What is Constructor?

The constructor is a method used to initialize an object's state in a class. It automatically called during the creation of an object in a class.

The concept of a constructor is the same in React. The constructor in a React component is called before the component is mounted. When you implement the constructor for a React component, you need to call **super(props)** method before any other statement. If you do not call super(props) method, **this.props** will be undefined in the constructor and can lead to bugs.

Syntax

```
Constructor(props){
    super(props);
}
```

In React, constructors are mainly used for two purposes:

1. It used for initializing the local state of the component by assigning an object to this.state.
2. It used for binding event handler methods that occur in your component.

> Note: If you neither initialize state nor bind methods for your React component, there is no need to implement a constructor for React component.

You cannot call **setState()** method directly in the **constructor()**. If the component needs to use local state, you need directly to use '**this.state**' to assign the initial state in the constructor. The constructor only uses this.state to assign initial state, and all other methods need to use set.state() method.

Example

The concept of the constructor can understand from the below example.

App.js

```
import React, { Component } from 'react';

class App extends Component {
  constructor(props){
    super(props);
```

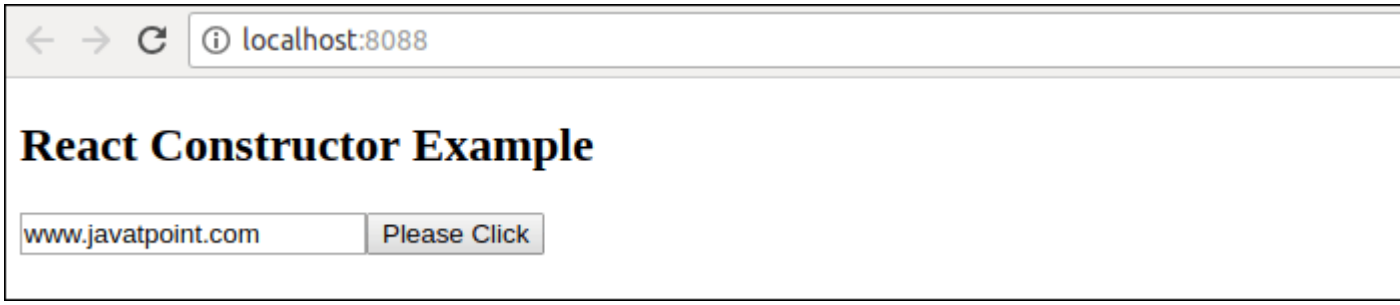
```
6.      this.state = {
7.          data: 'www.javatpoint.com'
8.        }
9.      this.handleEvent = this.handleEvent.bind(this);
10.   }
11.   handleEvent(){
12.     console.log(this.props);
13.   }
14.   render() {
15.     return (
16.       <div className="App">
17.     <h2>React Constructor Example</h2>
18.     <input type ="text" value={this.state.data} />
19.         <button onClick={this.handleEvent}>Please Click</button>
20.       </div>
21.     );
22.   }
23. }
24. export default App;
```

Main.js

```
1.  import React from 'react';
2.  import ReactDOM from 'react-dom';
3.  import App from './App.js';
4.
5.  ReactDOM.render(<App />, document.getElementById('app'));
```

Output

When you execute the above code, you get the following output.



The most common question related to the constructor are:

1. Is it necessary to have a constructor in every component?

No, it is not necessary to have a constructor in every component. If the component is not complex, it simply returns a node.

```
class App extends Component {
    render () {
        return (
            <p> Name: { this.props.name }</p>
        );
    }
}
```

2. Is it necessary to call super() inside a constructor?

Yes, it is necessary to call super() inside a constructor. If you need to set a property or access 'this' inside the constructor in your component, you need to call super().

```
class App extends Component {
    constructor(props){
        this.fName = "Jhon"; // 'this' is not allowed before super()
    }
    render () {
        return (
            <p> Name: { this.props.name }</p>
        );
    }
}
```

When you run the above code, you get an error saying **'this' is not allowed before super()**. So if you need to access the props inside the constructor, you need to call super(props).

Arrow Functions

The Arrow function is the new feature of the ES6 standard. If you need to use arrow functions, it is not necessary to bind any event to 'this.' Here, the scope of 'this' is

global and not limited to any calling function. So If you are using Arrow Function, there is no need to bind 'this' inside the constructor.

```
1. import React, { Component } from 'react';
2.
3. class App extends Component {
4.   constructor(props){
5.     super(props);
6.     this.state = {
7.         data: 'www.javatpoint.com'
8.       }
9.   }
10.  handleEvent = () => {
11.    console.log(this.props);
12.  }
13.  render() {
14.    return (
15.      <div className="App">
16.    <h2>React Constructor Example</h2>
17.    <input type ="text" value={this.state.data} />
18.        <button onClick={this.handleEvent}>Please Click</button>
19.      </div>
20.    );
21.  }
22. }
23. export default App;
```

We can use a constructor in the following ways:

1) The constructor is used to initialize state.

```
1. class App extends Component {
2.   constructor(props){
3.       // here, it is setting initial value for 'inputTextValue'
4.       this.state = {
5.           inputTextValue: 'initial value',
```

```
        };
    }
}
```

2) Using 'this' inside constructor

```
class App extends Component {
    constructor(props) {
        // when you use 'this' in constructor, super() needs to be called first
        super();
        // it means, when you want to use 'this.props' in constructor, call it as below
        super(props);
    }
}
```

3) Initializing third-party libraries

```
class App extends Component {
    constructor(props) {

        this.myBook = new MyBookLibrary();

        //Here, you can access props without using 'this'
        this.Book2 = new MyBookLibrary(props.environment);
    }
}
```

4) Binding some context(this) when you need a class method to be passed in props to children.

```
class App extends Component {
    constructor(props) {

        // when you need to 'bind' context to a function
        this.handleFunction = this.handleFunction.bind(this);
    }
}
```