligaments, and muscles hang on the skeleton and sheet rock and plumbing hang on frames.
- One of the foundations of the web itself, along with HTTP and the browser.
- HTML has been through several revisions, the current version is known as HTML5.
- The HTML specification is governed by the W3C.

CSS - Cascading Style Sheets

- Whereas HTML provides the content and structure of a document, CSS takes care of the presentation side of the document, how it appears in the browser.
- Uses selectors to style individual html elements like body, h1, p, etc.
- Can also define classes by prefixing the name of a class with a . (period)
- Styles can be defined in-line (not recommended), as a <style> block inside of an html document, or in a separate .css file that is linked in the html head.
- LESS and SASS are separate languages that compile-down to CSS by a CSS preprocessor. LESS is written in JavaScript, and SASS in Ruby.

Simple CSS example

```
<style>
p {
    text-align: center;
    color: red;
}
</style>
```

Classes and Id’s

- An html element can be tagged with multiple classes with each class name separated by a space. Ex: `<div class=”nav navbar navbar-reversed”>`
- An id can be assigned to only one element in the DOM. So something with an id is unique to that document.
- This use of class and id inside of the div tag is called an HTML attribute.
- Often it is desirable to perform some action to all elements of a certain class, or to get a single element based on its id. There is even a built-in function for doing so - document.getElementById(id)

document.getElementById

```
function getElementById(id) {
    walkTheDOM(document.body, function(node) {
        if (node.id === id) {
            return node;
        }
    });
}
```

document.getElementsByClassName

```
function getElementsByClassName(className) {
    var results = [];
    walkTheDOM(document.body, function(node) {
        var a, c = node.className, i;
        if (c) {
            a = c.split(' ');
            for (i = 0; i < a.length; i += 1) {
                if (a[i] ==  className) {
                    results.push(node);
                    break;
                }
            }
        }
    });
    return results;
}
```

jQuery

- Cross-platform JavaScript library
- Written by John Resig back in 2006, really took off around 2009
- Designed to simplify writing client-side JavaScript applications
- DOM element selection, traversal, and manipulation
- It is very helpful to know code will work the same across browsers
- Somewhat superseded by newer frameworks like Angular and React

jQuery example of utility functions

```
$.each([1,2,3], function() {
  console.log(this + 1);
});
```

jQuery Example of an Ajax function

```
$.ajax({
  type: 'POST',
  url: '/process/submit.php',
  data: {
    name : 'John',
    location : 'Boston',
  },
}).done(function(msg) {
  alert('Data Saved: ' + msg);
}).fail(function(xmlHttpRequest, statusText, errorThrown) {
  alert(
    'Your form submission failed.\n\n'
      + 'XML Http Request: ' + JSON.stringify(xmlHttpRequest)
      + ',\nStatus Text: ' + statusText
      + ',\nError Thrown: ' + errorThrown);
});
```

Additional Resources

- CodePen - http://codepen.io/
- HTML validation tool - http://validator.w3.org/
- Codecademy - https://www.codecademy.com/catalog/language/javascript
- Codeschool - https://www.codeschool.com/learn/javascript
- Tom Wilson's JavaScript workshop on 3/17/17 - https://www.youtube.com/watch?v=ArdT3_mz73U
- Douglas Crockford in depth DOM talk - https://www.youtube.com/watch?v=Y2Y0U-2qJMs