1. What is React?

React is a JavaScript library for building user interfaces. It allows developers to create reusable UI components and manage their state efficiently. React was developed by Facebook and is widely used in web development for building single-page applications.

Code Example 1: Hello World in React
```jsx
// Hello.js
import React from 'react';

const Hello = () => {
  return <h1>Hello, World!</h1>;
};

export default Hello;
```

Code Example 2: Rendering Hello World Component
```jsx
// App.js
import React from 'react';
import Hello from './Hello';

const App = () => {
  return (
    <div>
      <Hello />
    </div>
  );
};

export default App;
```

In the first example, we define a simple React component called `Hello` that renders a heading with the text "Hello, World!". In the second example, we import and render the `Hello` component within another component called `App`.

This demonstrates the basic structure of a React application where components are used to compose the UI.

Of course! Let's delve deeper into the introduction to React with additional examples:

2. Why use React?

React offers several advantages for building modern web applications:

- **Component-Based Architecture**: React encourages a component-based architecture where UIs are built from reusable and composable components. This makes the codebase easier to manage and scale.
- **Virtual DOM**: React uses a virtual DOM to optimize rendering performance. It only updates the parts of the actual DOM that have changed, resulting in faster rendering and a smoother user experience.
- **Declarative Syntax**: React uses a declarative syntax, which means developers describe the desired UI state and React takes care of updating the DOM to match that state. This leads to more predictable and maintainable code.

Code Example 1: Component-Based Architecture
```jsx
// Button.js
import React from 'react';

const Button = ({ onClick, label }) => {
  return <button onClick={onClick}>{label}</button>;
};

export default Button;
```

```jsx
// App.js
import React from 'react';
import Button from './Button';

const App = () => {
  const handleClick = () => {
    console.log('Button clicked!');
  };

  return (
    <div>
      <h1>Welcome to My App</h1>
      <Button onClick={handleClick} label="Click Me" />
    </div>
  );
};
```

```
export default App;
```

In this example, we create a `Button` component that accepts an `onClick` event handler and a `label` prop. We then use this `Button` component within the `App` component. This showcases the component-based architecture of React.

Code Example 2: Virtual DOM and Declarative Syntax

```jsx
// Counter.js
import React, { useState } from 'react';

const Counter = () => {
  const [count, setCount] = useState(0);

  const increment = () => {
    setCount(count + 1);
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
    </div>
  );
};

export default Counter;
```

```jsx
// App.js
import React from 'react';
import Counter from './Counter';

const App = () => {
  return (
    <div>
      <h1>React Counter App</h1>
      <Counter />
    </div>
  );
};
```

export default App;
```

In this example, we create a simple counter app using React. The `Counter` component manages its state using the `useState` hook. When the "Increment" button is clicked, the `count` state is updated, and React efficiently re-renders only the parts of the DOM that have changed, thanks to its virtual DOM implementation and declarative syntax.

These examples illustrate why React is a popular choice for building modern web applications, showcasing its component-based architecture, virtual DOM, and declarative syntax.
```