W05 - Testing and Debugging

Identify and Deal with errors quickly.

- Errors, exceptions, and warnings
- Importance of testing and debugging
- strict mode
- Debugging in browser
- Error Objects
- Throwing Exceptions
- Exception Handling
- Testing Frameworks

ERRORS, EXCEPTIONS, AND WARNINGS

- USUAL CAUSES OF ERRORS -
  - System error - system or external devices interaction with program
  - Programmer Error - incorrect syntax or faulty logic
  - User Error - data entered in format unfamiliar to program

→ Attempt to reduce impact of errors cannot control, avoid causing errors in code, try to predict user errors and deal with them

Exceptions - an error that returns a value useful for dealing with the error.

Stack Traces - a sequence of functions or method calls leading to the point the error originated.

Warnings - code errors aren't enough to cause a crash.

IMPORTANCE OF TESTING AND DEBUGGING

- JS still fails silently many times (causes me frustration)
- work to make failure more obvious ← LOUD!

Strict Mode - Produces more exceptions and warnings. Prohibits use of some depricated features.

W05 - Testing and Debugging

Strict mode simply requires a single line of code at the beginning of a js file: 'use strict';
- can also use strict mode per function
- use strict mode in a self invoking function.

Linting Tools - JS Lint, JS Hint, ES Lint ← use to test quality of JavaScript code.
http://jslint.com http://jshint.com http://eslint.org

FEATURE DETECTION - done using an if statement
ie. using a feature that may not be supported by placing calls to that feature in an if block:

```
if (window.holoDeck) {
    virtualReality.activate();
}
```

https://modernizr.com/docs http://caniuse.com

DEBUGGING IN THE BROWSER - create break points to halt program in certain place that allow closer inspection.

The Trusty Alert - use alert() to show a dialog at certain points of the code.
- Alert stops the program from running until ok is checked.

Using console - use to log information
- console.log() - can use within code blocks to log variables being used
- console.trace() - log an interactive stack trace in console

W05 - Testing and Debugging

Debugging tools - most major browsers have a debugging tool.

Keyword debugger pauses execution of code
- Remember to remove debugger command before shipping

Error Objects

```
const error = new Error( );
```

takes a parameter that is used as error message

1. EvalError not currently used
2. RangeError number is outside allowable range
3. ReferenceError reference is made to an item that does not exist.
4. SyntaxError error in code
5. TypeError value type does not match (string → number)
6. URIError problem encoding or decoding the URI.
7. InternalError non-standard error when error occurs in JavaScript engine.

```
const error = new TypeError('You need to use numbers in this function');
```

Safe error properties to use

'name' returns name of the error constructor function

'message' description - used as argument to the error constructor function

'stack' returns a stack trace - not safe to use in production sites

W05 - Testing and Debugging

Throwing Exceptions - throw statements - highlight and deal with problems

```
throw new Error('I've given 'er all she's got, cap'n!')
```

imaginary numbers are unsupported in JavaScript, so Math.sqrt() returns NaN on negative numbers:

```
function squareRoot(number) {
  'use strict';
  if (number < 0) {
    throw new RangeError('You cannot find the
      square root of a negative number')
  }
  return Math.sqrt(number);
};
```

Exception Handling - in production, we want to handle errors gracefully, responding in a way that enhances the UX.

try, catch, finally - wrap possible error block in a try block.

TESTS - important aspect of programming, good tests will strengthen code as it develops

test squareRoot function with a simple function

```
function isSquareRoots4() {
  return squareRoot(4) === 2;   << will return true
}
```

W05 - Testing and Debugging

TEST-DRIVEN DEVELOPMENT - writing tests before any actual code.

1. Write tests (initially fail)
2. Write code to pass the tests.
3. Refactor Code
4. Test refactored code
5. Write more tests for new features

RED - GREEN - REFACTOR

TESTING FRAMEWORKS - Provide a structure to write meaningful tests and run them.

Jest - in terminal type: npm install -g jest

Crunching some numbers