React

CS568 – Web Application Development I

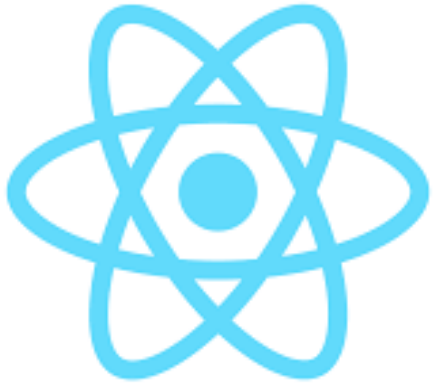
Computer Science Department

Maharishi International University

Maharishi International University - Fairfield, Iowa

Content

- React overview
- Create the first React app
  - App.js
  - Package.json
- React Element
- JSX
- React Component



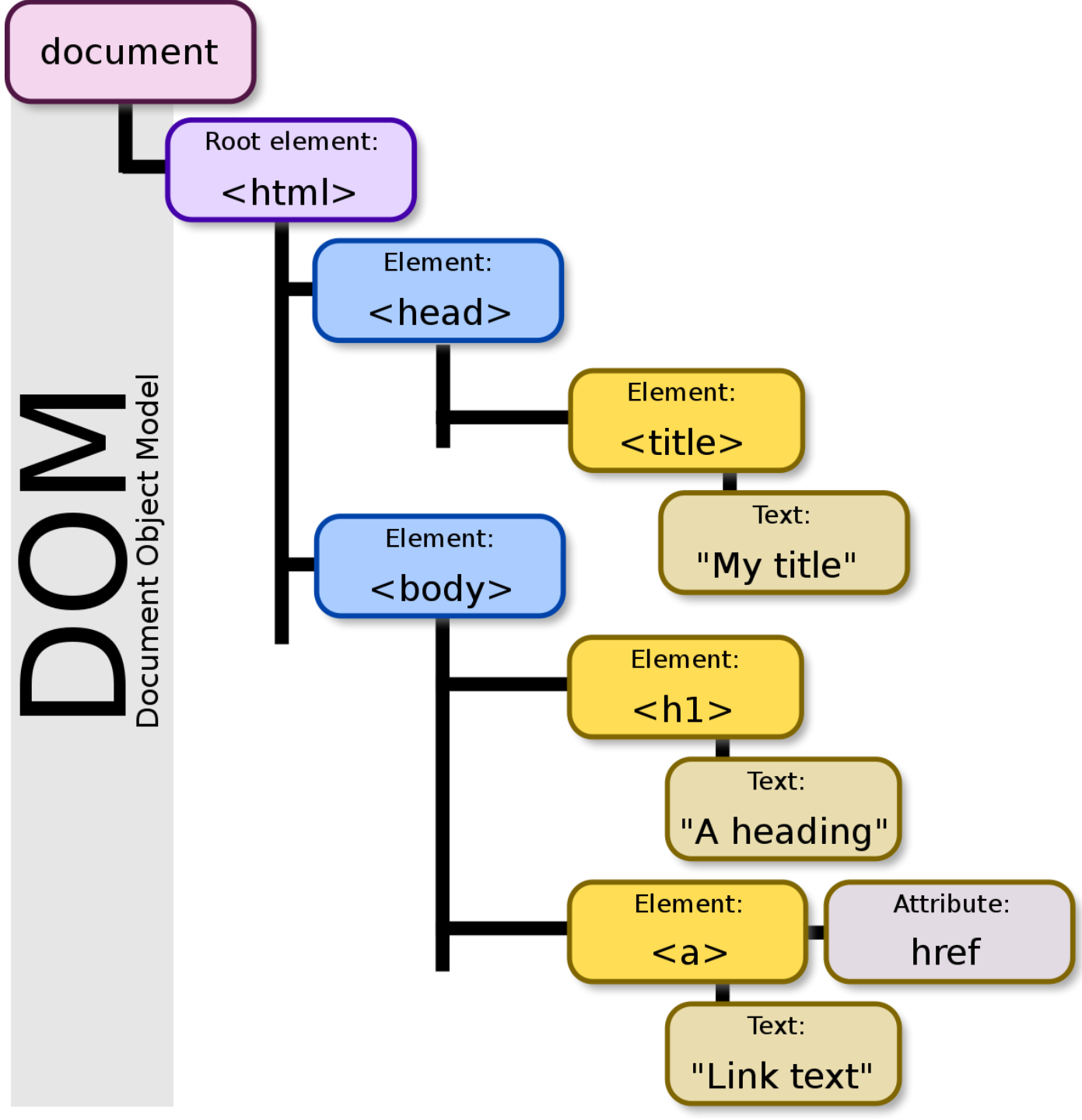
What is React?

React is a JavaScript **library** for building **user interfaces**.

- One of the most popular libraries, with over 100,000 stars on GitHub.
- React is **not a framework** (unlike Angular).
- React is an **open-source** project created by **Facebook**.
- React is used to build user interfaces (**UI**) on the **front end**.
- **Component-based programming model** which is a trending front-end programming model. There are similar front-end technologies out there such as Vue.

DOM

- Represents HTML as a tree. Each node is an object. You can manipulate DOM using JS to update HTML pages.
- Manipulating HTML elements through DOM is slower. CSS also has a tree structure. Updating CSS is also slower.
- Make DOM changes as less as possible for better performance.
- React is more performant than JQuery because it does less DOM updates.


DOM
Document Object Model
document
Root element:
<html>
Element:
<head>
Element:
<title>
Text:
"My title"
Element:
<body>
Element:
<h1>
Text:
"A heading"
Element:
<a>
Attribute:
href
Text:
"Link text"

Important Files

- **App.js**: This is the file for App Component. App Component is the main component in React which acts as a container for all other components.
- **Package.json**: This File has the list of node dependencies which are needed.

React Elements

An element is like a single frame in a movie. It represents the UI at a certain point in time.

```
import React from "react";

export default function App() {
  return React.createElement(
    "div",
    null,
    React.createElement(
      "p",
      { className: "App" },
      "Hello World. This is my first React App."
    )
  );
}
```

React.createElement()

It needs at least 3 arguments (component, props, ...children)

- The element we want to render to DOM
- Properties or an object for configuration
- Children

Configuration – Use camelCase naming standard:

- id
- className
- style

JSX

JSX just provides syntactic sugar for the React.createElement function. It is NOT a HTML. It is javascript!

```
function App() {
 return (
    <div className="App">
        <p>
          Hello World. This is my first React App.
        </p>
    </div>
 );
}
```

JSX

- User-Defined Components Must Be Capitalized.
- When an element type starts with a lowercase letter, it refers to a built-in elements like or and results in a string 'div' or 'span' passed to React.createElement
- Must return one parent item. Not more than one.

Embedding Expressions in JSX

Use curly bracket to refer a variable or call a function.

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Returning Multiple Elements

Wrap components and other HTML elements in a div

```
function App() {
 return (
   <div className="App">
       <p>
         Hello World. This is my first React App.
       </p>
       <p>
         It is fun !!!
       </p>
   </div>
 );
}
```

Returning Multiple Elements

use Fragment

```
function App() {
 return (
   <Fragment>
      <p>
         Hello World. This is my first React App.
       </p>
       <p>
         It is fun !!!
       </p>
   </Fragment>
 );
}
```

Fragment motivation

Fragments let you group a list of children without adding extra nodes to the DOM.

```
class Table extends React.Component {
    render() {
      return (
        <table>
          <tr>
            <Columns />
          </tr>
        </table>
      );
    }
  }
```

Error without Fragment

```
class Columns extends React.Component {
    render() {
      return (
        <div>
          <td>Hello</td>
          <td>World</td>
        </div>
      );
    }
  }
```

```
<!-- result -->
<table>
  <tr>
    <div>
      <td>Hello</td>
      <td>World</td>
    </div>
  </tr>
</table>
```

Solution with Fragment

```
render() {
    return (
      <>
        <td>Hello</td>
        <td>World</td>
      </>
    );
  }
}
```

```
<!-- result -->
<table>
  <tr>
    <td>Hello</td>
    <td>World</td>
  </tr>
</table>
```

React Components

- React separates concerns with loosely coupled units called “components” that contain both the markup (HTML) and logic (JS).
- Components let you split the UI into independent, **reusable** pieces.
- Components are “made of” elements.
- There are 2 types of components:
  - Functional – Stateless, dumb, presentational. Preferred.
  - Class – Stateful, smart, containers. Should override render() method.

Functional Components

- 90% cleaner code than class components.
- Class components are verbose.
- Class components get compiled. The compiled code could be messy.
- More **consistent** and easier to test.

Functional and Class Components

```
function Welcome() {
  return <h1>Hello world!</h1>;
}
```

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello world!</h1>;
  }
}
```

Extracting Components

Don’t be afraid to split components into smaller components!

```
function Comment(props) {
  return (
    <div className="Comment">
      <div className="UserInfo">
        <img className="Avatar"
          src={props.author.avatarUrl}
          alt={props.author.name}
        />
        <div className="UserInfo-name">
          {props.author.name}
        </div>
      ...
```

Creating an Avatar component

```
function Avatar(props) {
    return (
      <img className="Avatar"
        src={props.user.avatarUrl}
        alt={props.user.name}
      />
    );
  }
```

Including the Avatar component

```
function Comment(props) {
  return (
    <div className="Comment">
      <div className="UserInfo">
        <Avatar user={props.author} />
        <div className="UserInfo-name">
          {props.author.name}
        </div>
        …
```