DOMinating the Web

Eric Wood

October 9, 2012

A brief history of web development

- (1989) Tim Berners-Lee comes up with the idea of HTML for use at CERN and basically gives birth to the modern interwebs
- (1993) First HTML spec defined by the Internet Engineering TaskForce (IETF)
- (1994) The World Wide Web Consortium (W3C) is formed by Berners-Lee.
- (1994) Tim Berners-Lee names himself "Supreme Internet Overlord"
- (present) Everything is a mess, but things are looking up!

ALL HAIL THE SUPREME INTERNET OVERLORD!

Web pages: what are they made of?

HTML - HyperText Markup Language
: The page's content and basic structure

CSS - Cascading Style Sheets
: Controls the page's style and presentation

JavaScript
: Scripting language for manipulating the above, amongst other things...

The Basics

- Collection of "tags"
  - this text is now bold LOL
- Tags can be nested
  - this text is now bold *and* italic ... whoa...
- Tags can have attributes
  - This is an "anchor" tag

The Basics

At some point in the evening I mentioned that it was sad that Lynx was not going to be able to display many of the HTML extensions that we were proposing, I also pointed out that the only text style that Lynx could exploit given its environment was blinking text. We had a pretty good laugh at the thought of blinking text, and talked about blinking this and that and how absurd the whole thing would be. [...] Saturday morning rolled around and I headed into the office only to find what else but, blinking text. It was on the screen blinking in all its glory, and in the browser. How could this be, you might ask? It turns out that one of the engineers liked my idea so much that he left the bar sometime past midnight, returned to the office and implemented the blink tag overnight. He was still there in the morning and quite proud of it.

- Lou Montulli (creator of Lynx)

Important Attributes and Tags

- and tags are for dividing up portions of the document (more on this later)
- The ID attribute gives a tag a unique identifier to make it easy to find:
- The class attribute identifies multiple tags:

The Document Object Model (DOM)

- All of these tags form a tree structure called the DOM tree
- The root node is considered the "document"
- JavaScript traverses and modifies nodes in the DOM tree
- Note that the DOM is an abstraction that applies to other markup languages as well (XML, SGML, etc.)

Example HTML Document

```
<html>
  <head>
    <title>Example</title>
    <!-- This is an HTML comment! -->
    <!-- Metadata and other stuff goes here. -->
    <!-- This is where you import external CSS and JS -->
  </head>
  <body>
    <h1>Hello World!</h1>

    This is where content goes. Hooray!
  </body>
</html>
```

Selectors

Selectors are a syntax for selecting nodes in the DOM tree Here's some awesome examples:

- b (all <b> tags)
- .foo (all nodes with class="foo")
- #foo (all nodes with id="foo")
- b.foo (all <b> tags with class="foo")
- #foo b (every <b> tag inside a node with id="foo")
- Enumerating them sucks. Live code??

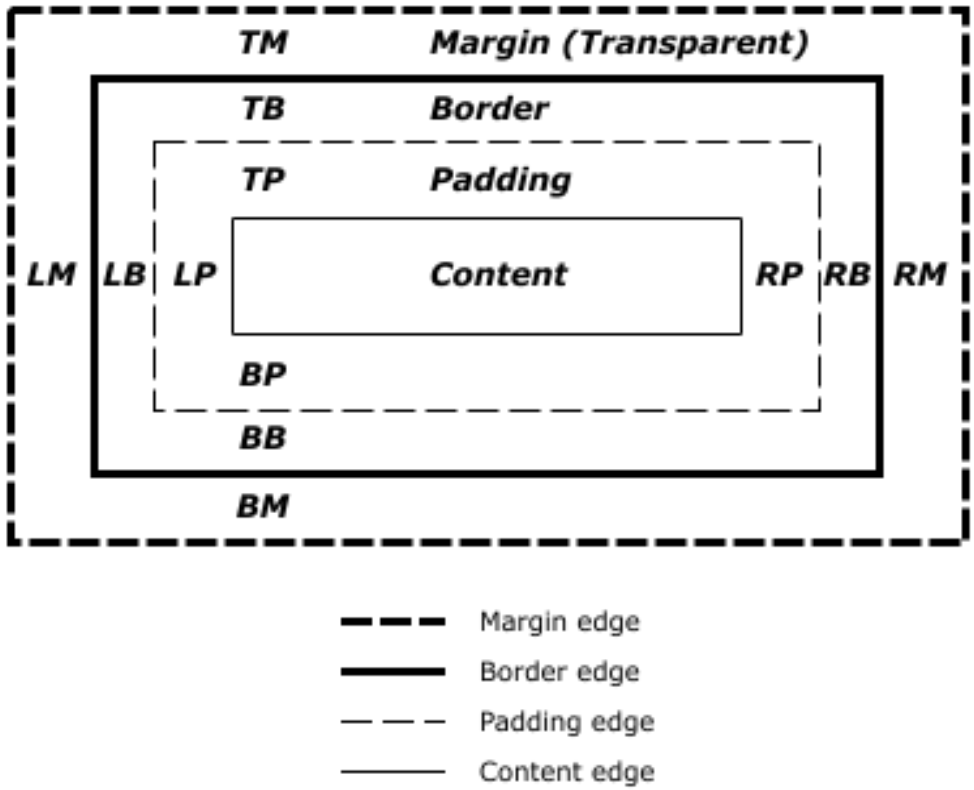
Attributes

- Elements have style attributes which are set via CSS
- Too many to discuss here, but I'll touch on them as I progress
- Here's an example thingy:

```
.foo {
  color: #FFF;
  border: solid #000 1px;
  padding: 5px;
  font-family: Helvetica;
}
```

Positioning and the Box Model

BOW BEFORE THE GLORIOUS BOX MODEL.



Positioning and the Box Model

- Block-level elements
  - , <p>, etc.
  - Elements above and below are on new lines
- Inline elements
  - , <b>, <i>, etc.
  - Appear inline with their surrounding elements; no automatic spacing or whatever

The Cascade

- The "cascade" part of the acronym refers to the way inheritance works
- CSS rules can be defined in several parts of the document and almost always overlap
- So...how exactly do elements inherit their styles?
- Lets discuss the pecking order!

The Cascade

1. Gather all selectors that apply to an element
2. Order them based on the following criteria:
   1. Source (ascending order)
      1. Browser defaults
      2. Author's stylesheets
   2. Specification method
      1. Linked stylesheet
      2. Embedded stylesheet (<style tag)
      3. Inline styles (style attribute on the tag itself)
   3. Element selector specificity
      1. Contextual selector depth
      2. Class
      3. ID
   4. The !important flag
3. Sort by order specified

Putting it all together

Live demo time. Lets build a badass webpage *from scratch* :D

Fundamentals

- JavaScript falls into the following programming language paradigms:
  - Object-oriented (classless aka. prototyped) - not like C++, mmk?
  - Functional (I hope you like anonymous functions!!!!)
  - Imperative (the paradigm you're most familiar with)
- Dynamic typing
- Objects are everything

Objects

- Pretty much everything is an object
- Basically a key-value store (a hash map, if you prefer that term)
- Prototypes
  - More advanced and beyond the scope of this talk
  - You won't use them very often unless you're really awesome
  - These are how inheritance and OOP happens in JS

Functions

- JS functions are *first-class* – they're treated like data (this is the definition of functional programming, omg)
- You can give them names, or define them inline as anonymous functions

Messin' with the DOM

- Very simple, but kind of powerful I guess
- The "document" JS object provides a means for accessing attributes and modifying the DOM tree
- document.getElementById()
- document.getElementsByClassName()
- The Element object provides access to tag attributes and functions for DOM traversal from that node in the tree
- Complete reference: `https://developer.mozilla.org/en-US/docs/DOM/element`

Caveats

- Lots of manual traversal (lame)
- If I need to modify attributes on a set of nodes I have to iterate over them
- Inconsistent between browsers (this applies to all aspects of web dev, though)

Is there a solution?!

A very emphatic *yes*. jQuery to the rescue!

- jQuery is a JS library that provides a developer-friendly (and cross-browser!) means for interacting with the DOM
- There are many other libraries (Prototype, Mootools, etc.), but I <3 jQuery the most (as do a majority of web devs), so I'll focus on it for the rest of this talk

The jQuery Object

- The $ function is used by jQuery for everything
- We can use the $() function along with CSS selectors to grab elements
  - $('#foo'), etc.
  - jQuery returns the result as an array object with additional functionality; this allows us to modify multiple nodes in a single function call! In the background it takes care of the iteration and DOM API calls for us
- The $() function can also generate new nodes
  - $(' '); returns a brand new tag
- Appending and modifying the DOM tree is easy with wrapper methods like appendTo()

Event Handling

- We can use jQuery event handlers to wrap the normal and less awesome DOM API
- JS is well-suited to event-driven programming in its use of functional programming

Lets try out a few event handlers!

AJAX - the buzzword we hate to love

- AJAX is a stupid acronym for Asynchronous JavaScript And XML
- It has nothing to do with XML, but it *is* asynchronous
- The XMLHttpRequest function allows us to retrieve content from a remote server
  - The content must come from the same server the page was loaded from for security reasons (same-origin policy)
- People usually prefer JSON (JavaScript Object Notation) over XML since it's more complex and is actually valid JS code

AJAX with XMLHttpRequest Example

```
var request = new XMLHttpRequest();
request.open('GET', 'http://www.mozilla.org', false);
request.send();

if (request.status === 200) {
  console.log(request.responseText);
}
```

AJAX in jQuery

- $.ajax() is a handy function that wraps XMLHttpRequest in a friendly manner
- Reduces the complexity of handling errors with XMLHttpRequest
- Automatically parse JSON securely

Putting it all together

Example time!

Useful resources

- Mozilla Developer Network - the everything reference
- quirksmode.org - the most comprehensive browser compatibility tables in the universe
- W3C Standards - the *official* specifications for HTML, CSS, and JavaScript