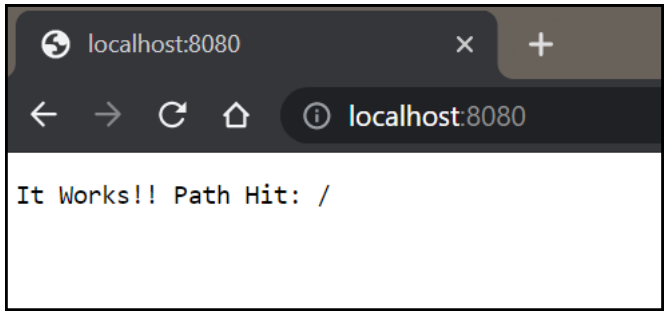
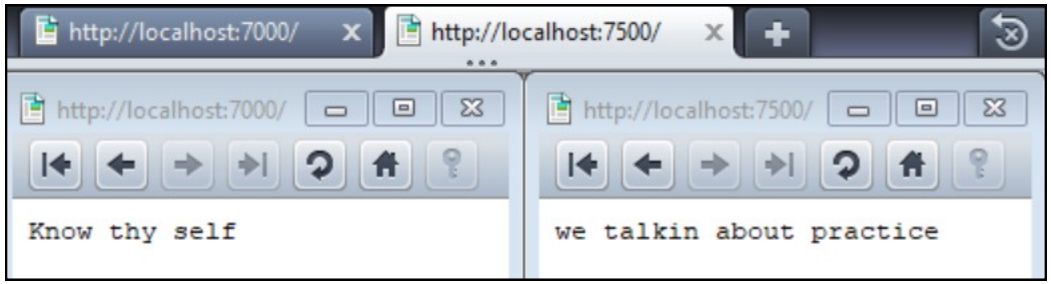
Servers

```
// Require/import the HTTP module
const http = require("http");

// Define a port to listen for incoming requests
const PORT = 8080;

// Create a generic function to handle requests and responses
const handleRequest = (request, response) => {

  // Send the below string to the client when the user visits the PORT URL
  response.end(`It Works!! Path Hit: ${request.url}`);
};

// Use the Node HTTP package to create our server.
// Pass the handleRequest function to empower it with functionality.
const server = http.createServer(handleRequest);

// Start our server so that it can begin listening to client requests.
server.listen(PORT, () => {

  // Log (server-side) when our server has started
  console.log(`Server listening on: http://localhost:${PORT}`);
});
```

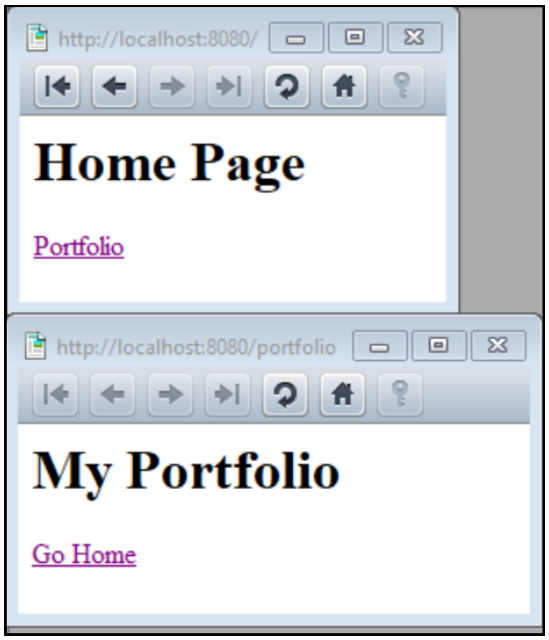


```
const http = require("http");

const server1 = http.createServer((req, res) => {
  res.end("Know thy self");
});

const server2 = http.createServer((req, res) => {
  res.end("we talkin about practice");
});

server1.listen(7000, () => {
  console.log("server1 started");
});

server2.listen(7500, () => {
  console.log("server2 started");
});
```



Portfolio

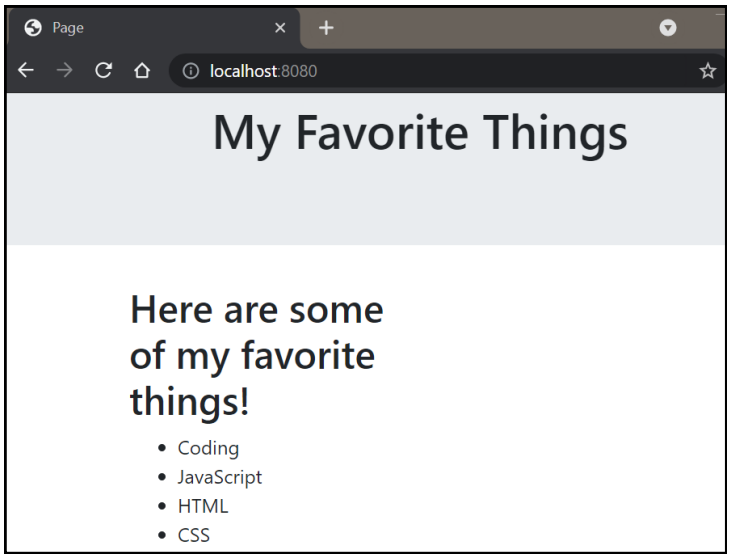
```
const http = require('http');

const PORT = 8080;

// When someone visits the "http://localhost:8080/portfolio" path, this function is run.
const displayPortfolio = (res) => {
  const myHTML = `
  <html>
    <body>
      <h1>My Portfolio</h1>
      <a href='/'>Go Home</a>
    </body>
  </html>`;
  // Configure the response to return a status code of 200 (meaning everything went OK), and to be an HTML document
  res.writeHead(200, { 'Content-Type': 'text/html' });
  // End the response by sending the client the myHTML string (which gets rendered as an HTML document
        thanks to the code above)
  res.end(myHTML);
};

// When someone visits the "http://localhost:8080/" path, this function is run.
const displayRoot = (res) => {
  const myHTML = `
  <html>
    <body>
      <h1>Home Page</h1>
      <a href='/portfolio'>Portfolio</a>
    </body>
  </html>`;
  // Configure the response to return a status code of 200 (meaning everything went OK),
        and to be an HTML document
  res.writeHead(200, { 'Content-Type': 'text/html' });
  // End the response by sending the client the myHTML string (which gets rendered as an HTML document
        thanks to the code above)
  res.end(myHTML);
};

// When someone visits any path that is not specifically defined, this function is run.
const display404 = (url, res) => {
  const myHTML = `
  <html>
    <body>
      <h1>404 Not Found </h1>
      <p>The page you were looking for: ${url} can not be found</p>
    </body>
  </html>`;
  // Configure the response to return a status code of 404 (meaning the page/
resource asked for couldn't be found), and to be an HTML document
  res.writeHead(404, { 'Content-Type': 'text/html' });
  // End the response by sending the client the myHTML string (which gets rendered as an HTML document
        thanks to the code above)
  res.end(myHTML);
};
```




```
// Create a function which handles incoming requests and sends responses
const handleRequest = (req, res) => {
  // Capture the url the request is made to
  const path = req.url;
  // Depending on the URL, display a different HTML file.
  switch (path) {
    case '/':
      return displayRoot(res);
    case '/portfolio':
      return displayPortfolio(res);
    default:
      return display404(path, res);
  }
};

// Assign our createServer method to a variable called "server"
const server = http.createServer(handleRequest);
// Start our server
server.listen(PORT, () => {
  // Callback triggered when server is successfully listening. Hurray!
  console.log(`Server listening on: http://localhost:${PORT}`);
});
```

Serving HTML

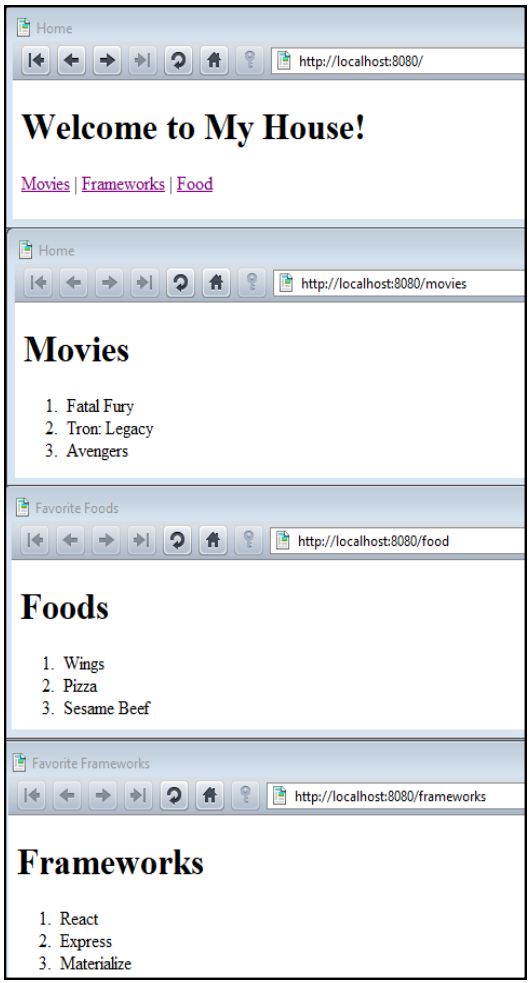
```
<!DOCTYPE html>
<html>
<head>
  <title>Page</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
</head>
<body>
  <div class="jumbotron">
    <h1 class="text-center">My Favorite Things</h1>
  </div>
  <div class="container">
    <div class="col-sm-6 col-sm-offset-3">
      <h2>Here are some of my favorite things!</h2>
      <ul>
        <li>Coding</li>
        <li>JavaScript</li>
        <li>HTML</li>
        <li>CSS</li>
      </ul>
    </div>
  </div>
</body>
</html>
```




```
// Dependencies
const http = require('http');
const fs = require('fs');

// Set our port to 8080
const PORT = 8080;

// Create a function for handling the requests and responses coming into our server
const handleRequest = (req, res) => {
  // Here we use the fs package to read our index.html file
  fs.readFile(`${__dirname}/index.html`, (err, data) => {
    if (err) throw err;
    // We then respond to the client with the HTML page by specifically telling the browser
          that we are delivering an html file.
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(data);
  });
};

// Create our server
const server = http.createServer(handleRequest);

// Starts our server
server.listen(PORT, () => {
  console.log(`Server is listening on PORT: ${PORT}`);
});
```

Serving Favorites

```
// Require dependencies
const http = require('http');
const fs = require('fs');
// Set our port to 8080
const PORT = 8080;

const handleRequest = (req, res) => {
  // Capture the url the request is made to
  const path = req.url;
  // When we visit different urls, read and respond with different files
  switch (path) {
    case '/food':
      return fs.readFile(`${__dirname}/food.html`, (err, data) => {
        if (err) throw err;
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(data);
      });
    case '/movies':
      return fs.readFile(`${__dirname}/movies.html`, (err, data) => {
        if (err) throw err;
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(data);
      });
    case '/frameworks':
      return fs.readFile(`${__dirname}/frameworks.html`, (err, data) => {
        if (err) throw err;
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(data);
      });
    // default to rendering index.html, if none of above cases are hit
    default:
      return fs.readFile(`${__dirname}/index.html`, (err, data) => {
        if (err) throw err;
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(data);
      });
  }
};

// Create the server, assign it to a variable called "server"
const server = http.createServer(handleRequest);
// Starts our server.
server.listen(PORT, () => {
  console.log(`Server is listening on PORT: ${PORT}`);
});
```



Bonus:

```
// Require dependencies
const http = require('http');
const fs = require('fs');
// Set our port to 8080
const PORT = 8080;

// function to take a filepath and respond with html
const renderHTML = (filePath, res) => {
  return fs.readFile(`${__dirname}${filePath}`, (err, data) => {
    if (err) throw err;
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(data);
  });
};

const handleRequest = (req, res) => {
  // Capture the url the request is made to
  const path = req.url;
  // When we visit different urls, call the function with different arguments
  switch (path) {
    case '/food':
    case '/movies':
    case '/frameworks':
      return renderHTML(`${path}.html`, res);
    default:
      return renderHTML('/index.html', res);
  }
};

// Create the server, assign it to a variable called "server"
const server = http.createServer(handleRequest);
// Starts our server
server.listen(PORT, () => {
  console.log(`Server is listening on PORT: ${PORT}`);
});
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
</head>
<body>
  <h1>Welcome to My House!</h1>
  <nav>
    <a href="/movies">Movies</a> |
    <a href="/frameworks">Frameworks</a> |
    <a href="/food">Food</a>
  </nav>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Home</title>
</head>
<body>
  <h1>Movies</h1>
  <ol>
    <li>Fatal Fury</li>
    <li>Tron: Legacy</li>
    <li>Avengers</li>
  </ol>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Favorite Frameworks</title>
</head>
<body>
  <h1>Frameworks</h1>
  <ol>
    <li>React</li>
    <li>Express</li>
    <li>Materialize</li>
  </ol>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Favorite Foods</title>
</head>
<body>
  <h1>Foods</h1>
  <ol>
    <li>Wings</li>
    <li>Pizza</li>
    <li>Sesame Beef</li>
  </ol>
</body>
</html>
```

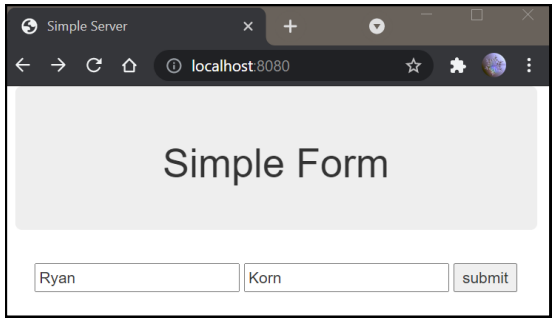
Request Methods

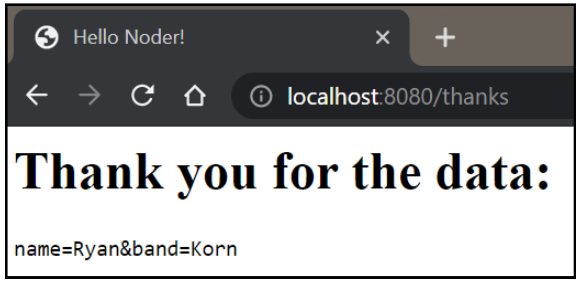
```
// Dependencies
const http = require('http');
const PORT = 8080;

const handleRequest = (req, res) => {
  // Saving the request data as a variable
  let requestData = '';
  // When the server receives data...
  req.on('data', (data) => {
    // Add it to requestData.
    requestData += data;
  });
  // When the request has ended...
  req.on('end', () => {
    // Log (server-side) the request method, as well as the data received!
    console.log(`You did a ${req.method}, with the data:\n, ${requestData}`);
    res.end();
  });
};

// Create the server, assign it to a variable called "server"
const server = http.createServer(handleRequest);

// Start our server
server.listen(PORT, () => {
  console.log(`Server listening on: http://localhost:${PORT}`);
});
```

```
C:\Users\const\Documents\UNCC BootCamp\Class\Week 11\11-express\
01-Activities\06-Request-Methods>node server.js
Server listening on: http://localhost:8080
You did a GET, with the data:
,
```

POST Method

```
// Dependencies
const http = require('http');
const fs = require('fs');
const PORT = 8080;

const renderWelcomePage = (req, res) => {
  fs.readFile(`${__dirname}/index.html`, (err, data) => {
    if (err) {
      res.writeHead(500, { 'Content-Type': 'text/html' });
      res.end(
        '<html><head><title>Oops</title></head><body><h1>Oops, there was an error</h1></html>'
      );
    } else {
      // We then respond to the client with the HTML page by specifically telling the browser that
              we are delivering an html file.
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.end(data);
    }
  });
};
const renderThankYouPage = (req, res) => {
  // Saving the request posted data as a variable.
  let requestData = '';
  let myHTML =
    "<html><head><title>Hello Noder!</title></head><body><h1>Oops, I didn't get any data</h1></body></html>";
  // When the server receives data, it will add it to requestData.
  req.on('data', (data) => {
    requestData += data;
    console.log('You just posted some data to the server:\n', requestData);
    myHTML = `
    <html>
      <head>
        <title>Hello Noder!</title>
      </head>
      <body>
        <h1>Thank you for the data: </h1>
        <code>${requestData}</code>
      </body>
    </html>
    `;
  });
  // When the request has ended...
  req.on('end', () => {
    res.writeHead(200, { 'Content-Type': 'text/html' });
    res.end(myHTML);
  });
};

const handleRequest = (req, res) => {
  const path = req.url;
  switch (path) {
    case '/thanks':
      return renderThankYouPage(req, res);
    default:
      return renderWelcomePage(req, res);
  }
};

// Create the server and assign it to a variable called "server"
const server = http.createServer(handleRequest);
// Starts our server.
server.listen(PORT, () => {
  console.log(`Server listening on: http://localhost:${PORT}`);
});
```





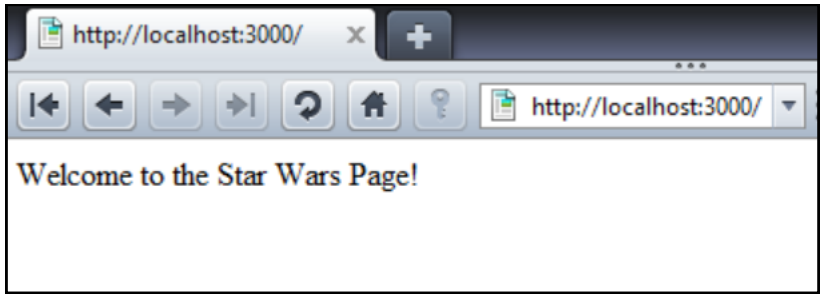

```
C:\Users\const\Documents\UNCC BootCamp\Class\Week 11\11-express\
01-Activities\07-POST-Method\Solved>node server.js
Server listening on: http://localhost:8080
You just posted some data to the server:
 name=Ryan&band=Korn
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Simple Server</title>
  <!-- Latest compiled and minified CSS -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>
  <div class='container'>
    <div class='jumbotron text-center'>
      <h1>Simple Form</h1>
    </div>
    <div class="row">
      <div class="col-sm-12 text-center">
        <form action="http://localhost:8080/thanks" method="POST">
          <input name="name" placeholder="enter your name">
          <input name="band" placeholder="enter your favorite band">
          <button type='submit'>submit</button>
        </form>
      </div>
    </div>
  </div>
</body>
</html>
```

Star Wars Model (pt.1&2)

```
// Dependencies
const express = require('express');
const app = express();
const PORT = 3000;

// Data
const characters = [
  {
    routeName: 'yoda',
    name: 'Yoda',
    role: 'Jedi Master',
    age: 900,
    forcePoints: 2000,
  },
  {
    routeName: 'darthmaul',
    name: 'Darth Maul',
    role: 'Sith Lord',
    age: 200,
    forcePoints: 1200,
  },
  {
    routeName: 'obiwankenobi',
    name: 'Obi Wan Kenobi',
    role: 'Jedi Knight',
    age: 60,
    forcePoints: 1350,
  },
];

// Routes
app.get('/', (req, res) => {
  res.send('Welcome to the Star Wars Page!');
});
app.get('/:character', (req, res) => {
  const chosen = req.params.character;
  // What does this log?
  console.log(chosen);
  res.end();
});

// Listener
app.listen(PORT, () => console.log(`App listening on PORT ${PORT}`));
```

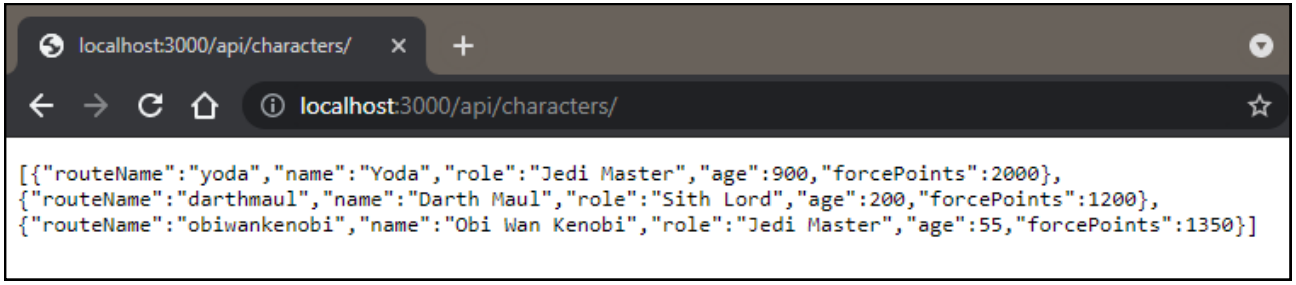
```
C:\Users\const\Documents\UNCC BootCamp\Class\Week 11\11-express\
01-Activities\09-StarWars-2>node server2.js
App listening on PORT 3000
yoda
```




```
// Directed Routes (From Pt. 1)
app.get("/", (req, res) => {
  res.send("Welcome to the Star Wars Page!");
});

app.get("/yoda", (req, res) => {
  res.json(yoda);
});

app.get("/darthmaul", (req, res) => {
  res.json(darthmaul);
});

app.get("/obiwan", (req, res) => {
  res.json(obiwan);
});
```

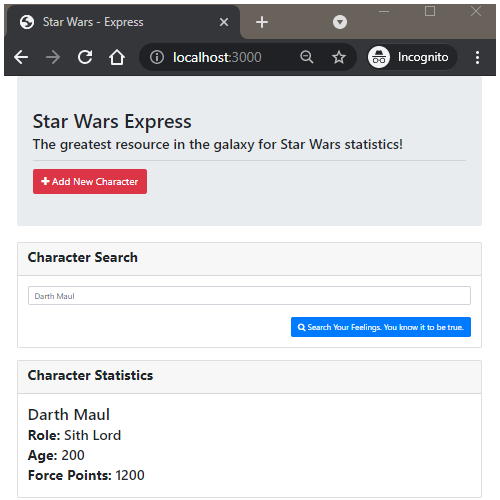
Star Wars Model (pt.3&4)

```
// Dependencies
const express = require("express");
const app = express();
const PORT = 3000;

// Data
const characters = [
  {
    routeName: "yoda",
    name: "Yoda",
    role: "Jedi Master",
    age: 900,
    forcePoints: 2000,
  },
  {
    routeName: "darthmaul",
    name: "Darth Maul",
    role: "Sith Lord",
    age: 200,
    forcePoints: 1200,
  },
  {
    routeName: "obiwankenobi",
    name: "Obi Wan Kenobi",
    role: "Jedi Master",
    age: 55,
    forcePoints: 1350,
  },
];

// Routes
app.get("/", (req, res) => res.send("Welcome to the Star Wars Page!"));

// Displays all characters
app.get("/api/characters", (req, res) => res.json(characters));

// Displays a single character, or shows "No character found"
app.get("/api/characters/:character", (req, res) => {
  // Grab the selected parameter
  const chosen = req.params.character;
  console.log(chosen);
  // Check each character routeName and see if the same as "chosen"
  for (let i = 0; i < characters.length; i++) {
    // If the statement is true, send the character back as JSON,
    if (chosen === characters[i].routeName) {
      return res.json(characters[i]);
    }
  }
  // otherwise send back "false"
  return res.json(false);
});
// Listener
app.listen(PORT, () => {
  console.log(`App listening on PORT ${PORT}`);
});
```



Star Wars Model (pt.5&6)

```
// Dependencies
const express = require('express');
const path = require('path');
// Sets up the Express App
const app = express();
const PORT = 3000;

// Sets up the Express app to handle data parsing
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

// Star Wars Characters (DATA)
const characters = [
  {
    routeName: 'yoda',
    name: 'Yoda',
    role: 'Jedi Master',
    age: 900,
    forcePoints: 2000,
  },
  {
    routeName: 'darthmaul',
    name: 'Darth Maul',
    role: 'Sith Lord',
    age: 200,
    forcePoints: 1200,
  },
  {
    routeName: 'obiwankenobi',
    name: 'Obi Wan Kenobi',
    role: 'Jedi Master',
    age: 55,
    forcePoints: 1350,
  },
];
// Routes
// res.send("Welcome to the Star Wars Page!")
app.get('/', (req, res) => res.sendFile(path.join(__dirname, 'view.html')));

// Displays all characters
app.get('/api/characters', (req, res) => res.json(characters));

// Displays a single character, or returns false
app.get('/api/characters/:character', (req, res) => {
  const chosen = req.params.character;
  console.log(chosen);

  /* Check each character routeName and see if the same as "chosen"
   If the statement is true, send the character back as JSON,
   otherwise send the boolean value false as JSON */
  for (let i = 0; i < characters.length; i++) {
    if (chosen === characters[i].routeName) {
      return res.json(characters[i]);
    }
  }
  return res.json(false);
});

// Create New Characters - takes in JSON input
app.post('/api/characters', (req, res) => {
  // req.body hosts is equal to the JSON post sent from the user
  // This works because of our body parsing middleware
  const newcharacter = req.body;
  console.log(newcharacter);
  // We then add the json the user sent to the character array
  characters.push(newcharacter);
  // We then display the JSON to the users
  res.json(newcharacter);
});

// Starts the server to begin listening
app.listen(PORT, () => console.log(`App listening on PORT ${PORT}`));
```




```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Star Wars - Express</title>
  <!-- Latest compiled and minified CSS & JS -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
        font-awesome/4.7.0/css/font-awesome.min.css" />
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
        bootstrap/4.5.2/css/bootstrap.min.css" />
</head>
<body>
  <div class="container">
    <div class="jumbotron">
      <h1>Star Wars Express</h1>
      <h3>The greatest resource in the galaxy for Star Wars statistics!
          </h3>
      <hr>
      <button class="btn btn-danger btn-lg"><span class="fa fa-plus">
          </span> Add New Character</button>
    </div>
    <div class="row">
      <div class="col-12">
        <div class="card mb-4">
          <div class="card-header">
            <h3><strong>Character Search</strong></h3>
          </div>
          <div class="card-body">
            <input type="text" id="character-search" class="form-control">
            <br>
            <div class="text-right">
              <button type="submit" class="btn btn-primary btn-md"
                  id="search-btn"><span class="fa fa-search"></span>
                Search
                Your Feelings. You know it to be true.</button>
            </div>
          </div>
        </div>

        <div class="card">
          <div class="card-header">
            <h3><strong>Character Statistics</strong></h3>
          </div>
          <div class="card-body">
            <h2 id="name">Yoda</h2>
            <div id="stats" style="display: none;">
              <h3><strong>Role:</strong> <span id="role">Jedi Master
                  </span></h3>
              <h3><strong>Age:</strong> <span id="age">900</span>
                  </h3>
              <h3><strong>Force Points:</strong>
                  <span id="force-points">2000</span></h3>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```
  <script type="text/javascript">
    const searchBtn = document.getElementById('search-btn');
    const statsEl = document.getElementById('stats');
    const nameEl = document.getElementById('name');
    const roleEl = document.getElementById('role');
    const ageEl = document.getElementById('age');
    const fpEl = document.getElementById('force-points');

    // What is this code doing?
    searchBtn.addEventListener('click', () => {
      let searchedCharacter = document
        .getElementById('character-search')
        .value.trim();

        // Using a RegEx Pattern to remove spaces
            from searchedCharacter
        // You can read more about RegEx Patterns later
            https://www.regexbuddy.com/regex.html
      searchedCharacter = searchedCharacter.replace(/\s+/
g, '').toLowerCase();

        // QUESTION: What does the fetch method do, and
            what parameters is it expecting
      fetch(`/api/characters/${searchedCharacter}`, {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json',
        },
      })
        .then((response) => response.json())
        .then((data) => {
          console.log(data);
          const { name, role, age, forcePoints } = data;
          if (data) {
            statsEl.setAttribute('style', 'display: inline');
            nameEl.textContent = name;
            roleEl.textContent = role;
            ageEl.textContent = age;
            fpEl.textContent = forcePoints;
          } else {
            nameEl.textContent =
              'The force is not strong with this one. Your character
                  was not found.';
            statsEl.setAttribute('style', 'display: none');
          }
        })
        .catch((error) => {
          console.error(error);
        });
    });
  </script>
</body>
</html>
```