COMP 307

Principles of Web Development

Lecture 15

Unit 4 – Servers

MERN (part 2) – Intro to React Programming

Contents

Single Page Website
MERN
Web Stack

Class Outline

- Single page websites
- React Programming
- About the website stack

Contents

Single Page Website
MERN
Web Stack

Readings

- MyCourses Resource Folder
  - N/A
- Internet Resources
  - https://reactjs.org/
  - https://www.w3schools.com/react/default.asp
  - https://en.wikipedia.org/wiki/React_(JavaScript_library)

Contents

Single Page Website
MERN
Web Stack

Single Page Websites

MERN (part 2)

Contents

Single Page Website
MERN
Web Stack

Single Page?

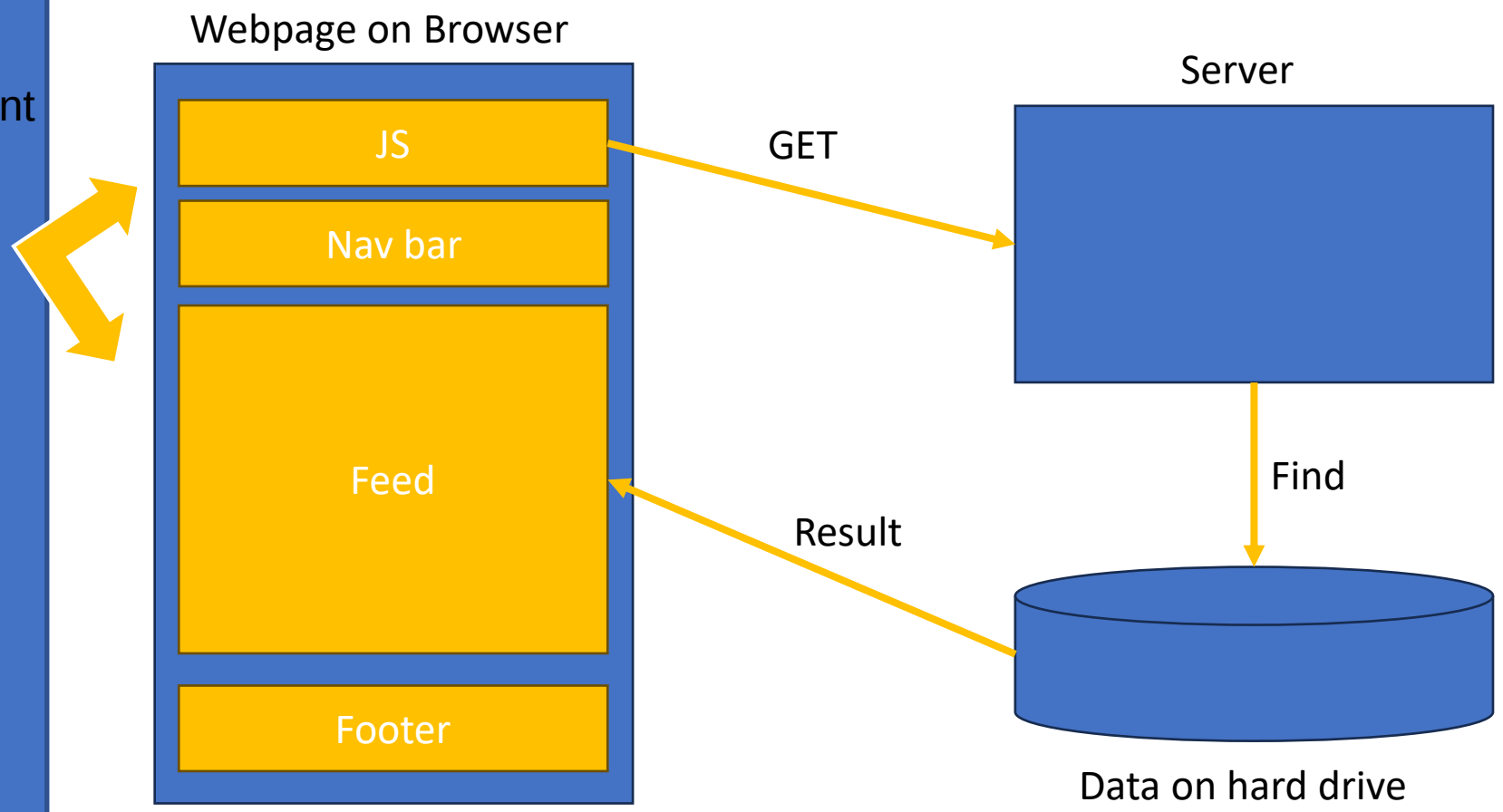
Not to be confused with webpages that are **actually** singly paged.

- https://christophermonnat.com/

Contents

Single Page Website
MERN
Web Stack

What is a single page website?



The user never really needs to go away from the “Feed”. They might switch to other views like “settings” or “groups”, but the design is normally a main feed with supporting popups, navbars, and forms that easily fit on a single screen and single HTML file. All content is loaded from a database.

Contents

Single Page Website
MERN
Web Stack

Non-React Version

Skeleton

```
<html>
    <script>
        function getContentFromDB(URL) {}
        function populateElementByID(ID, URL) {
                document.getElementByID(ID).innerHTML =
                                        getContentFromDB(URL);
        onload() { do initial populateElementByID("abc", "API" }
        }
    </script>
    <body>
        <div class="heading"></div>

        <div class="navbar"></div>

        <div class="feed">
                <element id="abc" onclick="populateElementByID('abc','API')">
        </div>

        <div class="popup" style="hidden" id="def" onclick="….."></div>

        <div class="footer"></div>
    </body>
</html>
```

Access Functions

Hidden elements

Notice that there is little or no content in this webpage, other than heading, nav, and footer. All other content is loaded from the DB at runtime. Popup's is ID'd meaning contents of can change radically.

Contents

Single Page Website
MERN
Web Stack

Examples

- Facebook…
- https://getrepeat.io/
- https://airbnb.io/

hybrid

same interface different content feed

Contents

Single Page Website
MERN
Web Stack

Architecture Implementations

- Apache
  - Browser-side
    - HTML onclick to JS sending REST + JSON or XML
    - Waits for response and modifies DOM only with result
  - Server-side
    - PHP code (or Python, C, Java) find info from DB or file
- Node.JS
  - Browser-side
    - React uses JS to send REST + JSON
    - Waits for response and modifiers DOM only with result
  - Server-side
    - JS running under Node to find info from DB or file

Same technologies, different engines

Contents

Single Page Website
MERN
Web Stack

Introduction to React

MERN (part 2)

Contents

Single Page Website
MERN
Web Stack

About React

maintains 2 DOMs
1 for browser
1 in JS memory

- React uses a Virtual DOM
- It only modifies the real DOM when needed
- Simple React apps can use public libraries
- Production apps must be installed on your computer and server using npm or npx and Node.js

Contents

Single Page Website
MERN
Web Stack

Simple React apps can use public libraries

React in HTML

Opensource libs & frameworks

Babel: JS browser compatibility compiler

```
<!DOCTYPE html>
<html>
 <head>
  <script src="https://unpkg.com/react@18/umd/react.development.js" crossorigin></script>
  <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js" crossorigin></script>
  <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
 </head>

 <body>

  <div id="mydiv"></div>

  <script type="text/babel">
   function Hello() {
    return <h1>Hello World!</h1>;
   }

   const container = document.getElementById('mydiv');
   const root = ReactDOM.createRoot(container);

   root.render(<Hello />)
  </script>

 </body>
</html>
```

What is happening?

Contents

Single Page Website
MERN
Web Stack

need npx + npm

Installing

1. Type: mkdir project, then cd project
2. Type: npx create-react-app my-react-app
3. Type: cd my-react-app
4. Type: npm start

This will cause your browser to launch and display a default app on the browser.

If you see this, then you succeeded.

Contents

Single Page Website
MERN
Web Stack

Next Steps

Now that the app exists. You can look at its code:

1. Type: cd my-react-app
2. Type: dir (Windows) or ls (Mac, Linux)
3. Type: cd src
4. Edit App.js

Next slide shows what you should see

Contents

Single Page Website
MERN
Web Stack

App.js

```
import logo from './logo.svg';
import './App.css';

function App() {
 return (
  <div className="App">
   <header className="App-header">
    <img src={logo} className="App-logo" alt="logo" />
    <p>
     Edit <code>src/App.js</code> and save to reload.
    </p>
    <a
     className="App-link"
     href="https://reactjs.org"
     target="_blank"
     rel="noopener noreferrer"
    >
     Learn React
    </a>
   </header>
  </div>
 );
}

export default App;
```

What is happening?

Contents

Single Page Website
MERN
Web Stack

App.js (version 2)

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
          <div className="App">
              <h1>Hello World!</h1>
          </div>
  );
}

export default App;
```

What does this display?
How does it know to run?

Let's strip the app

Contents

Single Page Website
MERN
Web Stack

Index.js

```
import React from ‘react’;
import ReactDOM from ‘react-dom/client’;
import ‘./index.css’;
import App from ‘./App’;
import reportWebVitals from ‘./reportWebVitals’;

const root = ReactDOM.createRoot(document.getElementById(‘root’));

root.render (
   <React.StrictMode>
      <App />
   </React.StrictMode>
);
```

Connection to real DOM

But where is it?

Contents

Single Page Website
MERN
Web Stack

my-react-app/public/index.html

```
<html>

  <head>
    <title>My App</title>
  </head>

  <body>
    <div id="root"></div>
  </body>

</html>
```

same but stripped

This is the actual webpage displayed on the browser. is populated by React.

Contents

Single Page Website
MERN
Web Stack

Example 1 - Basics

Our React program in a single file:

```
import React from 'react';
import ReactDOM from 'react-dom/client';

const Greeting = () => {
    return (
        <div className="hello-world">
            <h1>Hello, world!</h1>
        </div>
    );
};

const App = () => {
    return <Greeting />;
};

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(
    <React.StrictMode>
        <App />
    </React.StrictMode>
);
```

Notice how XML is used in React:
<App />
<React.StrictMode>
These tags are interpreted by the 'react' library.

Notice "root". This will form the connection to the landing page.

"render" will innerHTML the code into the landing page.

Source: wikipedia

Contents

Single Page Website
MERN
Web Stack

Example 1 - Basics

The end result:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>React App</title>
</head>
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
</html>
```

Inserted here

```
<div class="hello-world">
  <h1>Hello, world!</h1>
</div>
```

Source: wikipedia

Contents

Single Page Website
MERN
Web Stack

How to add React to a webpage?

Example 2 – Adding to HTML

Step 1: Add a id

```
<html>
  <body>
    <div id="myStuff"></div>
  </body>
</html>
```

Step 2: Add the libraries

```
<html>
  <head>
    <!-- Deploying, replace "development.js" with "production.min.js". -->
    <script src=https://unpkg.com/react@18/umd/react.development.js
                                                    crossorigin></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-
                                    dom.development.js" crossorigin></script>
    <script src="https://unpkg.com/@babel/standalone/babel.min.js">
                                                            </script>
  </head>

  <body>
    <div id="myStuff"></div>
  </body>
</html>
```

Contents

Single Page Website
MERN
Web Stack

How to add React to a webpage?

Example 2 – Adding to HTML

Step 3: Add React code (“component”)

```
<html>
  <head>
    <!-- Deploying, replace "development.js" with "production.min.js". -->
    <script src=https://unpkg.com/react@18/umd/react.development.js crossorigin></script>
    <script src=https://unpkg.com/react-dom@18/umd/react-dom.development.js crossorigin></script>
   <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script src="like_button.js"></script>
  </head>

  <body>
    <div id="myStuff"></div>
  </body>
</html>
```

Step 4: Create the file **like_button.js** file

```
<script type="text/babel">
  /* ------------  write your React program here ------------ */
    LikeButton {
          bla
          bla
    }
  /* -------- write the following to find your <div> -------- */

  ReactDOM.render(<LikeButton />, document.getElementById('myStuff'));

</script>
```

Contents

Single Page Website
MERN
Web Stack

Example 2 – Adding to HTML

Step 5: The like_button.js code

```
<script type="text/babel">

'use strict';

const e = React.createElement;

class LikeButton extends React.Component {
  constructor(props) {                          // props accesses "states"
    super(props);
    this.state = { liked: false };
  }

  render() {
    if (this.state.liked) {
      return 'You liked this.';
    }

    return e(
      'button',
      { onClick: () => this.setState({ liked: true }) },
      'Like'
    );
  }
}

ReactDOM.render(<LikeButton />, document.getElementById('myStuff'));

</script>
```

different on his slides

Contents

Single Page Website
MERN
Web Stack

Example 3 – Quiz

Quiz:

- See quiz.html example code
- https://meda.io/embed-react-into-an-html-web-page/

Contents

Single Page Website
MERN
Web Stack

Example 4 – Tic Tac Toe

Tic Tac Toe:

- See “Sample React.txt”
- Copy paste into example 2 to see it run.
- Full game: https://codepen.io/gaearon/pen/LyyXgK?editors=0010

Contents

Single Page Website
MERN
Web Stack

Website Stack

MERN (part 2)

Contents

Single Page Website
MERN
Web Stack

Importance of planning

- Decisions
  - Apache or Node.JS or Django or …
  - Libraries, templates, frameworks, languages
  - Frontend language same as backend language?
  - Mobile focus or platform focus
  - System load
  - Security
  - Number of servers, load balancers, database servers
- Different technology stacks require different types of planning
- Different use-cases require different technology stacks

Contents

Single Page Website
MERN
Web Stack

Importance of planning

- Plan early
- Think it through carefully before coding to foresee issues
- Validate your designs (on paper) with others before starting to code
  - Use techniques that help you design on paper that others can look at and read. This gives them time to consider your ideas and be critical without having to look at code.
  - Helps to come to an agreement.

Contents

Single Page Website
MERN
Web Stack

What goes into a website build

- Frontend:
  - storyboard, wireframes, shape/style/colour, coding the pages/screens, implementing/testing them locally, integrating with the backend.
- Tech stack:
  - frontend language/tools, communication rules/languages/libraries, backend language/tools, security, load balancing, server load and response time, implementation of communication algorithms, installation of server(s). Worst case planning.
- Backend:
  - designing API signatures and database schemas, implementing APIs, implementing database tables/documents, testing that it all works locally, automatic backups, integrating it with the communication libraries, integrating with the frontend.

Contents

Single Page Website
MERN
Web Stack

Steps

- **Zero**: preconditions/requirements for tech/stack, and/or specific performance requirements.
- **First**: storyboard the UI or storyboard the flow if there is no UI.
- **Second**: wireframe the UI & determine the API signatures.
- **Third**: select your stack, libraries, and tools. You are making your big decisions here (note step zero).
- **Fourth**: make a deliverables schedule like a series of assignments with due dates for each team member.
- **Fifth**: Built & test locally first then integrate online iteratively. Repo & wiki are good ideas. Bug sheet too.
- **Sixth**: Exhaustive online real-world testing.

Contents

Single Page Website
MERN
Web Stack

Prepare for Next Class

- Assignments
  - Mini 6 (how's it going?)
  - Project will be out Nov 7th
- Labs this week
  - Lab D – MERN and React.
- Do on your own
  - Install REACT and run the sample program, as shown in class.

Contents

Single Page Website
MERN
Web Stack