17. Introduction to React

Table of Content

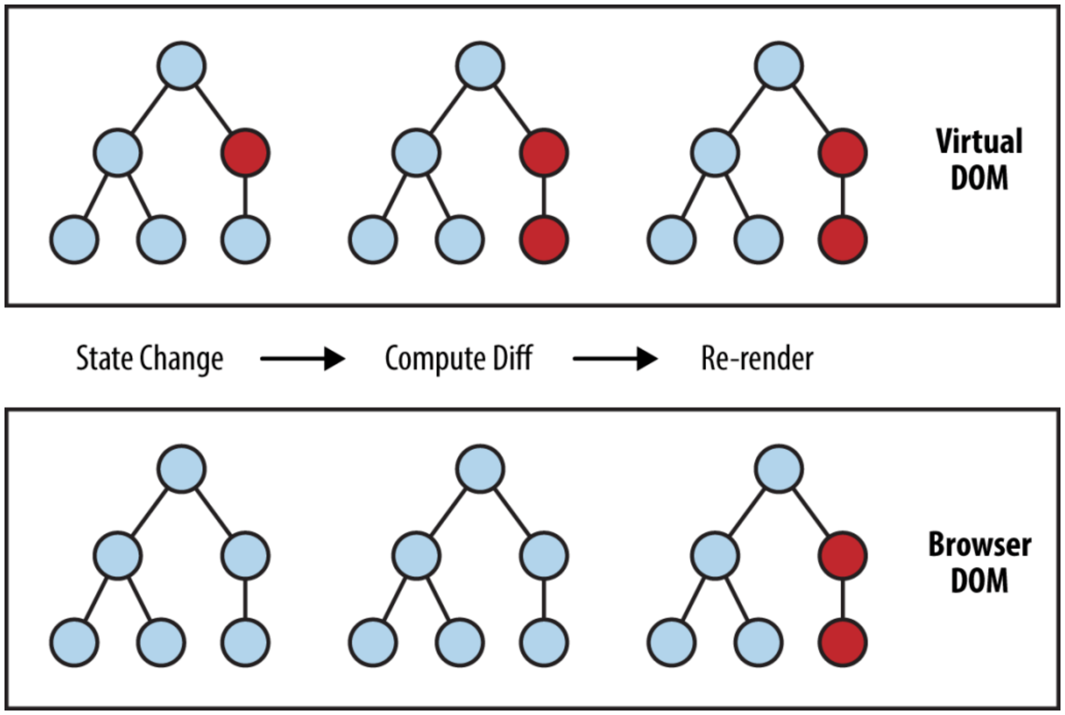
1. What is React?

React is a JavaScript library created by Facebook for building user interfaces. React is used to build single-page applications. React allows us to create reusable UI components.

▼ **How does React work?**

React creates a VIRTUAL DOM in memory.

Instead of manipulating the browser's DOM directly, React creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM.



React only changes what needs to be changed!

React finds out what changes have been made, and changes **only** what needs to be changed.

▼ **How to Create a React App using the `create-react-app` tool?**

1. Install Node JS (Node version > 14) https://nodejs.org/en/
2. Install `create-react-app` using npm

```
npm install create-react-app
```

3. Move to the Directory where you want to create your react application. Open Command prompt / Terminal and use this npx command. (npx is not a typo)

```
npx create-react-app my-react-app
```

This will create a react application named my-react-app.

2. JSX Syntax

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 converts HTML tags into react elements.

```
const myElement = <h1>This is JSX!</h1>;
```

JSX is an extension of the JavaScript language based on ES6, and is translated into regular JavaScript at runtime.

▼ Use parenthesis to insert a large block of JSX.

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

▼ One Top Level Element

The HTML code must be wrapped in *ONE* top level element.

So if you like to write *two* paragraphs, you must put them inside a parent element, like a `div` element.

▼ 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:

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

▼ The class attribute

The `class` attribute is a much-used attribute in HTML, but since JSX is rendered as JavaScript, and the `class` keyword is a reserved word in JavaScript, you are not allowed to use it in JSX.

Use attribute `className` instead.

JSX solved this by using `className` instead. When JSX is rendered, it translates `className` attributes into `class` attributes.

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

▼ Conditions - if statements

React supports `if` statements, but not *inside* JSX. Because the if statements are not *expressions* in Javascript. You could use a ternary expression instead:

```
const x = 5;

const myElement = <h1>{(x) < 10 ? "Hello" : "Goodbye"}</h1>;
```

▼ CSS Styling

You can use Inline Styling or a StyleSheet to style your elements in React.

1. **Inline Styling**

```
<h1 style={{color: "red"}}>Hello Style!</h1>
```

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 `{{}}`

You can also create an object with styling information, and refer to it in the style attribute:

```
const myStyle = {
    color: "white",
    backgroundColor: "DodgerBlue",
    padding: "10px",
    fontFamily: "Sans-Serif"
```

```
  };

const elem = <h1 style={myStyle}>Hello Style!</h1>;
```

2. **CSS Stylesheet**

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

   `import './App.css';`

---

3. Rendering Elements

Elements are the smallest building blocks of React apps.

```
const element = <h1>Hello, world</h1>;
```

Unlike browser DOM elements, React elements are plain objects, and are cheap to create. React DOM takes care of updating the DOM to match the React elements.

▼ Render with root.render()

Applications built with just React usually have a single root DOM node. To render a React element, first pass the DOM element to ReactDOM.createRoot(), then pass the React element to root.render():

```
const root = ReactDOM.createRoot(
  document.getElementById('root')
);
const element = <h1>Hello, world</h1>;
root.render(element);
```

▼ Updating the rendered element

React elements are immutable. Once you create an element, you can't change its children or attributes. An element is like a single frame in a movie: it represents the UI at a certain point in time.

With our knowledge so far, the only way to update the UI is to create a new element, and pass it to `root.render()`.

> 💡 In practice, most React apps only call root.render() once.

---

4. React Components and props

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation. This page provides an introduction to the idea of components.

> 💡 Conceptually, components are like JavaScript functions. They accept arbitrary inputs (called "props") and return React elements describing what should appear on the screen.

Previously, we only encountered React elements that represent DOM tags:

```
const element = <div />;
```

However, elements can also represent user-defined components:

```
const element = <Welcome name="Sara" />;
```

When React sees an element representing a user-defined component, it passes JSX attributes and children to this component as a single object. We call this object "props".

▼ **Two Types of Components**

1. **Functional Components**

   The simplest way to define a component is to write a JavaScript function:

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

   This function is a valid React component because it accepts a single “props” (which stands for properties) object argument with data and returns a React element. We call such components “function components” because they are literally JavaScript functions.

2. **Class Components**

   You can also use an ES6 class to define a component:

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

   The above two components are equivalent from React’s point of view.

▼ Composing Components

Components can refer to other components in their output. This lets us use the same component abstraction for any level of detail. A button, a form, a dialog, a screen: in React apps, all those are commonly expressed as components.

For example, we can create an `App` component that renders `Welcome` many times:

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

function App() {
  return (
    <div>
        <Welcome name="Sara" />
        <Welcome name="Cahal" />
```

```
        <Welcome name="Edite" />
    </div>);
}
```

Behind the scenes “Babel”

Browsers don’t understand JSX out of the box, so most React users rely on a compiler like Babel or TypeScript to **transform JSX code into regular JavaScript**. Many preconfigured toolkits like Create React App or Next.js also include a JSX transform under the hood.

For example, let’s say your source code looks like this:

```
import React from 'react';

function App() {
  return <h1>Hello World</h1>;
}
```

Under the hood, the old JSX transform turns it into regular JavaScript:

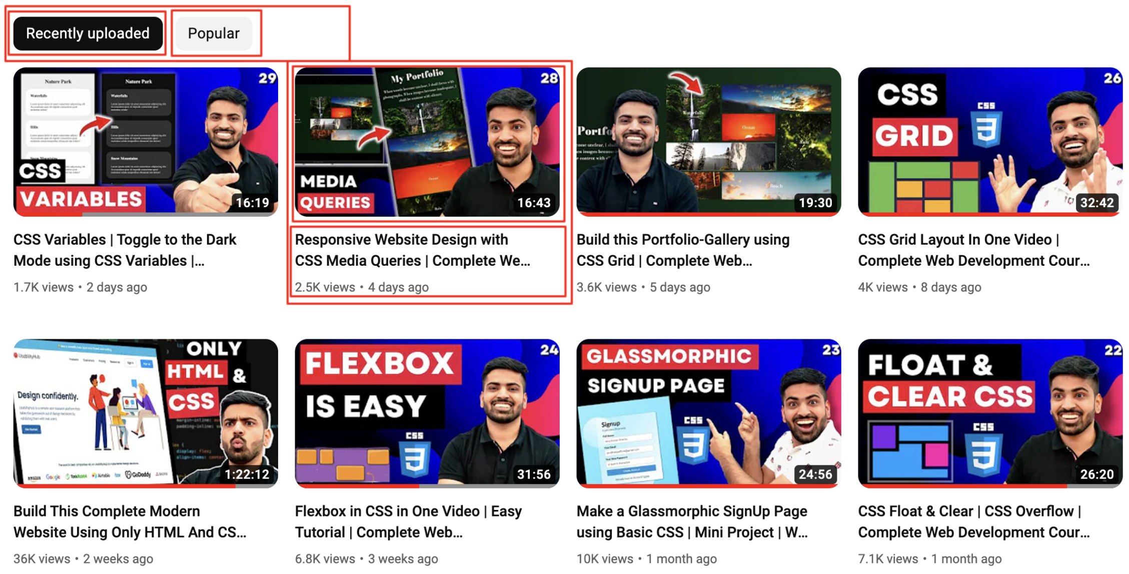
```
import React from 'react';

function App() {
  return React.createElement('h1', null, 'Hello world');
}
```

Assignment

1. Build an accordion component to display FAQs.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f9cc22c4-e0a7-4e41-8b96-b831e83e9a71/Screen_Recording_2022-12-05_at_2.10.46_PM.mov

2. Build a clone of YouTube video page section. Create Different components for all the red boxes.



CSS Variables | Toggle to the Dark Mode using CSS Variables |...
1.7K views • 2 days ago

Responsive Website Design with CSS Media Queries | Complete We...
2.5K views • 4 days ago

Build this Portfolio-Gallery using CSS Grid | Complete Web...
3.6K views • 5 days ago

CSS Grid Layout In One Video | Complete Web Development Cour...
4K views • 8 days ago

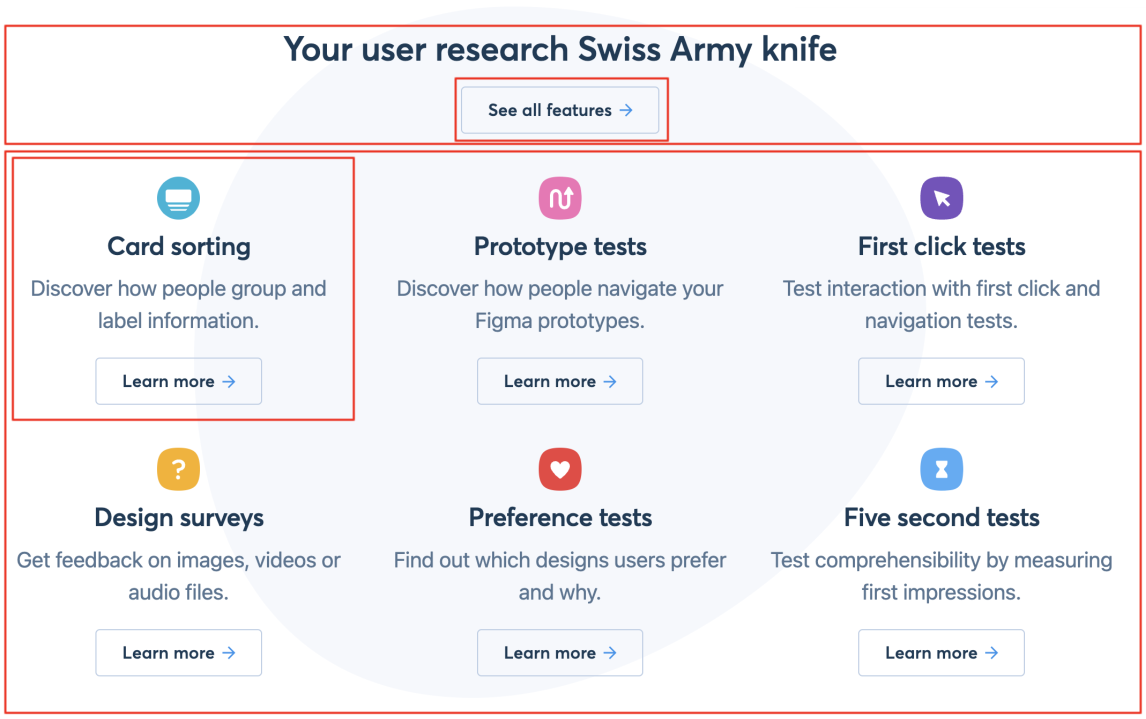
Build This Complete Modern Website Using Only HTML And CS...
36K views • 2 weeks ago

Flexbox in CSS in One Video | Easy Tutorial | Complete Web...
6.8K views • 3 weeks ago

Make a Glassmorphic SignUp Page using Basic CSS | Mini Project | W...
10K views • 1 month ago

CSS Float & Clear | CSS Overflow | Complete Web Development Cour...
7.1K views • 1 month ago

3. Build this section with red boxes as separate components.


Your user research Swiss Army knife
See all features
Card sorting
Discover how people group and label information.
Learn more
Prototype tests
Discover how people navigate your Figma prototypes.
Learn more
First click tests
Test interaction with first click and navigation tests.
Learn more
Design surveys
Get feedback on images, videos or audio files.
Learn more
Preference tests
Find out which designs users prefer and why.
Learn more
Five second tests
Test comprehensibility by measuring first impressions.
Learn more