Component Techniques

1. Lifecycle methods
2. Lifecycle methods usage
3. Optimizing UI updates
4. Component child content

1. Lifecycle Methods

- Overview
- Lifecycle methods available
- Implementing lifecycle methods

Overview

- When you define a component, React does all this:
  - Creates an instance of your component
  - Calls various lifecycle methods at interesting times
  - Calls `render()` to tell component to render its elems
- In this section we'll discuss the various lifecycle methods that react calls upon a component
  - We'll explain how, and why, to implement these methods

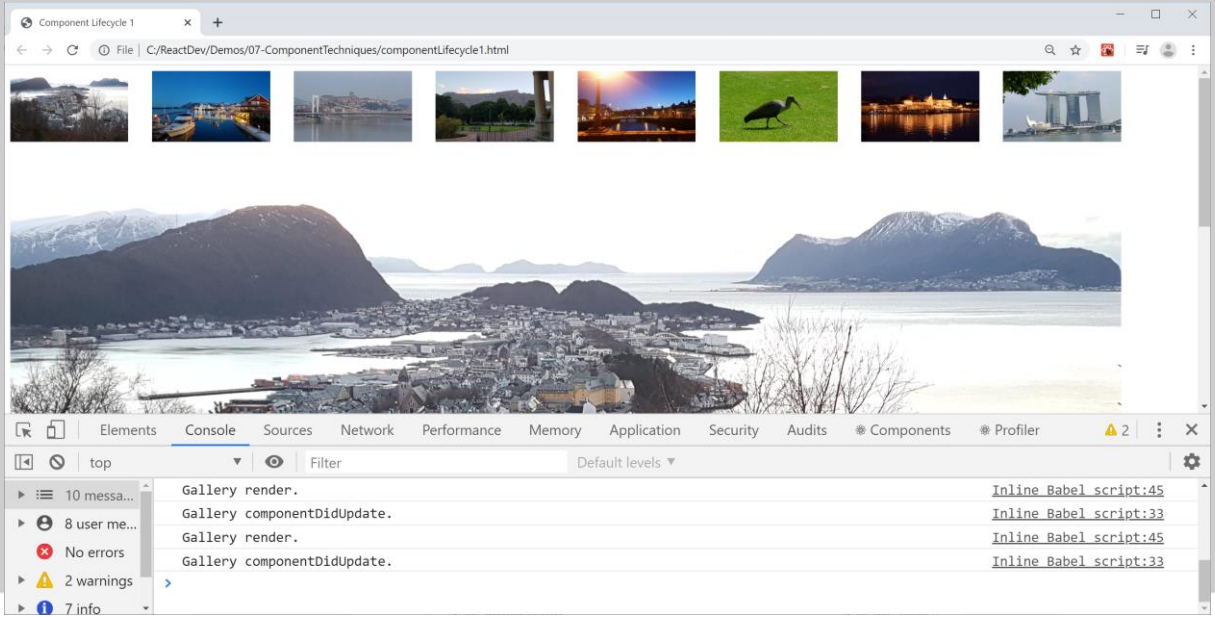
Lifecycle Methods Available

- Here are the most common lifecycle methods that React calls upon a component object
  - `componentDidMount()`
  - `componentDidUpdate()`
  - `componentWillUnmount()`

- For a full of lifecycle methods, see:
  - https://reactjs.org/docs/react-component.html

Implementing Lifecycle Methods

- See the example in `componentLifecycle1.html`
  - Displays info messages in the console



2. Lifecycle Methods Usage

- Overview
- Starting the REST service
- Testing the REST service
- Calling the REST service

Overview

- In this section we show a realistic example of why you'd implement a lifecycle method
- The example will call a REST service at the point when a component is being mounted
  - The component calls the REST service asynchronously
  - When the REST service returns with the data, the component will re-render itself

Starting the REST Service

- We've implemented the REST service in Node.js
  - Open a command window in the `server` folder
  - Run the following commands

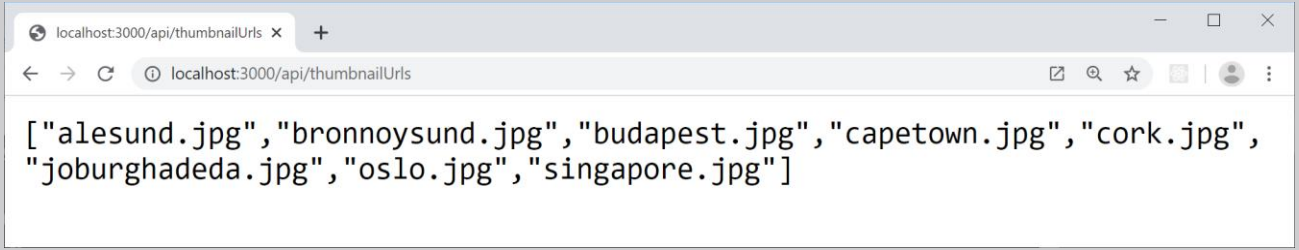
```
npm install
```

```
npm start
```

- The REST service starts up on port 3000

Testing the REST Service

- To test the REST service is working, browse here:
  - `http://localhost:3000/api/thumbnailUrls`



Calling the REST Service

- The client app calls the REST service after mounting
  - See `componentLifecycle2.html`
- After the component has mounted
  - i.e. in `componentDidMount()` …
- … it calls a function to invoke the REST service
  - See `getThumbnailUrlsFromRestService()`
  - It invokes the REST service asynchronously

3. Optimizing UI Updates

- Overview
- Is an update necessary?
- Accessing previous state

Overview

- In this section we show how to optimize UI updates, to reduce the amount of re-rendering
  - At the moment, when the user clicks a thumbnail icon, it renders that icon in large format on the page
  - It does this even if you click the same thumbnail several times - what a waste of effort!
- A better approach:
  - Only re-render if a different thumbnail is clicked

Is an Update Necessary?

- Implement `shouldComponentUpdate()` method
  - The method receives the provisional new properties and state for the component
- Return a boolean result
  - `true` if you want an update/render to happen
  - `false` if you decide there's no need to update/render
- Example - see `componentLifecycle3.html`

Accessing Previous State

- Implement the `componentDidUpdate()` method
  - The method receives the previous properties and state for the component
- What to do with this info?
  - E.g. stick it into some kind of "undo" store
  - E.g. log the change somewhere useful
- Example - see `componentLifecycle3.html`

4. Comp Child Content

- Overview
- Accessing children
- Example

Overview

- You can supply child content to a component

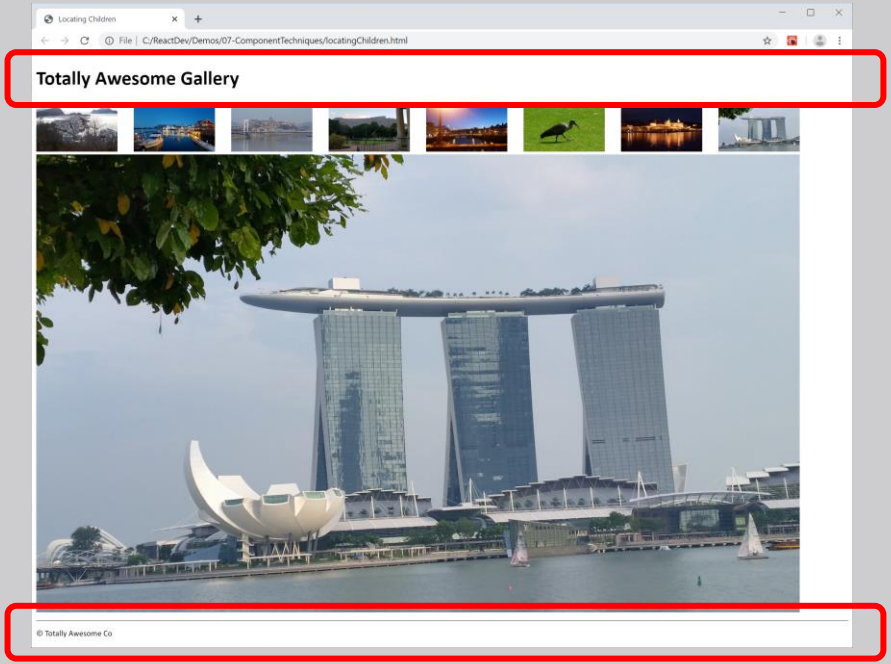
```
<Gallery baseAddress="… … …">
  <GalleryHeader>
    <h1>Totally Awesome Gallery</h1>
  </GalleryHeader>
  <GalleryFooter>
    <hr/>
    <small>&copy; Totally Awesome Co</small>
  </GalleryFooter>
</Gallery>
```

Accessing Children

- A component needs to be able to render the child content defined between the start/end tags
- To access the child content for a component:
  - Use `React.Children.toArray()`

Example

- See `locatingChildren.html`



Summary

- Lifecycle methods
- Lifecycle methods usage
- Optimizing UI updates
- Component child content