Ch. 9 the Window Object

- The window object is identified as the global object of JavaScript, and represents the browser window containing the web page.
- Both in this textbook and in videos that we've watched, pop-ups are highly discouraged. Why then are they so common still? Is it due to pressure from business executives? It just seems strange that I still see so much of something that is so obviously frustrating.
- Window.location seems worth looking into, seeing as it can be used to change the current URL.
- I need to remember that cookies are covered in this chapter, as they will likely be used in this class.
    - Allows me to store data on a user across multiple different sessions of a browser.

Content Template Element

- Essentially, templates represent sections of HTML that can be inserted into the document after it has initially been rendered. One example from the website was the use of templates to add rows to an HTML table.
- Note the browser compatibility at the bottom of the page.

Ch. 10 Testing and Debugging

- Types of error:
    - System
    - Programmer – our primary responsibility
    - User – arguably also a programmer error, in a roundabout way
- While code is being developed, errors should be "loud" and easy to spot. Once the code has been deployed, these errors should become more graceful.
- It is often worthwhile to develop JavaScript using "strict" mode, which is used by putting the following line at the top of a .js file:
    - 'use strict';
    - Note that this can also be done at the top of a function instead, if necessary.
- The "alert()" function can be used to simulate having a breakpoint in the code, as it will halt execution until the dialog is acknowledged.
    - However, they are no longer the recommended tool for debugging.
- The "debugger" command works well in tangent with most browser tools, as long as you remember to **remove all references to debugger statements *before* shipping.**
- Throw / catch / finally statements use the following syntax:
    - function imaginarySquareRoot(number) {

```
'use strict';
let answer;
try {
  answer = String(squareRoot(number));
```

```
    } catch(error) {
      answer = squareRoot(-number)+"i";
    } finally {
      return `+ or - ${answer}`;
    }
  }
```

- TDD through Jest seems to be a reliable method for developing concrete code.

Single-Page Applications

- I'll need to spend some more time with this, as the concept is somewhat difficult for me to wrap my head around. That being said, the main point of this article seems to be that this is a good way to make the webpage run more efficiently.

Problem with Single Page Apps

- This article takes the time to immediately bring my previous statement into question. From the sounds of it, using baked in features and following best practices is both less complex and just as efficient. These two methods seem worth discussing with my team, as we'll have to decide what to do assuming it isn't already defined for us.

The Template Element

- Like the other articles, this is just another possible method for writing more efficient code.