Roadmap to React 16.8

Agenda

- Render returning arrays, strings. [16.0]
- Error Boundaries [16.0]
- <Fragment /> [16.2]
- New Context API [16.3]
- React.lazy() & Code Splitting [16.6]
- Hooks [16.8]
  - useState()
  - useEffect()
- Fiber:New Reconciliation engine
- Overview of React, Relay , GraphQL

16.0

- Component returning arrays, strings.
- Improved error handling with introduction of "error boundaries“.
  - componentDidCatch() life cycle hook

16.0

- Components can now return arrays and strings from render.

```
render() {
 return [
  "Some text.",
  <h2 key="heading-1">A heading</h2>,
  "More text.",
  <h2 key="heading-2">Another heading</h2>,
  "Even more text."
 ];
}
```

16.2

- **<Fragment/>**
- However, this has some confusing differences from normal JSX:
  - Children in an array must be separated by commas.
  - Children in an array must have a key to prevent React's key warning.
  - Strings must be wrapped in quotes.

```
render() {
  return (
    <Fragment>
      Some text.
      <h2>A heading</h2>
      More text.
      <h2>Another heading</h2>
      Even more text.
    </Fragment>
  );
}
```

16.3

- New Context API
- Context designed to share data that can be considered “global” for a tree of React components,
  - such as the current authenticated user, theme, or preferred language any other props

```
import React, { Component } from 'react';
class ShoppingCart extends Component {
  state = {    likes: 100  }
  render() {
    return (
      <div className="container">
        <h1>Shopping Cart</h1>
        <Product likes={this.state.likes} />
      </div>
    )
  }
}
class Product extends Component {
  render() {
    return (
      <div>
        <h1>Product</h1>
        <Likes likes={this.props.likes} />
      </div>
    )
  }
}
```

Context API

React.createContext

```
const MyContext = React.createContext(defaultValue);
```

- Creates a Context object.
- When React renders a component that subscribes to this Context object it will read the current context value from the closest matching Provider above it in the tree.

Context.Provider

```
<MyContext.Provider value={/* some value */}>
```

Context.Consumer

```
<MyContext.Consumer>
  {value => /* render something based on the context value */}
</MyContext.Consumer>
```

16.6 React.lazy() & Code Splitting

- The React.lazy function lets you render a dynamic import as a regular component.
- React.lazy takes a function that must call a dynamic import()

- **Suspense**
- If the module containing the DynamicComponent is not yet loaded we must show some fallback content while we're waiting for it to load - such as a loading indicator.
- This is done using the Suspense component.

```
const OtherComponent = React.lazy(() => import('./OtherComponent'));

function MyComponent() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <OtherComponent />
      </Suspense>
    </div>
  );
}
```

16.8 Hooks

- Motivation : **It’s hard to reuse stateful logic between components**

- They let you use state and other React features without writing a class.
- **No Breaking Changes**
- **Completely opt-in**
- **100% backwards-compatible**
- **Available now [16.8]**

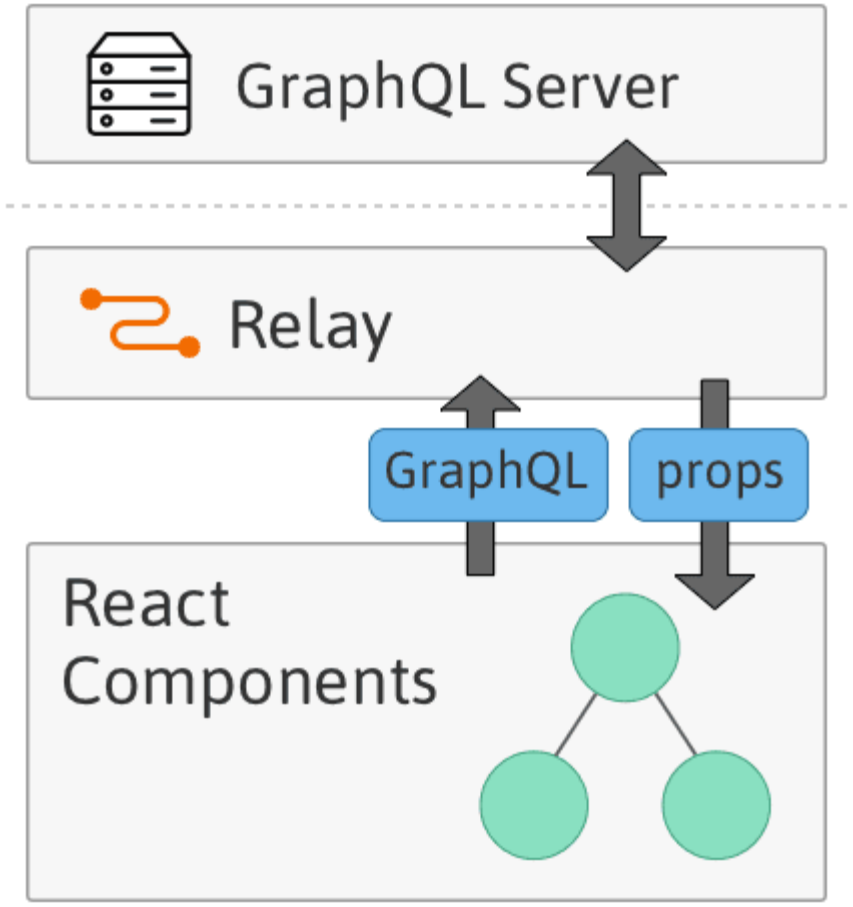
- **useState() Hook**
- **useEffect() Hook**

React's new core algorithm, React Fiber

- The goal is to increase its suitability for areas like animation, layout, and gestures.

- **incremental rendering**: the ability to split rendering work into chunks and spread it out over multiple frames
- React Fiber is a virtual stack frame, specialized for React Components

React, Relay & GraphQL



Relay : A JavaScript framework for building data-driven React applications

- JavaScript framework for building data-driven React applications powered by GraphQL
- It allows components to specify what data they need and the Relay framework provides the data.
- This makes the data needs of inner components opaque and allows composition of those needs.

GraphQL

Sample GraphQL Request

POST /graphql

```
query: {
  user(id: 1) {
    id,
    firstName,
    city
  }
}
```

```
{
  id: 1,
  firstName: 'Jason',
  city: 'Boston'
}
```