Errors, Exceptions, and Warnings

-System Error, Programmer Error, User Error

-User error could also be considered programmer error. Program for easy user experience.

-Exceptions & Warnings

-Exception - Produces stack trace. Back tracks to try to find exception source.

-Warnings - Not enough to cause program to crash. Could cause future problems.

-Will look different in different browsers

-Runtime error - HTML works, but javascript stops working

-Importance of Testing and Debugging

-Make sure your program fails loudly in test environment to prevent silent errors later

-Strict Mode

-considers previous "poor style" as errors. Helps prevent future errors

- 'use strict'; syntax to run in strict mode. Recommended use is in self invoking function

(not required in modules because they're already in strict mode)

-Linting Tools

Testing tools to highlight sloppy programming. Text editor plug-in

-Feature Detection

-Recommended way to check if a feature is in a specific browser. Use an if statement to see if the feature works before implementing the feature.

-Debugging in the Browser

-create breakpoints

-use alert() to show if something is running. Used to be only option

- use console.log() to show on the console. Use console.trace() to log a stack trace

- use debugger; in code to create breakpoints in certain browsers (always remove!)

-Error Object

-can create own or there's 7 standard ones. Can put string as argument to be message

-Throwing Exceptions

-“throw” statements cause the program to stop when certain things occur. Best practice to throw error object. “throw new Error(“Something went wrong”)

-Exception handling

-try, catch, finally

- if error is thrown in try block, catch block gets whatever error is in it’s parameter and displays what you tell it to instead of crashing or showing error message.

-finally can be added after catch in case something happens that isn’t in the try or catch. Kind of like default.

-Test Driven Development

-writing good tests helps to iron out errors early on.

-write tests first

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

-Use testing frameworks.

-Jest. Made by facebook. Uses expect() function .toBe()

-Book runs many good examples of how to use this process using jest.