React Intro

RUJUAN XING

What is React?

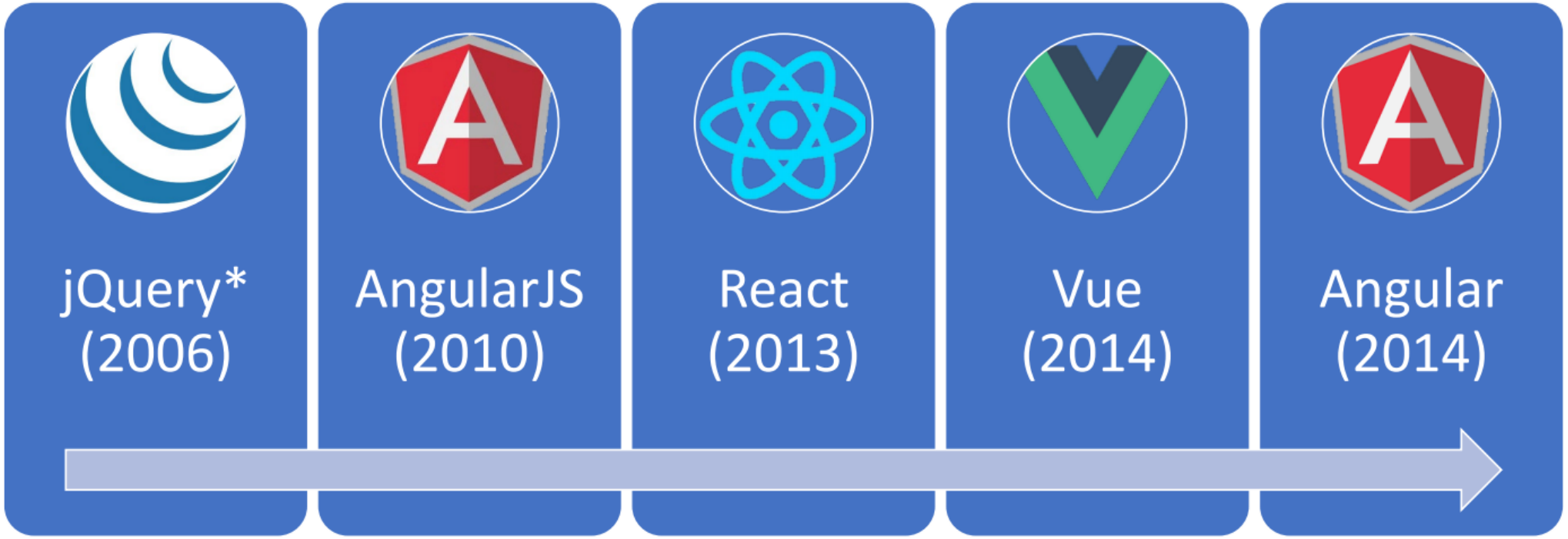
React.js, commonly referred to as React, is library for web and native user interfaces.

Example: If ask you display a collection of user stored in DB in a html page, what would you do?

1. Make request to fetch the user list
2. Process the data
3. Use DOM API to render the page

React is a JavaScript library that renders data into HTML.

Timeline of font-end JavaScript Frameworks/Libraries



Common tasks in front-end development

App state	Data definition, organization, and storage
User actions	Event handlers respond to user actions
Templates	Design and render HTML templates
Routing	Resolve URLs
Data fetching	Interact with server(s) through APIs and AJAX

Why React?

1. Complexity: Working with the DOM API directly often requires writing verbose and complex code, which can be error-prone and challenging to maintain.
2. Inefficiency with Frequent Updates: Continuous updates to the DOM can lead to a phenomenon called "reflow" or "layout thrashing," where the browser recalculates the layout and rendering of the entire page. This can be computationally expensive.
3. Performance Overhead: Direct manipulation of the DOM can be slow and resource-intensive, especially when dealing with a large number of elements. Frequent updates or changes to the DOM can lead to performance bottlenecks and affect the user experience.
4. Lack of Component Reusability: Reusing components across different parts of an application can be challenging.

Features of React

1. **Component-Based Architecture:** React is built around a component-based architecture where the user interface is divided into reusable, self-contained components. Components can be composed to create complex user interfaces, making it easier to manage and maintain code.
2. **Declarative Syntax**: React uses a declarative approach, where developers describe how the UI should look based on the current application state. This simplifies UI development and reduces the risk of bugs related to manual DOM manipulation.
3. **Virtual DOM + Diffing Algorithm**: React uses a virtual representation of the Document Object Model (DOM) called the Virtual DOM. This allows React to efficiently update the actual DOM by only re-rendering components when their state or props change. This results in improved performance and responsiveness.
4. **React Native**: React Native is a framework that extends React to build native mobile applications for iOS and Android using the same component-based approach. This enables code sharing between web and mobile applications.

Hello World

1. react.development.js: is the core library for building user interfaces with a focus on component-based architecture and declarative rendering.
2. react-dom.development.js: specializes in rendering React Components into web browsers.
3. babel.min.js: JSX allows developers to write declarative user interfaces in a more concise and readable manner. Babel can transpile JSX code into regular JavaScript so that it can be executed in web browsers.

The order of scripts matters.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
    <!-- React Core Library -->
    <script type="text/javascript" src="./libraries/react.development.js"></script>

    <!-- React DOM - used by react to do DOM operations -->
    <script type="text/javascript" src="./libraries/react-dom.development.js"></script>

    <!-- Used to transfer jsx to js -->
    <script type="text/javascript" src="./libraries/babel.min.js"></script>
</head>
<body>
    <!-- container is used to by virtual dom to render the content-->
    <div id="container"></div>

    <!-- make sure type is text/babel, because we wrote jsx syntax here -->
    <script type="text/babel">
        // make sure we don't have quotes on the value, because it's not a String
        const vDOM = <h1>Hello World</h1>;
        // React 17
        // ReactDOM.render(vDOM, document.getElementById("container"));

        //  React 18
        const root = ReactDOM.createRoot(document.getElementById("container"));
        root.render(vDOM);
    </script>

</body>
</html>
```

Two ways to create Virtual DOM

```
<body>
<!-- container is used to by virtual dom to render the content-->
<div id="container"></div>
<!-- React Core Library -->
    <script type="text/javascript" src="./libraries/react.development.js"></script>

    <!-- React DOM - used by react to do DOM operations -->
    <script type="text/javascript" src="./libraries/react-dom.development.js"></script>

    <!-- Used to transfer jsx to js -->
    <script type="text/javascript" src="./libraries/babel.min.js"></script>

<!-- make sure type is text/babel, because we wrote jsx syntax here -->
<script type="text/babel">
    // make sure we don't have quotes on the value, because it's not a String
    const vDOM = (
        <h1 id="title">
            <span>Hello World</span>
        </h1>
    );

    //  React 18
    const root = ReactDOM.createRoot(document.getElementById("container"));
    root.render(vDOM);
</script>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World</title>
    <!-- React Core Library -->
    <script type="text/javascript" src="./libraries/react.development.js"></script>

    <!-- React DOM - used by react to do DOM operations -->
    <script type="text/javascript" src="./libraries/react-dom.development.js"></script>

</head>
<body>
<!-- container is used to by virtual dom to render the content-->
<div id="container"></div>

<script type="text/javascript">
    const spanVDOM = React.createElement('span', null, 'Hello World');
    const vDOM = React.createElement('h1', {id: 'title'}, spanVDOM);
    //  React 18
    const root = ReactDOM.createRoot(document.getElementById("container"));
    root.render(vDOM);
</script>

</body>
</html>
```

Two ways to create Virtual DOM

1. JSX
   - **User-Friendly Syntax**: JSX is a JavaScript extension that provides a syntax similar to HTML. It allows you to define React elements using a familiar markup-like structure, making it more accessible to developers, especially those with web development experience.
   - **Readability**: JSX code tends to be more readable and maintainable, as it resembles the structure of the rendered UI. It makes it easier to understand the component's hierarchy.
   - **Babel Transpilation**: JSX code needs to be transpiled by tools like Babel to convert it into standard JavaScript that browsers can understand. This transpilation step is a common part of the modern web development workflow.

2. JavaScript – `React.createElement`
   - **Programmatic Approach**: `React.createElement` is a JavaScript function provided by React that allows you to create React elements programmatically.
   - **Less User-Friendly**: The syntax of React.createElement is less user-friendly compared to JSX. It can be more verbose and harder to read, especially for complex component hierarchies.
   - **No Transpilation Required**: Since React.createElement is standard JavaScript, you don't need to transpile it.

Playground: https://infoheap.com/online-react-jsx-to-javascript/

Virtual DOM vs Real DOM

1. Virtual DOM is a JavaScript object stored in memory.
2. Virtual DOM is lightweight, Real DOM is heavyweight. (Compare their properties)
3. Virtual DOM is a representation of the real DOM, will be transferred into Real DOM and render on the page.

```
<script type="text/babel">
   // make sure we don't have quotes on the value, because it's not a String
   const vDOM = (
      <h1 id="title">
         <span>Hello World</span>
      </h1>
   );

   const root = ReactDOM.createRoot(document.getElementById("container"));
   root.render(vDOM);
   console.log("Virtual DOM", vDOM);
   console.log(typeof vDOM);
   console.log(vDOM instanceof Object);
   const realDOM = document.getElementById("container");
   console.log("Real DOM", realDOM);
   debugger;
</script>
```

Create React App

REACT

Create React App

"Create React App" (CRA) is a popular and officially supported tool for quickly setting up and bootstrapping React.js applications. It's an open-source project maintained by Facebook and the React community, designed to simplify the process of starting, configuring, and developing React applications without the need for complex build configurations or setup.

1. **Zero Configuration**: Create React App allows you to create a new React application with zero configuration. It sets up a development environment with all the necessary tools and build configurations, including Webpack, Babel, and ESLint, without requiring you to manually configure them.
2. **Opinionated Defaults**: CRA comes with a set of opinionated defaults and best practices for structuring your project, such as placing components in a src directory and organizing files in a recommended way. These defaults help maintain consistency across React projects.
3. **Built-in Development Server**: It includes a development server that enables hot-reloading, allowing you to see immediate updates as you make changes to your code. No need to manually set up a server or refresh the browser.
4. **Scripts**: CRA provides a set of predefined scripts (e.g., npm start, npm build, npm test) that streamline common development tasks. These scripts handle tasks like starting the development server, building the production version of your app, and running tests.
5. **Configurability**: While CRA aims to be zero-config, it also offers advanced configuration options. You can "eject" from the default configuration to gain full control over the build process if your project requires customizations.
6. **Dependencies**: Create React App manages project dependencies, including React, React DOM, and other commonly used libraries, ensuring that you start with the latest versions.

Install and Start

1. Create a new project
   - `npx create-react-app hello-react`
2. Switch to project folder
   - `cd hello-react`
3. Start project
   - `npm start`

You'll need to have Node >= 14 on your local development machine.

npx is a command-line tool that comes with npm (Node Package Manager) and is used for executing Node.js packages. It's particularly useful for running packages that are installed as dependencies but don't have globally installed command-line tools.

Folder Structure – public folder

public ---- static resources

favicon.icon ------ favorite icon

index.html -------- main page

logo192.png ------- logo

logo512.png ------- logo

manifest.json ----- plays a crucial role in defining the behavior and appearance of the PWA when users install it on their devices

robots.txt -------- robots.txt is not specific to React.js but is a standard file used in web development to instruct web crawlers (like search engine bots) on how to interact with a website. It's used to define rules and permissions for web crawlers, specifying which parts of the site they are allowed or disallowed to access and index.

Folder Structure – src folder

src ---- source code

App.css -------- Style of App component

App.js --------- App Component

App.test.js ---- App testing file

index.css ------ Style

index.js ------- Project Entry

logo.svg ------- logo

reportWebVitals.js

--- measure and report key performance metrics that can help developers and site operators analyze and optimize the user experience. (need web-vitals library)

setupTests.js

---- unit tests of Components (need jest-dom library)

JSX

JSX (JavaScript XML) is a syntax extension for JavaScript that is commonly used in React to describe the structure and elements of the user interface.

Pros:

- The declarative template syntax of HTML
- write HTML-like code within JavaScript, easier to define and render React components

```
function App() {
  return (
    <div className="App">
      <header className="App-header">
          Learn React
      </header>
    </div>
  );
}
```

The essence of JSX

JSX is not standard JavaScript syntax; it is a syntax extension of JavaScript. Browsers cannot recognize it on their own; it needs to be parsed by a parsing tool before it can run in the browser.

```
1 <div>
2   this is div
3 </div>
4
```


```
import { jsx as _jsx } from
"react/jsx-runtime";
/*#__PURE__*/_jsx("div", {
  children: "this is div"
});
```

JSX

JSX (JavaScript XML) is a syntax extension for JavaScript that is commonly used in React to describe the structure and elements of the user interface. JSX allows you to write HTML-like code within JavaScript, making it easier to define and render React components.

Rules:

1. Don’t use quote when define virtual DOM.
2. JavaScript Expressions: To insert dynamic data, variables, or expressions into the rendered content, curly braces `{}` inside JSX elements. You cannot use statements.
3. Class Name: Use the `className` attribute to set the CSS class of an element.
4. Inline Style: Use the `style` attribute to set inline style with syntax `{{key:value}}`.
5. Only One Top-Level Element: all elements within a JSX expression must be wrapped in a single parent element.
6. Must have closing tag. Be aware of Self-Closing Tags.
7. The first letter of a tag
    1) If it’s lower case, it’ll be translated to HTML element with the same tag name. If couldn’t find in HTML, throw error but still display.
    2) If it’s upper case, react will render the component, if no component, throw error.
8. Comments: use syntax below, must inside the top-level element.

```
{/* THis is a comment*/}
```

JSX Exercise

Create a page display a list of locations which stored in Array.

- Fairfield
- Ottumwa
- Iowa City

Main Points

A good design reflects re-usability and adaptability and most importantly traceability of requirements. A good web design promotes stability in the business application and flexibility in the presentation layer. The Field of Pure Creative Intelligence is characterized by the qualities of stability and flexibility.

Transcendental consciousness is the experience of pure consciousness, the unified field of physics. Just by having this experience one gains this wholeness of knowledge and actions will be accord with all the laws of nature.