Testing and Debugging

JavaScript can be placed into Strict mode using “use strict;” for either the whole file or just a function.

```
‘use strict’;
```

Beware that this can impact the code executing on frameworks like node.js and Angular.

Linting tools ensure that your code applies good practices and conventions, ensure that you apply the correct lint setup and keep to it!

The console and browser built-in debugging tool can be used in interactively find and fix bugs in code. Know how to use the various features in your preferred browser.

Error objects are created when Exceptions are thrown; Exceptions can be deliberately thrown using the **throw** command. Any code placed within a try block experiencing an exception can be caught using the **catch** block. The **finally** block will always execute.

Test Driven Design (TDD), applies best practices to ensure that code is consistent with expectations. First, write failing tests, then code that passes the tests, then refactoring of the code when new features are added to ensure consistency of tests continuing to pass. Keeping code and test scripts in separate files can be challenging to maintain, keep to a well understood and implemented test standard / framework.

Error Objects can be created, using:

const error = new Error();

Important Error Objects are:

- EvalError
- RangeError
- ReferenceError
- SyntaxError
- TypeError
- URIError
- InternalError

const error = new TypeError(‘You need to use a number in this function’);

REMOVE Debugging Code Prior to Shipping Code,

Using the **debugger;** (browser) automatically invoke the breakpoint so that executing code is paused, allowing you to trace the variable states. While the **alert;** provides a useful pause and display confirmation process, but is discouraged in development as too heavy handed.

Feature Detection can be used to determine if a browser supports new API’s, consider using Modernizer.