# React

***CS568 – Web Application Development I***

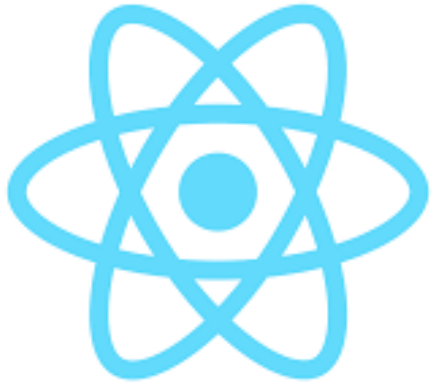
***Computer Science Department***

***Maharishi International University***

# Maharishi International University - Fairfield, Iowa

# Content

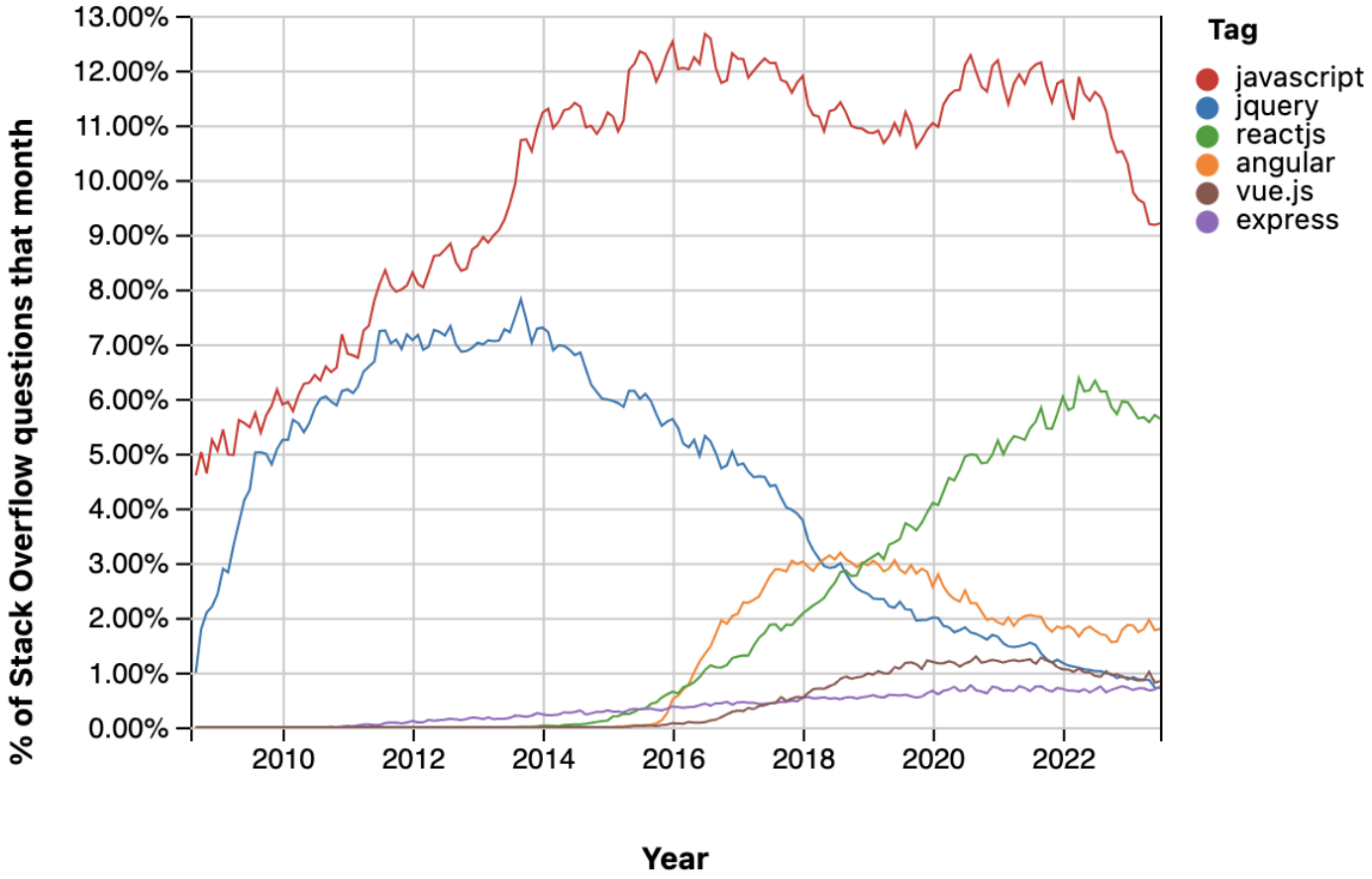
- React overview
- Create the first React app
- React.createElement
- JSX
- React Component
- React Props



# React Overview

- Single Page Application (SPA) Technology
- Developed and Maintained by Facebook
- JavaScript Library for Building Web GUI
- Reusable Components for HTML Pages
- Built on HTML, CSS, JavaScript

# Frontend technologies for Web applications



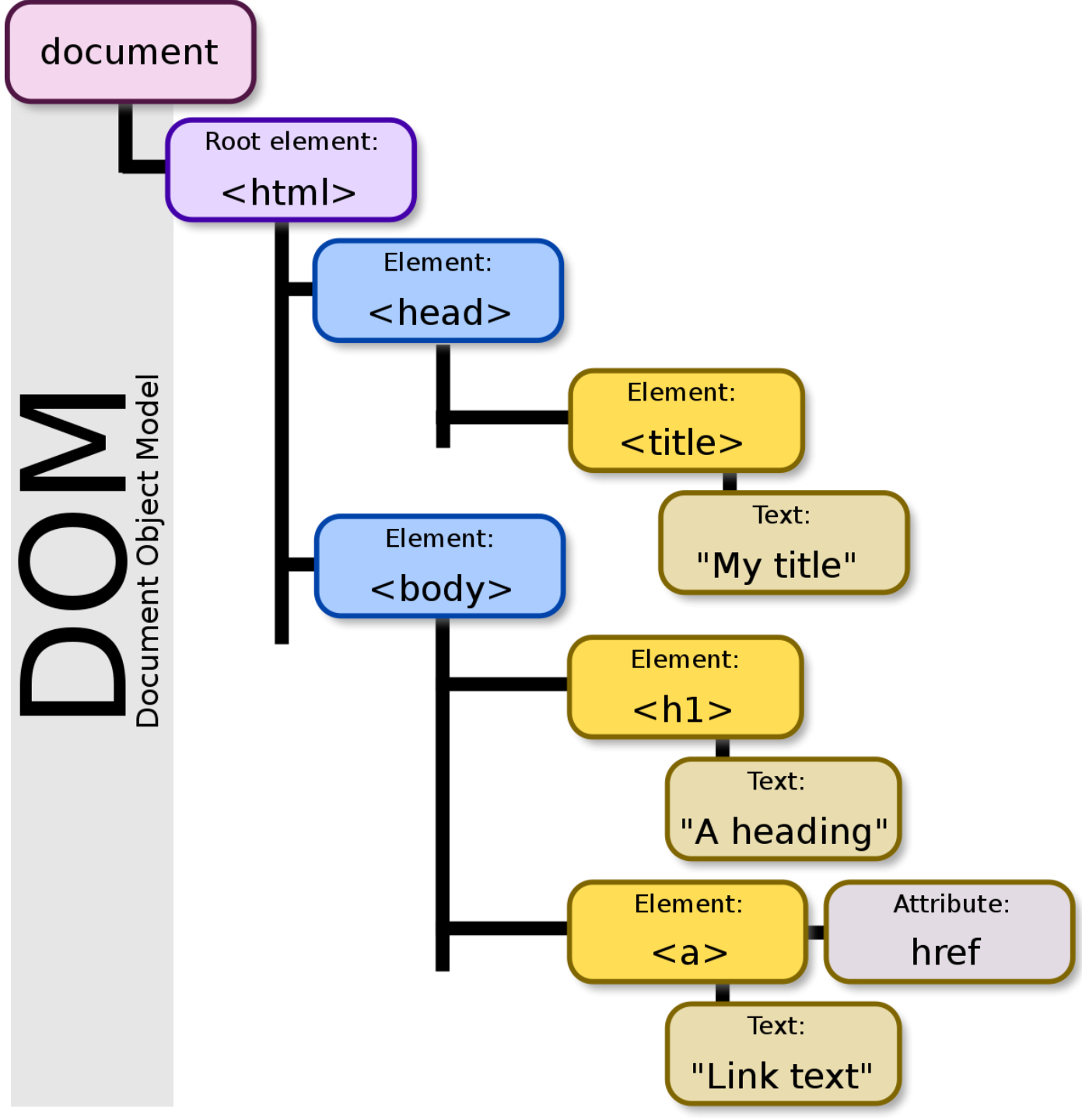

- Refer: https://insights.stackoverflow.com/trends?tags=jquery%2Cangular%2Creactjs%2Cjavascript%2Cexpress%2Cvue.js

# Basic HTML tags

- Container: 
- Inline text: 
- Block text: <p>, <h1> - <h6>
- Image: <img>
- List: <ol>, <ul>
- Table: <table>
- Form: <form>
- Input: <input>, <textarea>, radio, checkbox
- Select, option
- Button: <button>

# DOM (Document Object Model)

- Represents an HTML page 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 these entire DOM and CSS trees is slow.
- React updates only the required portion needed to be updated. It doesn’t redraw the DOM tree.
- Bottom line is, make DOM changes as less as possible for better performance. React is more performant than jQuery (it was so popular before React) 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"

# Create your first React App

- Install a LTS NodeJS: https://nodejs.org/en
- Check Node version: node - -version
- Create the first React app: npx create-react-app myapp
- Go into the app: cd myapp
- Run the app: npm start
- Open browsers and type: http://localhost:3000
- By default, this app runs on the port 3000 (you can change it if you want)
- Suggested IDE: Visual Studio Code

# Important Files in a React project

- **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.
- **Package-lock.json**: In package.json, you only define dependencies you need. Those dependencies also have dependencies. The package-lock stores all dependencies needed to run your app with **the exact verions**. Exact versions are very important. If versions mismatch, your app doesn't even start!
- **README.md** – App doc. Better write if you have time.
- **node_modules** – Actual library code. You can regenerate with npm install. So **don't push it to git**. Include it in gitignore file.
- public/**index.html** – Your app is rendered in this html.
- **Index.js** – imported in the index.html. Calls the app.js or the entry point. If you instantiate a variable here, that will be available across your app.

# React - createElement()

```
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
- onClick

# JSX (JavaScript XML)

JSX just provides syntactic sugar for the React.createElement function. It is JavaScript, not HTML.

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

# JSX (JavaScript XML)

- JS (JavaScript): For regular JavaScript code
- JSX (JavaScript XML): Syntax extension. Write JS code and XML-like code. Able to mix between JavaScript code and HTML
- A user-defined component is reusable and self-contained code. This component must be **capitalized**. E.g. ProductList. The component should return an element.
- An element is a representation of information which will be rendered on browsers.
- A component, starting with a lowercase letter, refers to a HTML built-in elements like  or  and results in a string 'div' or 'span' passed to React.createElement.
- One component must return one parent item. Not more than one.

# Embedding Expressions in JSX

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

```
function App() {
const name = "Josh";
const Welcome = <h1>hello {name}</h1>;
return (
<div className="App">
{Welcome}
<input type="text" placeholder='Enter name'/>
<button>Test</button>
</div>
);
}
```

# 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

You can use empty opening and closing tags for Fragment or (<></>)
Fragments let you group a list of children without adding extra nodes to the DOM.

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

# 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
- A component is a function or a class
  - Single responsibility
  - If code is repeated, you make it a function and call from many places.
  - Separate the GUI to multiple components to easy to reuse and maintain

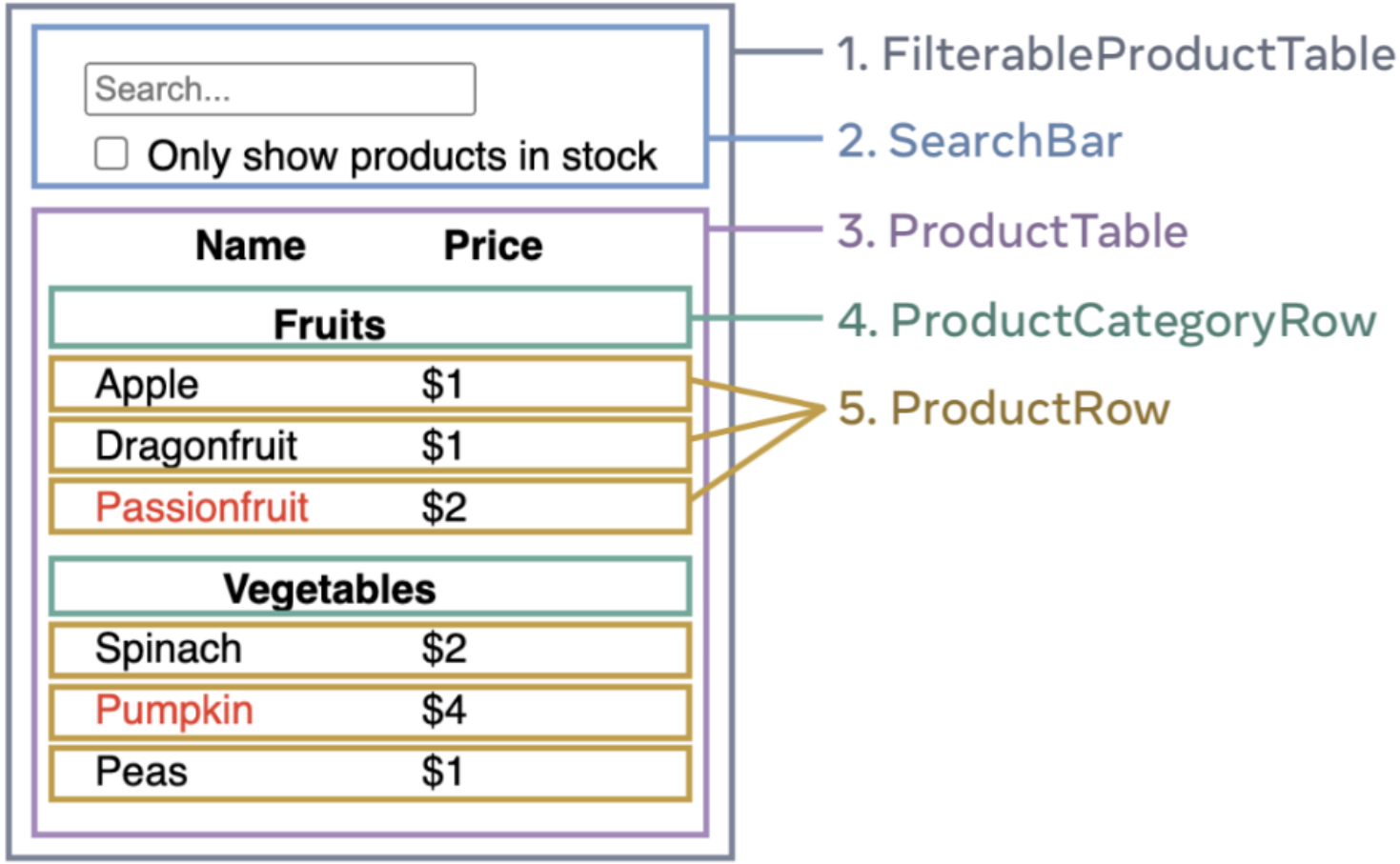
# Describing the UI

Components are reusable and nestable. To reuse them, you need to export and import the components (or functions).

From web sites to phone apps, everything on the screen can be broken down into components.

For more: https://react.dev/learn/describing-the-ui

# Example of designing components in React




https://react.dev/learn/thinking-in-react

# 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!

Split if the code gets too big or is used frequently.

```
function Comment() {
  return (
    <div>
      <div>
        <img src={avatarUrl}/>
        <p>
          {name}
        </p>
      ...
```

# Creating an Avatar component

```
function Avatar() {
    return (
      <img src={avatarUrl}/>
    );
  }
```

# Including the Avatar component

```
function Comment() {
  return (
    <div>
      <div>
        <Avatar/>
        <p>
          {name}
        </p>
        …
```

# Props

- Props is an object that can have many properties. Developers can destructure it.
- You can include anything in the props such as another component or event handler.
- Props (properties) are arguments passed to components.
- Can be used to pass data from the parent component to child components. One way, parent to child.
- They are passed like HTML properties.
- Props are read-only!
- “Props are parameters from the parent.”

# Props in Functional Component

advancedHello.js

```
function AdvancedHello(props) {
   return (<h1>Hello {props.name} </h1>);
}

export default AdvancedHello;
```

App.js

```
function App() {
 return (
   <div className="App">
    <AdvancedHello name='Umur'></AdvancedHello>
    <AdvancedHello name='Unubold'></AdvancedHello>
   </div>
 );
}
```

# Props in Class-Based Component

advancedGreeting.js

```
import React from 'react';

class AdvancedGreeting extends React.Component
{
   render() {
       return <h1>Hello {this.props.name}</h1>
   };
}

export default AdvancedGreeting;
```

App.js

```
function App() {
 return (
   <div className="App">
     <AdvancedGreeting name='Erol'></AdvancedGreeting>
     <AdvancedGreeting name='Ayse'></AdvancedGreeting>
   </div>
 );
}
```

# Children Prop

Children is a special property of React components which contains any child elements defined within the component.

```
class AdvancedGreeting extends React.Component {
    render() {
        return <h1>Hello {this.props.name} {this.props.children}</h1>
    };
}

export default AdvancedGreeting;
```

# Children Prop

```
function App() {
 return (
   <div className="App">
     <AdvancedGreeting name='Erol'>How are you?</AdvancedGreeting>
     <AdvancedGreeting name='Ayse'>We all miss you</AdvancedGreeting>
   </div>
 );
}
```

# Summary

- React
- JSX
- Component vs. Element
- Functional components & class-based components
- Props: children