Welcome!

Agenda

- Node
- NPM
- Parcel
- Modules
- React

Review

Before Node

JavaScript Engines

- JavaScript Engines are pieces of software that execute JavaScript code
- They are typically created by Web Browser Vendors:
  - V8 is for Google Chrome
  - SpiderMonkey is for Firefox
  - JavaScriptCore is for Safari
  - Chakra is for Internet Explorer (and a fork of it was for Edge)

Why are we talking about this?

- Node.js is the JavaScript Engine (V8)
  - Taken out of the browser, with a little bit extra
- It's the same programming language as the JavaScript you know, it just has extra back-end related functionality
  - e.g. Work with file system, work with databases etc.

Node

What is Node?

- A back-end programming language
- Open-source and cross-platform
- Created in 2009 by Ryan Dahl while he was working at Joyent
- Can be run interactively (as a REPL) or as files
- Very popular (Uber, Netflix, LinkedIn, Twitter, Paypal, eBay etc.)

When we install Node:

We install command line tools, too:

- Node
- NPM

How to run it

To open up a REPL (CTRL + C to quit):

```
node
```

To run a file:

```
node index.js
```

NPM

What is NPM?

- A command line tool
- The Node Package Manager
- The largest software library (Package Registry)
  - With over 1.3 million packages
- A tool for managing software projects
  - It helps install things, manage dependencies and structure our apps - as well as making it discoverable (if we want)

`npm init`

Used to set up a new Node/NPM project

```
npm init
```

This will create a **`package.json`** file. This file will describe our project, its scripts and its dependencies

npm install

How we install packages into our projects

```
# Install all required packages
npm install

# Install and save a package as a dependency
npm install --save package_name

# Install and save a package as a dependency for development
npm install --save-dev package_name
```

npm run

How we run scripts defined in our `package.json`

```
npm run script_name

# Run the start script in package.json
npm run start
```

Modules

What are modules?

Modules allow us to build big JavaScript projects by allowing us to easily work in multiple files, and to bring in any dependencies that are necessary:

- We can export code from one file
- And we can require (essentially import) any files or dependencies that we need in another file

Exporting Code

```
function sayHello(name) {
  console.log(`Hello ${name}`);
}

module.exports = sayHello;
```

Exporting Code

```
function add(x, y) {
  return x + y;
}

function subtract(x, y) {
  return x - y;
}

module.exports = {
  add: add,
  subtract: subtract
};
```

Importing Code

```
const maths = require("./maths");

const auth = require("../auth/local");

const React = require("react");
```

If you require a file path, it will store the exported code in the variable

If you require a package or module name, it will do the same (note that it is not a file path)

```javascript
function hello(name) {
  let msg = `Hello ${name}`;
  console.log(msg);
}

module.exports = hello;
```

```javascript
const hello = require("./hello");

hello();
```

```javascript
function add(a, b) {
  return a + b;
}

function subtract(a, b) {
  return a - b;
}

module.exports = {
  add: add,
  subtract: subtract
};
```

```javascript
const maths = require("./maths");

maths.add();

maths.subtract();
```

Parcel

What is Parcel?

It is a Build System, a Bundler, a server and a Compiler

- **Build System**: It automates tasks for us
- **Bundler**: It will combine multiple files into one
- **Compiler**: It takes our code, transforms it and then returns a new version

It does a lot of other things too - things like minifying and optimizing our code, minifying images etc.

It is a command line that also mostly doesn't require configuration

Why do we need a compiler?

Because we are using a lot of new features, and soon, we will be using things that aren't a part of JavaScript itself (we will see this with React).

Some of the things it may translate for us:

- SCSS -> CSS
- New, fancy JavaScript -> Compatible JavaScript
- Large images -> Optimized Images

Are there any alternatives?

- Webpack (one of the most popular)
- Grunt
- Gulp
- Snowpack
- FuseBox
- RollUp
- Browserify
- Plus many more

Installing

Installing Parcel

```
# Set up the NPM project
npm init

# Add parcel as a development dependency
npm install --save-dev parcel
```

Starting your Build System

Add a script in your `package.json`

```
1  {
2    "name": "parcel-install",
3    "version": "0.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "parcel app/index.html"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC"
12 }
```

Now, execute `npm run dev` in your terminal

Building for Production

Add a script in your `package.json`

```
1  {
2    "name": "parcel-install",
3    "version": "0.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "parcel app/index.html",
8      "build": "parcel build app/index.html"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC"
13 }
```

Now, execute **`npm run build`** in your terminal

JSX

What is JSX?

- A syntax extension of JavaScript
  - It's not part of the normal language!
- Something made popular by React
  - It provides us with shortcuts to create elements
- JSX looks like HTML, and we can mostly treat it is as such
  - But it is turned into JavaScript before it reaches the browser
- It's an easy way to describe UI

What does it look like?

```
const element = <h1>Hello, world!</h1>;
```

This data isn't a string or HTML

It is an extension of JavaScript, that is eventually turned into regular JS. It is turned into something like this:

```
const element = React.createElement("h1", {}, "Hello World");
```

Just a shortcut - something that tries to make everything a little clearer

JSX

```
<h1 id="hello">Hello World</h1>

// Compiles to...

React.createElement(
  "h1",
  { id: "hello" },
  "Hello World"
);
```

JSX

```
<img src="http://fillmurray.com/400/400" id="bill">

// Compiles to...

React.createElement(
  "img",
  { src: "http://fillmurray.com/400/400", id: "bill" },
  null
);
```

Single Page Applications

What are Single Page Applications?

- Apps that don't require reloading
- The page is loaded, then JavaScript takes control
- SPAs typically rely heavily on APIs and AJAX
  - As well as frameworks to organise the code, because they are very JS-heavy

Pros of SPAs

- They often have a very nice user experience
- They tend to promote the use of APIs
  - Meaning apps are decoupled and back-ends can be used in multiple environments (e.g. web and mobile apps)
- Interactions happen very quickly and updates seem instantaneous
- Loading screens and page transitions are much more possible
- There is much less load on the server

Cons of SPAs

- They require lots of JavaScript and only work if JavaScript is enabled
- Search Engine Optimisation is much more difficult
- The initial load, if not managed correctly, can take much longer
- They tend to be less secure
- They tend to mean developers have to take control of things they wouldn't otherwise have to (e.g. browser history etc.)
- They are resource-heavy for the browser which can slow performance and put load on devices

React

What is React?

What is React?

- An open-source JavaScript library/framework for building applications (particularly Single Page Applications)
  - It was created by Facebook
- It is declarative - we describe patterns and React does the heavy-lifting
- It is component-based - React makes it easy to break applications down into lots of pieces then compose them
- Learn once, write anywhere

What can it do?

What can it do?

Anything! Lots of companies use it - Facebook, Instagram, Uber, AirBnB, Netflix, Pinterest, Shopify, Twitter, Atlassian, Codecademy, Khan Academy as well as many, many more.

It's used in every context, for every type of app

It is certainly the most popular front-end framework - but it does take time to get used to!

React Alternatives

React Alternatives

- Vue
- Svelte
- Angular
- Ember
- Meteor
- Backbone
- Plus many more

Advantages of React

- Really easy to see the structure of your app
- Very good at managing state
- Performant
- Virtual DOM
- Data Binding
- Easy to test
- Isomorphic (can be rendered server-side)
- Agnostic (you can use it with all sorts of other libraries as React is just the view layer)
- Learn once, write everywhere

Disadvantages of React

- A big library
- Lots of magic
- It is just the view layer
- Typically requires a transformation step
- A steep learning curve
- It changes incredibly regularly

Installing

Installing React

```
npm init

npm install --save react react-dom
```

Breaking Pages Down

Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More

Tweet

React
2,397 Tweets

Following

React
@reactjs

React is a JavaScript library for building user interfaces.

reactjs.org Joined July 2013

270 Following **505.6K** Followers

Followed by Ashlee Boyer, Danijela.js |Software engineer|MERN|Mentor, and 468 others you follow

Tweets | Tweets & replies | Media | Likes

Pinned Tweet

React @reactjs · Jun 9

Our next major release is React 18. It will include out-of-the-box improvements like automatic batching, new APIs like startTransition, and a new streaming server renderer with built-in support for React.lazy.

The Plan for React 18 – React Blog
The React team is excited to share a few updates: We've started work on the React 18 release, which ...
reactjs.org

80 1.6K 4.8K

Show this thread

React @reactjs · Aug 21

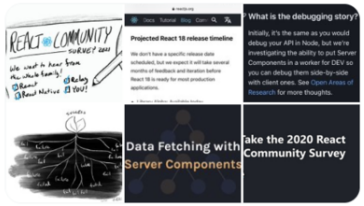
Don't forget, the React Conf CFP closes Monday! At React Conf we want to highlight what you're building with React—for your community, your family, your business. We'd love to hear and give a platform for your React Story ❤️

REACT CONF 2021 CALL FOR SPEAKERS
Submit your CFP for the React Conference 2021!
reactconf2021cfp.splashthat.com

2 24 137

React Retweeted

Search Twitter



You might like

Vue.js @vuejs Follow

Tailwind CSS @tailwindcss Follow

JavaScript @JavaScript Follow

Show more

What's happening

COVID-19 · LIVE
Updates on COVID-19 in Australia

World news · LIVE
The latest news from Afghanistan as the Taliban take control of the country

Trending in Australia
Pete Evans

The Sunday Times · Yesterday
An interview with Lorde: Ecstasy, the divine, and leaving the past behind

Trending in Australia
#AustraliaHasFallen
32.5K Tweets

Show more

Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More
Tweet

React
2,397 Tweets

Following

React
@reactjs

React is a JavaScript library for building user interfaces.

reactjs.org · Joined July 2013

270 Following **505.6K** Followers

Followed by Ashlee Boyer, Danijela.js |Software engineer|MERN|Mentor, and 468 others you follow

Tweets | Tweets & replies | Media | Likes

Pinned Tweet

React @reactjs · Jun 9

Our next major release is React 18. It will include out-of-the-box improvements like automatic batching, new APIs like startTransition, and a new streaming server renderer with built-in support for React.lazy.

The Plan for React 18 – React Blog
The React team is excited to share a few updates: We've started work on the React 18 release, which ...
reactjs.org

80 1.6K 4.8K

Show this thread

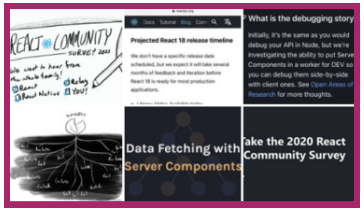
React @reactjs · Aug 21

Don't forget, the React Conf CFP closes Monday! At React Conf we want to highlight what you're building with React—for your community, your family, your business. We'd love to hear and give a platform for your React Story ❤️

REACT CONF 2021 CALL FOR SPEAKERS
Submit your CFP for the React Conference 2021!
reactconf2021cfp.splashthat.com

2 24 137

Search Twitter



You might like

Vue.js @vuejs Follow
Tailwind CSS @tailwindcss Follow
JavaScript @JavaScript Follow

Show more

What's happening

COVID-19 · LIVE
Updates on COVID-19 in Australia

World news · LIVE
The latest news from Afghanistan as the Taliban take control of the country

Trending in Australia
Pete Evans

The Sunday Times · Yesterday
An interview with Lorde: Ecstasy, the divine, and leaving the past behind

Trending in Australia
#AustraliaHasFallen
32.5K Tweets

Show more

Pages broken down into components

When thinking about apps through a React lens, we want to break them down into lots of individual components.

Ideally, those components should follow FIRST principles, meaning they should be:

- Focused
- Independent
- Reusable
- Small
- Testable

So, what are components?

- They can be written in two different ways:
  - They can be a function that returns JSX (essentially markup to render)
  - Or they can be a class with a render method that returns JSX
- We will be focussing on the function approach

Our First Component

Our First Component

```
function Hello() {
  return <h1>Hello</h1>;
}
```

Rendering

Rendering

```
import React from "react";
import ReactDOM from "react-dom";

function Hello() {
  return <h1>Hello</h1>;
}

ReactDOM.render(<Hello />, document.body);
```

Interpolation

Interpolation with JSX

```
1 function Hello() {
2   let name = "Jacques Cousteau";
3   return (
4     <div>
5       <h1>Hello {name}</h1>
6       <h2>Hello (in capitals) {name.toUpperCase()}</h2>
7     </div>
8   );
9 }
```

Curly brackets mean interpolation in JSX (very similar to `${}` in template literals)

Props

What are Props?

- Props are very similar to parameters in functions
  - They are a way for us to provide data to a component
- They are immutable (meaning they can't change)
- From a parent component, we can pass data down using props
- They look very similar to HTML attributes

Props

```
1  const React = require("react");
2  const ReactDOM = require("react-dom");
3
4  function Hello(props) {
5    let name = props.name;
6    return (
7      <div>
8        <h1>Hello {name}</h1>
9      </div>
10   );
11 }
12
13 ReactDOM.render(
14   <Hello name="Jacques Cousteau" />,
15   document.body
16 );
```

Events

Event Handlers

```
function MyComponent() {
  function onButtonClick() {
    console.log("The button was clicked");
  }

  return (
    <div>
      <h1>Hello World</h1>
      <button onClick={onButtonClick}>Click Me</button>
    </div>
  )
}
```

State

What is State?

- State is the way we work with data that can change within individual components
  - It is mutable (meaning it can change) data that is local to a component - it can't be accessed by parent components by detail
- It's the way we make our components interactive

Home
Explore
Notifications
Messages
Bookmarks
Lists
Profile
More
Tweet

React
2,397 Tweets

Following

React
@reactjs

React is a JavaScript library for building user interfaces.

reactjs.org · Joined July 2013

270 Following **505.6K** Followers

Followed by Ashlee Boyer, Danijela.js |Software engineer|MERN|Mentor, and 468 others you follow

Tweets | Tweets & replies | Media | Likes

Pinned Tweet

React @reactjs · Jun 9

Our next major release is React 18. It will include out-of-the-box improvements like automatic batching, new APIs like startTransition, and a new streaming server renderer with built-in support for React.lazy.

The Plan for React 18 – React Blog
The React team is excited to share a few updates: We've started work on the React 18 release, which ...
reactjs.org

80 1.6K 4.8K

Show this thread

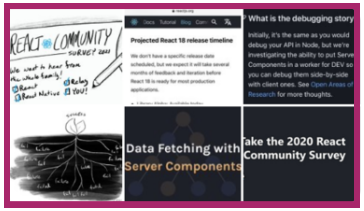
React @reactjs · Aug 21

Don't forget, the React Conf CFP closes Monday! At React Conf we want to highlight what you're building with React—for your community, your family, your business. We'd love to hear and give a platform for your React Story ❤️

REACT CONF 2021 CALL FOR SPEAKERS
Submit your CFP for the React Conference 2021!
reactconf2021cfp.splashthat.com

2 24 137

Search Twitter



You might like

Vue.js @vuejs Follow
Tailwind CSS @tailwindcss Follow
JavaScript @JavaScript Follow

Show more

What's happening

COVID-19 · LIVE
Updates on COVID-19 in Australia

World news · LIVE
The latest news from Afghanistan as the Taliban take control of the country

Trending in Australia
Pete Evans

The Sunday Times · Yesterday
An interview with Lorde: Ecstasy, the divine, and leaving the past behind

Trending in Australia
#AustraliaHasFallen
32.5K Tweets

Show more

Hooks

What are hooks?

- They are the way that we will manage state in our application (that mutable data that is local to each component)
- Hooks are functions that React defines for us*
- They are run inside a function component
- They maintain their value even when the component re-renders (meaning updates)
- A lot of errors can arise from creating or running hooks within conditionals or loops (so don't do this)

What hooks does React provide?

- useState
- useEffect
- useContext
- useReducer
- useCallback
- useMemo
- useRef
- useImperativeHandle
- useLayoutEffect
- useDebugValue

But we will likely only be using:

- useState
- useEffect

Destructuring Assignment

What is Destructuring Assignment

- It is a new feature of JavaScript that provides us with a shorthand for accessing data within objects and arrays
  - For the moment, it needs to be translated into something that is compatible in browsers (we have Parcel doing that for us)

Object Destructuring

```
let person = {
  firstName: "Jacques",
  lastName: "Cousteau"
};

let firstName = person.firstName;
let lastName = person.lastName;

// That's a bit repetitive, isn't it?
// We could replace it with Destructuring Assignment

let { firstName, lastName } = person;
```

Array Destructuring

```
let alphabet = ["A", "B"];

let letterA = alphabet[0];
let letterB = alphabet[1];

// That's a bit repetitive, isn't it?
// We could replace it with Destructuring Assignment

let [letterA, letterB] = alphabet;
```

useState

`useState`

- A function that React provides for us
- We need to import it before we can use it
- It receives an initial value
- It returns an array with two pieces of data
  1. The current value
  2. A function to update the current value

useState

```
1  import React, { useState } from "react";
2
3  function ClickCounter() {
4    // Set the starting value to be 0
5    const [count, setCount] = useState(0);
6
7    function onButtonClick() {
8      setCount(count + 1);
9    }
10
11   return (
12     <div>
13       <h1>You have clicked {count} times</h1>
14       <button onClick={onButtonClick}>Click Me</button>
15     </div>
16   );
17 }
```

useState

```
1  import React, { useState } from "react";
2
3  function ClickCounter() {
4    // Set the starting value to be 0
5    const [count, setCount] = useState(0);
6
7    function onButtonClick() {
8      setCount(count + 1);
9    }
10
11   return (
12     <div>
13       <h1>You have clicked {count} times</h1>
14       <button onClick={onButtonClick}>Click Me</button>
15     </div>
16   );
17 }
```

useState

```
1  import React, { useState } from "react";
2
3  function ClickCounter() {
4    // Set the starting value to be 0
5    const [count, setCount] = useState(0);
6
7    function onButtonClick() {
8      setCount(count + 1);
9    }
10
11   return (
12     <div>
13       <h1>You have clicked {count} times</h1>
14       <button onClick={onButtonClick}>Click Me</button>
15     </div>
16   );
17 }
```

Handling User Input

useState

```
1  function LogInForm() {
2    const [email, setEmail] = useState("");
3    function updateEmail(event) {
4      setEmail(event.target.value);
5    }
6    return (
7      <form>
8        <p>Your email is {email}</p>
9        <input
10         type="text"
11         value={email}
12         placeholder="Email"
13         onChange={updateEmail}
14       />
15     </form>
16   );
17 }
```

useState

```
1  function LogInForm() {
2    const [email, setEmail] = useState("");
3    function updateEmail(event) {
4      setEmail(event.target.value);
5    }
6    return (
7      <form>
8        <p>Your email is {email}</p>
9        <input
10         type="text"
11         value={email}
12         placeholder="Email"
13         onChange={updateEmail}
14       />
15     </form>
16   );
17 }
```

useState

```
1  function LogInForm() {
2    const [email, setEmail] = useState("");
3    function updateEmail(event) {
4      setEmail(event.target.value);
5    }
6    return (
7      <form>
8        <p>Your email is {email}</p>
9        <input
10         type="text"
11         value={email}
12         placeholder="Email"
13         onChange={updateEmail}
14       />
15     </form>
16   );
17 }
```

useState

```
1  function LogInForm() {
2    const [email, setEmail] = useState("");
3    function updateEmail(event) {
4      setEmail(event.target.value);
5    }
6    return (
7      <form>
8        <p>Your email is {email}</p>
9        <input
10         type="text"
11         value={email}
12         placeholder="Email"
13         onChange={updateEmail}
14       />
15     </form>
16   );
17 }
```

Working with APIs

```
function MovieSearch() {
  const [title, setTitle] = useState("");
  const [data, setData] = useState(null);
  console.log(data);
  function updateTitle(event) {
    setTitle(event.target.value);
  }
  function searchForMovie(e) {
    e.preventDefault();
    fetch(`http://www.omdbapi.com/?apikey=88e15bed&t=${title}`)
      .then(function (r) {
        return r.json();
      })
      .then(function (data) {
        setData(data);
      });
  }
  return (
    <form onSubmit={searchForMovie}>
      <input type="text" value={title} onChange={updateTitle} />
      <button>Search</button>
    </form>
  );
}
```

```
function MovieSearch() {
  const [title, setTitle] = useState("");
  const [data, setData] = useState(null);
  console.log(data);
  function updateTitle(event) {
    setTitle(event.target.value);
  }
  function searchForMovie(e) {
    e.preventDefault();
    fetch(`http://www.omdbapi.com/?apikey=88e15bed&t=${title}`)
      .then(function (r) {
        return r.json();
      })
      .then(function (data) {
        setData(data);
      });
  }
  return (
    <form onSubmit={searchForMovie}>
      <input type="text" value={title} onChange={updateTitle} />
      <button>Search</button>
    </form>
  );
}
```

```
function MovieSearch() {
  const [title, setTitle] = useState("");
  const [data, setData] = useState(null);
  console.log(data);
  function updateTitle(event) {
    setTitle(event.target.value);
  }
  function searchForMovie(e) {
    e.preventDefault();
    fetch(`http://www.omdbapi.com/?apikey=88e15bed&t=${title}`)
      .then(function (r) {
        return r.json();
      })
      .then(function (data) {
        setData(data);
      });
  }
  return (
    <form onSubmit={searchForMovie}>
      <input type="text" value={title} onChange={updateTitle} />
      <button>Search</button>
    </form>
  );
}
```

useEffect

`useEffect`

- This hook allows us to perform "side effects" in our components (meaning things outside of the component itself)
  - Side effects include data fetching, manually changing the DOM and setting up subscriptions
- It is a function that React provides for us, and that we need to import into our projects
  - It receives a callback function and a dependencies array

`useEffect`

The useEffect callback function is executed based upon on the dependencies array (the second parameter)

- If there is no dependencies array, it will run after the component renders every time
- If there is an empty array as the dependencies array, it will run only once when the component is first rendered
- If the dependencies array contains things, the useEffect callback will run whenever items in that array change

React Router

Class Components

Homework

- Learn about Destructuring Assignment
- Learn about Arrow Functions
- Read about JSX
- Go through this Video Series
- Go through the React Tutorial
  - It will do things slightly differently to how we do it, but it will still cover the concepts
- Read Tinselcity whys:packers

What's next?

- React
- More React
- More React
- THREE.js

Thank you!