

Notes on React

Creating and rendering elements

src > JS index.js > ⓧ setInterval() callback

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3
```

```
4 function time() {
```

```
5   /**
```

```
6   * React.createElement(
```

```
7     type,
```

```
8     [props],
```

```
9     [... children]
```

```
10    )
```

```
11   */
```

```
12   return React.createElement(
```

```
13     'h1',
```

```
14     null,
```

```
15     <span>{new Date().toLocaleTimeString()}</span>,
```

```
16     <h1> Children 2 </h1>
```

```
17   );
```

```
18 }
```

```
20 const rootDiv = document.getElementById('root');
```

```
22 setInterval(() => {
```

```
23   /**
```

```
24   ReactDOM.render(
```

```
25     element,
```

```
26     container[,
```

```
27     callback]
```

```
28   )
```

```
29   */
```

```
30   ReactDOM.render(time(), rootDiv);
```

```
31 }, 1000);
```

type : argument can be either
a tag name eg. div,span,
A React.Component or
A React.Fragment

[props] : a config for the
the ReactElement such as style

[...children] : multiple
children can be passed as it's
rest operator

React.craeteElement : Returns
a ReactElement Which is the smalles
building block in react

ReactDOM/Native.render renders

React element/s in the
supplied

container and execute

Callback if provided

with JSX

```
src > js index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  function time() {
5    return (
6      <div style={{ color: 'red' }}>
7        <span>{new Date().toLocaleTimeString()}</span>
8        <h1> Children 2 </h1>
9      </div>
10    );
11  }
12
13  const rootDiv = document.getElementById('root');
14
15  setInterval(() => {
16    ReactDOM.render(time(), rootDiv);
17  }, 1000);
18
```

JSX syntax

JS expression inside curly braces

ReactDOM escapes any values embedded in JSX before rendering them. This helps prevent XSS (cross-site-scripting) Attacks

Functional Component

User-Defined Components
Must Be Capitalized

React considered lower-
case tags a HTML tags

```
src > JS index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  // eslint-disable-next-line no-unused-vars
5  function Time() {
6    return (
7      <div style={{ color: 'red' }}>
8        <span>{new Date().toLocaleTimeString()}</span>
9        <h1> Children 2 </h1>
10      </div>
11    );
12  }
13
14  const rootDiv = document.getElementById('root');
15
16  setInterval(() => {
17    ReactDOM.render(<Time />, rootDiv);
18  }, 1000);
19
```

The Time function considered
as a component

While the returned element
by a component considered
as a ReactElement

Props

```
src > .js index.js > ...
```

```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  function Time(props) {
5    const { color, locale } = props;
6
7    return (
8      <div style={{ color }}>
9        <span>{new Date().toLocaleTimeString(locale)}</span>
10     </div>
11   );
12 }
13
14 const rootDiv = document.getElementById('root');
15
16 setInterval(() => {
17   ReactDOM.render(
18     <div>
19       <Time color="blue" locale="bn-BD" />
20       <Time color="green" locale="ar-EG" />
21       <Time color="red" locale="ko-KR" />
22     </div>,
23     rootDiv
24   );
25 }, 1000);
26
```

Read props inside the child component

Don't miss the pair of { and } curlyies inside of (and) when declaring props: or destructring the objects received

Passing props to the child components as comma separated values

props are immutable—a term from computer science meaning “unchangeable”. When a component needs to change its props (for example, in response to a user interaction or new data), it will have to “ask” its parent component to pass it different props—a new object!

Class Component

```
rc > js index.js > ...
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  class Time extends React.PureComponent {
5    render() {
6      console.log(this.props);
7      const { style, locale, children } = this.props;
8      return (
9        <div style={style}>
10          <span>{new Date().toLocaleTimeString(locale)}</span>
11          {children}
12        </div>
13      );
14    }
15  }
16
17  const rootDiv = document.getElementById('root');
18
19  ReactDOM.render(
20    <div>
21      <Time style={{ border: '1px solid red' }} font="50px" locale="bn-BD">
22        <h1>This is test</h1>
23        <h2>This is test</h2>
24      </Time>
25    </div>,
26    rootDiv
27  );
28
```

while rendered Class.render gets called

props passed via the React.Component to this.props and nested elements are in this.props.children

State and lifecycle

we pass
props to
the base
constructor

Class components
should always
call the
base
constructor
with props.

Initial State of a component
should initialize through
constructor

State is similar to props,
but it is private and fully
controlled by the component.

lifecycle methods

setState() called with given
values updates the components
with newer states

```
src > components > JS Clock.js > ...
1  import React from 'react';

2  class Time extends React.Component {
3    constructor(props) {
4      super(props);
5      this.state = { date: new Date() };
6    }
7
8    componentDidMount() { Life Cycle starts on 'mount'
9      this.clockTimer = setInterval(() => this.tick(), 1000);
10    }
11
12    componentWillUnmount() { Life Cycle ends on 'unmount'
13      clearInterval(this.clockTimer);
14    }
15
16
17    tick() {
18      this.setState({
19        date: new Date(),
20      });
21    }
22
23    render() {
24      const { style, locale, children } = this.props;
25      const { date } = this.state;
26      return (
27        <div style={style}>
28          <span>{date.toLocaleTimeString(locale)}</span>
29          {children}
30        </div>
31      );
32    }
33  }
34
35  export default Time;
```

