Creating a React Web App

1. Creating a simple app
2. Creating many elements
3. Data-driven UIs

Pearson

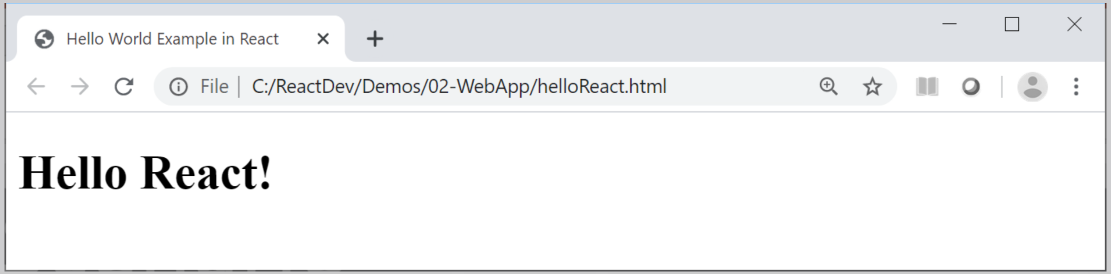
1. Creating a Simple App

- Scenario
- Defining a target element
- Including React libraries
- The virtual DOM
- Creating React elements
- Rendering React elements
- Viewing the React virtual DOM
- Logging a React element

Scenario

- In this chapter we show how to create simple React apps in pure React, using ES6++ language features

- See the `Demos/02-WebApp` folder
  - The first example we'll look at is `helloReact.html`
  - This is a minimalistic "Hello World" React app ☺


Hello World Example in React
File
C:/ReactDev/Demos/02-WebApp/helloReact.html

Hello React!

Defining an HTML Target Element

- A React web app has a single top-level HTML element into which React will render the UI

- You typically define it like this

```
<div id="osl-container"></div>
```

  - Give it a suitable `id`
  - You'll refer to this id when you render content (see later)

Including React Libraries

- To use React in a web page, you need 2 libraries:
  - React - Creates views
  - ReactDOM - Renders views in the web browser

- The following code downloads these libraries directly
  - Later, we'll see how to use npm to manage these libraries

```
<script src="https://unpkg.com/react@16/umd/react.development.js"></script>
<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
```

The Virtual DOM

- In a web app, you can manipulate elements using DOM
  - Create elements, append child element, etc.
- DOM is a very low-level API
  - It takes a lot of code to achieve simple tasks
  - It can also be really slow at run time!
- React introduces the concept of the <u>virtual DOM</u>
  - You create React elements (lightweight JS objects)
  - You manipulate the lightweight JS objects, and React renders the appropriate HTML as efficiently as possible

Creating React Elements

- You can create a React element programmatically by calling `React.createElement()`
  - 1st argument specifies the type of element to create
  - 2nd argument specifies the element's properties
  - 3rd argument specifies the element's children

```
<script>
  const msg = React.createElement(
    'h1',
    {id: 'msg-0', 'title': 'My message'},
    'Hello React!')
</script>
```

Rendering React Elements

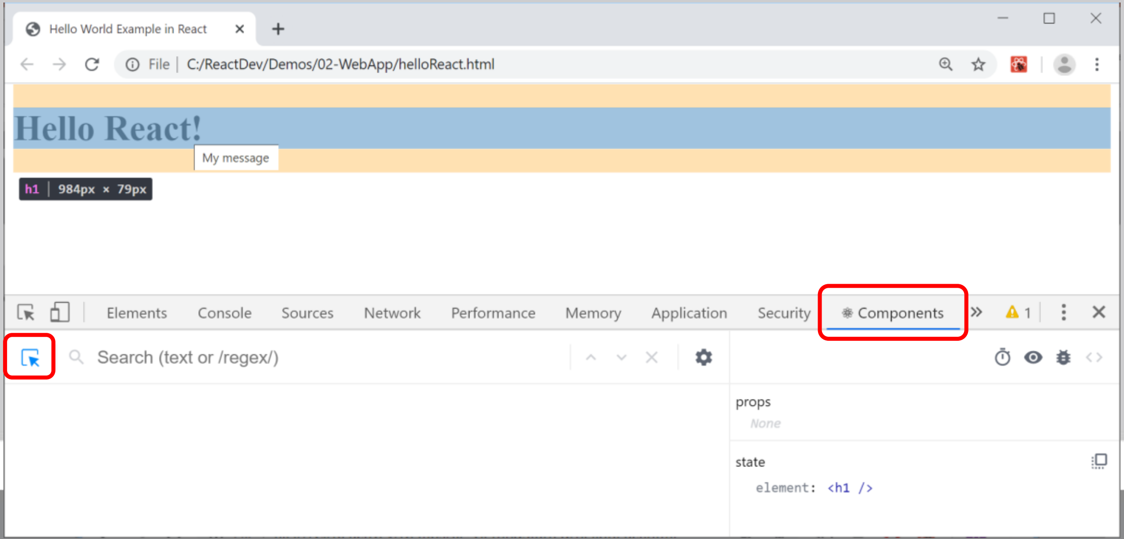
- Render your top-level React element into a target node on the web page, using `ReactDOM.render()`
  - 1st argument is your top-level React element
  - 2nd argument is the target node on the web page

```
<script>
  …
  ReactDOM.render(
      msg,
      document.getElementById('osl-container'))

</script>
```

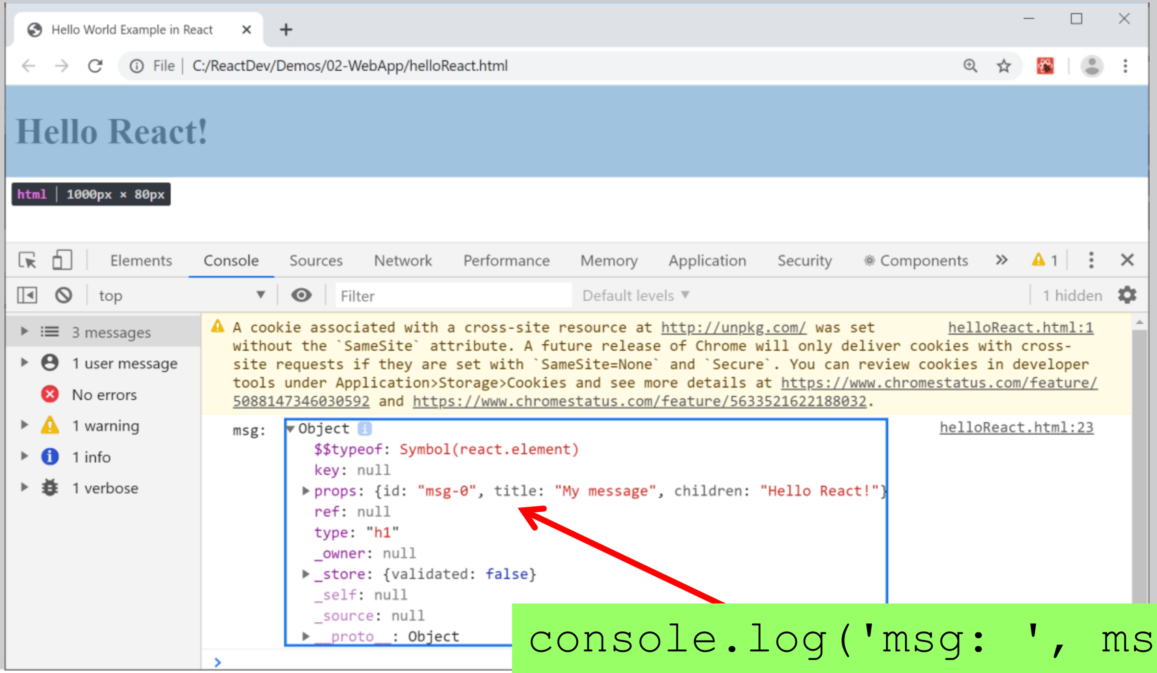
Viewing the React Virtual DOM

- Chrome (and other browsers) allow you to view the React elements in the virtual DOM
  - Install the React Developer Tools extension
  - Then in DevTools, click Components and select element



Logging a React Element

- A React element is a lightweight object, as shown here



```
⚠ A cookie associated with a cross-site resource at http://unpkg.com/ was set        helloReact.html:1
without the `SameSite` attribute. A future release of Chrome will only deliver cookies with cross-
site requests if they are set with `SameSite=None` and `Secure`. You can review cookies in developer
tools under Application>Storage>Cookies and see more details at https://www.chromestatus.com/feature/
5088147346030592 and https://www.chromestatus.com/feature/5633521622188032.
msg:  ▼Object                                                                        helloReact.html:23
         $$typeof: Symbol(react.element)
         key: null
        ▶props: {id: "msg-0", title: "My message", children: "Hello React!"}
         ref: null
         type: "h1"
         _owner: null
        ▶_store: {validated: false}
         _self: null
         _source: null
        ▶__proto__: Object
```

```
console.log('msg: ', msg)
```

2. Creating Many Elements

- Overview
- Hierarchy of React elements
- View the page in the browser
- View the virtual DOM

Overview

- In this section we'll see how to create a more ambitious virtual DOM tree, containing nested React elements
  - Then we'll render the root React element to the DOM

- Actually that's an important point…
  - You only ever render the root React element to the DOM

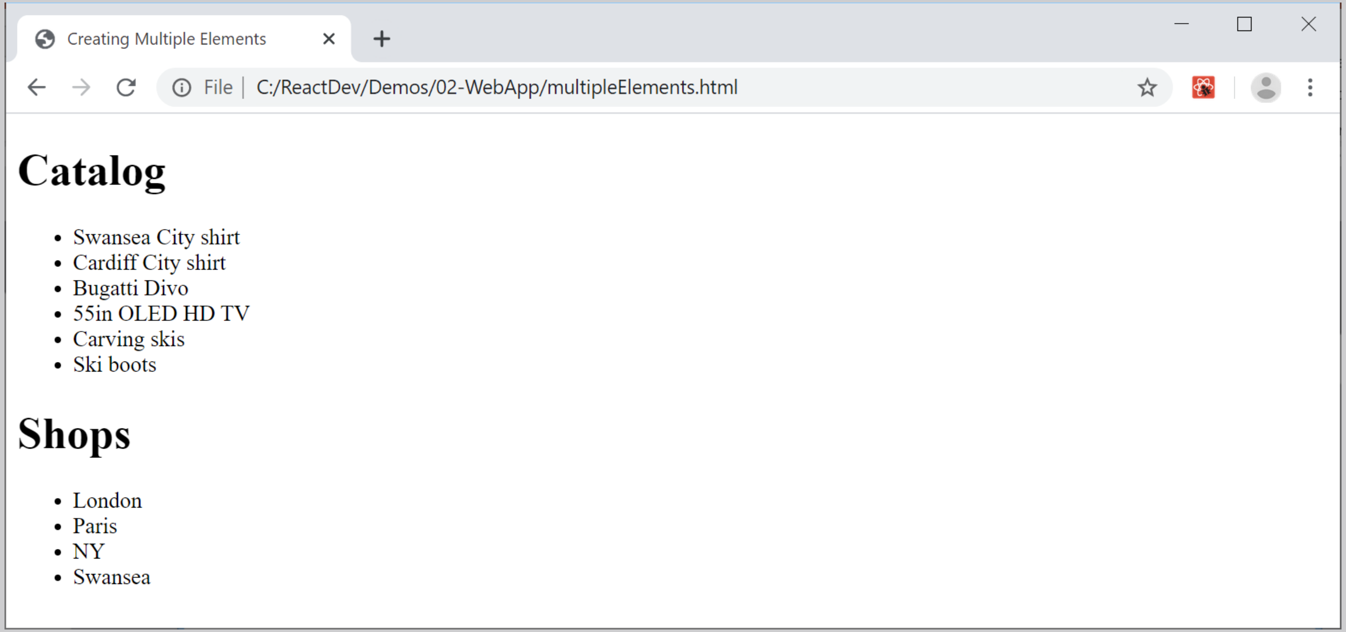
Hierarchy of React Elements

- `createElement()` can take a variadic list of child elements, so you can create a hierarchy of elements

```
const ul =
  React.createElement('ul', null,
     React.createElement('li', null, 'Item1'),
     React.createElement('li', null, 'Item2'),
     React.createElement('li', null, 'Item3'))
```

- For an example, see `multipleElements.html`

View the Page in the Browser


Creating Multiple Elements
File
C:/ReactDev/Demos/02-WebApp/multipleElements.html

Catalog

- Swansea City shirt
- Cardiff City shirt
- Bugatti Divo
- 55in OLED HD TV
- Carving skis
- Ski boots

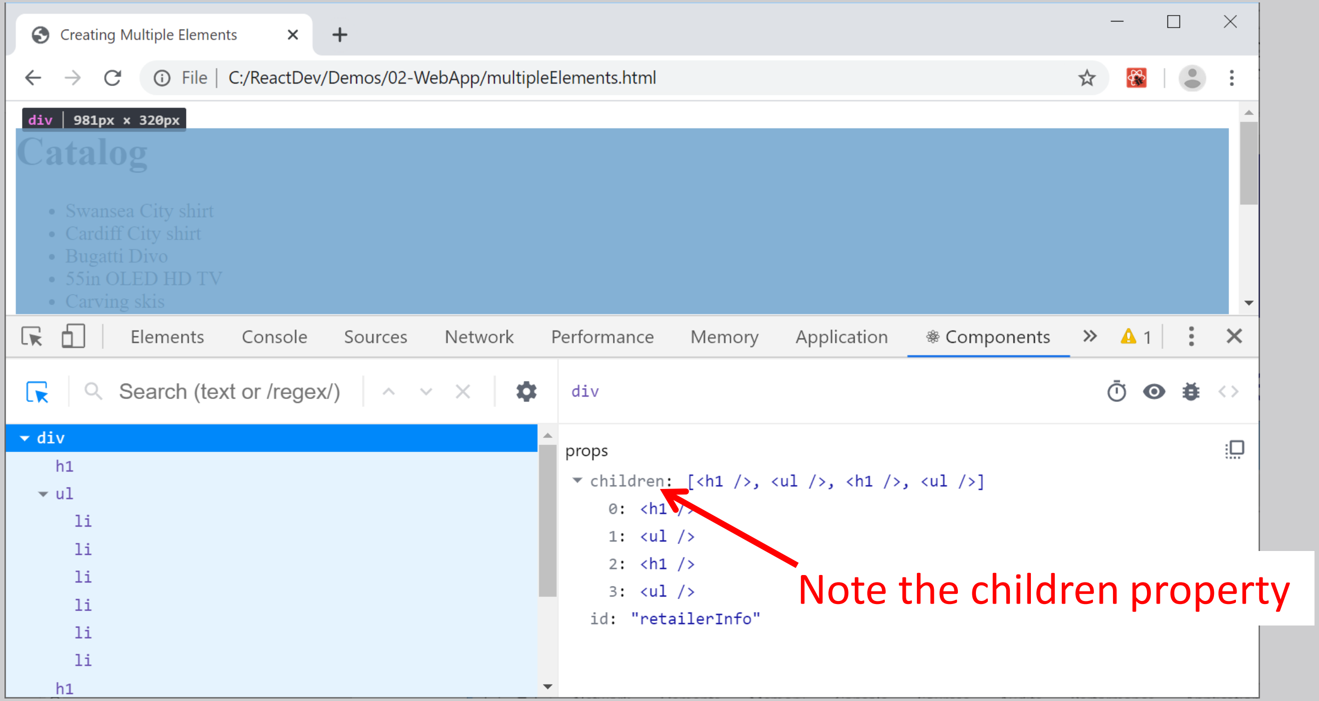
Shops

- London
- Paris
- NY
- Swansea

View the Virtual DOM (1 of 2)

- To view components in the virtual DOM, e.g. using React Developer Tools in Chrome:
  - Show the DevTools window (F12)
  - Click the Components tab
  - In the Search window, click the Settings icon
  - In the popup window, select the Components tab
  - Deselect the "Hide components where" option
  - You should now see a list of the components

View the Virtual DOM (2 of 2)



```
▾ div
    h1
  ▾ ul
      li
      li
      li
      li
      li
      li
    h1
```

```
div
props
▾ children: [<h1 />, <ul />, <h1 />, <ul />]
    0: <h1 />
    1: <ul />
    2: <h1 />
    3: <ul />
  id: "retailerInfo"
```

Note the children property

3. Data-Driven UIs

- Overview
- Defining data
- Mapping data to elements
- Example
- View the page in the browser

Pearson

Overview

- The previous section created a hard-coded hierarchy of React elements

- In a real app, you'll adopt a data-driven approach
  - The elements you create will depend on data
  - We'll see how to create a data-driven UI in this section
  - See `multipleElementsViaData.html`

Defining Data

- We'll have an array of products and an array of shops

```
const products = [
  'Swansea City shirt',
  'Cardiff City shirt',
  …
]
```

```
const shops = [
  'London',
  'Paris',
  …
]
```

Mapping Data to Elements

- You can use `map()` to map array item to a React elem

```
someArray.map((arrayItem, index) =>
  React.createElement(elem,
                      {key:index},
                      arrayItem.someProperty)
)
```

- `map()` takes a function that typically has 2 args
  - Arg 1 is an array item, arg 2 is the index of the array item
  - The function creates a React element (with unique key)

Example (1 of 2)

- Let's map `products` and `shops` arrays to `<ul>`'s

```
let prodList = React.createElement('ul',  null,
  products.map((p, i) =>
        React.createElement('li', {key: i}, p))
)
```

```
let shopList = React.createElement('ul',  null,
  products.map((s, i) =>
        React.createElement('li', {key: i}, s))
)
```

Example (2 of 2)

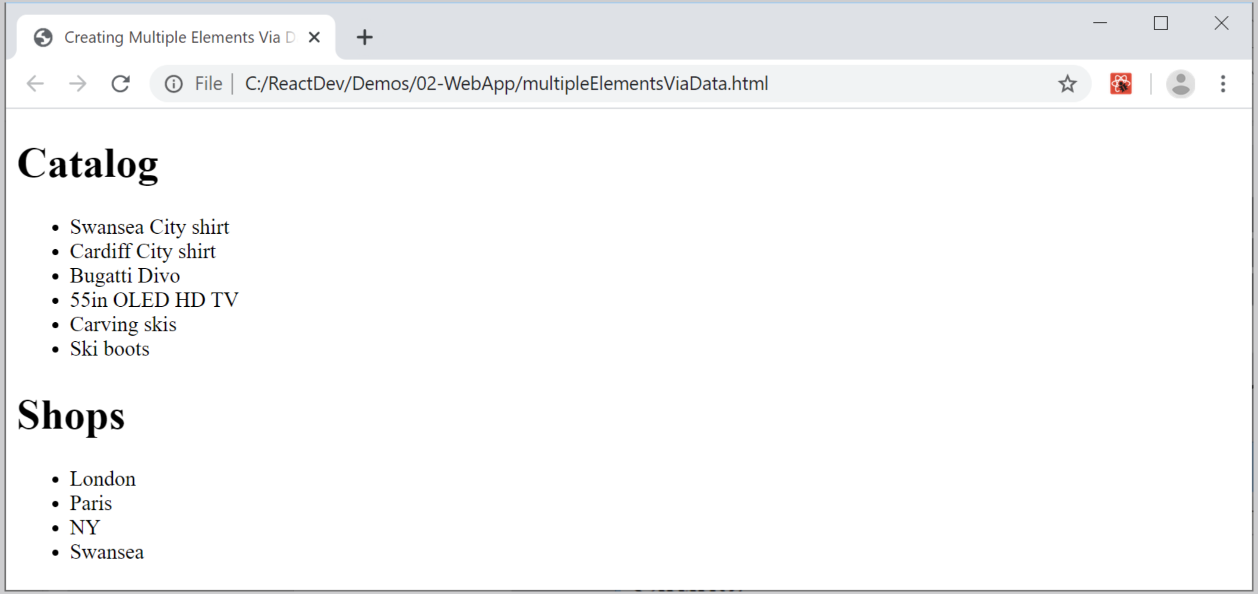
- Now let's put it all together

```
let retailer = React.createElement('div', null,
  React.createElement('h1', null, 'Catalog'),
  prodList,
  React.createElement('h1', null, 'Shops'),
  shopList
)

ReactDOM.render(
  retailer,
  document.getElementById('osl-container'))
```

Viewing the Page in the Browser

- Here's how the page looks in the browser


Creating Multiple Elements Via D
File | C:/ReactDev/Demos/02-WebApp/multipleElementsViaData.html

Catalog

- Swansea City shirt
- Cardiff City shirt
- Bugatti Divo
- 55in OLED HD TV
- Carving skis
- Ski boots

Shops

- London
- Paris
- NY
- Swansea

Summary

- Creating a simple app
- Creating many elements
- Data-driven UIs