Component Based UI w/ React

LIGHTHOUSE LABS

AGENDA

Declarative UI

JSX deeper dive

Storybook Demo

LIGHTHOUSE LABS

Declarative UI

LIGHTHOUSE LABS

React is a declarative UI framework

- Tell the system **what** we want the UI to show, not **how** to achieve that
- When your data changes - react will internally determine how to change the UI to reflect the data

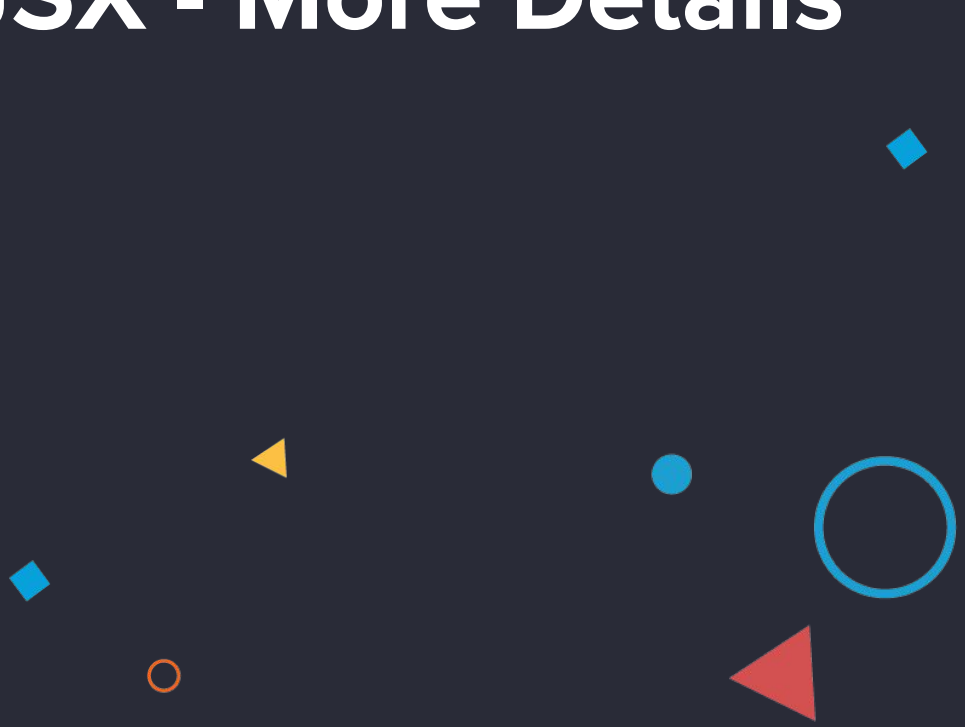
Therefore

- You don't tell React when to render - React chooses for you
- Data ownership is a key concept in understanding React

The Virtual DOM

- The Virtual DOM is an in-memory copy of the DOM tree that React keeps
- The Virtual DOM is a performance optimisation inside React
- Whenever React detects that a component needs re-rendering
    a. React renders the component
    b. React compares the newly rendered component, to the previously rendered content
    c. React updates the DOM, only if this render results in changes to the DOM

JSX - More Details



LIGHTHOUSE LABS

Components have one root

```
return (
  <div className="App">
    <header className="App-header">
      <NewBookForm createNewBook={createNewBook}/>
      <div>
        <h4>Books</h4>
        {books.map(book => <Book key={book.id} {...book} />)}
      </div>

    </header>
  </div>
);
```

✓One Root Element

LIGHTHOUSE LABS

Components have one root

```
export default function Book({title, author}) {
    return (
        <h5>{title} - {author}</h5>
        <span>Published in {releaseDate}</span>
    )
}
```

❌ Multiple Roots

LIGHTHOUSE LABS

Fragments to the rescue

```
export default function Book({title, author, releaseDate}) {
    return (
        <>
            <h5>{title} - {author}</h5>
            <span>Published in {releaseDate}</span>
        </>
    )
}
```

✓One Root Element

- Fragments will not appear in your final html
- Used in the virtual DOM only

LIGHTHOUSE LABS

JSX elements child arrays

```
const books = [
    {
        id: 0, title: 'Frankenstein', author: 'Shelly, Mary'
    },
    {
        id: 1, title: 'Wizard Of Earthsea', author: 'Ursula K. LeGuin'
    },
    {
        id: 2, title: 'The Vegetarian', author: 'Han Kang'
    },
    . . .
];
```

LIGHTHOUSE LABS

JSX elements child arrays

```
import Book from './Book';

export default function Bookshelf (props) {
    return (<div className="bookshelf">
        <Book book={props.books[0]} />
        <Book book={props.books[1]} />
        <Book book={props.books[2]} />
        <Book book={props.books[3]} />
    </div>)
}
```

LIGHTHOUSE LABS

JSX elements child arrays

```
import Book from './Book';

export default function Bookshelf (props) {
    return (<div className="bookshelf">
        {[<Book book={props.books[0]} />,
        <Book book={props.books[1]} />,
        <Book book={props.books[2]} />,
        <Book book={props.books[3]} />]}
    </div>)
}
```

LIGHTHOUSE LABS

JSX elements child arrays

```
const books = [
    {id: 0, title: 'Frankenstein', author: 'Shelly, Mary'},
    {id: 1, title: 'Wizard Of Earthsea', author: 'Ursula K. LeGuin'},
    {id: 2, title: 'The Vegetarian', author: 'Han Kang'},
    …
]
books.map(book => <Book key={book.id} book={book} />)
```

```
[
<Book key={0} book={{id: 0, title: 'Frankenstein'}} />,
<Book key={1} book={{id: 1, title: 'Wizard Of Earthsea'} />,
<Book key={2} book={{id: 2, title: 'The Vegetarian'} />
]
```

LIGHTHOUSE LABS

JSX elements child arrays

```
import Book from './Book';

export default function Bookshelf (props) {
    return (<div className="bookshelf">
        {props.books.map(book => <Book key={book.id} book={book} />)}
    </div>)
}
```

LIGHTHOUSE LABS

JSX elements child arrays

```
import Book from './Book';

export default function Bookshelf (props) {
    return (<div className="bookshelf">
        {props.books.map(book => <Book key={book.id} book={book} />)}
    </div>)
}
```

- Key is used by React in order to be able to determine if the virtual DOM has changed
- Use a unique ID for the `key` value
- Avoid using the index in the array as your `key` value
- Using bad keys can result in incorrect data being shown, or performance regressions

LIGHTHOUSE LABS

You can spread props

```
<Book book={book} />
```

← Instead of this

```
props = {
    book: {
        id: 0,
        title: "Frankenstein",
        author: "Mary Shelly"
    }
}
```

```
<Book {...book} />
```

←Do this

```
props = {
    id: 0,
    title: "Frankenstein",
    author: "Mary Shelly"
}
```

LIGHTHOUSE LABS

Props handling

```
export default function Book(props) {
    return (
        <div>
            <h5>{props.title} - {props.author}</h5>
        </div>
    )
}
```

- Props is always an object
- We access props with dot-notation

```
props = {
    id: 0,
    title: "Frankenstein",
    author: "Mary Shelly"
}
```

LIGHTHOUSE LABS

Props destructuring

```
export default function Book(props) {
    const {title, author} = props;
    return (
        <div>
            <h5>{title} - {author}</h5>
        </div>
    )
}
```

- Use object destructuring to keep your template as simple as possible

```
props = {
    id: 0,
    title: "Frankenstein",
    author: "Mary Shelly"
}
```

LIGHTHOUSE LABS

Props destructuring

```
export default function Book({title, author}) {
    return (
        <div>
            <h5>{title} - {author}</h5>
        </div>
    )
}
```

- Destructure in the params list!

```
props = {
    id: 0,
    title: "Frankenstein",
    author: "Mary Shelly"
}
```

LIGHTHOUSE LABS

Combine the two for a slick API

In combination with prop destructuring, we result in very readable, clear code

```
<Book {...book} />
```

```
export default function Book({title, author}) {
    return (
        <div>
            <h5>{title} - {author}</h5>
        </div>
    )
}
```

LIGHTHOUSE LABS

Questions?

LIGHTHOUSE LABS