# Introduction

## What is Node.js?

* A platformer which allows us to run JavaScript on a computer/server
* Read, delete and update files
* Easily communicate with a database

## Why use Node.js?

* It uses JavaScript
* Very fast
* Huge ecosystems of open source packages
* Great for real-time services

## What I need to know?

* HTML
* JavaScript

# The V8 Engine

## JavaScript Engines

* Computers don’t understand JavaScript
* A Javascript engine takes JavaScript, and converts it into something it does understand, like machine code

1. JavaScript
2. C++
3. Assembly Language
4. Machine Code

* Node.js is written in C++
* At the heart of Node.js is the V8 Engine
* The V8 engine converts our JS into machine code

# The Global Object

* Global object - an object that always exists in the global scope. We can use them anywhere in the file.
  + In JavaScript, there's always a global object defined.
  + In a web browser, when scripts create global variables defined with the var keyword, they're created as members of the global object.
  + Global object functions and variables documentation: <https://nodejs.org/dist/latest-v18.x/docs/api/globals.html>
* A global object in node is called a global
  + This allows us to use methods straight out of the box in Node.js

# Function Expressions

* Function Expression - a function that is created as a part of an expression
* Function Declaration - a function that is declared as a separate statement in the main code flow

### Function Expression Example

| var sayBye = function() {  console.log('Bye')  }  sayBye() |
| --- |

### Function Declaration Example

| function sayHi(){  console.log("Hi")  }  sayHi() |
| --- |

### Call Function Example

* A function can be used as an argument/parameter

| function callFunction(func){  func()  }  var sayBye = function() {  console.log('Bye')  }  callFunction(sayBye) |
| --- |

# Modules and require()

* Modules - a software design technique that emphasizes separating the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality
  + As of now all it is is just another JavaScript file
* Require() - is used to load and cache JavaScript modules
  + Reads a JavaScript file, executes it, and then proceeds to return the export object. require() not only allows adding built-in core NodeJS modules but also community-based and local modules.

### Require example

* The require needs a path to the file

| var counter = require('./stream')  console.log(counter(['Dakota', 'Hat', 'Doom', 'Phone', 'JavaScript'])) |
| --- |

### Module example (stream.js)

* Modules.exports is needed for export the function/method

| var counter = function(arr){  return 'There are ' + arr.length + ' elements in this array'  }  module.exports = counter |
| --- |

# Module Patterns