香 港 中 文 大 學
The Chinese University of Hong Kong

CSCI2720 2023-24 Term 1: Building Web Applications

Lab 5: Basic React

Dr Colin Tsang

Outline

- Tools to prepare
- Hello World
- Component by component
- Looping through an array

- Additional materials:
  - Handling events
  - Using states
  - Conditional rendering

Tools to prepare

- You need to get these ready:
  - Google Chrome
  - React Developers Tools (a Google Chrome extension)
    - https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi
  - Simple Web Server
    - Otherwise, you will run into CORS error

Hello World

- Download the zip file *lab05.zip* from Blackboard
- The zip file includes:
  - **index.html**
    - React, reactDOM, Babel, and Bootstrap are already included
  - **Images**
    - Some pictures of CUHK to showcase in your React app later
- You need to prepare a JSX file *app.jsx* in the same directory
  - The names of *app.jsx* and *#app* in the HTML are arbitrary
  - You could use any name you wish in your own development

Hello World

- In *app.jsx*, you only need one statement now for testing:

```
const root = ReactDOM.createRoot(document.querySelector("#app"));
root.render( <h1>Hello World</h1> );
```

- Setup a local web server via the Simple Web Server. It should contain:
  - A *folder named images* holding 5 CUHK images
  - *index.html*
  - *app.jsx*
- View the website via Simple Web Server (i.e., http://localhost:xxxx)
  - You should see the “Hello World” rendered.
- Check out the React Developer Tools with the new *Components tab* and *Profiler tab*
  - The new tabs appear in the right-hand-side
  - Nothing in these tabs yet

The first component

- Now, put this into your jsx file:

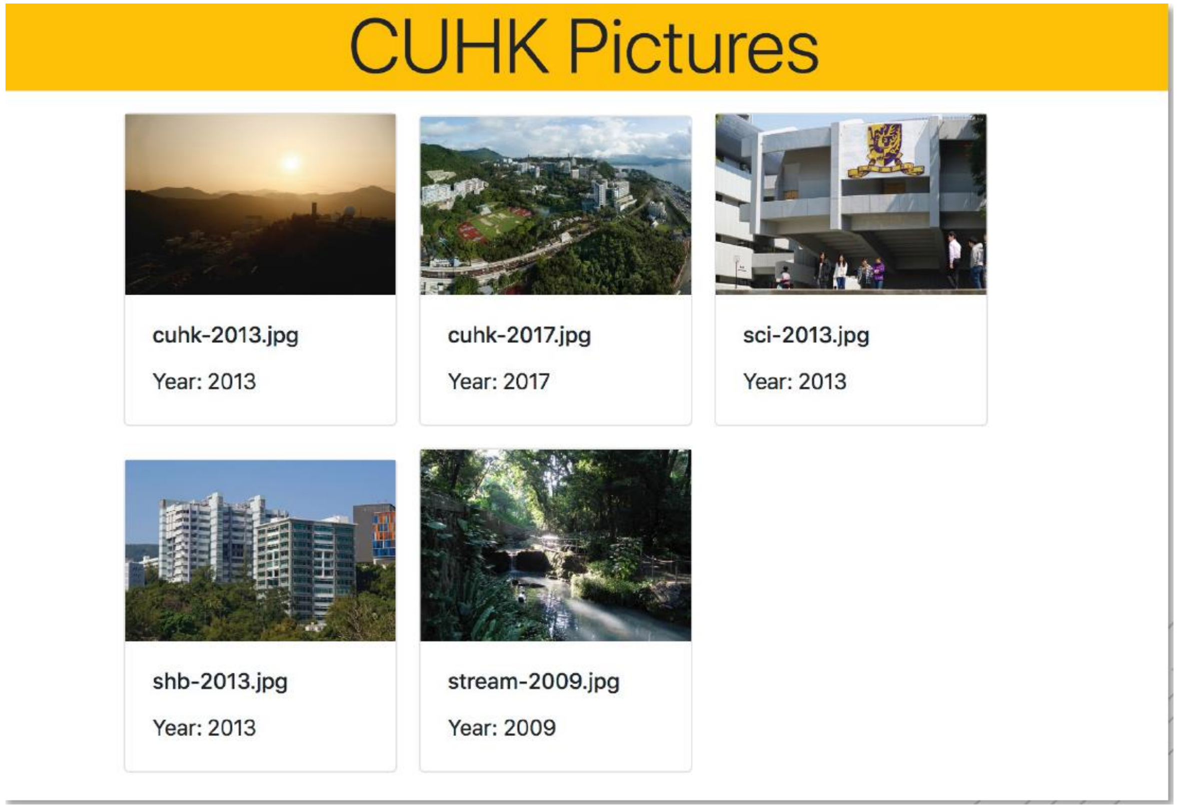
```
class App extends React.Component{
    render(){
        return <h1>Hello World</h1>;
    }
}
```

- And adjust your **root.render()** line to **root.render(<App />);**
  - This line must come after the definition of the App class, otherwise *<App />* cannot be found

- Now, you should be able to see the component *App* appears in the React Developer Tool's *Components tab*.

Start creating our app

- Our goal looks like this:



CUHK Pictures

cuhk-2013.jpg

Year: 2013

cuhk-2017.jpg

Year: 2017

sci-2013.jpg

Year: 2013

shb-2013.jpg

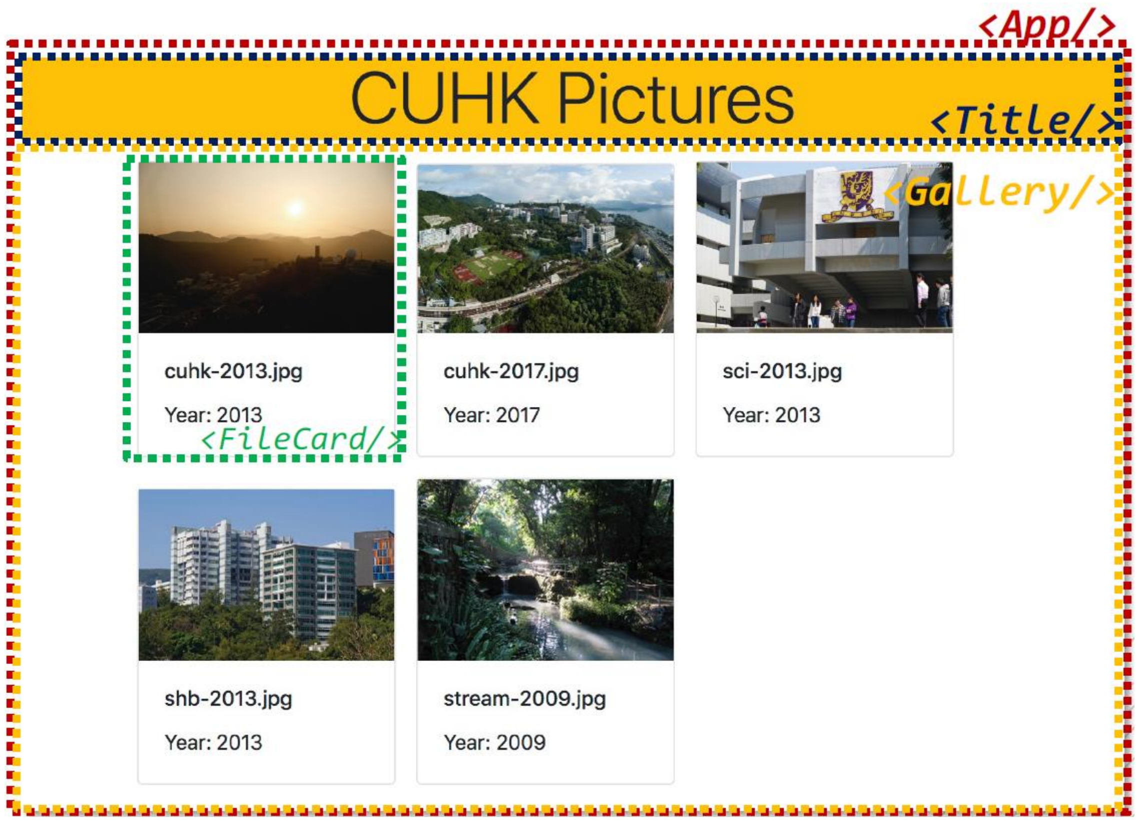
Year: 2013

stream-2009.jpg

Year: 2009

Start creating our app

- You need to divide your app into components and build them one by one



The <App/> component

- Use this code for your class App

```
class App extends React.Component{
    render(){
        return(
            <>
                <Title name={this.props.name}/>
                <Gallery />
            </>
        )
    }
}
```

- The *name props* comes from an attribute setting in the parent:
  - `root.render( <App name="CUHK pictures" />);`
  - You only need one line of **root.render()**, so add the attribute by editing but not adding an extra one.
- You can't see the result yet, as **Title** and **Gallery** are *not yet defined*

The <Title/> component

- The title component inherits the *name props* from App (passing by parent)
- Here the styling is done with Bootstrap classes
  - Note: use *className* instead of *class* for the CSS classes.

```
class Title extends React.Component {
    render() {
        return (
            <header className="bg-warning">
                <h1 className="display-4 text-center">{this.props.name}</h1>
            </header>
        );
    }
}
```

The <Gallery/> component

- The gallery component is merely a container for the contents we build later
- We better put some debugging text here before moving on

```
class Gallery extends React.Component {
    render() {
        return (
            <main className="container">Is my code ok?</main>
        );
    }
}
```

- Now refresh you page in Chrome, and you should be able to see the skeleton rendered
    - More components are seen under *Developer Tools >> Components tab*

Preparing the data

- Set up a simple data variable for the file information
  - Hardcoding isn't a good idea for actual production, but good enough for this lab.

```
const data = [
    {filename: "cuhk-2013.jpg", year:2013, remarks: "Sunset over CUHK"},
    {filename: "cuhk-2017.jpg", year:2017, remarks: "Bird's-eye view of CUHK"},
    {filename: "sci-2013.jpg", year:2013, remarks: "The CUHK Emblem"},
    {filename: "shb-2013.jpg", year:2013, remarks: "The Engineering Buildings"},
    {filename: "stream-2009.jpg", year:2009, remarks: "Nature hidden in the campus"},
    ];
```

- This is an array of objects
- This global **const** variable should be in the top of the file

Bootstrap cards

- We would like to show each image as a Bootstrap card
  - Ref: https://getbootstrap.com/docs/5.2/components/card/
  - For one card (i.e., **data[0]**), the structure would be

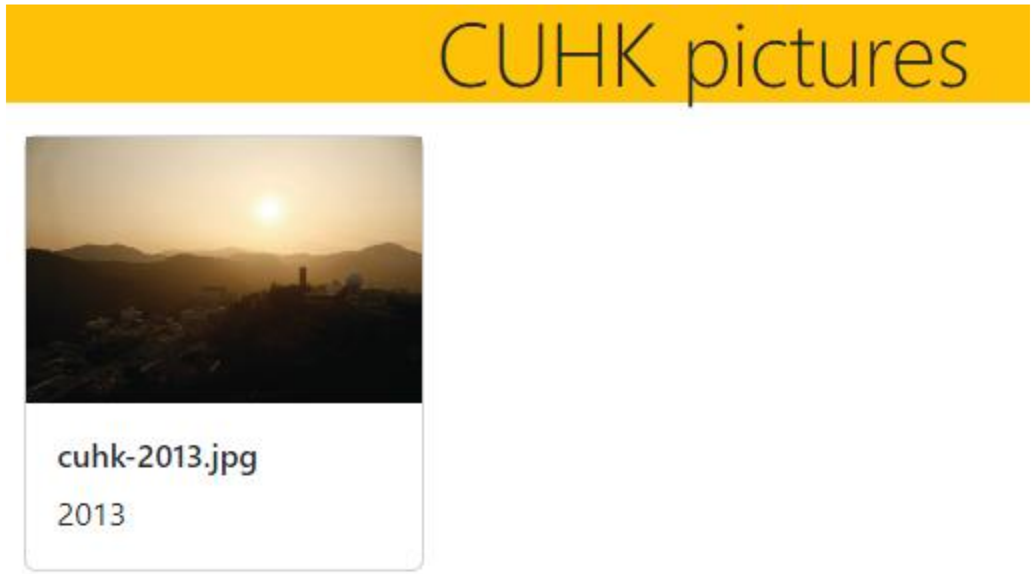
```
<div className="card d-inline-block m-2" style={{width:200}}>
   <img src={"images/"+data[0].filename} className="w-100" />
   <div className="card-body">
      <h6 className="card-title">{data[0].filename}</h6>
      <p className="card-text">{data[0].year}</p>
   </div>
</div>
```

  - Note: mind the special closing of **<img/>**

- Create a new class called **<FileCard/>** and render the above code.
- Inside **<main>** in **<Gallery/>**, render the **<FileCard/>**

Bootstrap cards

- You should see a result looks like this:



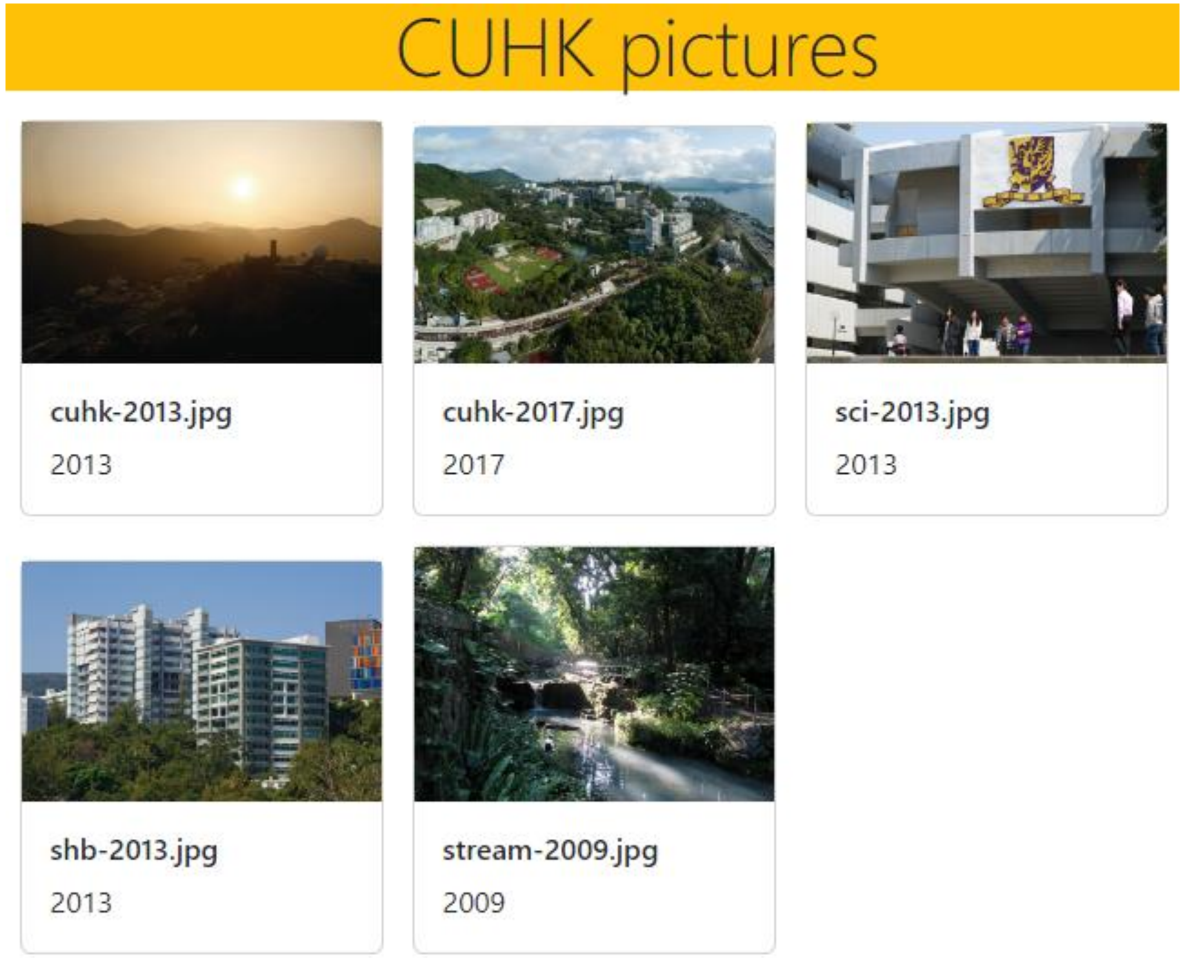
- How to loop through the data array so that we can display all images?

Looping through the data array

- To show all images, loop through the array in **<Gallery/>**
  - **.map()** is an efficient way to generate the result
- Inside **<main>** in **<Gallery/>**, use this the following to render multiple **<FileCard/>** Components
  - `{data.map((file,index) => <FileCard i={index} key={index}/>)}`
  - *key={index}* allows React to identify the elements in the ReactDOM for efficient re-rendering
- Adjust your code in **<FileCard/>** to the following:

```
render() {
   let i = this.props.i;
   return (
      // change data[0] to data[i]
   );
```

Done!



CUHK pictures

cuhk-2013.jpg

2013

cuhk-2017.jpg

2017

sci-2013.jpg

2013

shb-2013.jpg

2013

stream-2009.jpg

2009

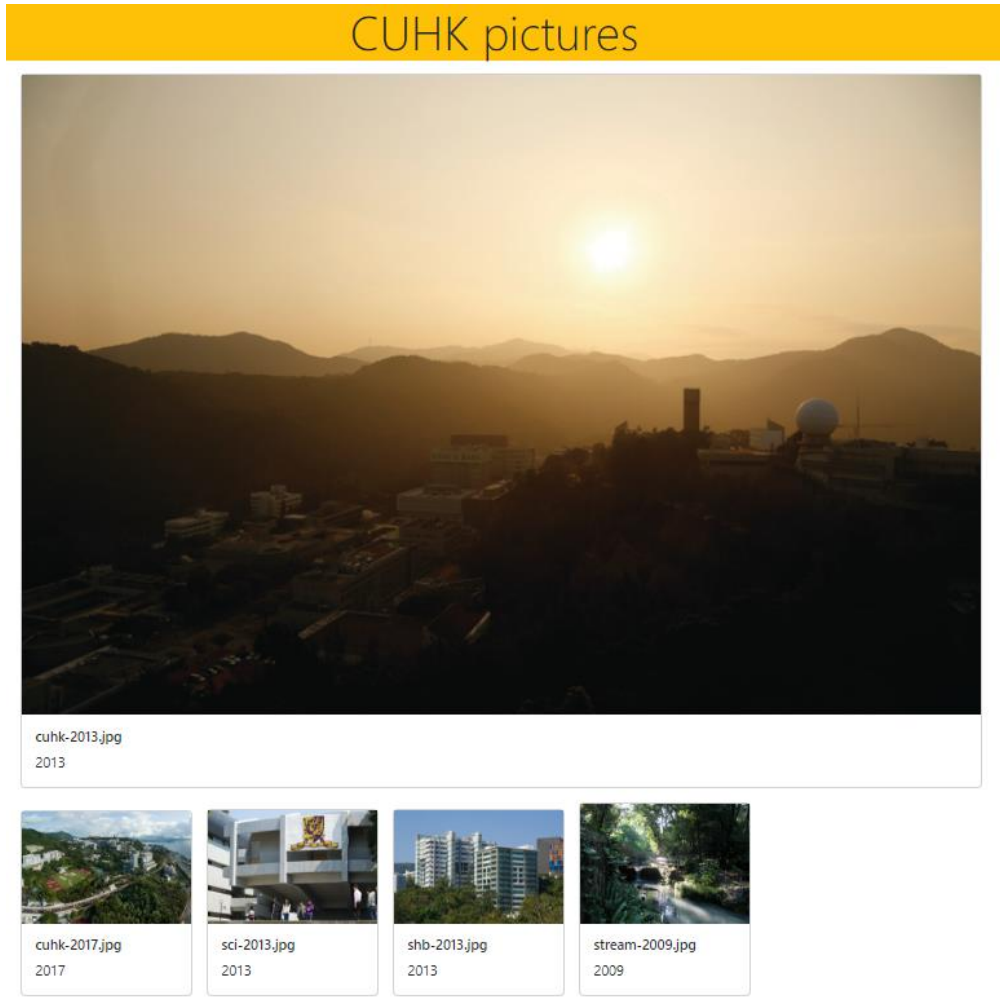
Submission

- No submission is needed for labs
- That you have done could be useful for your further exploration or the upcoming assignments
- Please keep your own file safely

- Try to finish the additional materials too!

Additional materials

- In the additional materials, we want to add one more feature to our app.
- Click and enlarge the image!



Additional materials – handling events

- Set up an event handler inside **<FileCard/>**

```
handleClick() {
    console.log("clicked");
}
```

  - This handler should be on top of the **render()**

- Next, put the **onClick** handler in the card div
  - `onClick={this.handleClick}`
  - The name *handleClick* isn’t important as long as they match.

- Check if you can see the console’s message when clicking

Additional materials – handling events

- However, since we want to send the *index i* too, we need to use this for *onClick*
  - `onClick={(e) => this.handleClick(i, e)}`
- And of course, adjust the event handler too:

```
handleClick(index, e) {
    console.log(index);
}
```

- Are you able to see the index printed when clicking?

Additional materials – using states

- To use states, you need to set it up in the class constructor of **<FileCard/>**

```
constructor(props) {
    super(props);
    this.state = { selected: -1 };
}
```

  - The syntax *this.state* should only be used in the constructor, otherwise **this.setState()** must be used.

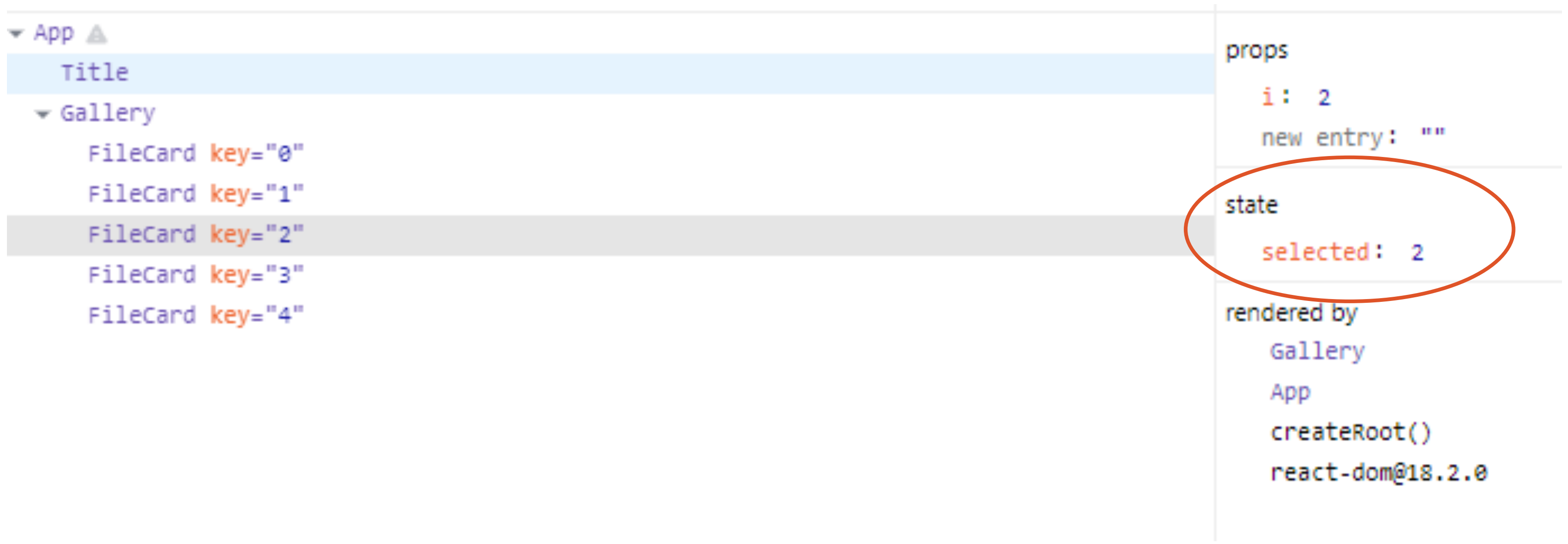
- In the event handler, you could do this (with proper JavaScript) with **this.setState()**

```
/* If this.state.selected is not index
        set selected state to index
    Else
        set selected state to -1 */
```

  - Can you write a JavaScript for the above pseudo-code?

Additional materials – using states

- Now, when clicking the cards, you can see a change (state from -1 to the corresponding index) in the *developer tools >> Components tab*



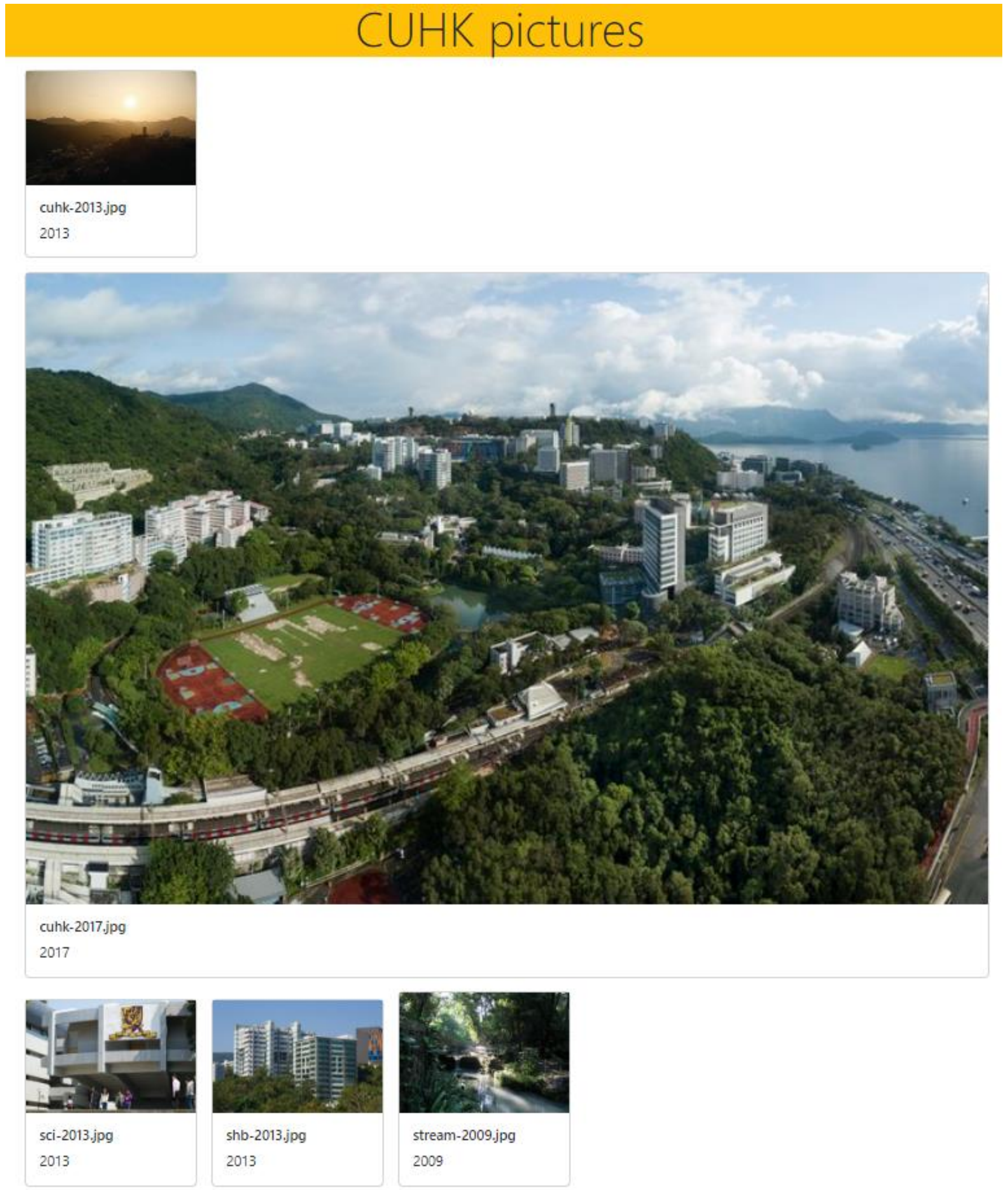
```
▾ App
    Title
  ▾ Gallery
      FileCard key="0"
      FileCard key="1"
      FileCard key="2"
      FileCard key="3"
      FileCard key="4"
```

```
props
  i: 2
  new entry: ""

state
  selected: 2

rendered by
    Gallery
    App
    createRoot()
    react-dom@18.2.0
```

Conditional rendering

- There are different ways to render conditionally in React.
- One easy way is to use the ternary operator ? … : …
  - `style={{width:this.state.selected==i ? '100%' : 200}}`

Done!


CUHK pictures

cuhk-2013.jpg
2013

cuhk-2017.jpg
2017

sci-2013.jpg
2013

shb-2013.jpg
2013

stream-2009.jpg
2009