HTML Node Server

One aspect of Web Servers is that they typically serve HTML pages, which is what a user sees when they visit a website.

How do we take the web server we wrote in the previous lesson and have it serve HTML?

```
const http = require('http');

const server = http.createServer((request, response) => {
  response.statusCode = 200;
  response.setHeader('Content-Type', 'text/plain');
  response.end("Hello World");
});

server.listen({ port: 3000, host: 'localhost' }, () => {
  console.log('Up and Running!');
});
```

Well for one thing, we can change the content type response:

```
response.setHeader('Content-Type', 'text/html');
```

We'll also want to actually serve some HTML, not `"Hello World"`.

For a very quick approach we can store a basic web page in a string and then send that back:

```
const HTML = `
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>My Hello World</title>
    <style>
      body {
        background-color: black;
        color: yellow;
        text-align: center;
        font-size: 40px;
      }
    </style>
  </head>
  <body>
    Hello World
  </body>
</html>
`;
```

Then we'll serve the HTML instead of `"Hello World"`:

```
response.end(HTML);
```

You can run this server by once again running `node index` (provided your file is called `index.js`). Go to http://localhost:3000 in your web-browser and voila! You'll see an HTML page with a Yellow Hello World on a black background:

Ok, so it's not the prettiest website in the world

We can work on that later! Let's spruce up our server a bit more.

Serving a HTML File

Rather than serving our HTML file from a string, let's create a separate file to store the HTML! Let's call the file `index.html`, then copy and paste the HTML into the file from our `HTML` string:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>My Hello World</title>
    <style>
      body {
        background-color: black;
        color: yellow;
        text-align: center;
        font-size: 40px;
      }
    </style>
  </head>
  <body>
    Hello World
  </body>
</html>
```

This will make it much easier to modify! Coding text editors will also detect HTML files by the `.html` extension and even highlight the HTML syntax as seen above!

Next we'll modify our server:

```
const http = require('http');
// add the fs library for reading from the file system
const fs = require('fs');

const server = http.createServer((request, response) => {
  // we'll attempt to read from the index.html file
  fs.readFile('index.html', function(err, content) {
    if(err) {
        // readFile will return an error if it was unable to successfully read the content
        // if this happens, let's return an error response back from the server
        response.statusCode = 500;
        response.end("Could not serve index.html");
    }
    else {
        // if there is no error, we'll serve the HTML we read from the file!
        response.statusCode = 200;
        response.setHeader('Content-Type', 'text/html');
        response.end(content);
    }
  });
});
```

We're using the Node.js fs library to read from the file system and retrieve the contents of `index.html`! This makes the code much cleaner and extensible

> You can find the documentation for the method `readFile` in the fs documentation.

Create a CSS file

We can take this one step further by serving a CSS file. We'll take the styling from our `index.html`:

```
body {
  background-color: black;
  color: yellow;
  text-align: center;
```

```
  font-size: 40px;
}
```

Let's move the CSS into it's own file `style.css` !

In the `index.html` , replace the entire `<style>...</style>` node with:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

This `link` element will tell the browser to request a `"style.css"` file from our server. So what's our next step?

We have to **serve the CSS** file from our server of course!

Let's update our server code:

```
const server = http.createServer((request, response) => {
  // by default we'll serve index.html
  let filename = "index.html";
  let contentType = "text/html";
  // if the client is requesting style.css, we'll serve it instead
  if(request.url === "/style.css") {
    filename = "style.css";
    contentType = "text/css";
  }
  fs.readFile(filename, function(err, content) {
    if(err) {
        response.statusCode = 500;
        response.end(`Could not serve ${filename}`);
    }
    else {
        response.statusCode = 200;
        response.setHeader('Content-Type', contentType);
        response.end(content);
    }
  });
});
```

Yikes! This is starting to get a bit messy

At this point, we should be thinking about generalizing this code a bit

- It should know where to find our static assets like images, CSS, JavaScript files, etc..
- It should recognize extensions like `.css` and change the contentType accordingly
- It should know when the client wants the `.index.html` versus another file
- It should also be able to handle requests for creating, retrieving, updating and deleting data

That will take quite a bit of work to do on our own!

Fortunately, there are great Node.js frameworks like Express that help us serve static assets and create routes for data requests.

✓ **Complete**

Next: **Course Completion** >