ReactJS Training Course (Session 2)

TUAN MAI CHUNG / NashTech

Sep/2020

Agenda

1. PROPS / STATES
2. LIFECYCLE
3. HANDLING EVENTS

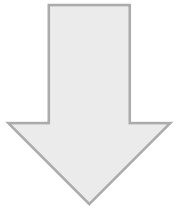
Props / States?

What is prop?

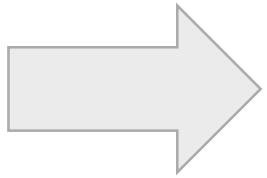
```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

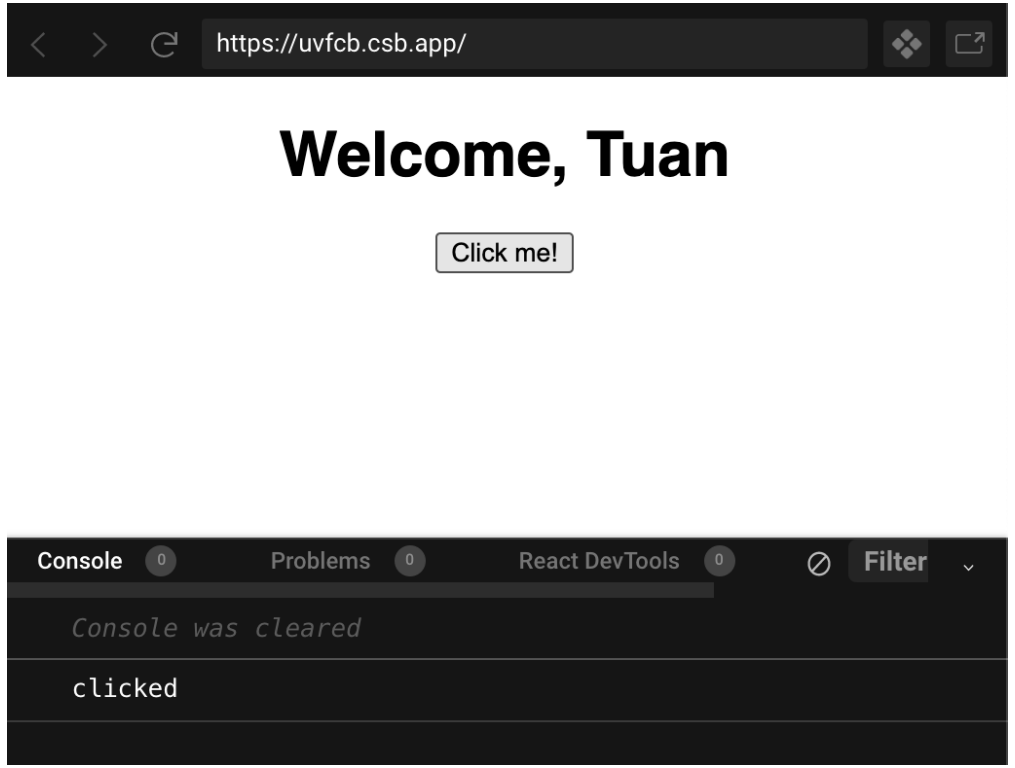
Class component with props

```jsx
export default class App extends React.Component {
  render() {
    return (
      <div className="App">
        <Welcome title="Tuan" clicked={() => console.log('clicked')} />
      </div>
    );
  }
}
```



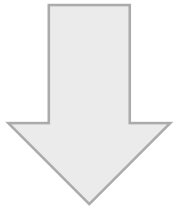
```jsx
export default class Welcome extends React.Component {
  render() {
    return (
      <div>
        <h1>Welcome, {this.props.title}</h1>
        <button onClick={() => this.props.clicked()}>Click me!</button>
      </div>
    );
  }
}
```



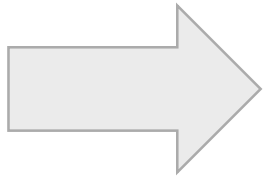


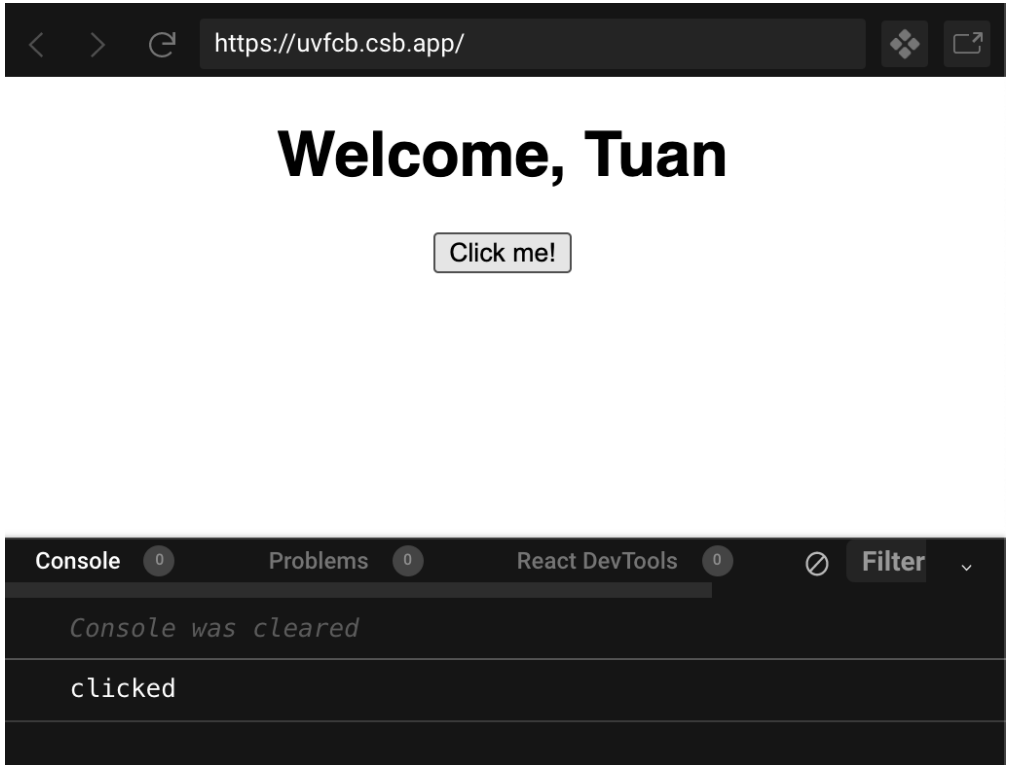
Function component with props

```
export default class App extends React.Component {
  render() {
    return (
      <div className="App">
        <Welcome title="Tuan" clicked={() => console.log('clicked')} />
      </div>
    );
  }
}
```



```
const Welcome = (props) => {
    return (
        <div>
            <h1>Welcome, {props.title}</h1>
            <button onClick={() => props.clicked()}>Click me!</button>
        </div>
    )
}
```





What is prop?

- Input (parameter) of a component
  - From Parent to Child
    - Passing a value
  - From Child to Parent
    - Passing a function
- Immutable

What is state?

- Local variable of a component
- Use **setState()** to update component's states
- Page will be re-rendered when state changed

Example of state

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

Example of state

```
class Welcome extends React.Component {
  constructor(props) {
    super(props);
    this.state = { name: "Tuan" };
  }

  updateName(newName) {
    this.setState({
      name: newName,
    });
  }

  render() {
    return (
      <div>
        <h1>Welcome, {this.state.name}</h1>
        <button onClick={() => this.updateName("Samantha")}>Update Name</button>
      </div>
    );
  }
}
```

```
class Welcome extends React.Component {
  constructor(props) {
    super(props);
    this.state = { name: "Tuan" };
  }

  updateName(newName) {
    this.state.name = newName
  }

  render() {
    return (
      <div>
        <h1>Welcome, {this.state.name}</h1>
        <button onClick={() => this.updateName("Samantha")}>Update Name</button>
      </div>
    );
  }
}
```

Example of useState()

```jsx
import React, { useState } from "react";

function Welcome() {
    const [name, setName] = useState('Tuan');

    const updateName = (newName) => {
        setName(newName);
    };

    return (
        <div>
            <h1>Welcome, {name}</h1>
            <button onClick={() => updateName('Samantha')}>Update Name</button>
        </div>
    );
}

export default Welcome;
```

Summary

- Prop is used for **external communication** with other components
- State is used for **internal communication** inside a component

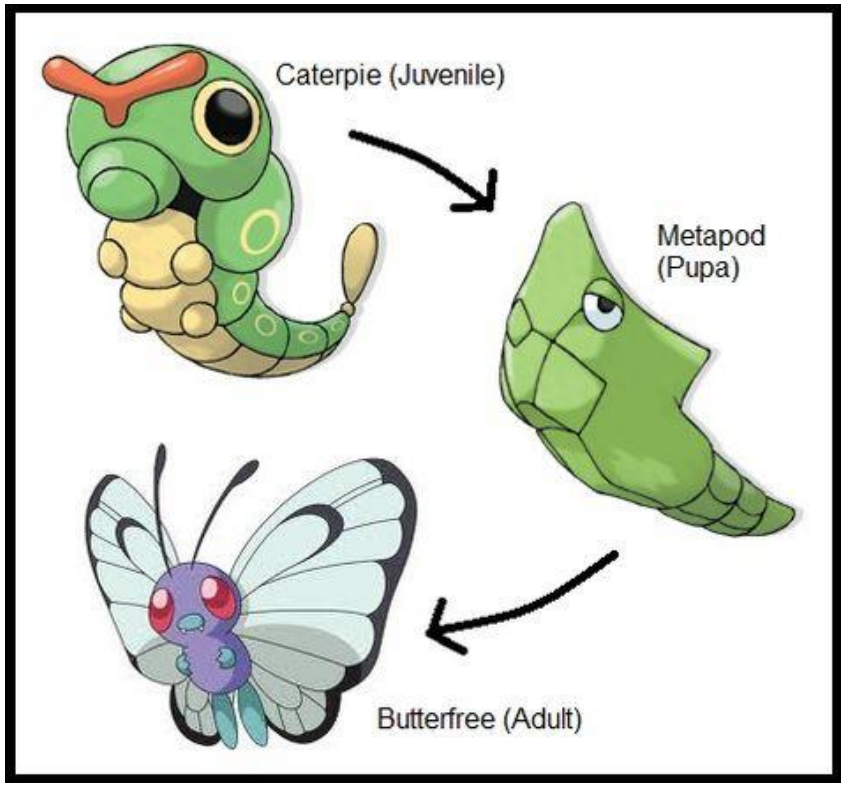
Exercise

- Create new state **cartItem** in **App.js** with following initial value:
  - *cartItem = 0*
- Add *cart* button in **Header** component
- Bind data of **cartItem** to *cart* button

Lifecycle

What is lifecycle?

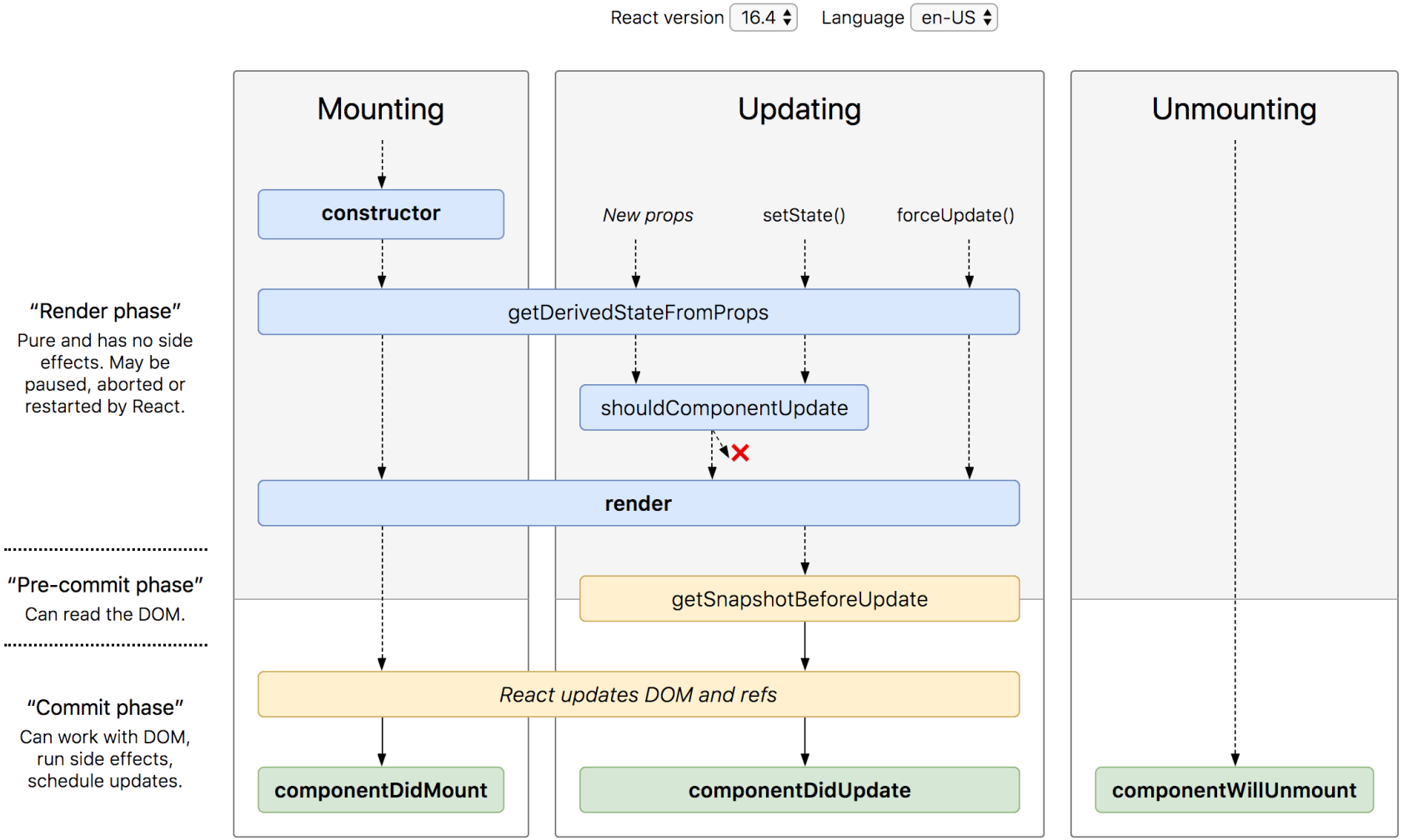
What is lifecycle?



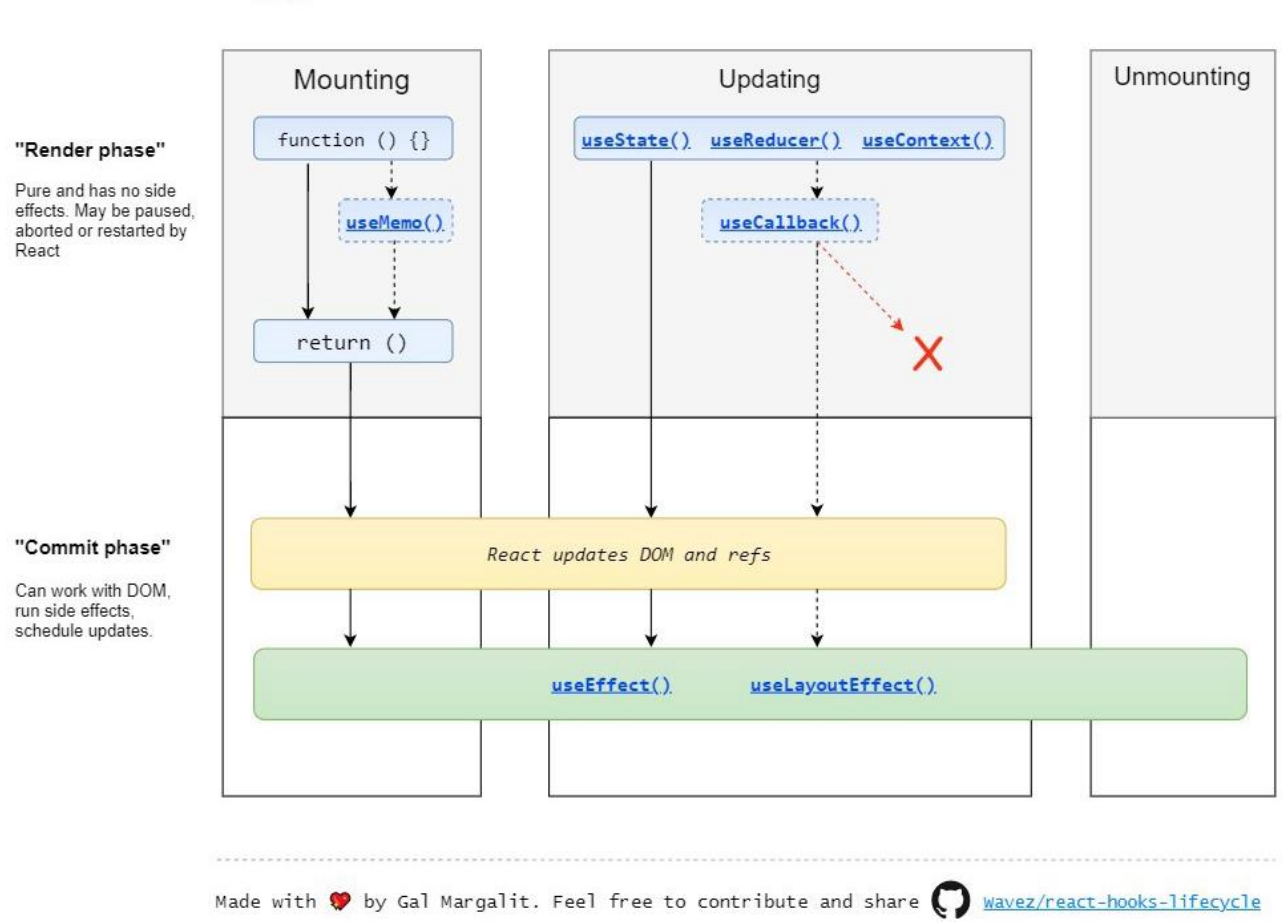
What is lifecycle?

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }

  componentDidMount() {
  }

  componentWillUnmount() {
  }

  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
}
```

Lifecycle in React



Lifecycle in React Hook



```jsx
import { useState, useEffect } from "react";

export default function App() {
  const [state, setState] = useState(0);

  console.log("Run1");

  useEffect(() => {
    console.log("Effect");
    return () => console.log("Clean up");
  });

  useEffect(() => {
    console.log("Mounted");
    return () => console.log("Unmouting");
  }, []);

  console.log("Run2");

  return (
    <div className="App">
      <h1>Hello CodeSandbox</h1>
      <h2>Start editing to see some magic happen!</h2>
      <h3>{state}</h3>
      <button onClick={() => setState((i) => i + 1)}>Click</button>
    </div>
  );
}
```

- What is the order of *console.log?*
- What is the order after clicked on button?

Exercise

- Create new states **bookItem** in **App.js**
- Apply lifecycle (*componentDidMount or useEffect()*) in **App.js** to fetch data from *bookItems.json* file to previous state
- Create `**ProductList**` in *components* folder
- Use data of **bookItem** to generate **ProductList**'s items

Handling events

Handling events

- HTML

```
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

- React

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

Handling events

- Without arrow function:
  - Need to bind `this` to *constructor* of your component
- With arrow function:
  - Do not need to bind `this` (it is automatically bound to your component)

Handling events – without arrow function

```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};

    // This binding is necessary to make `this` work in the callback
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(state => ({
      isToggleOn: !state.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? 'ON' : 'OFF'}
      </button>
    );
  }
}

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

Handling events – with arrow function

```
class LoggingButton extends React.Component {
  handleClick() {
    console.log('this is:', this);
  }

  render() {
    // This syntax ensures `this` is bound within handleClick
    return (
      <button onClick={() => this.handleClick()}>
        Click me
      </button>
    );
  }
}
```

Passing arguments to event handlers

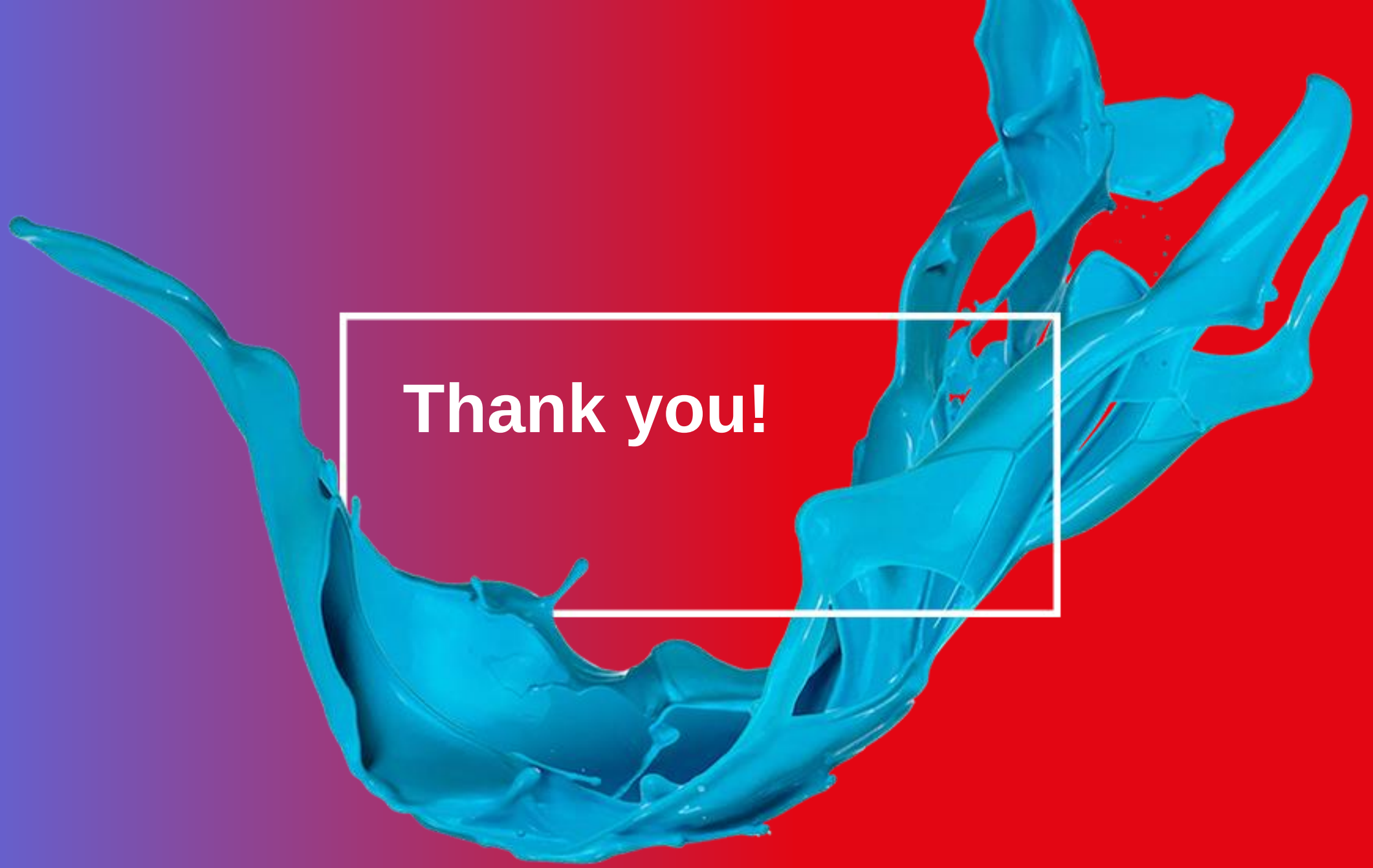
```
function App (props) {
    function handleClick(param) {
        alert('Hello, ', + param)
    }

    return (
        <div>
            <button onClick={() => handleClick('John')}>Click me</button>
        </div>
    )
}
```

Passing arguments to event handlers

```
function Welcome(props) {
  return (
    <div>
      <button onClick={() => props.clicked('John')}>Click me</button>
    </div>
  );
}

function App() {
  function handleChildClick(param) {
    alert('Hello, ' + param);
  }

  return <Welcome clicked={e => handleChildClick(e)} />;
}
```

Exercise

- Create ***Buy Now*** button for each product items
- Add event '*onItemClick(productItem)*' for ***Buy Now***
- Update cart value when user click on ***Buy Now***



Thank you!