Brick-Walls

https://github.com/alienbrains/Brick-Walls

Src

App.js

```
import React from "react";
import Wall from "./Components/Wall/index";
import "./App.css";

function App() {
 return <Wall />;
}

export default App;
```

Here the App is a functional component.
There are 2 types of components.
Class Based / Stateful Components - Which has its own state.
Functional / Stateless Components - which does not have its state.
Here, App is a stateless component as it doesn’t handle state.

Components

Components/Wall/index.js

```
import React from "react";

import Brick from "../Brick/index";

import "./style.css";

class Wall extends React.Component {
 constructor(props) {
   super(props);
   this.state = {
     bricks: [1, 2, 3, 4] // initial bricks to display
   };
 }

 addBrick = () => {
   //  first copying the old data
   const newBricks = [...this.state.bricks];

   //  adding new data to previously copied data
   newBricks.push(newBricks.length + 1);

   //  updating the view using setState
   this.setState({
     bricks: newBricks
   });
 };

 //  executed when the user clicks on a brick
 onBrickClick = (number) => {
   window.alert("You Clicked Brick " + number);
 };
 //  render function returns what to display to the user
 render = () => {
   return (
     <div className="wall-container">
       {this.state.bricks.map((num) => {
         return <Brick onClick={this.onBrickClick} number={num} />;
       })}

       <div className="add-btn-box">
         <button onClick={this.addBrick} className="add-btn">
           +
         </button>
       </div>
     </div>
   );
 };
}

export default Wall;
```

The Wall component is a stateful component as it has the `this.state` which holds the data of the Wall component. The state holds the bricks array which is used to display the number of bricks on the wall.

The render() function
The purpose of this render function is to display the html code which is written inside it. Any HTML code returned from it will be displayed in the web browser.

The function addBrick
Now, on clicking the button to add a brick the function `addBrick` gets called.
We first copy the elements of the array `this.state.bricks` to a new array named `newBricks`.

After that we push a new element `newBricks.length + 1` into the `newBricks` array.
So, if initially `newBricks` was [1, 2, 3, 4] now it is [1, 2, 3, 4, 5].

We can only update our UI to show new data by using `setState`.
`setState is a function provided by React.Component to update the state and re-render the view.`
We update the state now by calling the `setState` method, which updates my state of bricks.

The function onBrickClick
This function when called will show an alert on the window displaying the brick number which you have clicked.

The map() function
The map function iterates over each element and returns a new array.
For example:

```
const array1 = [1, 4, 9, 16];

// pass a function to map
const map1 = array1.map(x => x * 2);
```

Here, x in array1.map refers to each element. So what it does is multiplies each element with 2 and returns the new array.
So, map2 here will be [2, 8, 18, 32] as each element in array1 is multiplied with 2. For more info on map
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

```
{this.state.bricks.map((num) => {
         return <Brick onClick={this.onBrickClick} number={num} />;
       })}
```

this.state.bricks.map() iterates over each element of the array and displays the corresponding jsx tag that number of times. So if there are 4 elements in the array, 4 bricks will be displayed. Similarly if there are 5 elements in the array 5 bricks will be displayed.

Now, this num is the individual element of the array, and this.onBrickClick is the function which is passed down to the <Brick /> element as props.

num is passed down to <Brick/> as number, i.e number={num}
this.onBrickClick is passed down to <Brick/> as onClick, i.e onClick={this.onBrickClick}

So, what are props?
Props in react means properties, these are the properties which are passed from one parent component to child component. Their value cannot be modified; they can only be read. Here, we are passing num and this.onBrickClick from the component Wall to the component Brick.

Components/Brick/index.js

```
import React from "react";

import "./style.css";

const Brick = (props) => {
 return (
   <div onClick={() => props.onClick(props.number)} className="brick">
     <span>{props.number}</span>
   </div>
 );
};

export default Brick;
```

This is also a stateless component.
As you can see that this component takes in an argument called **props.**
This props is passed down from the parent Wall Component.
The props is an object and consists of two key value pairs. One is onClick which is a function and the other is number.
They are accessed as props.onClick() and props.number.
On clicking the div we are calling the parent's **onBrickClick** function which was passed down to the child <Brick /> component as **onClick.**
We are passing **props.number** as an argument so that when one particular brick is clicked its value is shown as an alert in the window.

So the basic component hierarchy goes like this.
The outermost Component is the App.js
Inside it we are rendering the Wall Component.
The Wall Component is composed of 2 sections : The Brick Component and a button to add more bricks.
Since the number of bricks is stored as an array, we use the map function to display that many Brick Components as there are a number of elements in the array.
The + button is used to add more bricks in the wall.
On clicking a particular brick we display the number of the brick as an alert.

The Probable Questions

Q1. What is a class ?
Answer: Classes are a template for creating objects. They encapsulate data with code to work on that data. Classes in JS are built on prototypes but also have some syntax and semantics that are not shared with ES5 classalike semantics.
In classes we can add methods which can be accessed using objects.

Q2. What is a constructor ?

Answer: A constructor is the first method which gets executed. It is used for creating objects of a particular class. The statement super(props) is used in React as it calls the constructor of the class React.Component.

Q3. Why are we extending from React.Component ?
Answer: Extending from React.Component allows us to use the methods of the React Component class. One such method which we use is this.setState(), to update our state. Lifecycle methods and the render method which form the core of React are also provided by this Component Class.

Q4. Why do we use className ?
Answer: className is used to refer to the particular CSS class, which is used to apply styling to our particular element.

Q5. What is setState?
Answer: We use this function to update the state and cause a re-render of the UI. For example, in this project we used setState to change the number of bricks which was reflected in the UI, when we pressed the button to add more bricks. setState is asynchronous. Which means that it does not follow the normal synchronous way of programming. When we call the setState function it re-renders the entire component tree, i.e it updates the webpage to show the necessary changes that have been made in the UI. It accepts a callback. Which means

```
this.setState({
      bricks: newBricks
}, ( state )=> {
      // This is a callback function
      console.log(state)
});
```

The console.log statement will be executed after the state has been updated.