Chapter 10: Testing and Debugging

Errors, Exceptions, & Warnings

Error are caused when something goes wrong in a program
- System errors – there's a problem w/ the system or external devices w/ which the program is interacting
- Programmer error - the program contains incorrect syntax or faulty logic; it could even be as simple as a typo
- User error - the user has entered data incorrectly, which the program is unable to handle

Programmer errors are our responsibility
- limit user errors by predicting any possible interactions that may throw an error

Exceptions
- an error that produces a return value that can then be used by the program to deal w/ the error

Stack Traces
- exception will also produce a stack trace
- a sequence of functions or method calls that lead to the point where the error occurred
- will work backwards from the point @ which the error occurred to identify the original function or method that started the sequence

Warnings
- warnings & exceptions are presented differently in various environments
- when a runtime error occurs in the browser, the HTML will still appear, but the JS code will stop working in the background

The Importance of Testing and Debugging
- JS is a fairly forgiving language when it comes to errors
- Failing silently makes errors difficult to spot & longer to track down
- programmers should ensure that the code they write fails loudly in development
- programmer should try to make the code fail gracefully

Strict Mode
- produces more exceptions & warnings & prohibits the use of some 'deprecated features'
- can also help improve its clarity & speed
- not using strict mode is often referred to as 'sloppy mode'
- encourages a better quality of JS
- simply requires the following string to be added to the first line of a JS file:

'use strict';
- you can even use strict mode on a per-function basis by adding the line inside a function. Strict mode will then only be applied to any thing inside that function:

```
function strictly(){
'use strict';
//function code goes here
}
```

Linting Tools

- such as JS Lint, JS Hint, & ES Lint
- can be used to test the quality of JS code, beyond simply using strict mode
- highlight any sloppy programming practices or syntax errors
- also useful for enforcing a programming style guide

"Passing a lint test is no guarantee that your code is correct, but it will mean it will be more consistent and less likely to have problems"

Feature Detection

- the recommended way to determine browser support for a feature is to use feature detection
- guarentees that the method is only called if it actually exists & fails gracefully, w/out any exceptions being thrown, if the method doesn't exist
- 'old-school' way of checking for browser support was known as browser sniffing
- using the string returned by window.navigator.userAgent property that we met last chapter to identify the user's browser
- cannot be relied upon to be accurate

Debugging in the Browser

- the process of finding out where bugs occur in the code & then dealing w/ them

- it can be useful to create what are known as breakpoints, which halt the progress of the code and allow us to view the value of different variables at that point in the program.

The Trusty Alert

- most basic form of debugging is to use the alert() method to show a dialog @ certain points in the code

"Using alerts for debugging was the only option in the past, but JavaScript development has progressed since then and their use is discouraged for debugging purposes today."

Using the Console

"One of the most useful commands is the debugger keyword. This will create a breakpoint in your code that will pause the execution of the code and allow you to see where the program is currently up to."

Remove Debugging Code Prior to Shipping

"remove any references to the debugger command before shipping any code"

Error Objects

- error object can be created by the host environment when an exception occurs, or it can be created in the code using a constructor function
- 7 more error objects used for specific errors:

- EvalError is not used in the current ECMAScript specification and only retained for backwards compatibility. It was used to identify errors when using the global eval() function.
- RangeError is thrown when a number is outside an allowable range of values.
- ReferenceError is thrown when a reference is made to an item that doesn't exist. For example, calling a function that hasn't been defined.
- SyntaxError is thrown when there's an error in the code's syntax.
- TypeError is thrown when there's an error in the type of value used; for example, a string is used when a number is expected.
- URIError is thrown when there's a problem encoding or decoding the URI.
- InternalError is a non-standard error that is thrown when an error occurs in the JavaScript engine. A common cause of this is too much recursion."

- All error objects have a number of properties, but they're often used inconsistently across browsers. The only properties that are generally safe to use are:

- The name property returns the name of the error constructor function used as a string, such as 'Error' or 'ReferenceError'.
- The message property returns a description of the error and should be provided as an argument to the Error constructor function.
- The stack property will return a stack trace for that error. This is a non-standard property and it's recommended that it is not safe to use in production sites"

Throwing Exceptions

- also possible to throw your own exceptions using the throw statement

Exception Handling

- when an exception occurs, the program terminates w/ an error message
- it is possible to handle exceptions gracefully by catching the error

try, catch, and finally

"If we suspect a piece of code will result in an exception, we can wrap it in a try block. This will run the code inside the block as normal, but if an exception occurs it will pass the error object that is thrown onto a catch block"

"code inside the catch block will only run if an exception is thrown inside the try block. The error object is automatically passed as a parameter to the catch block. This allows us to query the error name, message and stack properties"

"finally block can be added after a catch block. This will always be executed after the try or catch block, regardless of whether an exception occurred or not"

Tests

- can simply be a function that tests a piece of code runs as it should

Test-driven development

- process of writing tests before any actual code
- write some code to make the tests pass
- code is refactored to make it faster, more readable, and remove any repitition

Workflow:

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

- often referred to as "red-green-refactor"
- TDD mindset can be hard to always use, & @ the end of the day, any tests are better than no tests @ all

Testing Frameworks

- write your own tests
- can be a laborious process
- provides a structure to write meaningful tests & then run them

Jest

- makes it easy to create & run tests by providing helper methods for common test assertions
- install it using npm, enter the following:

```
npm install -g jest
```

"To check everything worked okay, try running the following command to check the version number that has been installed:

```
jest -v
<< v19.0.2
```

"run this test, simply navigate to the folder that contains the file squareRoot.test.js and enter the following command:

```
jest -c {}
```

This will run all files that end in 'test.js' within that folder. The -c {} flag at the end is shorthand for 'configuration'. We don't need any extra configuration, so we simply pass it an empty object.

Chapter Summary

- Bugs are unavoidable in code, and it's best to find them early rather than later.
- JavaScript can be put into strict mode using the string "use strict". This can be used in a whole file or just a single function.
- Linting tools can be used to ensure your code follows good practice and conventions.
- Feature detection can check whether a method is supported before calling it, helping to avoid an exception being thrown.
- The console and browser's built-in debugging tool can be used to interactively find and fix bugs in code.
- Exceptions can be thrown using the throw statement.
- An error object is created when an exception occurs.
- Any code placed inside a try block will pass any error objects to a catch block when an exception occurs. Any code inside a finally block will run if an exception does or does not occur.
- Test-driven development is the practice of writing tests that fail, then writing the code that passes the test, then refactoring the code every time a new feature is implemented.
- The Jest framework can be used to test your code."

Questions:

1. Why is it called Linting?
2. Whats the advantage to using 'strict' in each function instead of in the program as a whole?
3. Should we go ahead and install Jest? Will we be wanting to use this for any projects this semester?