HAZZA INSTITUTE OF TECHNOLOGY

iqracity

MERN – ES6 + React

Module 13: React

Outline

Module 13

HAZZA INSTITUTE OF TECHNOLOGY

- HTML Basics
- Explaining Tagging concept
- List creation and display
- Divs and Spans creation
- Attributes and information display

Outline

React

HAZZA INSTITUTE OF TECHNOLOGY

- React Basics
  - React Render
  - React JSX
  - React Components
  - React Props
  - React Events
  - React Conditional Rendering
  - React Lists
  - Styling React Using CSS

React Basics

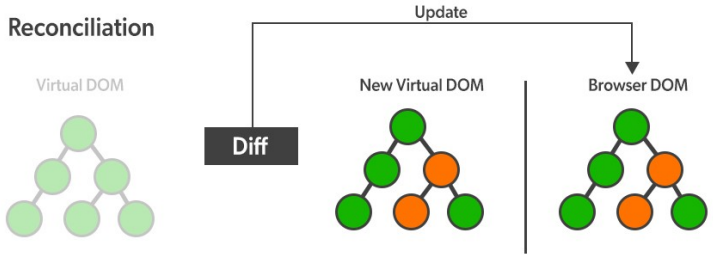
React Introduction

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- What is React?
  - Static vs Dynamic HTML
  - React, sometimes referred to as a frontend JavaScript framework, is a **JavaScript library** created by Facebook.
  - React is a tool for **building UI components**.

- How does React Work?
  - React creates a **VIRTUAL DOM** in memory.
  - React only changes what needs to be changed!



- React.JS History
  - Current version of React.JS is V18.0.0 (April 2022).
  - Initial Release to the Public (V0.3.0) was in July 2013.

React Getting Started

HAZZA INSTITUTE OF TECHNOLOGY

React Basics

- 01-React Directly in HTML

- 02-Setting up a React Environment
  - If you have npx and **Node.js installed**, you can create a React application by using create-react-app.
  - **npx create-react-app my-react-app**
- Run the React Application
  - Run this command to run the React application my-react-app:
  - **npm start**

- React ES6

React Render HTML

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- The Render Function
  - React renders HTML to the web page by using a function called ReactDOM.render().
  - The ReactDOM.render() function takes **two** arguments, HTML code and an HTML element.

Example

Display a paragraph inside an element with the id of "root":

```
ReactDOM.render(<p>Hello</p>, document.getElementById('root'));
```

The result is displayed in the `<div id="root">` element:

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

React JSX

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- What is JSX?
  - JSX stands for JavaScript XML.
  - JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
  - JSX follows XML rules, and therefore HTML elements must be properly closed.

Example 1

JSX:

```
const myElement = <h1>I Love JSX!</h1>;

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

React JSX

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- Expressions in JSX
  - With JSX you can write expressions inside curly braces { }.
  - The expression can be a React **variable**, or **property**, or any other valid **JavaScript expression**. JSX will execute the expression and return the result:

Example

Execute the expression `5 + 5`:

```
const myElement = <h1>React is {5 + 5} times better with JSX</h1>;
```

React JSX

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- Inserting a Large Block of HTML in JSX
  - To write HTML on multiple lines, put the HTML inside parentheses ():

Example

Create a list with three list items:

```
const myElement = (
  <ul>
    <li>Apples</li>
    <li>Bananas</li>
    <li>Cherries</li>
  </ul>
);
```

React JSX

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- Inserting a Large Block of HTML in JSX
  - To write HTML on multiple lines, put the HTML inside parentheses ():
  - **One Top-Level Element**
    - The HTML code must be wrapped in ONE top-level element or fragment <></>.

Example

Wrap two paragraphs inside a fragment:

```
const myElement = (
  <>
    <p>I am a paragraph.</p>
    <p>I am a paragraph too.</p>
  </>
);
```

React JSX

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- Inserting a Large Block of HTML in JSX
  - To write HTML on multiple lines, put the HTML inside parentheses ():
  - **One Top-Level Element**
    - The HTML code must be wrapped in ONE top-level element or fragment <></>.
  - Attribute class = **className**

Example

Use attribute `className` instead of `class` in JSX:

```
const myElement = <h1 className="myclass">Hello World</h1>;
```

React Components

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- What are React Components?
  - Components are like functions that return HTML elements.
  - Class components and **Function components**
- Create Your First Component
  - The component's name MUST start with an **upper case** letter.
  - Class Component

Example

Create a Class component called `Car`

```
class Car extends React.Component {
  render() {
    return <h2>Hi, I am a Car!</h2>;
  }
}
```

React Components

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- What are React Components?
  - Components are like functions that return HTML elements.
  - Class components and **Function components**
- Create Your First Component
  - The component's name MUST start with an **upper case** letter.
  - Class Component
  - Function Component

Example

Create a Function component called `Car`

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}
```

React Components

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- What are React Components?
  - Components are like functions that return HTML elements.
  - Class components and **Function components**
- Create Your First Component
  - Function Component
- Rendering a Component

Example

Create a Function component called `Car`

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}
```

Display the `Car` component in the "root" element:

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

React Components

HAZZA INSTITUTE OF TECHNOLOGY

- Components in Components
  - We can refer to components inside other components:

Example

Use the Car component inside the Garage component:

```
function Car() {
  return <h2>I am a Car!</h2>;
}

function Garage() {
  return (
    <>
      <h1>Who lives in my Garage?</h1>
      <Car />
    </>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```

React Components

HAZZA INSTITUTE OF TECHNOLOGY

React Basics

- Components in Files
  - React is all about re-using code, and it is recommended to split your components into separate files.
  - To do that, create a new file with a .js file extension and put the code inside it:
  - Note that the **filename must start with an uppercase character**.

Example

This is the new file, we named it "Car.js":

```
function Car() {
  return <h2>Hi, I am a Car!</h2>;
}

export default Car;
```

Now we import the "Car.js" file in the application, and we can use the `Car` component as if it was created here.

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

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Car />);
```

React Props

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- What are React Props?
  - props stand for **properties**
  - React Props are like **function arguments** in JavaScript and **attributes** in HTML.
  - To send props into a component, use the same syntax as HTML attributes:

Example

Add a "brand" attribute to the Car element:

```
const myElement = <Car brand="Ford" />;
```

- The component receives the argument as a **props object**:

Use the brand attribute in the component:

```
function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}
```

React Props

HAZZA INSTITUTE OF TECHNOLOGY

React Basics

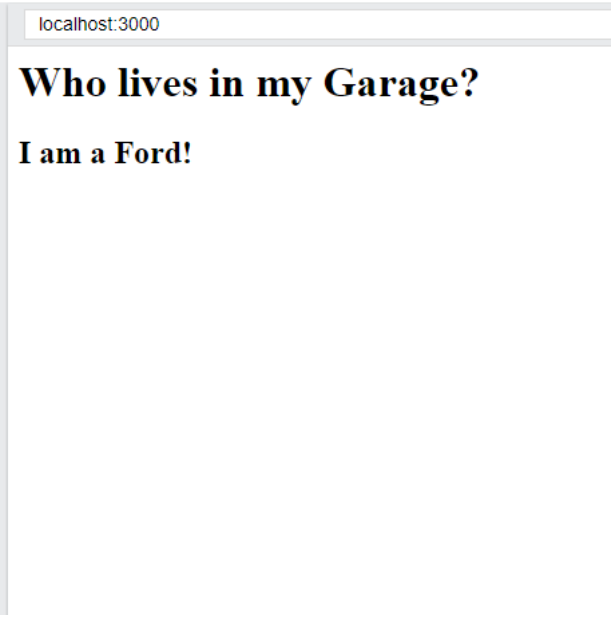
- Pass Data using React Props
  - React Props are like function arguments in JavaScript and attributes in HTML.
  - Props are also how you pass data from one component to another, as parameters.

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

function Car(props) {
  return <h2>I am a { props.brand }!</h2>;
}

function Garage() {
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand="Ford" />
    </>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```



React Props

HAZZA INSTITUTE OF TECHNOLOGY

- Pass Data using React Props
  - If you have a variable/object to send and not a string, just put the variable/object name inside **curly brackets {}**:

Example

Create an object named `carInfo` and send it to the `Car` component:

```
function Car(props) {
  return <h2>I am a { props.brand.model }!</h2>;
}

function Garage() {
  const carInfo = { name: "Ford", model: "Mustang" };
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <Car brand={ carInfo } />
    </>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```

Note: React Props are **read-only**! You will get an error if you try to change their value.

React Events

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- What are React Events?
  - Just like HTML DOM events, React can perform actions based on user events.
  - React has the same events as HTML: click, change, mouseover etc.

- Adding Events
  - React events are written in **camelCase** syntax:
  - onClick instead of onclick
  - React event handlers are written inside **curly braces{}**:
  - onClick={shoot} instead of onClick="shoot()"

React:

```
<button onClick={shoot}>Take the Shot!</button>
```

Tryit Editor

React Conditional Rendering

HAZZA INSTITUTE OF TECHNOLOGY

- In React, you can **conditionally** render components.

▶ if Statement

Example:

We'll use these two components:

```
function MissedGoal() {
  return <h1>MISSED!</h1>;
}

function MadeGoal() {
  return <h1>Goal!</h1>;
}
```

Now, we'll create another component that chooses which component to render based on a condition:

```
function Goal(props) {
  const isGoal = props.isGoal;
  if (isGoal) {
    return <MadeGoal/>;
  }
  return <MissedGoal/>;
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Goal isGoal={false} />);
```

React Conditional Rendering

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- In React, you can **conditionally** render components.

▶ Ternary Operator `condition ? true : false`

Example:

We'll use these two components:

```
function MissedGoal() {
  return <h1>MISSED!</h1>;
}

function MadeGoal() {
  return <h1>Goal!</h1>;
}
```

Return the `MadeGoal` component if `isGoal` is `true`, otherwise return the `MissedGoal` component:

```
function Goal(props) {
  const isGoal = props.isGoal;
  return (
    <>
      { isGoal ? <MadeGoal/> : <MissedGoal/> }
    </>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Goal isGoal={false} />);
```

React Lists

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- In React, you will render lists with some type of loop.
- The JavaScript map() array method is generally the preferred method.

Example:

Let's render all of the cars from our garage:

```
function Car(props) {
  return <li>I am a { props.brand }</li>;
}

function Garage() {
  const cars = ['Ford', 'BMW', 'Audi'];
  return (
    <>
      <h1>Who lives in my garage?</h1>
      <ul>
        {cars.map((car) => <Car brand={car} />)}
      </ul>
    </>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Garage />);
```

Styling React Using CSS

HAZZA INSTITUTE OF TECHNOLOGY

- There are many ways to style React with CSS, this tutorial will take a closer look at three common ways:

- Inline styling
- CSS stylesheets
- CSS Modules

- camelCased Property Names
  - Since the inline CSS is written in a JavaScript object, properties with hyphen separators, like background-color, must be written with camel case syntax:

  - Use backgroundColor instead of background-color

Styling React Using CSS

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- Inline Styling
  - To style an element with the inline style attribute, the value must be a **JavaScript object**:

Example:

Use `backgroundColor` instead of `background-color` :

```
const Header = () => {
  return (
    <>
      <h1 style={{backgroundColor: "lightblue"}}>Hello Style!</h1>
      <p>Add a little style!</p>
    </>
  );
}
```

Note: In JSX, JavaScript expressions are written inside curly braces, and since JavaScript objects also use curly braces, the styling in the example above is written inside two sets of curly braces {{}}

Styling React Using CSS

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- Styling using JavaScript Object
  - You can also create an **object with styling information**, and refer to it in the style attribute:

Example:

Create a style object named `myStyle` :

```
const Header = () => {
  const myStyle = {
    color: "white",
    backgroundColor: "DodgerBlue",
    padding: "10px",
    fontFamily: "Sans-Serif"
  };
  return (
    <>
      <h1 style={myStyle}>Hello Style!</h1>
      <p>Add a little style!</p>
    </>
  );
}
```

Styling React Using CSS

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- CSS Stylesheet
  - You can write your CSS styling in a **separate file**, just save the file with the .css file extension and import it in your application.

App.css:

Create a new file called "App.css" and inser

```css
body {
  background-color: #282c34;
  color: white;
  padding: 40px;
  font-family: Sans-Serif;
  text-align: center;
}
```

index.js:

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import './App.css';

const Header = () => {
  return (
    <>
      <h1>Hello Style!</h1>
      <p>Add a little style!.</p>
    </>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<Header />);
```

Styling React Using CSS

React Basics

HAZZA INSTITUTE OF TECHNOLOGY

- CSS Modules
  - CSS Modules are convenient for components that are placed in separate files.
  - Create the CSS module with the .module.css extension, example: my-style.module.css.

Create a new file called "my-style.module.css"

my-style.module.css:

```
.bigblue {
  color: DodgerBlue;
  padding: 40px;
  font-family: Sans-Serif;
  text-align: center;
}
```

Car.js:

```
import styles from './my-style.module.css';

const Car = () => {
  return <h1 className={styles.bigblue}>Hello Car!</h1>;
}

export default Car;
```

Summary

Module 13

HAZZA INSTITUTE OF TECHNOLOGY

- HTML Basics
- Explaining Tagging concept
- List creation and display
- Divs and Spans creation
- Attributes and information display