What is React

What is React?

React is a JavaScript library for building interactive user interfaces (UIs). Which can be clicked on-screen by the users.

In terms of library, React provide helpful APIs to build the UI! It is up to the developer on how to use those APIs to build what they want to build to present to the user.

However, using React by itself, requires some configuration that the developer would have to do, web frameworks such as **Next.js** configures React for you right out of the box, and provide additional structure, features, and optimization for your application.

Rendering User Interface

Having some solid understanding of how browser interprets the user interface is fundamental to understanding how React works.

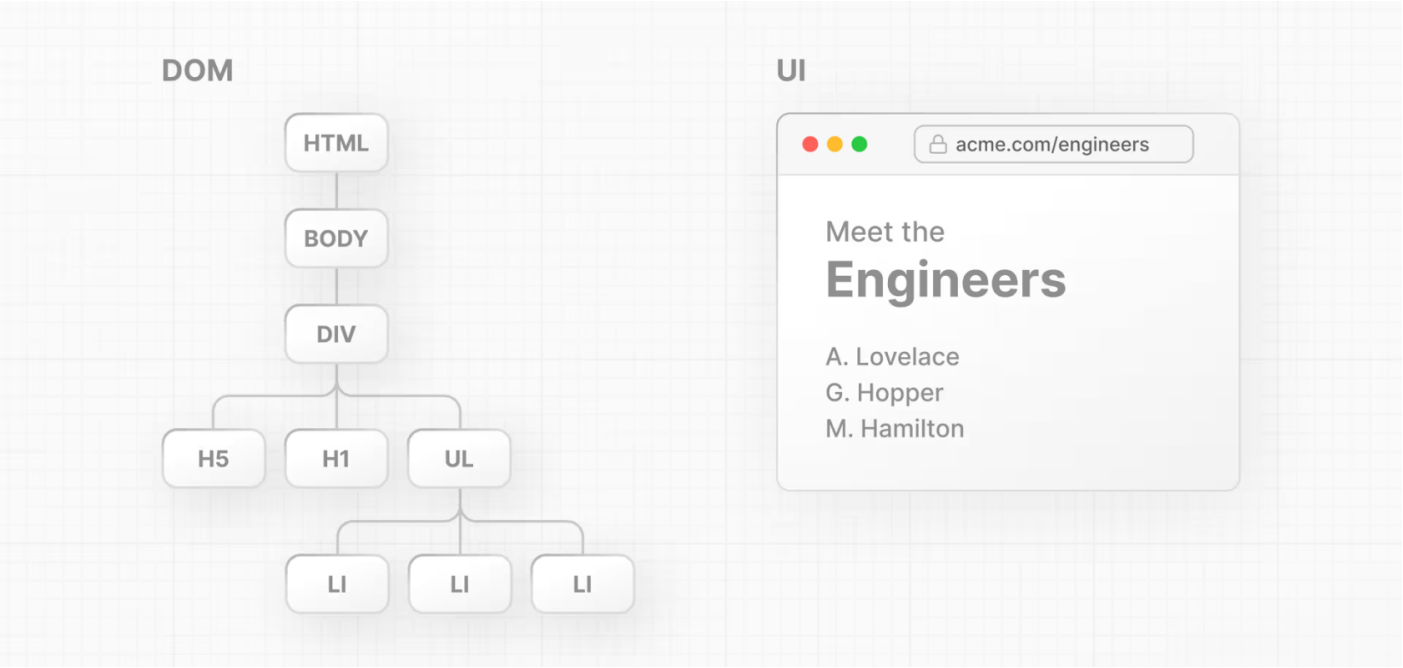
When a user visits a web page, the server returns a HTML page back to the client (Browser in this case) which may look something like

```
<html>
    <body>
        <p>Hello world!</p>
    </body>
</html>
```

The browser then reads the HTML and creates the Document Object Model (DOM).

What is DOM?

The DOM (Document Object Model) is an object representation of the HTML elements. It acts as the bridge between your code and the user interface and has a tree like structure with parent and child relationships.



You can manipulate the DOM by using JavaScript APIs, in which then the browser will render the result back to the user.

Using Plain JavaScript vs React

If you use plain JavaScript to say create a page that will insert a header text it will look something like below:

```
<html>
    <body>
        <div id="app"></div>
        <script type="text/javascript">
            const app = document.getElementById("app");

            const header = document.createElement("h1");

            const text = "This is a sample text hello world!";
            const headerContent = document.createTextNode(text);

            header.appendChild(headerContent);

            app.appendChild(header);
        </script>
    </body>
</html>
```

However, as you can see this is pretty verbose just to append a header element into the DOM. With React it simplifies a lot of the flow.

Now without having to install node or do any compiler configuring, you can start using React by just importing the main library scripts from CDNs:

```
<script src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
```

- **react**: This is the core library
- **react-dom:** This provides DOM-specific methods that let you use React with DOM

Now we can try to translate the verbose pure JavaScript code that we wrote to using React.

```
<html>
    <body>
        <div id="app"></div>
        <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
        <script src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
        <script type="text/javascript">
            const app = document.getElementById("app");
            const root = ReactDOM.createRoot(app);
            root.render(<h1>Hello world!</h1>);
        </script>
    </body>
</html>
```

Does It Work?

Now if you open the file in the browser, there will be nothing rendered and if you inspect the console output it will look something like:

```
Uncaught SyntaxError: expected expression, got '<'
```

This is because we are writing JSX, but the browser doesn't understanding JSX since it can only understand JavaScript.

What is JSX

JavaScript XML, it allows you to embed HTML into JavaScript without having to explicitly create the element yourself.

You are not required to use JSX in React, but it makes writing React application much easier if you can embed HTML into JavaScript.

It is effectively just a syntactic sugar for:

```
<MyButton color="blue" shadowSize={2}>
  Click Me
</MyButton>
```

Is compiled into:

```
React.createElement(
  MyButton,
  {color: 'blue', shadowSize: 2},
  'Click Me'
)
```

> React.createElement takes in couple arguments (type, [props], [...children])
> The type can be either a string which are HTML elements, or it could be a React component both a class or function or a React fragment.

In order to leverage JSX, you will need to add in **babel** CDN:

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

More Examples

If you choose to not use JSX and you want to create a header element and insert it into the page, then you will have to do the following:

```
const myElement = React.createElement('h1', {}, 'I do not use JSX!');

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

But if you choose to use JSX, then you can write HTML code, it gets compiled into JavaScript, and is much easier since you're embedding the exact content in the source code.

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

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

To see transpilation in real time you can visit this link: Babel

Closing Note

Now after you have all of those CDN added in you should be able to the page render with the same message! How awesome is that!

Revision #5
Created 2025-07-29 01:52:24 UTC by Tamarine
Updated 2025-11-20 03:00:07 UTC by Tamarine