Browsers: They're The BOM Dot Com!

If the Internet exists, but there's no way to browse it, does it even really exist? Unless you've been living under a rock for the past couple decades, you should know what a browser is, and you probably use it multiple times a day. Browsers are something most people take for granted, but behind the scenes is a complex structure working to display information to users who browse the Web.

Web developers rely on browsers constantly. They can be your best friend or your worst enemy. (*Yes, we're looking at you, IE!*) Spending some time learning about browsers will help you get a higher-level understanding of how the Web operates, how to debug, and how to write code that works across browsers. In this reading, we'll learn about the BOM (Browser Object Model), how it's structured, and how it differs from the DOM (Document Object Model).

The DOM vs. the BOM

By now, you've learned about the **DOM, or Document Object Model**, and that it contains a collection of nodes (HTML elements), that can be accessed and manipulated. In essence, the `document` object is a Web page, and the DOM represents the object hierarchy of that document.

How do we access a document on the Web? Through a browser, of course! If we took a bird's-eye view of the browser, we would see that the document object is part of a hierarchy of browser objects. This hierarchy is known as the **BOM, or Browser Object Model**.

The chief browser object is the `window` object, which contains properties and methods we can use to access different objects within the window. These include:

- `window.navigator`
  - Returns a reference to the navigator object.
- `window.screen`
  - Returns a reference to the screen object associated with the window.
- `window.history`
  - Returns a reference to the history object.
- `window.location`
  - Gets/sets the location, or current URL, of the window object.
- `window.document`, which can be shortened to just `document`
  - Returns a reference to the document that the window contains.

Note how we can shorten `window.document` to `document`. For example, the `document` in `document.getElementById('id')` actually refers to `window.document`. All of the methods above can be shortened in the same way.

The browser diagram

We started in the DOM, and we stepped outside it into the BOM. Now, let's take an even higher view of the browser itself. Take a look at this diagram depicting a high-level structure of the browser, from html5rocks.com:

- **User interface**: This is the browser interface that users interact with, including the address bar, back and forward buttons, bookmarks menu, etc. It includes everything except for the requested page content.
- **Browser engine**: Manages the interactions between the UI and the rendering engine.
- **Rendering engine**: Displays, or renders, the requested page content. If the requested content is HTML, it will parse HTML and CSS and render the parsed content.
- **Networking**: Handles network calls, such as HTTP requests.
- **Javascript interpreter**: Parses and executes JavaScript code.
- **UI backend**: Used for drawing basic widgets like combo boxes and windows; uses operating system user interface methods.
- **Data storage**: The persistence of data stored in the browser, such as cookies.

What we learned:

- Review of the DOM, or Document Object Model
- How the BOM differs from the DOM
- The window object and related methods