# 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

## Require example

| var stuff = require('./stuff')  console.log(stuff.counter(['Dakota', 'Hat', 'Doom', 'Phone', 'JavaScript']))  console.log(stuff.adder(1, 5))  console.log(stuff.pi)  console.log(stuff.adder(stuff.pi, 5)) |
| --- |

## Export module example 1

| var counter = function(arr){  return 'There are ' + arr.length + ' elements in this array'  }  var adder = function(a, b){  return `The sum of the two numbers is ${a + b}`  }  var pi = 3.14159  module.exports.counter = counter  module.exports.adder = adder  module.exports.pi = pi |
| --- |

## Export module example 2

| module.exports.counter = function(arr){  return 'There are ' + arr.length + ' elements in this array'  }  module.exports.adder = function(a, b){  return `The sum of the two numbers is ${a + b}`  }  module.exports.pi = 3.14159 |
| --- |

## Export module example 3

| var counter = function(arr){  return 'There are ' + arr.length + ' elements in this array'  }  var adder = function(a, b){  return `The sum of the two numbers is ${a + b}`  }  var pi = 3.14159  module.exports = {  counter: counter,  adder: adder,  pi: pi  } |
| --- |

# The Node Event Emitter

## Emitter example 1

| var events = require('events') //events is a core module  //element.on('click', function(){}) //when this element finds this click event the function will activate  var myEmitter = new events.EventEmitter() // this is our own event emitter object, basically event constructor  myEmitter.on('someEvent', function(mssg){ //event and function with parameter  console.log(mssg)  })  myEmitter.emit('someEvent', 'the event was emitted') //event and argument |
| --- |

## Emitter example 2

| var events = require('events') //events is a core module  var util = require('util') //util is a core module  var Person = function(name){ //we want this to inherit the event emitter  this.name = name  }  util.inherits(Person, events.EventEmitter)  var james = new Person('James')  var mary = new Person('Mary')  var ryu = new Person('Ryu')  var people = [james, mary, ryu]  people.forEach(function(person){ //we're taking each person and giving them an event listener  person.on('speak', function(mssg){  console.log(person.name + " said: " + mssg)  })  })  james.emit('speak', 'Hey dudes')  mary.emit('speak', 'What\'s up?')  ryu.emit('speak', 'Chicken sounds good right now') |
| --- |

# Reading & Writing Files (fs)

## Synchronous

### readFileSync

| var fs = require('fs') //fs is a core module  var readMe = fs.readFileSync('files/lorem.txt', 'utf8')  //file location (if it's in the same folder you don't need to specify), character encoding  //this will write the file before it runs any code after  console.log(readMe) |
| --- |

### writeFileSync

| var fs = require('fs') //fs is a core module  var readMe = fs.readFileSync('files/lorem.txt', 'utf8')  //file location (if it's in the same folder you don't need to specify), character encoding  //this will write the file before it runs any code after  fs.writeFileSync('files/new-lorem.txt', readMe)  //file location, what we want to write  //this will write the file before it runs any code after |
| --- |

## Asynchronous

| var fs = require('fs') //fs is a core module  fs.readFile('lorem.txt', 'utf8', function(err, data){  fs.writeFileSync('new-lorem.txt', data)  })  //file location (if it's in the same folder you don't need to specify), character encoding, call back function  //we're not blocking the code, as in the code below will fire while this is reading the file  //we're going to write the file once this is fully read |
| --- |

# Creating and removing Directories

## Deleting files

* I suggest using unlinkSync
* This will only delete if the file exists, if it doesn’t it will throw you an error

| var fs = require('fs') //fs is a core module  fs.unlinkSync('files/new-lorem.txt') |
| --- |

## Make directories sync

| var fs = require('fs') //fs is a core module  fs.mkdirSync('stuff') //this will create a directory named stuff |
| --- |

## Remove directories sync

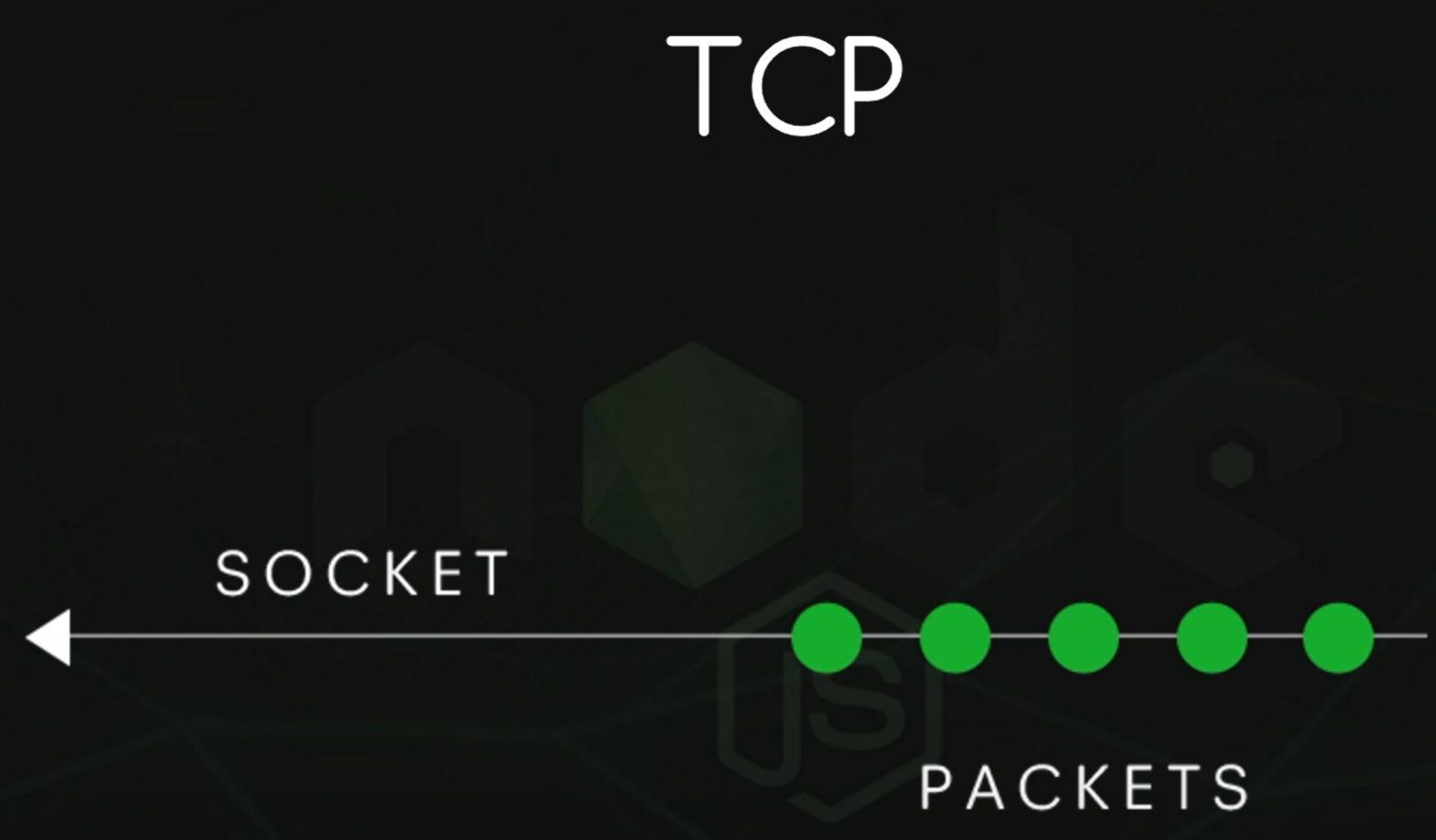
| var fs = require('fs') //fs is a core module  fs.rmdirSync('stuff') //this will remove a directory named stuff |
| --- |

# Clients and Servers



* Protocols - a set of communication rules, that two sides agree to use when communicating
  + The client and the server





* TCP - Transmission Control Protocol (TCP) is a standard that defines how to establish and maintain a network conversation by which applications can exchange data.
  + TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other.
* Ports - a program running on a computer can listen for requests sent to a particular port number
  + E.g. 172.24.86.76:3000