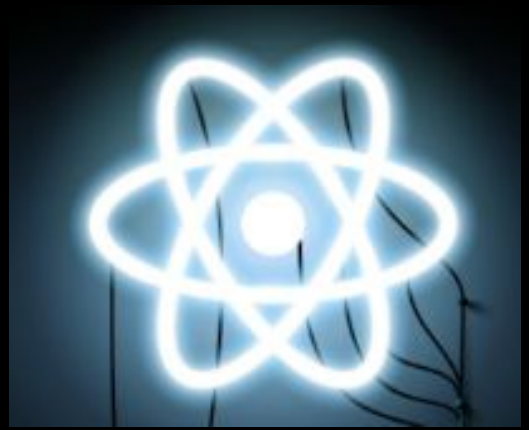
Full Stack IV

Virtual DOM & JSX

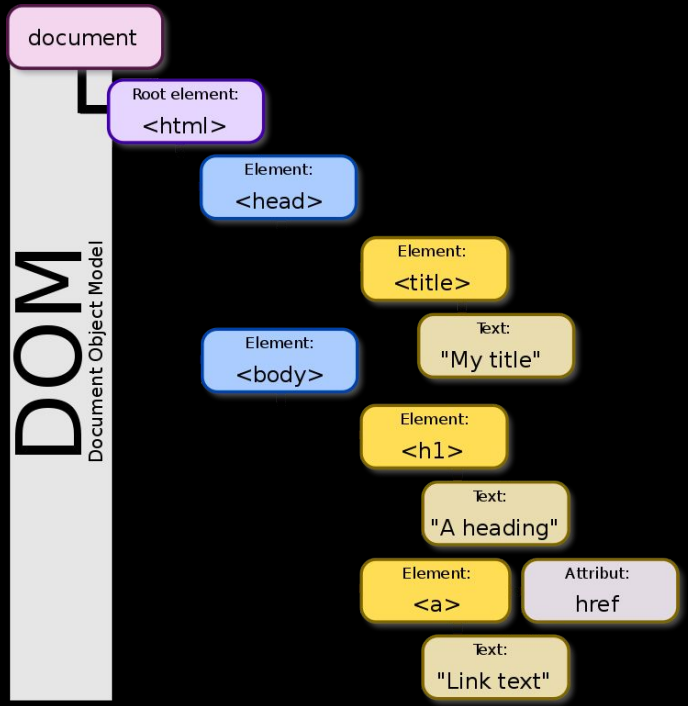


Topics

- Virtual DOM
- JSX
- React Fragments

Virtual DOM

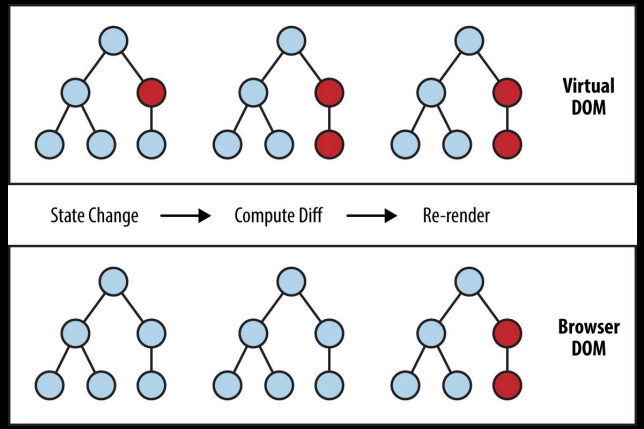
Virtual DOM



Updating the DOM is slow

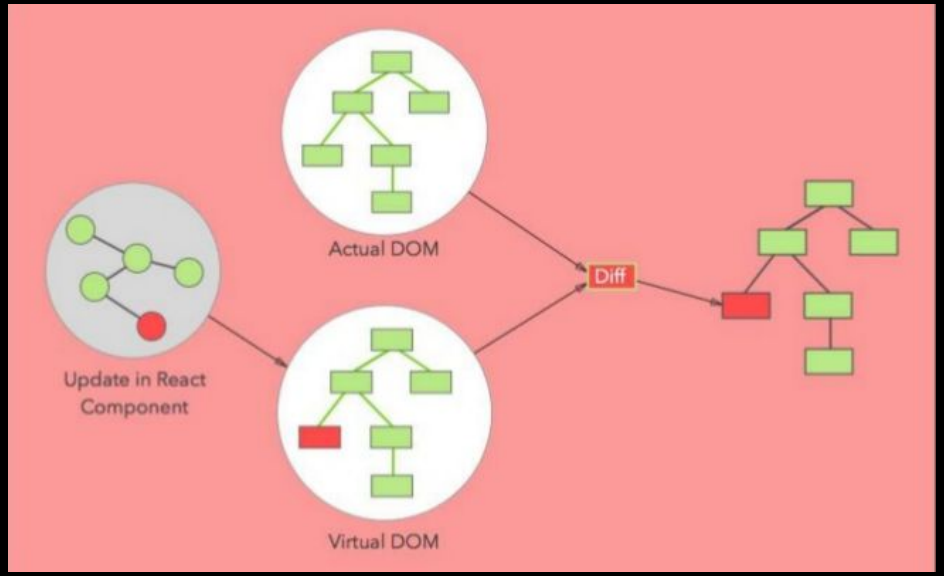
- React handles the rendering using the virtual DOM.
- The virtual DOM (VDOM) is a programming concept where a virtual representation of the UI is kept in memory and synced with the browser by ReactDOM
- VDOM is a lightweight JavaScript object which is simply a copy of Real DOM.
-
- Manipulating the DOM is slow. (Same application redraw the entire page, for minor changes)
- Manipulating the virtual DOM in memory is faster, because the screen does not get redrawn.

Virtual DOM cont..



- React will compare the updated virtual DOM with a snapshot of the previous VDOM before the update.
- Using a process called "diffing", React will compare the new VDOM with the previous state version and figure out the changes.
- When completed React will update only those changes in the real DOM.
- React maintains these 2 virtual DOM at any time

VDOM Strategy



On every update to component:

- React Builds New Virtual DOM Subtree
- Diff with the actual DOM
- Computes minimal DOM mutation
- Batch executes updates

JSX

What is JSX?

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

```
<div>
  <a
    href="#"
    onClick={sayHi}>Say Hi</a>
</div>
```

- JSX (aka JavaScript eXtension) is a JavaScript syntax extension that allows us to write JavaScript that looks likes XML.
- Similar to XML, JSX has tag name, attributes and children.
- JSX is converted into browser-compatible JavaScript using transpilers (like React JSX, jsx-transform and Babel)
  https://babeljs.io/en/repl

JSX = createReact.Element()

- Fundamentally, JSX just provides syntactic sugar for the function React.createElement(component, props, ...children)

JSX code:

```
<div className="sidebar" />
```

Compiles into:

```
React.createElement(
  'div',
  {className: 'sidebar'},
  null
)
```

JSX code:

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

Compiles into:

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

JSX Syntax

- We can mix the JSX syntax in with our React JavaScript components.
- Since class and for are reserved keywords in JavaScript, we need to use JSX attributes.
- Attributes should be camel-cased, as in classname will be className. htmlFor instead of for.

```
function MyButton() {
  return (
    <button id="main-btn" className="btn btn-small">
      Click me!
    </button>
  );
}
```

JSX Expressions

- JavaScript expressions can be embedded in JSX, by wrapping them in curly braces.
- This is how values like objects or numbers can be passed-in as props or how we can interpolate values within text content.
- Any JavaScript expression can be embedded inside curly braces, so you'll often see expressions using the ternary operator

```
function Greeter(props) {
  return (
    <div>
      {props.user ? <h2>Hello {props.user.name}</h2> : <h2>{ 1 + 2 + 3 }</h2>}
    </div>
  );
}
```

JSX in Class Component

- The following JSX code in the class component before being transpiled.

```
class Card extends React.Component {
  render() {
    const { title, content, scale, message } = this.props;
    return (
      <CardWrapper scale={scale}>
        <Header title={title} />
        <Main content={content} />
        <Footer>{message}</Footer>
      </CardWrapper>
    );
  }
}
```

JSX in Class Component *(transpiled)*

```
class Card extends React.Component {
  render() {
    const { title, content, scale, message } = this.props;
    return React.createElement(
      CardWrapper,
      { scale: scale },
      React.createElement(Header, { title: title }),
      React.createElement(Main, { content: content }),
      React.createElement(
        Footer,
        null,
        message
      )
    );
  }
}
```

One Root JSX element

- To prevent the React compilation error *'Adjacent JSX elements must be wrapped in a enclosing tag'* , you must have one root element that wraps all the child JSX elements.

Wrong!

```
return (
    <Comp1 />
    <Comp2 />
    <p>Hello!</p>
    )
```

Correct

```
return (
    <div>
        <Comp1 />
        <Comp2 />
        <p>Hello!</p>
    </div>
    )
```

More Correct

```
return (
    <React.Fragment>
        <Comp1 />
        <Comp2 />
        <p>Hello!</p>
    </React.Fragment>
    )
```

JSX Comments & Whitespace

- Comments inside JSX syntax with the /* ... */ comment syntax wrapped in curly braces
- White space can be a problem in HTML. To force a space between two elements in JSX use curly braces and the space you need.

JSX Comments

```
function Greeter() {
  return (
    <div>
      {/* Click for Hello!  */}
      Click Me?
      <MyButton />
    </div>
  );
}
```

JSX Space

```
function Greeter() {
  return (
    <div>
      Click me?{' '}Now
      <MyButton />
    </div>
  );
}
```

JSX Import React

- JSX gets transpiled to normal function calls to React.createElement

- To avoid compilation errors, the React library will need to be in scope in any files where you use JSX.

```
import React from 'react'; // don't forget to import React
import MyButton from './MyButton';

function Greeter() {
  return (
    <div>
      Hello!
      <MyButton />
    </div>
  )
}

export default Greeter;
```

Using Dot Notation for JSX Type

- React component can be referred to using dot-notation from within JSX
- This is convenient if you have a single module that exports many React components.

```
import React from 'react';

const MyComponents = {
  DatePicker: function DatePicker(props) {
    return <div>Imagine a {props.color} datepicker here.</div>;
  }
}

function BlueDatePicker() {
  return <MyComponents.DatePicker color="blue" />;
}
```

Fragments

Fragments

- A common pattern is for a component to return a list of children (nested components)

Child component <Columns/>

```
class Table extends React.Component {
  render() {
    return (
      <table>
        <tr>
          <Columns />
        </tr>
      </table>
    );
  }
}
```

Multiple <td> wrapped in parent

```
class Columns extends React.Component {
  render() {
    return (
      <div>
        <td>Hello</td>
        <td>World</td>
      </div>
    );
  }
}
```

Result is invalid HTML

```
<table>
  <tr>
    <div>
      <td>Hello</td>
      <td>World</td>
    </div>
  </tr>
</table>
```

Fragments cont..

- React Fragments solve this problem by not adding a HTML tag to the DOM

React.Fragment

```
class Columns extends React.Component {
  render() {
    return (
      <React.Fragment>
        <td>Hello</td>
        <td>World</td>
      </React.Fragment>
    );
  }
}
```

Short Syntax

```
class Columns extends React.Component {
  render() {
    return (
      <>
        <td>Hello</td>
        <td>World</td>
      </>
    );
  }
}
```

Resulting valid HTML!

```
<table>
  <tr>
    <td>Hello</td>
    <td>World</td>
  </tr>
</table>
```