# Lab 05
# React Review & Cookies

Software Studio

DataLab, CS, NTHU

2021 spring

# Outline

- React
- Routing
- AJAX
- Cookies
- Lab & Homework

# Outline

- React
- Routing
- AJAX
- Cookies
- Lab & Homework

# React

- Write HTML in JS
- Component based

```
render() {
    return (
        <div className={`today weather-bg ${this.state.group}`}>
            <div className={`mask ${this.state.masking ? 'masking' : ''}`}>
                <WeatherDisplay {...this.state}/>
                <WeatherForm city={this.state.city} unit={this.props.unit} onQuery={this.handleFormQuery}/>
            </div>
        </div>
    );
}
```

# Component Based in React

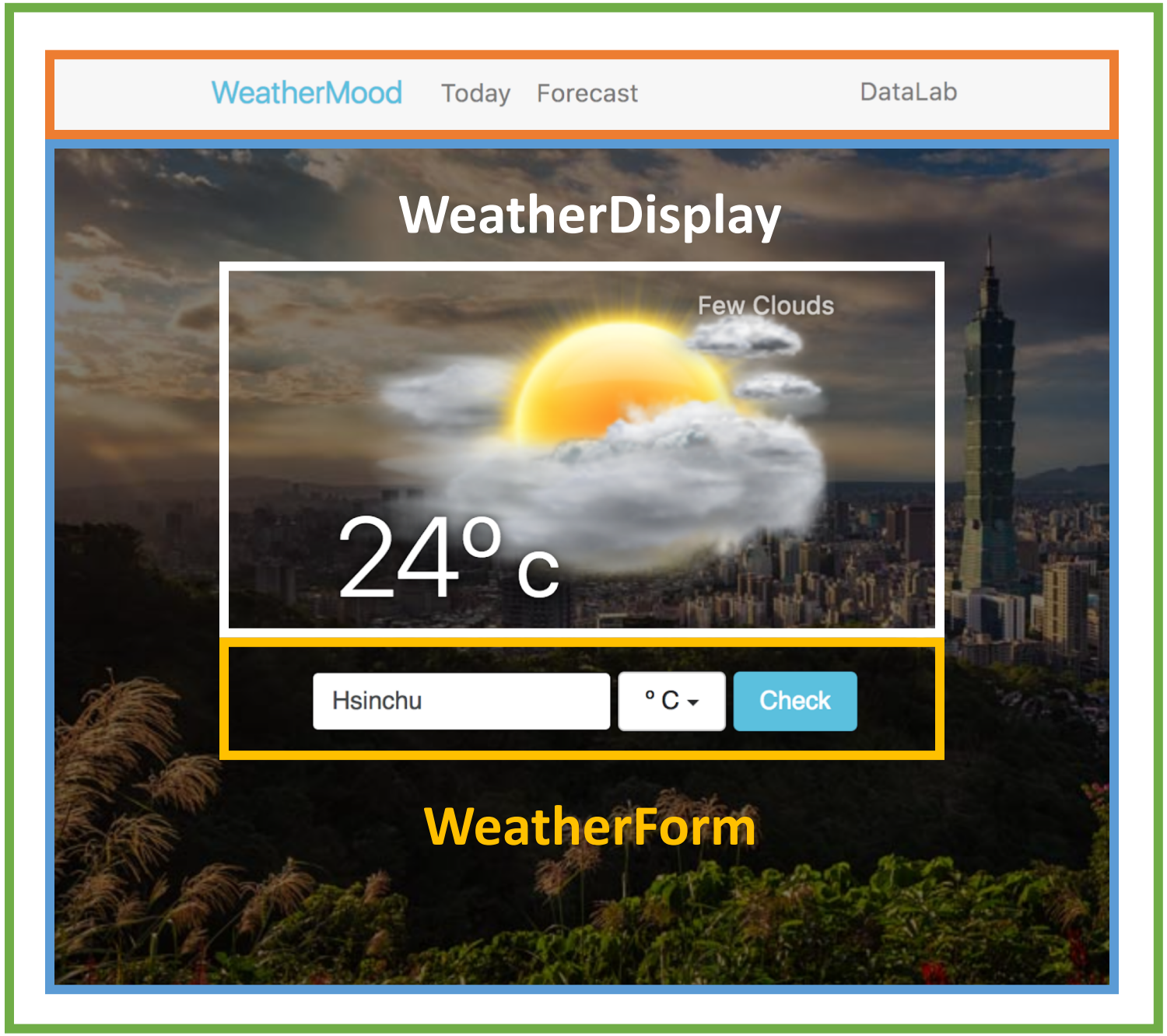
- Class

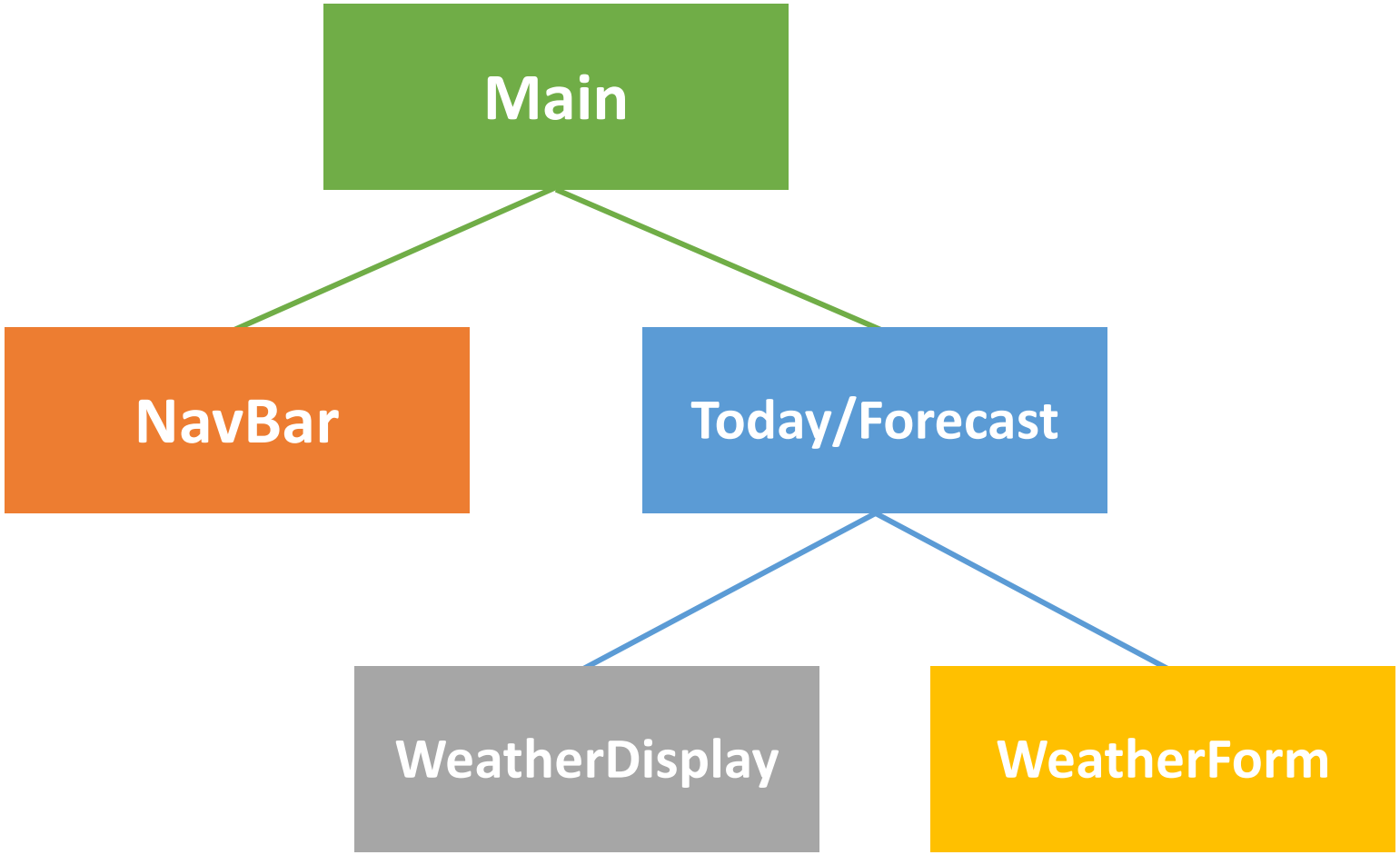
```
export default class WeatherDisplay extends React.Component {

    constructor(props) {
        super(props);

    }
```

- New an object

```
render() {
    return (
        <div className={`today weather-bg ${this.state.group}`}>
            <div className={`mask ${this.state.masking ? 'masking' : ''}`}>
                <WeatherDisplay {...this.state}/>
                <WeatherForm city={this.state.city} unit={this.props.unit} onQuery={this.handleFormQuery}/>
            </div>
        </div>
    );
}
```

# How to Communicate Between Components?


WeatherMood
Today
Forecast
DataLab
NavBar
WeatherDisplay
Few Clouds
24° c
Hsinchu
° C
Check
WeatherForm
Main
Today/Forecast


Main
NavBar
Today/Forecast
WeatherDisplay
WeatherForm

# Parent to child - Props

- Read-only
- Property from parent

# Parent to child - Props

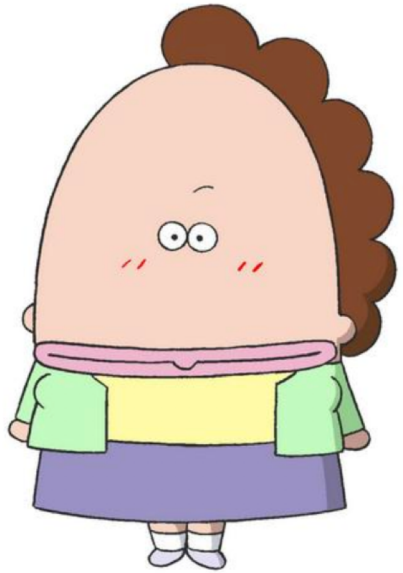
- Parent pass their states to child

```
render() {
    return (
        <div className={`today weather-bg ${this.state.group}`}>
            <div className={`mask ${this.state.masking ? 'masking' : ''}`}>
                <WeatherDisplay {...this.state}/>
                <WeatherForm city={this.state.city} unit={this.props.unit} onQuery={this.handleFormQuery}/>
            </div>
        </div>
    );
}
```

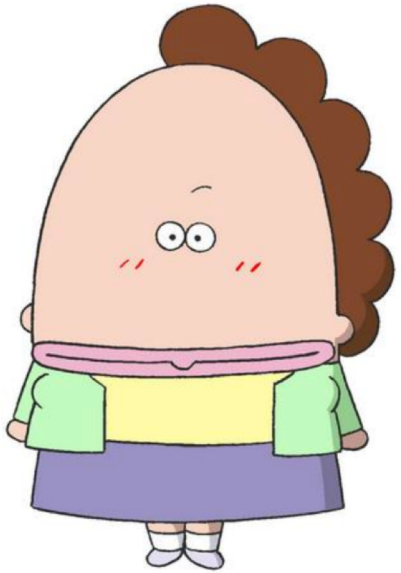
# Parent to child - Props

- Child use these properties to display

```
render() {
    return (
        <div className={`weather-display ${this.props.masking
            ? 'masking'
            : ''}`}>
            <img src={`images/w-${this.props.group}.png`}/>
            <p className='description'>{this.props.description}</p> 
            <h1 className='temp'>
                <span className='display-3'>{this.props.temp.toFixed(0)}&ordm;</span>
                 {(this.props.unit === 'metric')
                    ? 'C'
                    : 'F'}
            </h1>
        </div>
    );
}
```



CHILD



**Render!**

# Child to Parent - State

- Can be modified
- If state is used in render(), when you call setState(), it will trigger re-render automatically.

# Child to Parent - State

- Parent will have their own states

```
export default class Today extends React.Component {

    static getInitWeatherState() {
        return {
            city: 'na',
            code: -1,
            group: 'na',
            description: 'N/A',
            temp: NaN
        };
    }

    constructor(props) {
        super(props);

        this.state = {
            ...Today.getInitWeatherState(),
            loading: true,
            masking: true
        };

        this.handleFormQuery = this.handleFormQuery.bind(this);
    }
```

# Child to Parent - State

- Parent will have their own states

```
export default class Today extends React.Component {

    static getInitWeatherState() {
        return {
            city: 'na',
            code: -1,
            group: 'na',
            description: 'N/A',
            temp: NaN
        };
    }

    constructor(props) {
        super(props);

        this.state = {
            ...Today.getInitWeatherState(),
            loading: true,
            masking: true
        };

        this.handleFormQuery = this.handleFormQuery.bind(this);
    }
```

# Child to Parent - State

- They will set a function to handle the event

```
getWeather(city, unit) {
    this.setState({
        loading: true,
        masking: true,
        city: city // set city state immediately to prevent input text (in WeatherForm) from blinking;
    }, () => { // called back after setState completes
        getWeather(city, unit).then(weather => {
            this.setState({
                ...weather,
                loading: false
            }, () => this.notifyUnitChange(unit));
        }).catch(err => {
            console.error('Error getting weather', err);

            this.setState({
                ...Today.getInitWeatherState(unit),
                loading: false
            }, () => this.notifyUnitChange(unit));
        });

handleFormQuery(city, unit) {
    this.getWeather(city, unit);
}
```

# Child to Parent - State

- They will send this property to the child but actually, it is a handle function

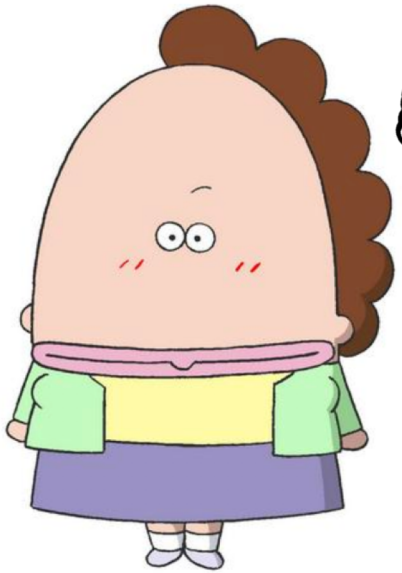
```
render() {
    return (
        <div className={`today weather-bg ${this.state.group}`}>
            <div className={`mask ${this.state.masking ? 'masking' : ''}`}>
                <WeatherDisplay {...this.state}/>
                <WeatherForm city={this.state.city} unit={this.props.unit} onQuery={this.handleFormQuery}/>
            </div>
        </div>
    );
}
```

# Child to Parent - State

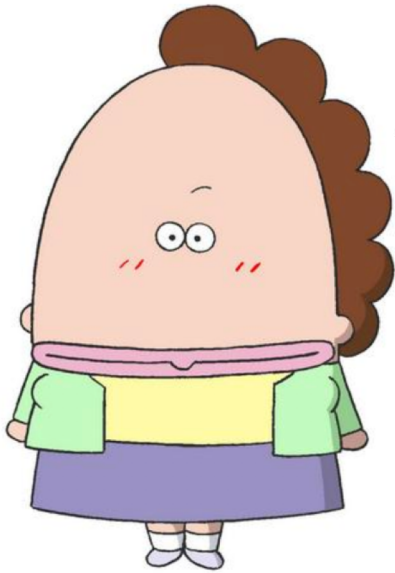
- The child only need to call this property to handle the event

```
handleSubmit(e) {
    e.preventDefault();

    this.inputEl.blur();
    if (this.state.inputValue && this.state.inputValue.trim()) {
        this.props.onQuery(this.state.inputValue, this.state.unit);
    } else {
        this.state.inputEl = this.props.city;
    }
}
```

codebook

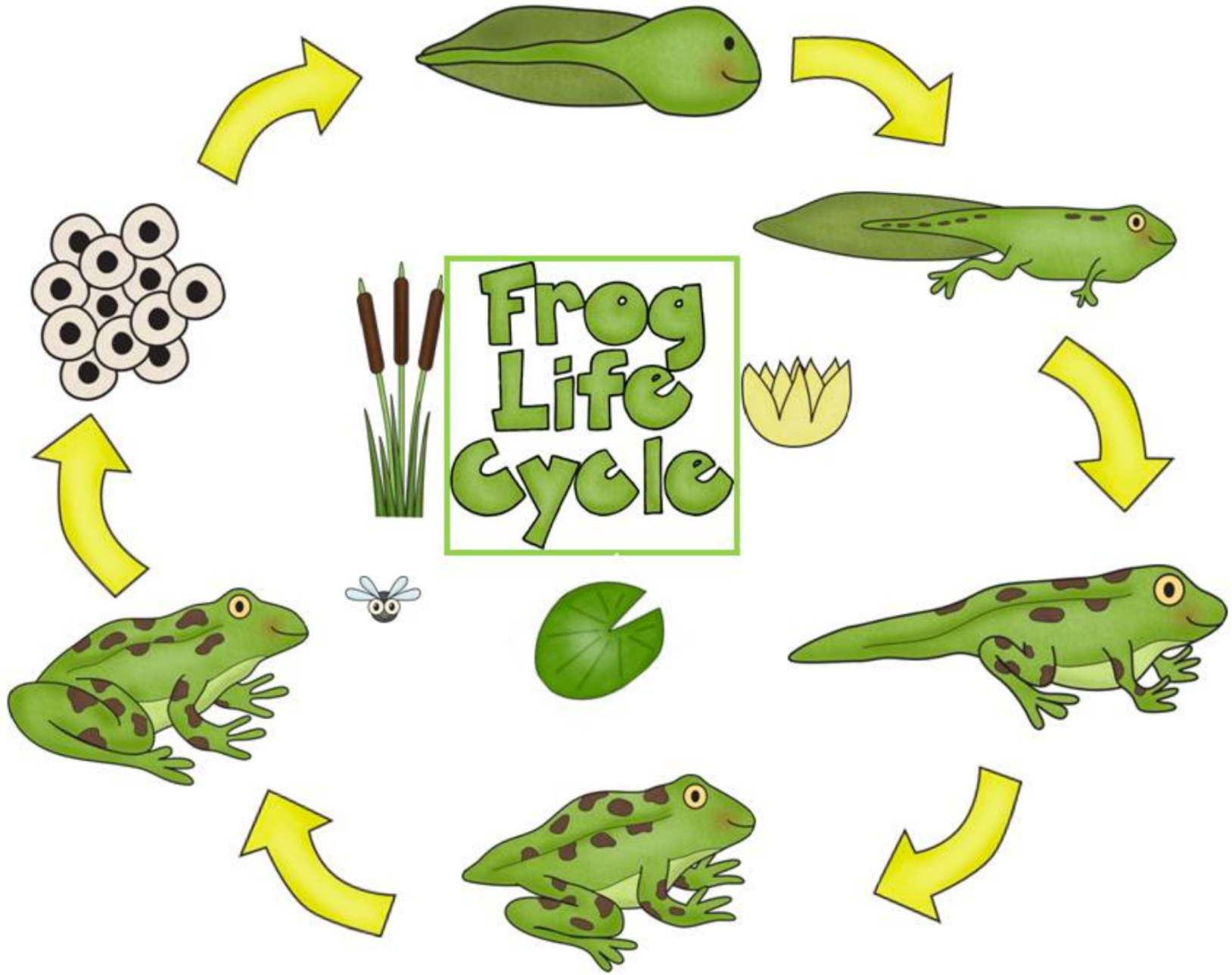


1!

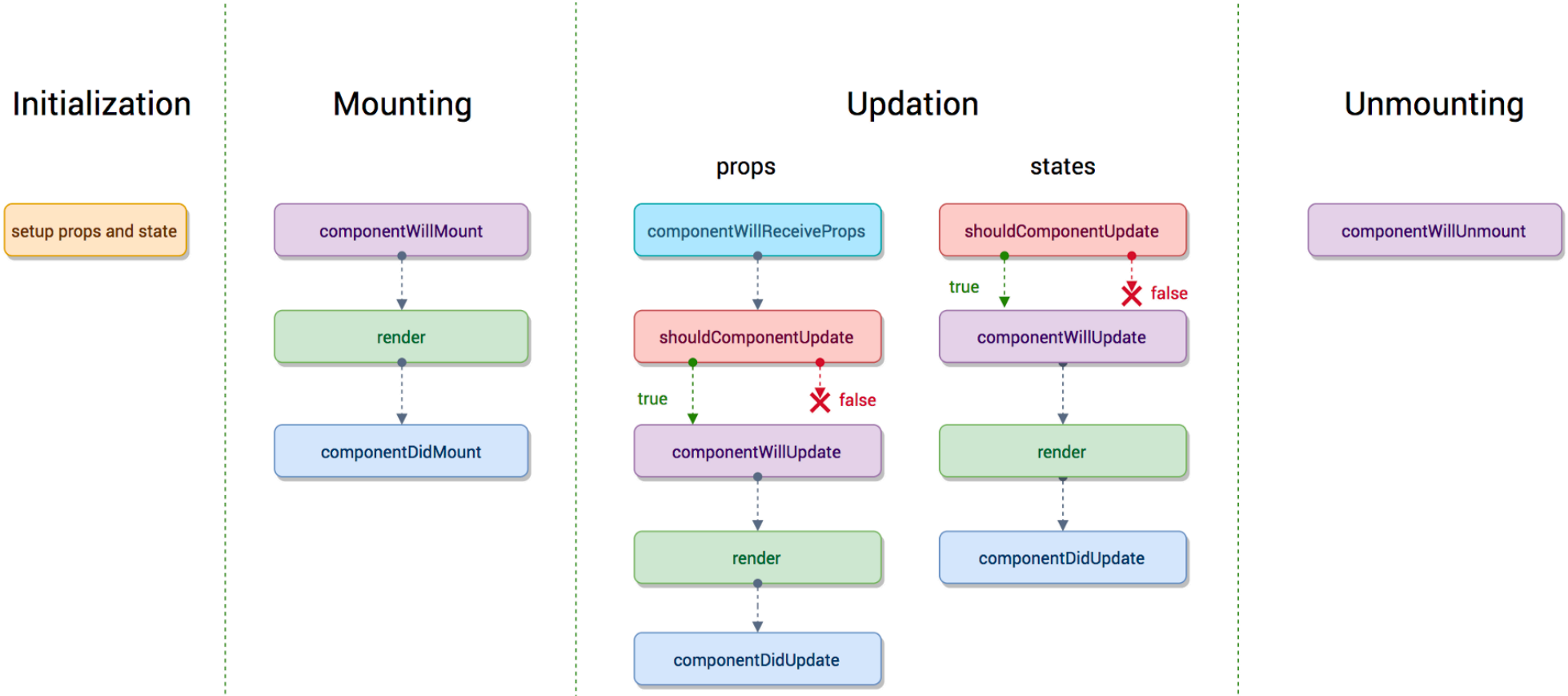


codebook

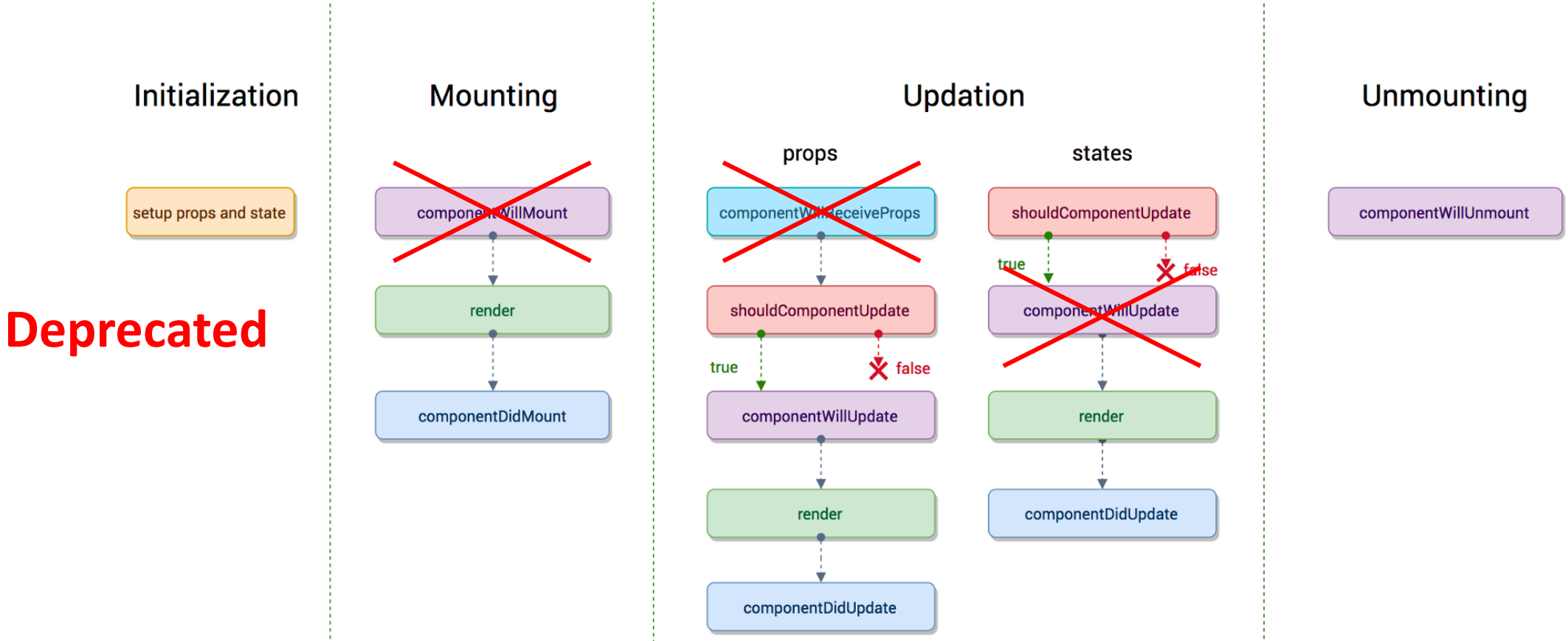
# If I Want to Call Method In the Different Stage of Rendering?


Frog
Life
Cycle

# Component Life Cycle



# Component Life Cycle



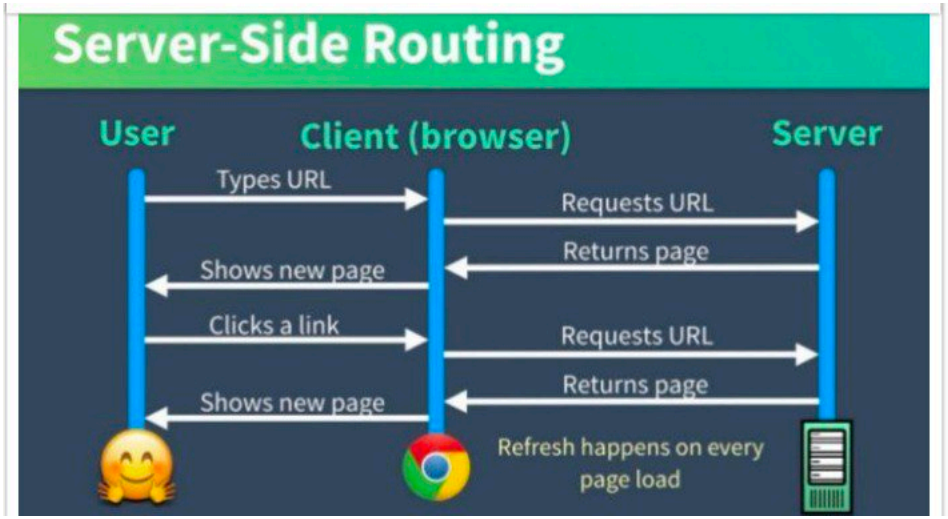

**Deprecated**

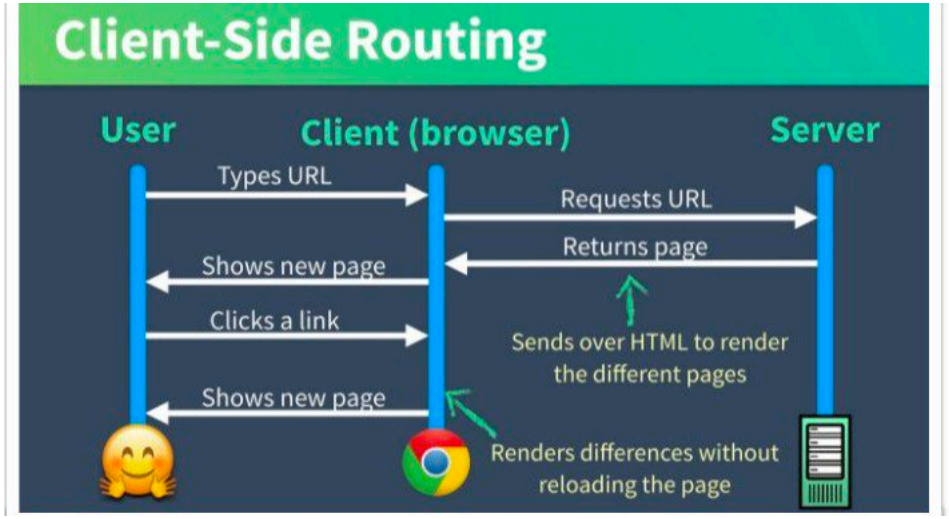
# Outline

- React
- Routing
- AJAX
- Cookies
- Lab & Homework

# Routing

- Server-side routing
  - Send a HTTP Request again to link to the target page
- Client-side routing
  - Change the component in the page by firing an event



# Client-side routing

- A Link component
  - Fire an event when clicked
  - Tells which path to go
- Container component can mount or unmount relevant component

# Client-side routing

- Main.jsx

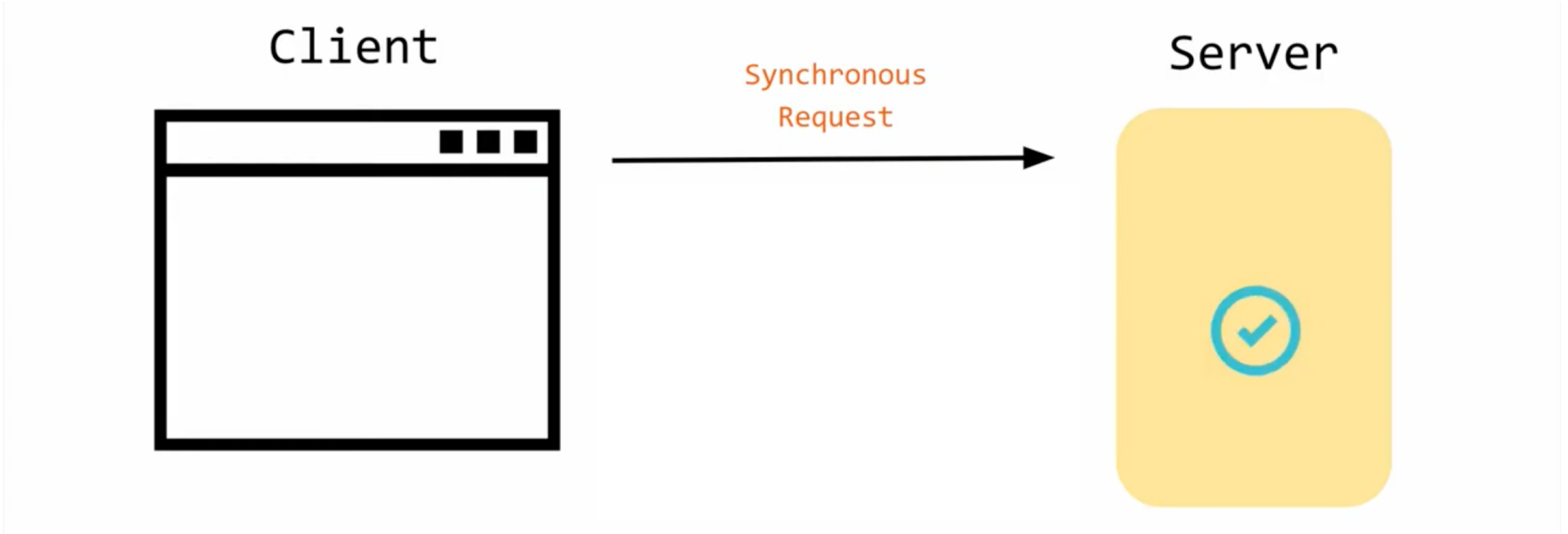
```
render() {
    return (
        <Router>
            <div className={`main bg-faded ${this.state.group}`}>
                <div className='container'>
                    <Navbar color="faded" light expand="md">
                        <NavbarBrand className='text-info' href="/">WeatherMood</NavbarBrand>
                        <NavbarToggler onClick={this.handleNavbarToggle}/>
                        <Collapse isOpen={this.state.navbarToggle} navbar>
                            <Nav navbar>
                                <NavItem>
                                    <NavLink tag={Link} to='/'>Today</NavLink>
                                </NavItem>
                                <NavItem>
                                    <NavLink tag={Link} to='/forecast'>Forecast</NavLink>
                                </NavItem>
                            </Nav>
                            <span className='navbar-text ml-auto'>DataLab</span>
                        </Collapse>
                    </Navbar>
                </div>

                <Route exact path="/" render={() => (
                    <Today unit={this.state.unit} onUnitChange={this.handleUnitChange} />
                )}/>
                <Route exact path="/forecast" render={() => (
                    <Forecast unit={this.state.unit} onUnitChange={this.handleUnitChange} />
                )}/>
            </div>
        </Router>
    );
}
```

# Outline

- React
- Routing
- AJAX
- Cookies
- Lab & Homework

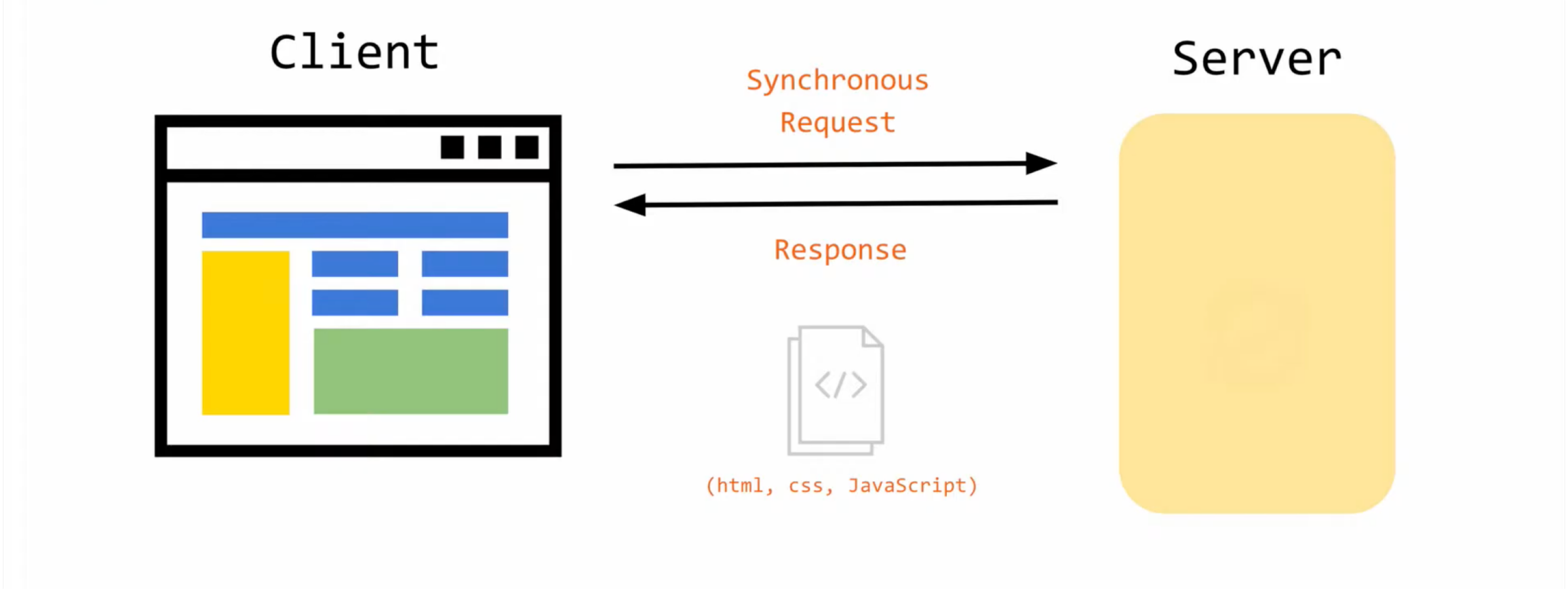
# Request

- Synchronous request



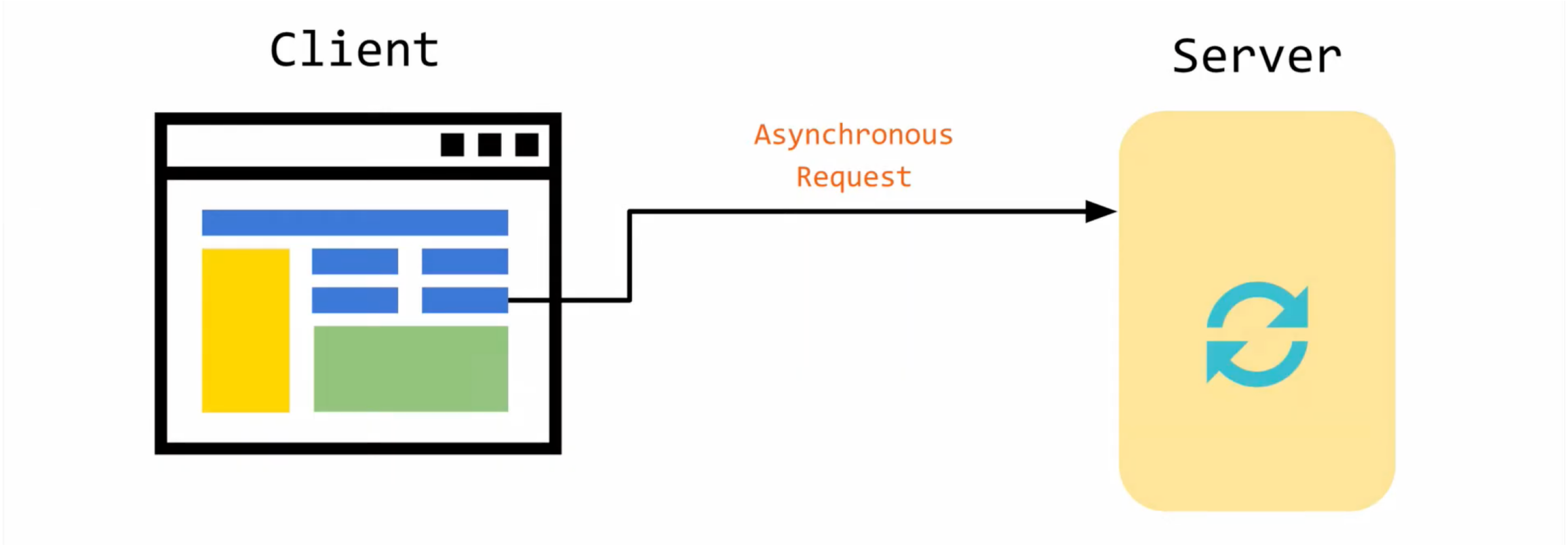
# Request

- Synchronous request



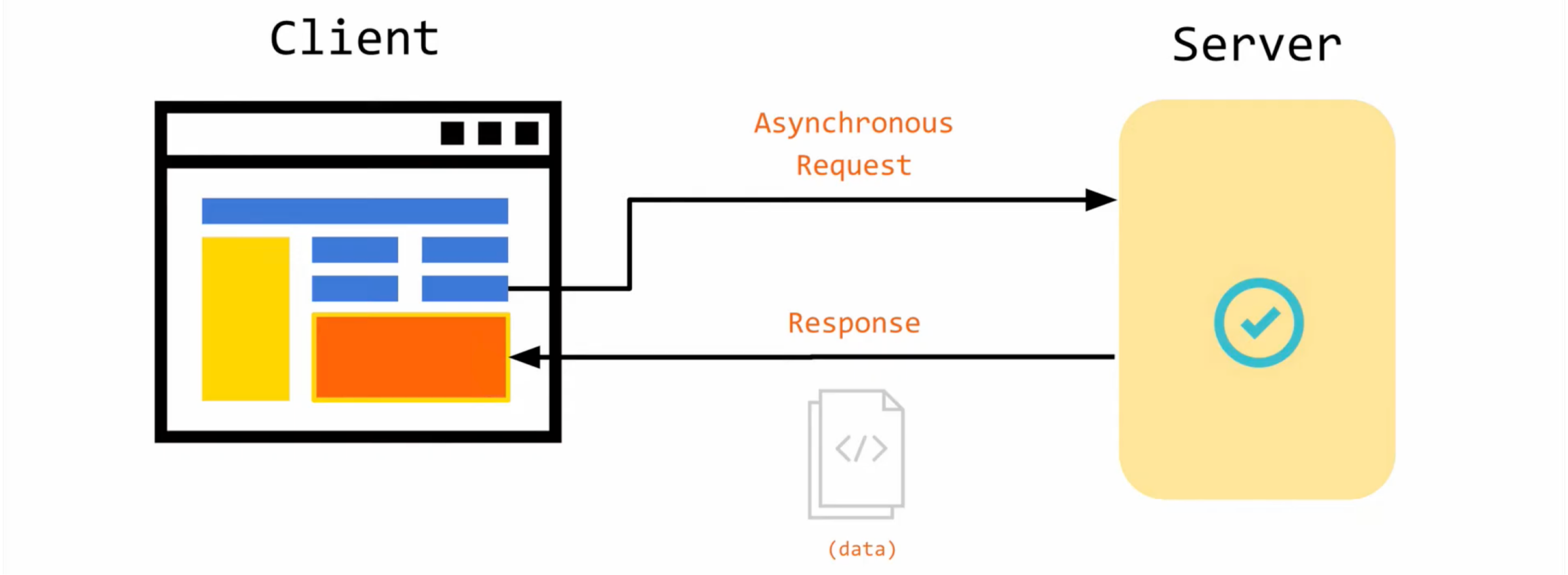
# Request

- Asynchronous Request



# Request

- Asynchronous Request



# Promise

- An object represents the eventual completion (or failure) of an asynchronous operation and its resulting value

```
const promise1 = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('foo');
  }, 300);
});

promise1.then((value) => {
  console.log(value);
  // expected output: "foo"
});

console.log(promise1);
// expected output: [object Promise]
```

```
> [object Promise]
> "foo"
```

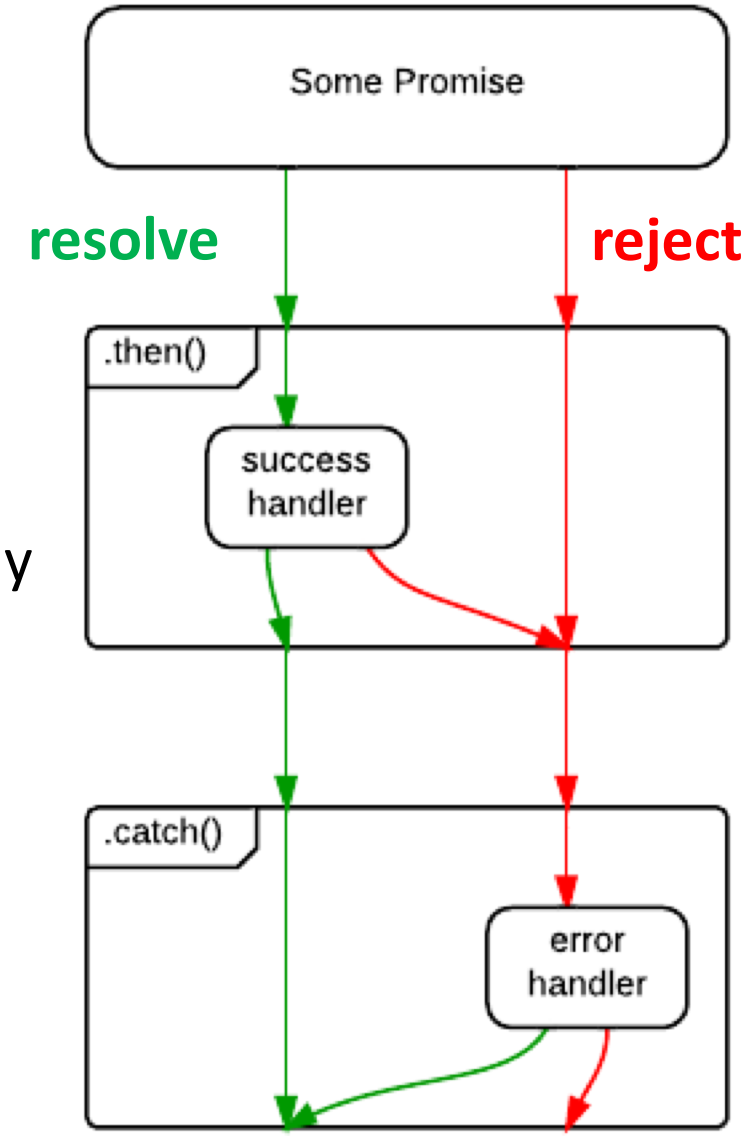
# Promise

```
// in method1()
const p = new Promise((resolve, reject) =>
{ ... // do asynchronous job here
  if (success) resolve(data);
  else reject(err);
});
return p;
```

```
// in method2(p)
const p2 = p.then(data => {
   ... // process data
  return data2
}); // always returns a new Promise
return p2;

// in method3(p2)
p2.then(data2 => {
  ... process data2
}).catch(err => {
  ... // handle err
}); // always returns a new Promise
```

# Promise - Execution State

- Pending
  - Initial state, neither fulfilled nor rejected
- Fulfilled
  - Meaning that the operation was completed successfully
- Rejected
  - Meaning that the operation failed



# Axios

- Promise based library that can send an asynchronous request

```
const axios = require('axios');

// Make a request for a user with a given ID
axios.get('/user?ID=12345')
  .then(function (response) {
    // handle success
    console.log(response);
  })
  .catch(function (error) {
    // handle error
    console.log(error);
  })
```

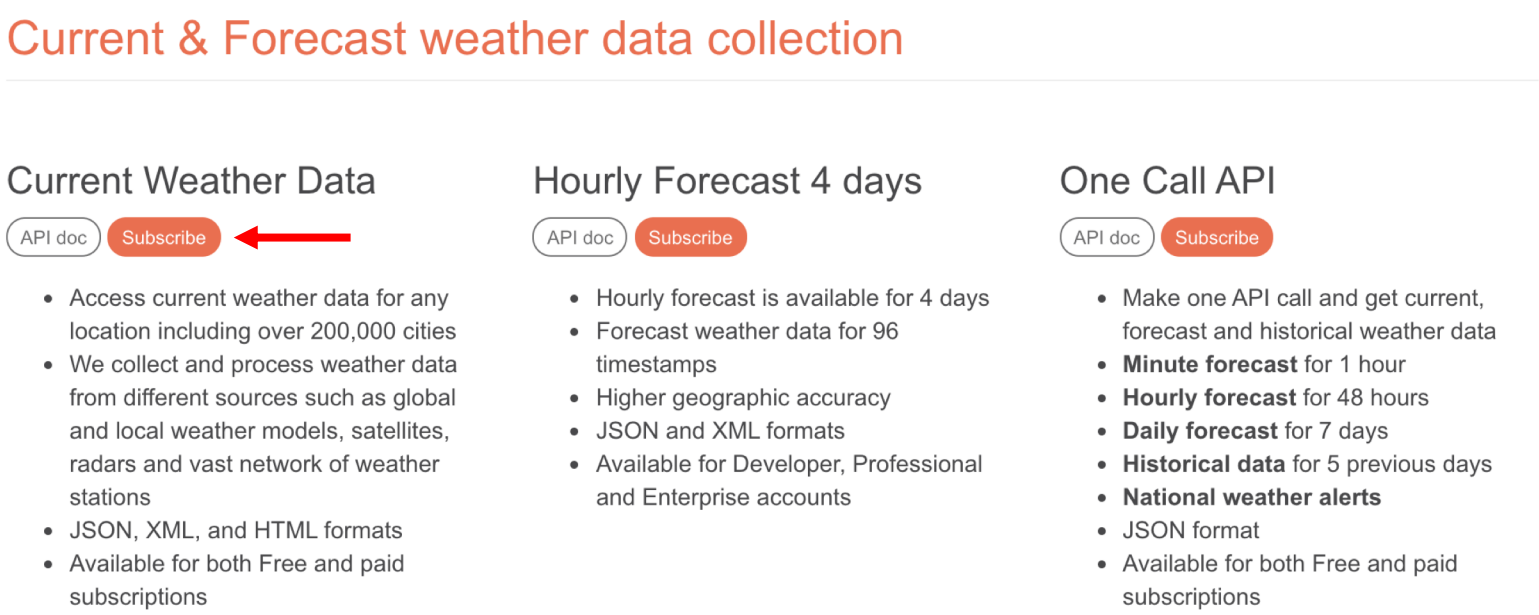
# Axios

- Cancel token

```
const CancelToken = axios.CancelToken;
const source = CancelToken.source();

axios.get('/user/12345', {
  cancelToken: source.token
}).catch(function (thrown) {
  if (axios.isCancel(thrown)) {
    console.log('Request canceled', thrown.message);
  } else {
    // handle error
  }
});

axios.post('/user/12345', {
  name: 'new name'
}, {
  cancelToken: source.token
})

// cancel the request (the message parameter is optional)
source.cancel('Operation canceled by the user.');
```

# API Key

- Open weather



# API Key

## Current weather and forecasts collection

| Free | Startup | Developer | Professional | Enterprise |
|---|---|---|---|---|
| | 40 USD / month | 180 USD / month | 470 USD / month | 2,000 USD / month |
| Get API key | Subscribe | Subscribe | Subscribe | Subscribe |
| **60 calls**/minute<br>**1,000,000 calls**/month | **600 calls**/minute<br>**10,000,000 calls**/month | **3,000 calls**/minute<br>**100,000,000 calls**/month | **30,000 calls**/minute<br>**1,000,000,000 calls**/month | **200,000 calls**/minute<br>**5,000,000,000 calls**/month |
| **Current Weather** | Current Weather | Current Weather | Current Weather | Current Weather |
| **Minute Forecast 1 hour*** | Minute Forecast 1 hour** | Minute Forecast 1 hour | Minute Forecast 1 hour | Minute forecast 1 hour |
| **Hourly Forecast 2 days*** | Hourly Forecast 2 days** | Hourly Forecast 4 days | Hourly Forecast 4 days | Hourly Forecast 4 days |

# API Key

- Open open-weather-map.js file
- Paste your api key

```
// TODO replace the key with yours
const key = "{Your own key}";
const baseUrl = `http://api.openweathermap.org/data/2.5/weather?appid=${key}`;
```

# API Key

- JSON file


```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 282.55,
    "feels_like": 281.86,
    "temp_min": 280.37,
    "temp_max": 284.26,
    "pressure": 1023,
    "humidity": 100
  },
  "timezone": -25200,
  "id": 420006353,
  "name": "Mountain View",
  "cod": 200
}
```

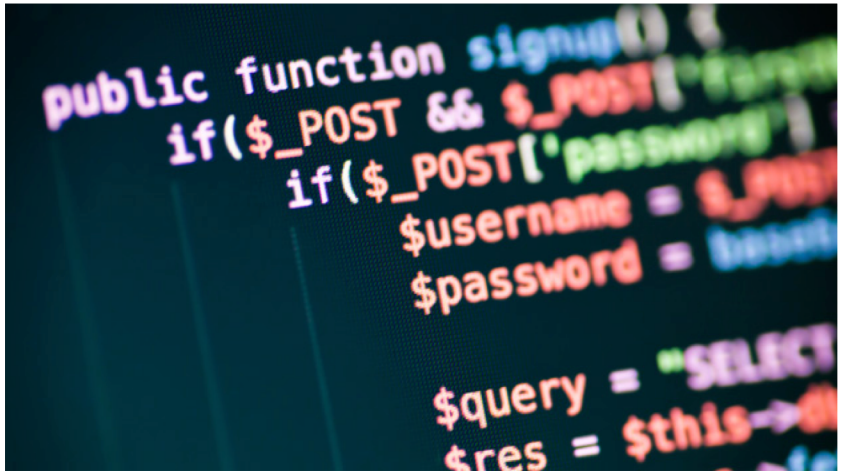


```
{
  "cod":401,
  "message": "Invalid API key..."
}
```

# Outline

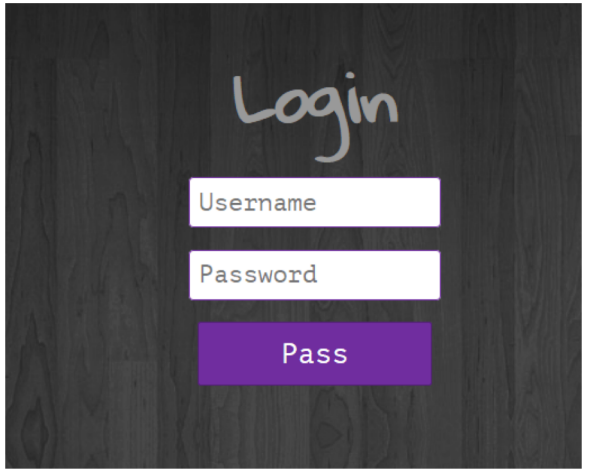
- React
- Routing
- AJAX
- Cookies
- Lab & Homework

# How do We Keep User's Info?




# Cookies

- Small piece of data stored locally
- String only
- Expire when turn off the browser
- name=value; expires=date; path=path; domain=domain; secure



# Some constraint

- Too small (only 4KB)
- Pass as http request header
- Not secure

# Using Cookie via JS

```
<script>
    if(navigator.cookieEnabled) {
        console.log("網頁的 cookie: " + document.cookie);
    } else {
        console.log("不支援 cookie");
    }
</script>
```

# Using Cookie via React

- Step 1: npm install react-cookie
- Step 2: import
- Step 3: use their api call
  - cookie.save(name, val)
  - cookie.load(name)
  - cookie.remove(name)

# Using Cookie via React

```jsx
// Root.jsx
import React from 'react';
import App from './App';
import { CookiesProvider } from 'react-cookie';

export default function Root() {
  return (
    <CookiesProvider>
      <App />
    </CookiesProvider>
  );
}
```

```jsx
// App.jsx
import React, { Component } from 'react';
import { instanceOf } from 'prop-types';
import { withCookies, Cookies } from 'react-cookie';

import NameForm from './NameForm';

class App extends Component {
  static propTypes = {
    cookies: instanceOf(Cookies).isRequired
  };

  constructor(props) {
    super(props);

    const { cookies } = props;
    this.state = {
      name: cookies.get('name') || 'Ben'
    };
  }

  handleNameChange(name) {
    const { cookies } = this.props;

    cookies.set('name', name, { path: '/' });
    this.setState({ name });
  }

  render() {
    const { name } = this.state;
```

# Lab & HW