- Asynch JS Prerequisites:
  - Computer literacy
  - Reasonable understanding of JS

- Asynch enables a program to start a possible long-running task and still be responsive to other events while that task is running
  - So you do not have to wait until that task has finished
  - Program presents result when task is finished

- Browsers provided functions:
  - HTTP requests with fetch()
  - Asking user to select files showOpenFilePicker()

- Even though may not have to implement own async functions often, need to use them correctly

- Synchronous programming:

```
const name = "Miriam";
const greeting = `Hello, my name is ${name}!`;
console.log(greeting);
// "Hello, my name is Miriam!"
```

  - Declares string called name
  - Declares string called greeting that uses name string
  - Outputs result in JS console

- Browser goes through program one line at a time as it is written
  - Browser waits for each line to finish before moving on to the next line
  - Each line depends on work in the lines before it
- makeGreeting() is a synchronous function as the caller must wait for the function to finish and return a value before the caller can continue

- Long-running synchronous function:
  - Trouble with long-running synchronous function is that they can become unresponsive if the lines do not return a result in the console before moving to the next time
- Start a long-running operation by calling a function.
- Have that function start the operation and return immediately, so that our program can still be responsive to other events
- Notify us with the result of the operation when it eventually completes.

- Event handlers:
  - Form of asynchronous programming
  - Provide a function that will be called when the event happens
  - If the even is the asynchronous operation has completed then that event is used to notify the caller about the result of an async function call

- Callbacks:

- Event handler is a particular type of callback
- Callback is a function that's passed into another function with the expectation that the callback is called at the right time
- Callback based code can be hard to understand when the callback itself has to call functions that accept a callback
- We get deeply nested doOperation() function which is much harder to read and debug
  - Referred to as the "callback hell" or the "pyramid of doom"
  - Hard to handle errors
- Most async APIs don't use callbacks