Frontend Web Dev Primer

NYU Makerspace, October 29, 2018

About me

Karan Rajpal 🙋‍♂️
Moat (Oracle Data Cloud)
Web Dev for about 4 years
Dressed up as Elvis last Halloween 👑

The stack I work with
Frontend - ES6 JavaScript, React, Redux, SCSS, Webpack
Backend - Node, Express, Typescript, SQL, Sequelize

What you need

1. Laptop with Npm and Node installed
2. Visual Studio Code
3. Basic HTML, CSS and Javascript knowledge

What you will learn

1. Npm, Node and package.json
2. ES6 JavaScript
3. Weird Wide Web
4. The magic of Parcel
5. Project setup
6. React fundamentals
7. Routing to make a multi-page app
8. Redux fundamentals
9. Sassy CSS

Npm and Node and package.json

Node - Javascript runtime environment outside the browser.
Npm - Node package manager. Also a repository of packages online.

If you're not sure if you installed it, run

`node -v` and `npm -v`

package.json - Manifest file that includes config and list of packages for a project.

Start a node project by running this in your terminal

```
mkdir test && cd test
npm init
```

What is a package?
A module someone wrote, that you can download and use in your project.

```
npm install cowsay --save
```

Installs in the `node_modules` folder.

```
cd node_modules/cowsay
./cli.js What does the cow say?
```

ES6 JavaScript

The ECMAScript committee updates the specifications for JavaScript every year. They add features and improvements under the hood and change syntax for the better(Synctactic sugar).
ES2015 = ES6

const and let

`const` is a constant variable
`let` is like var but with block scope and cannot be used before it is defined

Arrow Functions

```
// ES2015 arrow function
const jump = (count = 0) => {
  console.log(count);
};
```

```
// ES5 equivalent
var jump = function jump() {
  var count = arguments.length > 0 && arguments[0] !== undefined ? arguments[0] :
0;

  console.log(count);
};
```

Classes

JavaScript now has Classes. Anyone who comes from an Object oriented language like Java or C will find this very familiar. We could achieve Object oriented behavior even earlier but now it's formalized in the language as a Class.

Template Strings

Compose strings with variables using backtick. Instead of complex concatenation. Also good for multi-line text.

```
function hello(firstName, lastName) {
  return `Good morning ${firstName} ${lastName}!
How are you?`
}
```

Spread operator

Makes a shallow copy of an object or an array with simple syntax. Also very easy to override certain values. Important operator when trying to copy by value instead of reference.

```
const person1 = {
    name: 'Karan',
    profession: 'Surfer',
    city: 'New York'
};

const person2 = {
    ...person1,
    name: 'Cherisha'    // Creates a NEW object with the same fields except name
};
```

Destructuring

Enables extraction of variables from keys of an object.

```
const printPerson = ({ name, city }) => {
    console.log(`${name} is in ${city}`);
};

printPerson(person1);
```

import and export

Easily import and export modules across different parts of your project or node_modules.

Default imports vs named imports

```
import DefaultImport from 'some-module';
import { NamedImport } from 'some-module';
```

Default exports vs named exports

```
const Module = { }
export default Module;

export const someFunction = () => { }
```

Weird Wide Web

Web standards come up with changes and feature improvements to the langauge.

Who has to respect these standards? Browsers.

Different browsers adopt different feaures at different speeds - Fragmented web.

Enter Babel

Babel is a code transformer. Works by using polyfills.

1. Converts code from ES6(and later) to ES5.
2. Injects news features into the output ES5.

Example transformation - http://bit.ly/babel12

The magic of Parcel ✨

Parcel.js is a zero configuration web application bundler. The zero configuration is the magic part.

Main alternative = Webpack.

After installing parcel, it's as simple as running

```
parcel src/index.html
```

It has Babel transforms by default.
It has hot module replacement.
Lot of other awesome features!

https://parceljs.org/

Project setup

Get the code at http://bit.ly/resume-maker

We are attempting to build a Resume builder today. Frontend-only project.

Run `npm install` to install all the dependencies.

If not already present, add this script to the scripts section

```
"scripts": {
    "start": "parcel src/index.html"
},
```

and run `npm start` to start the app.

React fundamentals

Non-opinionated front-end framework.

A simple way to remember it is that React lets you compose custom reusable frontend components.

You can compose a React component from simple html elements or other React elements.

Using a React element is as simple as `<MyAwesomeComponent />`

Uses something known as Shadow DOM internally for performance improvements.

props

Every React component can take in props to display information. A prop is like an attribute you pass to an element. Think `<a href='http://google.com'>`

props is a special object that a React component receives, that contains all of the attributes passed into it.

```
<Section
    title='Experience'
    items={data}
    color='#333'
/>
```

Defining a React Component

React lets you define components as classes or functions.
Classes give you more functionality.
Defining a React Component as a function is more like a shorthand and simplifies code.

Class Component

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

You HAVE to define the `render` function.

Functional Component

```
const Welcome = (props) => {
    return <h1>Hello, {props.name}</h1>;
};
```

Notice the difference between `this.props.name` and `props.name`

Lifecycle of a React component

Every React component is "mounted" onto the UI. There are lifestyle events like `constructor`, `render` and `componentDidMount` which we will use when we want advanced behavior.

Dumb components vs smart components

Also called presentational components and container components.

Dumb components

Only take in props and display data.
Usually written with shorthand notation.

Smart components

Concerned with the behavior of the component in the context of the app.
Talks to the application state.

propTypes

Library to enforce type-checking of the props that a component receives.

Read more: https://reactjs.org/docs/hello-world.html

Redux fundamentals

The application state. There is one centralised application state and this state controls exactly how the UI should behave. In other words, the UI depends on the current **state** of the app to display.

Every single action that is taken in the UI that is to be remembered or interacts with other parts of the application needs to talk to the state. It's like a common pool of information that all the different parts of the app read from and write to.

What are some examples?

Clicking on a button. It needs to lead to a change in the UI like maybe showing a popup.

Another is if the user sorts a list of items. How would you do it without a framework? You would say

```
<div className='sort-button'
    onClick={() => {
        // code to sort the list
    }}
>
Sort
</div>
```

But with the new Redux way, it would be more like this

```
<div className='sort-button'
    onClick={() => {
        // call a special function that records the sort direction on to the state
    }}
>
Sort
</div>
```

Actions

Plain JavaScript objects.
Send data from your application to your store.
You send them to the store using `store.dispatch()`
Describe what happened.

Must have a `type` property that indicates the type of action being performed.

```
{
    type: 'OPEN_MODAL'
}
```

Can include other optional data needed for the operation.

```
{
    type: 'CHANGE_TEXT',
    text: 'New random text'
}
```

Action Creators

Function that creates actions

```
export const CHANGE_TEXT = 'CHANGE_TEXT';

const addItem = (text) => {
  return {
    type: CHANGE_TEXT,
    text
  };
}
```

Reducers

Listen for actions and decide how the application's state changes based on received actions.
Returns new state.
Do not mutate the state.

```
import {
  CHANGE_TEXT,
} from 'actions';

const initialState = {
    text: 'Initial Text',
    otherInformation: 'Other',
};

const AppReducer = (state = initialState, action) => {
    switch(action.type): {
        case CHANGE_TEXT:
            return {
                ...state,   // Very important to return a NEW copy of the state
                text: action.text,
            };
        default:
            return state;
    }
};
```

Further reading - Combining Reducers - https://redux.js.org/basics/reducers

Connecting React and Redux

Library called react-redux that lets react speak to redux. React needs to be able to communicate with redux in 2 ways.

1. Read from Redux state
2. Call Redux Actions

connect function

It connects a component to the Redux Store. It gets access to the Redux store if some ancestor component is wrapped in a `<Provider></Provider>` tag.

mapStateToProps - Maps Redux state to the component props. Takes state as parameter.
mapDispatchToProps - Allows us to define functions that invoke redux actions using dispatch.

Sassy CSS

SCSS is backward compatible with CSS. All CSS is valid SCSS.

Allows nesting.
Allows variables.

SCSS

```
$blue-bg: #1111fe;

body {
    background: $blue-bg;
    .title {
        width: 100px;

        &__title-text {
            height: 10px;
        }
    }
}
```

Corresponding CSS

```
body {
  background: #1111fe;
}
body .title {
  width: 100px;
}
body .title__title-text {
  height: 10px;
}
```

The End.

Reach out to say hi, or discuss life.

Email - kr377@cornell.edu
Twitter - karanrajpal_
Linkedin - in/karanrajpal1/