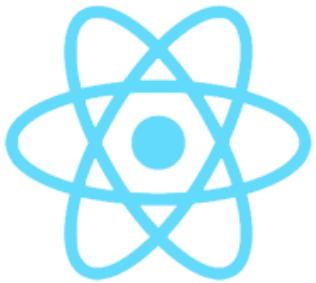
YASH Technologies™

React Life Cycle Methods



React

Presented by: Pankaj Sharma

Topics to be covered

- LifeCycle method of Reactjs and their use
- ComponentDidMount – How to make Ajax Request

LifeCycle methods of ReactJS and their use

Each component has several life cycle methods.

- These life cycle methods are important to run a particular code at particular time in the process.
- Methods prefixed with “will” are called right before something happens.
- Methods prefixed with “did’ are called right after something happens.

All Lifecycle methods can be split in four phases : initialization, mounting, updating and unmounting.

Ref :

- https://medium.com/react-ecosystem/react-components-lifecycle-ce09239010df
- https://facebook.github.io/react/docs/react-component.html

Prog : open : life_cycle_demo

Up till now we have already used two lifecycle methods in our applications.

- constructor()
- render()

These two methods comes under the category of Mounting. We have two more methods in this category

- componentWillMount()
- componentDidMount()

To understand the behaviour let us write the application, to check which method will trigger at what time.

LifeCycle methods of ReactJS and their use

Create a component and name it as LifeCycleDemoComponent.

```
Life-cycle-demo/src/index.js
-----------------------------------
import React from 'react';
import ReactDOM from 'react-dom';
class LifeCycleDemoComponent extends React.Component{
render(){
return (
<div>
<h1>Demo of LifeCycle Methods</h1>
</div>
)
}
}
ReactDOM.render(<LifeCycleDemoComponent />, document.getElementById('root'));
```

LifeCycle methods of ReactJS and their use

Now create constructor(), componentWillMount(), componentDidMount() methods, and for testing write console.log() in all the four methods, and check on browser console the sequense of execution.

```
Life-cycle-demo/src/index.js
-----------------------------------
import React from 'react';
import ReactDOM from 'react-dom';

class LifeCycleDemoComponent extends React.Component{

constructor(){
super();
this.state={
methods:['constructor()','componentWillMount()','render()','componentDidMount
()']
}
console.log('------constructor called-------');
}

componentWillMount(){
console.log('------componentWillMount called-------');
}

componentDidMount(){
console.log('------componentDidMount called-------');
}
```

LifeCycle methods of ReactJS and their use


```
Life-cycle-demo/src/index.js
-----------------------------------
render(){
console.log('------render called-------');
return (
<div>
<h1>Demo of LifeCycle Methods</h1>
<hr/>
<h2>Open the browser console to check the output</h2>
<h3>
Mounting
</h3>
<p>These methods are called when an instance of a component is being created
and instered into the DOM.</p>
<ul>
{this.state.methods.map((method)=>{return <MethodNameComponent key ={method}
method={method}/>})}
</ul>
</div>
)
}
}
```

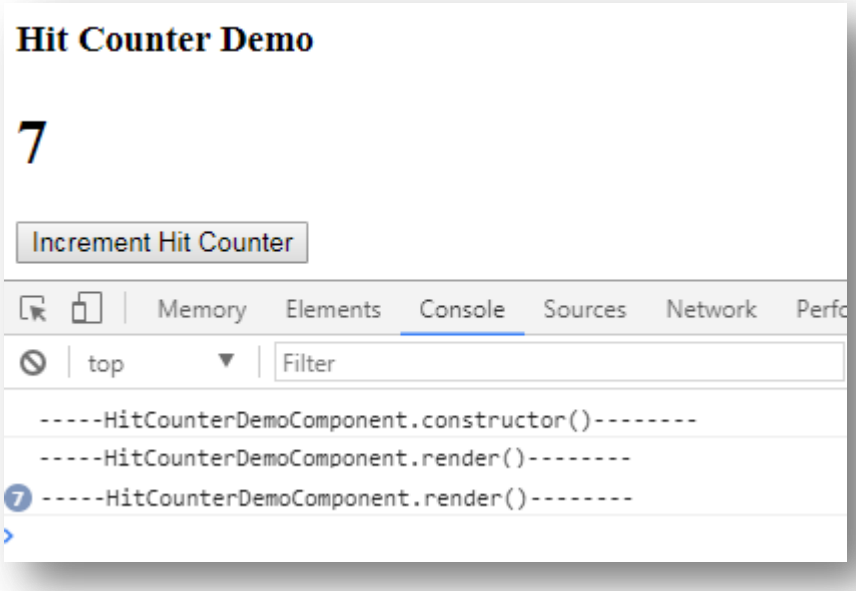
LifeCycle methods of ReactJS and their use


```
Life-cycle-demo/src/index.js
-----------------------------------
class MethodNameComponent extends React.Component{
render(){
return(
<li>{this.props.method}</li>
)
}
}

ReactDOM.render(<LifeCycleDemoComponent />, document.getElementById('root'));
```

LifeCycle Method Assignment

Assignment : Create a HitCounter that need to increase the number of hits on button click.



componentDidMount () - AJAX Request

Now here we will learn how to make AJAX Request. We will be using the Jquery support here to make AJAX Request.
To add the Jquery in your project run below command on command prompt in componentDidMount-Ajax directory.

npm i jquery –s

Our plan is to take JSON data from some other server and render the information from available JSON.

https://jsonplaceholder.typicode.com/

We will be using the todos json from here.

https://jsonplaceholder.typicode.com/todos

componentDidMount () - AJAX Request

Create basic application structure

componentDidMount-AJAX/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
class App extends React.Component{
render(){
return(
<div>
<h1>AJAX-Demo</h1>
<hr/>
</div>
)
}
}
ReactDOM.render(<App />, document.getElementById('root'));
```

componentDidMount () - AJAX Request

Now we will be getting some todos data from live server, to hold this data in our application we will require one todos array, it should be instantiated in state, that we mention in constructor. Follow below code.

componentDidMount-AJAX/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
class App extends React.Component{
constructor(props){
console.log('-------constructor--------');
super(props);
this.state={
todos:[]
}
. . . .
}
ReactDOM.render(<App />, document.getElementById('root'));
```

componentDidMount () - AJAX Request

Now improvise the render() method to show title from todos in list.

componentDidMount-AJAX/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
. . . .
render(){
console.log('-------render--------');
const {todos} =this.state;
return(
<div>
<h1>AJAX-Demo</h1>
<hr/>
<ul>
{
todos.map((todo)=>{
return <li>{todo.title}</li>
})
}
</ul>
</div>
)
}
}
ReactDOM.render(<App />, document.getElementById('root'));
```

What is the purpose of using => function here?

componentDidMount () - AJAX Request

Now import $ from jquery

componentDidMount-AJAX/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import $ from 'jquery'
. . . .
ReactDOM.render(<App />, document.getElementById('root'));
```

componentDidMount () - AJAX Request

Now add componetDidMount() method, and get data from server and assign it in todos array.

componentDidMount-AJAX/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import $ from 'jquery'
. . . . // constructor
componentDidMount(){
console.log('-------componentDidMount--------');
$.ajax({
url:'https://jsonplaceholder.typicode.com/todos',
success:(data)=>{
this.setState({
todos:data
})
}
})
}
. . . .  // render
ReactDOM.render(<App />, document.getElementById('root'));
```

componentDidMount () - AJAX Request : Assignment

Take below url and display below data in tabular form.

- Name
- Username
- Email

https://jsonplaceholder.typicode.com/users

Thank You!

Email: info@yash.com
Web: www.yash.com

Presented by: Pankaj Sharma