Node-Solid : Solidify Node Skills

Executing Node

Node API

Strems

Asycn Node

Executing Node

Node overview

- The most popular JavaScript runtime to emerge in the last few years
- Node is used by millions of developers to build web applications
  - Twitter, Facebook, Google, Instagram, etc.
- we can build simple web application to large scale application
- using electron we can build desktop application
- node can also be build hardware embedded application
- build apps end-to-end with one language that is JavaScript

background

- A page is a collection of elements that are rendered on the screen.
  - it consists of HTML, CSS, and JavaScript
  - and DATA from the Server side
- some server side languages to serve data
  - PHP, Python, Ruby, Java, etc.
- Node will allows us to write both client and server side code in one language

node.js features

- Node.js has computer features like accessing files and writing to files
    - it can also access the network
    - it can also access the hardware
    - it can also access the database
- these are c++ specific features that interact with os directly
- when JS works with C++ to control the computer features it is called Node.js

JavaScript Node and Computer

JavaScript does three things (1 with c++s help)

1. Saves the data and functionality [numbers, strings, objects]
2. Executes the data and functionality
3. Has a way to communicate with the computer using c++ features

- var, let and const are stored in memory
- function are stored in the heap, heap is a large memory space
    - function have a execution context
        - this is the scope of the function and the variables that are in the function
        - the function execution context is created when the function is called its temporary and destroyed when the function is done

Node API

calling Node in JS

- `http` - is a server side library that allows us to create a server that can be accessed from the internet
- there are 6400 entry points to the http library
- we ask c++ to create a server and then we can access the server from the internet, so we set the port to 80

```
const http = require('http'); // require is a function that is used to import a module
const server = http.createServer(); // create a server
server.listen(80); // listen to port 80
```

Calling methods in node

- if we want to run a function to run we pass it in createServer and then we call the listen method
- `executeFunction(incomingData, setOutGoingData)` a custom function to log to the console

```
const http = require('http'); // require is a function that is used to import a module
const server = http.createServer(executeFunctions); // create a server and execute the logToConsole function
executeFunction(incomingData, setOutGoingData) {
  console.log(incomingData);
  setOutGoingData().end('Hello World'); // end is a node function that sends the data to the client
};

server.listen(80); // listen to port 80
```

calling function under the hood

- node will execute the function and then return the result
- it will also add the arguments to the function
- node automatically create TWO JS OBJECTS FOR US
  - incomingData - the data that is coming in from the client
  - setOutGoingData - the data that is going out to the client
- `end` - is a node method that will send data to node which will then be sent to client
- `setOutGoingData.end('welcome')` - will send the data to the client

Q&A

- Q: how do we know what method to use to send data to the client?
- A: we look up node docs and find the method that is used to send data to the client is `end`
- Q: who do we know we have to use port 80?
- A: 80 is the default port that node will use if we don't specify a port and that is what we are using in the code

Req and Res

- 3 parts of HTTP Req
  - request type - GET, POST, PUT, DELETE
  - headers - meta data about the data that is sent to the server
  - body - the data that is sent to the server

- We have to look into in bound request we have to conditionally answer the request.
- Express is a framework that we can use to create a server that can be accessed from the internet, that helps us set up middle ware that analyzes the inbound request and response
- middle ware patters passes the object through the function and then passes it to the next function to send back the right data
- Express is a web framework that allows us to create a server that can be accessed from the internet
- Express is a node module that we can use to create a server `const express = require('express')`
- it can be used to create a server that can be accessed from the internet.

HTTPRequestObject & HTTPResPonseObject

```
const tweets = [
{ id: 1, text: 'Hello World' },
{ id: 2, text: 'Hello Universe' },
{ id: 3, text: 'Hello Galaxy' },
];
function doThisOnIncomingData(incomingData, functionToSetOutgoingData) {
  const tweet = tweets[incomingData.id];
  functionToSetOutgoingData(tweet);
}
const http = require('http'); // require is a function that is used to import a module
const server = http.createServer(doThisOnIncomingData); // create a server and execute the logToConsole function
server.listen(3000); // listen to port 80
```

Error Handling in Node

- `try` - is a block of code that we want to try to run
- `catch` - is a block of code that we want to run if the try block fails we will change our function `doThisOnIncomingData` only when there is a `request` coming in from the client
- `server.on` - is a method that we can use to listen to a certain event
- events like
    - `request` - is a event that is fired when a request is coming in from the client
    - `clientError` - is a event that is fired when there is an error with the client

error handling code

```
const tweets = [
{ id: 1, text: 'Hello World' },{ id: 2, text: 'Hello Universe' },{ id: 3, text: 'Hello Galaxy' },];
function doThisOnIncomingData(incomingData, functionToSetOutgoingData) {
  const tweet = tweets[incomingData.id];
  functionToSetOutgoingData(tweet);
}
const http = require('http'); // require is a function that is used to import a module
const server = http.createServer();
server.on('request', (req, res) => {
  try {
    doThisOnIncomingData(req, res);
  } catch (error) {
    res.statusCode = 500;
    res.end('Something went wrong');
  }
});
server.on('clientError', (err, socket) => {
  socket.end('HTTP/1.1 400 Bad Request\r\n\r\n');
}
server.listen(3000); // listen to port 80
```

folder structure

- `/` - is the root folder
- `./` - is the current folder
- `./public/img` - current folder and then the folder called public and then the folder called img
- `../` - is the parent folder
- `../../` - is the grandparent folder

JSON object

- JSON is a data format that is used to store data in a file
- `JSON.stringify` - is a method that we can use to convert a JS object to a JSON string
- `JSON.parse` - is a method that we can use to convert a JSON string to a JS object

```
// json object
const jsonObject = {
  name: 'John',
  age: 30,
  isMarried: false,
};
```

the FS module

- `fs` - is a module that we can use to read and write files
- `fs.readFile` - is a method that we can use to read a file
- `fs.readFileSync` - is a method that we can use to read a file synchronously

example of reading tweets.json and writing to output.html

```
const fs = require('fs'); // require is a function that is used to import a module
const tweets = JSON.parse(fs.readFileSync('tweets.json')); // read the file and convert the data to a JS object
const html = tweets.map(tweet => `<li>${tweet.text}</li>`).join(''); // create an array of HTML elements and join them together
fs.writeFileSync('output.html', `<ul>${html}</ul>`); // write the data to a file
```

Callback Stack, Callback Queue, Event Loop

- call Stack - Js keeps track of the functions that are being called and when they are called, whenever a function is called, it is added to the call stack
- call queue - any function that is delayed from running are added to the call back queue, when the background tasks are done, the functions in the call back queue are called
- Event Loop - Determines what function/code to run next from the queue

Node directory access

- `__dirname` - is a variable that is set to the absolute path of the current directory
- `__filename` - is a variable that is set to the absolute path of the current file
- `process.cwd()` - is a method that is used to get the current working directory
- `process.chdir()` - is a method that is used to change the current working directory
- `process.env` - is a variable that is set to the environment variables of the current process
- `process.exit()` - is a method that is used to exit the process

node response methods

- `res.writeHead` - is a method that is used to set the headers of the response
- `res.sendFile` - is a method that is used to send a file to the client
- `res.json` - is a method that is used to send a JSON object to the client
- `res.send` - is a method that is used to send a string to the client
- `res.end` - is a method that is used to end the response