

Frontend Framework

React

What is React?

- UI Library
- Component-based
 - Reusable
- Declarative API
 - <https://codeburst.io/declarative-vs-imperative-programming-a8a7c93d9ad2>

React components

- JSX
- Element & component
- State & Props
- Virtual DOM

Setting up Tools and Environment

- Node.JS
 - <https://nodejs.org/en/>
- Yarn
 - Package manager
 - Installing Node.js' dependencies
 - For Node.js version higher than 16.10
 - Enable [corepack] via corepack enable
 - Run the above command as an Admin
- Visual Studio Code
 - Install plugins:
 - ESLint
 - Prettier-Code formatter
 - Setting-> search format -> editor: Format On Save ☒
 - Vscodestyled-components
- Chrome Extension
 - React Developer Tools:
<https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en>

Create React project

- Change to target folder
- Run command
 - `yarn create react-app project_name`
- After complete
 - VS menu open folder -> project_name
 - From menu Terminal -> new
 - `yarn start`

Concept

- UI is the function of state

$$UI = f(state)$$

JavaScript XML (JSX)

- Using XML in JavaScript
- Use `{ }` to evaluate the expression

Element

- In react, html element can be defined using const as shown:

```
const h = <div><h1>SW Studio News03</h1></div>;
```


Header

CSS

```
import './App.css';  
import { Component } from 'react';
```

Component object

Component creation (ES6 and earlier)

- Import React from 'react';

```
var NewsHead04 = React.createClass({  
  render: function() {  
    return(  
      <div>  
        <h1>SW Studio News</h1>  
      </div>  
    );  
  }  
});
```

Component creation (ES6)

- import { Component } from 'react';
- To create Component
 - Class's name must begin with capital letter

Example:

Class component

```
class NewsHead01 extends Component {  
  render() {  
    return (  
      <div>  
        <h1>SW Studio News</h1>  
      </div>  
    );  
  }  
}
```

Function component

```
function NewsHead02() {  
  return (  
    <div>  
      <h1>SW Studio News</h1>  
    </div>  
  );  
}
```

Component

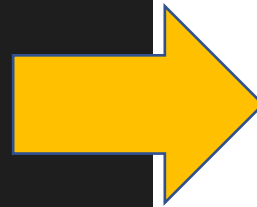
- import React from 'react';

```
class NewsHead01 extends React.Component {  
  render() {  
    return (  
      <div>  
        <h1>SW Studio News</h1>  
      </div>  
    );  
  }  
}
```

Component

- To create element with class

```
class NewsHead01 extends React.Component {  
  render() {  
    return (  
      <div className='class01'>  
        <h1>SW Studio News</h1>  
      </div>  
    );  
  }  
}
```



```
<div class='class01'>  
  <h1>SW Studio News</h1>  
</div>
```


Using Component

```
class Link01 extends Component {  
  render() {  
    return(  
      <a href="https://sv1.picz.in.th/images/2022/02/06/nM1SyZ.jpg" >Click here</a>  
    );  
  }  
}  
  
class Block01 extends Component {  
  render() {  
    return(  
      <div>  
        <Link01 />  
      </div>  
    );  
  }  
}
```



Using JavaScript with element

```
class Link03 extends Component {  
  display(ev) {  
    alert(ev.target.href);  
  }  
  render() {  
    return(  
      <a href="https://sv1.picz.in.th/images/2022/02/06/nM1SyZ.jpg"  
        onMouseOver={this.display}>Click here</a>  
    );  
  }  
}
```



The diagram consists of two blue arrows. One arrow points from the text 'Method' to the `display(ev)` method definition in the `Link03` class. The second arrow points from the `this.display` property access within the `onMouseOver` attribute of the `<a>` tag in the `render()` method back to the `display` method definition.

Event handling

```
const changecolor = event => {  
  const e = event.target;  
  e.style.color = 'blue';  
  e.style.background = 'yellow';  
}  
  
class Link01 extends Component {  
  render() {  
    return(  
      <a href="https://sv1.picz.in.th/images/2022/02/06/nM1SyZ.jpg"  
        onMouseOver={(event) => changecolor(event)}>Click here</a>  
    );  
  }  
}
```


Event handling (cont.)

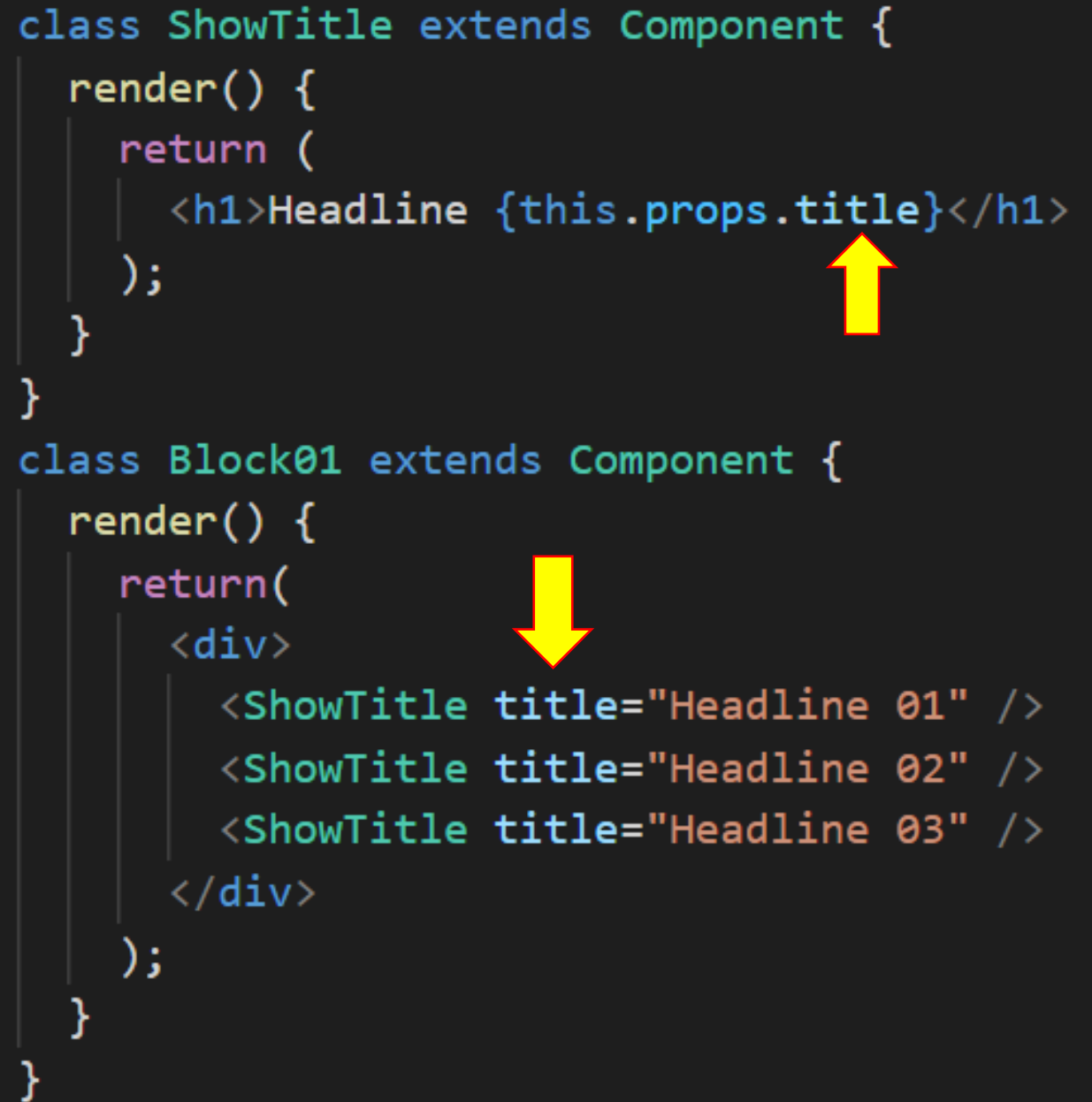
```
class Link02 extends Component {  
  display(ev) {  
    ev.target.style.color='blue';  
    ev.target.style.background = 'yellow';  
  }  
  render() {  
    return(  
      <a href="https://sv1.picz.in.th/images/2022/02/06/nM1SyZ.jpg"  
        onMouseOver={this.display}>Click here</a>  
    );  
  }  
}
```

Props

- Props = Properties
- Props is an object that stores the value
- Props can be passed from parent component to child component
 - Uni-directional (parent to child only)
 - Props' data cannot be modified by the child component (read only)

Props example

```
class ShowTitle extends Component {  
  render() {  
    return (  
      <h1>Headline {this.props.title}</h1>  
    );  
  }  
}  
  
class Block01 extends Component {  
  render() {  
    return(  
      <div>  
        <ShowTitle title="Headline 01" />  
        <ShowTitle title="Headline 02" />  
        <ShowTitle title="Headline 03" />  
      </div>  
    );  
  }  
}
```





The diagram illustrates the flow of props in a React application. A yellow arrow points from the `title` prop in the `Block01` component's render method to the `this.props.title` in the `ShowTitle` component's render method, demonstrating how data is passed from a parent component to a child component.

State

- Data storage within component
- Creates and manages by component
- State cannot be passed to other components
- ****used inside component only****
- When state changes, DOM re-rendering will occur
 - Just a part of DOM will be updated
 - Or more specific, only updated component will be re-rendered

State example

```
class ShowTitle extends Component {  
  state = {  
    datetime: new Date()   
  }  
  
  updatetime() {  
    this.setState({datetime: new Date()});   
  }  
  
  render() {  
    return (  
      <div>  
        <p>{this.state.datetime.toString()}</p>  
        <a href="https://sv1.picz.in.th/images/2022/02/06/nM1SyZ.jpg"  
          onMouseOver={this.updatetime.bind(this)}>Headline {this.props.title} </a>  
      </div>  
    );  
  }  
}
```

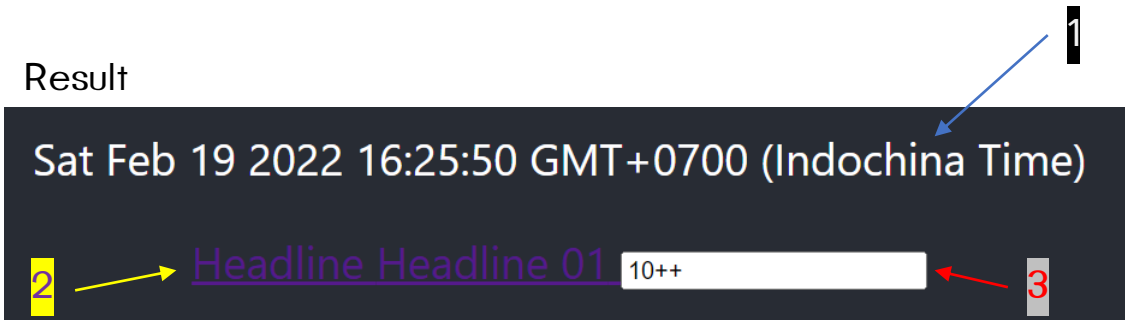


State & Props

	Props	State
Usage	Passing data	Inside the component only
Data modification	Read only	Yes, locally

More with state

Result



```
class ShowTitle extends Component {
  state = {
    ticker: false,
    count: 0,
    datetime: new Date()
  }

  updatetime() {
    var num = this.state.count;

    num++;
    if (num > 10)
      this.setState({ ticker: true });

    this.setState({
      count: num,
      datetime: new Date()
    });
  }

  render() {
    var textval = "";
    if (this.state.ticker === true)
      textval = '10++';
    else
      textval = this.state.count;

    return (
      <div>
        <p>{this.state.datetime.toString()}</p>
        <a href="https://sv1.picz.in.th/images/2022/02/06/nM1SyZ.jpg"
          onMouseOver={this.updatetime.bind(this)}>Headline {this.props.title}</a>
        <input type="text" value={textval} readOnly></input>
      </div>
    );
  }
}
```

Passing data

- From parent to child
 - Props

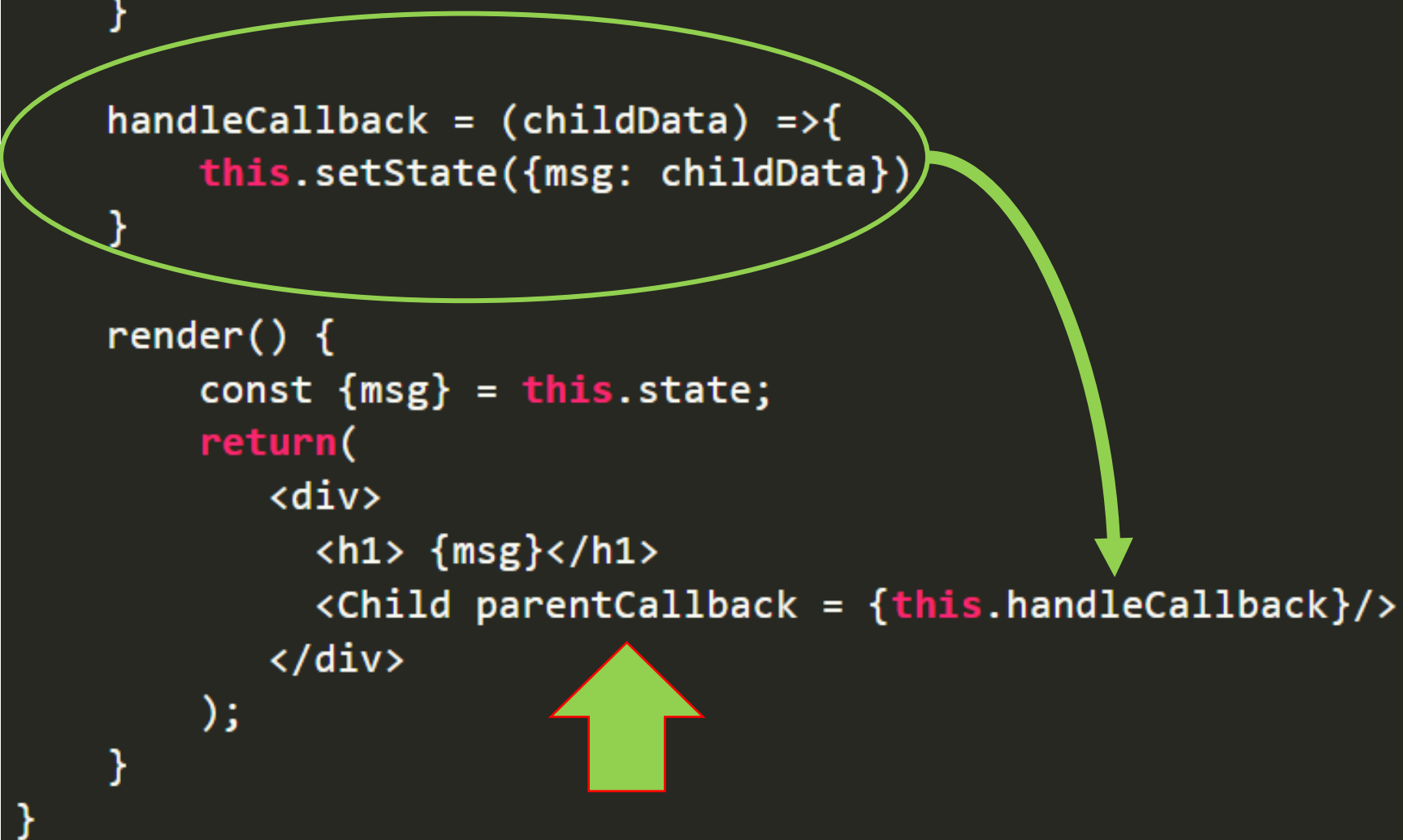
Passing data

- From child to parent
 - Callback function

```
class Parent extends React.Component{
  state = {
    msg: "",
  }

  handleCallback = (childData) =>{
    this.setState({msg: childData})
  }

  render() {
    const {msg} = this.state;
    return(
      <div>
        <h1> {msg}</h1>
        <Child parentCallback = {this.handleCallback}/>
      </div>
    );
  }
}
```



```
class Child extends React.Component {  
  onTrigger = () => {  
    this.props.parentCallback("Welcome to GFG");  
  };  
  
  render() {  
    return (  
      <div>  
        <br></br> <br></br>  
        <button onClick={this.onTrigger}>Click me</button>  
      </div>  
    );  
  }  
}
```

A green arrow originates from the `parentCallback` property in the `props` object within the `onTrigger` method and points to the `onTrigger` method definition in the `render` method. A yellow arrow points upwards to the `parentCallback` property. The `this.onTrigger` expression in the `onClick` prop is circled in green.