1. HTML defines content 2. CSS specifies layout 3. JavaScript programs behaviour of web pages		
JavaScript		
Places were it is used: - Web pages - Desktop and server programs (e.g. Node.js) - Databases		
Math		
Math.sin()		
Math.PI		
parseInt('123', 10); //123 + '42'; // 42	Convert string to an integer - Optional second argument: base for conversion	E.g. convert binary to integer parseInt('11', 2); // 3
Others		
isNan();	Test for NaN	
isFinite();		
undefined	Value of type 'undefined', an initialised value - A constant	
prompt('Question answer?'	Store input from user as variable	
typeof var		

function fun_name() { … }
var s = new class_name(…, …)
fun_name.call(s);
	Sister function of *apply()*	Also equivalent to:
		s.fun_name = fun_name;
		s.fun_name();

Strings		
.length		
.charAt(<int>)		
.replace(<to replace>, <replace with>)		
.toUpperCase()		
Boolean		
false	- Equivalent to 0, "", NaN, null, undefined	
Boolean('');	Convert to Boolean	
Variables		
- If declare variables without defining, is of type undefined		

let	Declare block-level variables
	- Variable declared is available from *block* enclosed in
		let a;
		let name = 'Simon';
		for (let myLetVar = 0; myLetVar < 5; myLetVar++) {
		// variable only visible here
		}

const	Declare variables whose values are fixed - Variable available *from* block declared in - Cannot reassign new value, will throw error	const Pi = 3.14 Pi = 1; // error, cannot change constant variable
var	Variable declared is available from *function* declared in	

Operators		

==
!=
	If different types, operator performs type coercion	123 == '123' // true
		1 == true // true

===
!==
	Avoids type coercion	123 === '123' // false
		1 === true // false

Control structures		

if (…) {
} else if (…) {
} else {
}		

while (…) {
}		

do {
} while (…);
	do-while loops ensure that body of loop is executed at least once	

for (***) {
}
	for…of
	for…in
		for (let value of array) {
		}
		for (let property in object) {
		}

switch (var) {
    case value1:
    case value2:
    default:
}		

Objects		
- Similar to dictionaries (key-value pairs)		

var obj = new Object();
var obj = {};
	Create an empty object	

var obj = {
attr1: value1
attr2: value2
}
	Initialise object	var obj = {
		name = 'Carrot',
		for: 'Max', // 'for' is a reserved word
		details: {
		colour: 'orange',
		size: 12
		}
		}

function obj_fun(arg1, arg2) {
this.attr1 = value1;
this.attr2 = value2;
}
	Create an **object prototype**
	- Similar to creating a class, take in values for constructor	

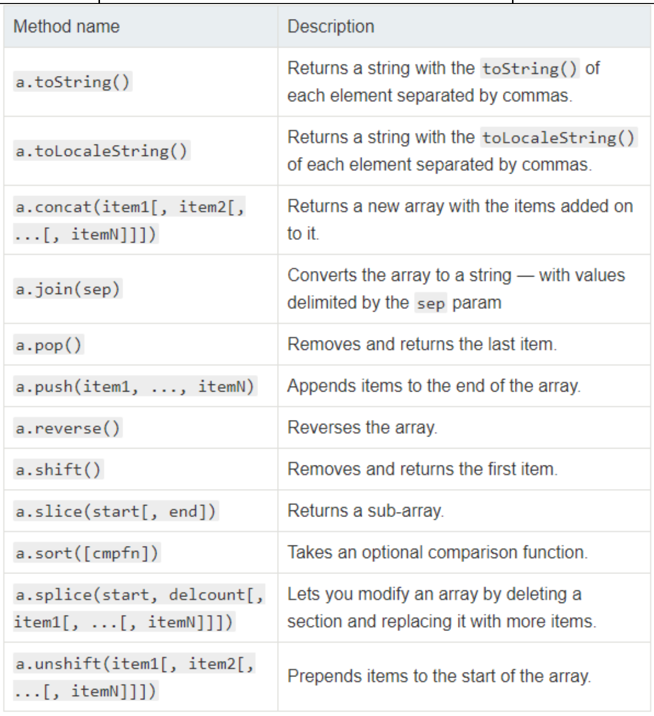
var var1 = new obj_fun(val1, val2)	Create instance of prototype	

obj.attr1.attr1_1
obj['attr1']['attr1.1']
	Access object properties	obj.details.colour; // orange
		obj['details']['size']; // 12

var key = "name" obj[key] = value	Define a new key-value pair in obj	

Arrays		
new Array(); […, …, …]		
Array(num_ele).fill(value)	Creates an array of specified length, filled with the same elements	
a[0]; a[1]; a[2];		
a.length;	Length of array is one more than the highest index - NOT the number of items in the array	var a = ['santa', 'rudolf', 'present'] a[100] = 'cookie' a.length; // 101
a.push(item);	Append item to array	

for (const currValue of a) {
 	
or
 	
a.forEach(function(…) {})
	Iterate over array	



Method name	Description
`a.toString()`	Returns a string with the `toString()` of each element separated by commas.
`a.toLocaleString()`	Returns a string with the `toLocaleString()` of each element separated by commas.
`a.concat(item1[, item2[, ...[, itemN]]])`	Returns a new array with the items added on to it.
`a.join(sep)`	Converts the array to a string — with values delimited by the `sep` param
`a.pop()`	Removes and returns the last item.
`a.push(item1, ..., itemN)`	Appends items to the end of the array.
`a.reverse()`	Reverses the array.
`a.shift()`	Removes and returns the first item.
`a.slice(start[, end])`	Returns a sub-array.
`a.sort([cmpfn])`	Takes an optional comparison function.
`a.splice(start, delcount[, item1[, ...[, itemN]]])`	Lets you modify an array by deleting a section and replacing it with more items.
`a.unshift(item1[, item2[, ...[, itemN]]])`	Prepends items to the start of the array.

arr →Array.prototype → Object.prototype → null	Prototype chain of an array	
arr.slice()	Create a copy of the array	
Functions		

function fun_name(arg1, arg2) {
    return …;
}		

arguments arguments.length arguments[i]	Additonal variable which can be accessed within the body of a function - An array-like object holding all of the values passed to the function	
function fun_name(…args) {	To pass in any number of arguments	

for (let value of args) {
}
}
	- Use **rest parameter operator (…)**	

function fun_name(arr) {
}
	Function that takes in an array	

fun_name.apply(null, […])	Call function with an arbitrary array of arguments - first argument is the object that should be treated as *this*	

var fun_name = function() {
};
	Anonymous function	

function class_name(…) {
return {
attr1: value1;
attr2: value2;
fun_name1: function() {
return …;},
fun_name2: function() {
return …;},
};
}
		Functions as classes
		- use *this* within the function to refer to the current object

function fun_class(arg1, arg2)
{
this.attr1 = arg1;
this.attr2 = arg2;
this.fun_name =
function() {
return …;
}
}
var a = new fun_class(…, …)
	☹: Each time create object, creating 2 brand new function objects	

new	1. Creates a brand new empty object 2. Calls the function specified 3. *this* set to that new object - Functions designed to be called by *new*: constructor functions - Common practice: capitalise these functions	

fn_or_class.prototype.attr =
…
	*** A better way of creating an object ☺
	- Can add extra methods to existing objects at runtime
	- Modify the properties of the class directly (all objects created from this class will have the new property)
		function Person(first, last) {
		this.first = first;
		this.last = last;
		}
		Person.prototype.fullName =
		function() {
		return this.first + ' ' + this.last;
		}

function fun_name(arg1 = default_value) {
};
	Default parameter for a function	

Closures		

	Function defined inside another function	function makeAdder(a) {
		    return function(b) {
		        return a + b;
		    };
		}
		var x = makeAdder(5);
		x(6); // 11

Prototype chain		
When trying to access a property, - check first if object has that property, - if not – check if the object's prototype has that property - if not, go deeper and check the prototype's prototype until the end of the prototype chain If there are multiple prototypes with the same property, the property found higher up is returned		

Example:
// Create object o from function f with its own properties a and b
let f = function() {
    this.a = 1;
    this.b = 2;
}
 	
// Add properties in function f's prototype
f.prototype.b = 3;
f.prototype.c = 4;
 	
// Full prototype chain:
// {a: 1, b: 2} → {b: 3, c: 4} → Object.prototype → null
// o.[[Prototype]] has properties b and c
// o.[[Prototype]].[[Prototype]] is Object.prototype
// o.[[Prototype]].[[Prototype]].[[Prototype]] is null (end of prototype chain)
		// Do not set prototype like this
		f.prototype = {b:3, c:4}

console.log(o.a); // 1, there is an 'a' own property on b
console.log(o.b); //2, there is a 'b' own property on o
// prototype also has 'b' property but not visited here (Property Shadowing)
console.log(o.d) // no property found, undefined		

Destructuring		
const {attr/key} = obj;	Extract **value of a key** (of an object) and assign it to a variable simultaneously	**Instead of:** const name = person.name;
const [attr1, attr2, …]	**Assign values to multiple variables simultaneously**	

import React, { Component} from ‘react’;
 	
class yourComponent extents Component {
}
	Extract Component directly from ‘react’ import	**Instead of:**
		import React from ‘react’;
		class yourComponent extends React.Component {
		}

Example:
 	
let TodoItem = ({taskName, isDone}) => (
    <li>
        <p>{taskName}<p>
        <input type=”checkbox” value={isDone} />
    </li>
);
	Destructuring function arguments	**Instead of:**
		TodoItem = (props) => (
		    <li>
		        <p>{props.taskName}</p>
		        <input type=”checkbox” value={props.isDone} />
		    </li>
		);

Cloning		
- Useful for when want to clone an object and modify the clone directly		

const class_name = {
    attr1: ‘valu1’,
    attr2: ‘valu2’
};
const clone_name = {
    attr1: class_name.attr1,
    attr2: class_name.attr2
};
clone_name.attr3 = ‘valu3’
	Common way
	- Only clone is changed to have an additional property	

Object.assign()
Object.assign({ }, …, …)
 	
const obj_name = {
    attr1: ‘valu1’,
    attr2: ‘valu2’,
};
const clone_name = Object.assign({ }, obj_name);
	Other way to clone objects
	- Takes in:
	    1. Object to clone to (e.g. empty object with no properties, { })
	    2. A variable number of objects
	- Can clone properties from an arbitrary number of objects
	- If same properties show up, previous values will be overwritten (i.e. properties cloned from left to right)
	- **Does a shallow clone** (i.e. if values from original object changed, cloned value remains)	

Tags		
<span></span>	<form></form>	<label></label>
<input type=”text” value={this.state.content}/>	<nav className=”nav-wrapper”></nav>	<div className=”container”></div>
<ul><li></li></ul>		

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>React Basics</title>
  <script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
  <script crossorigin src="https://unpkg.com/react@16/umd/react.development.js"></script>
  <script crossorigin src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>
</head>
<body>
  <div id="app"></div>
  <script type="text/babel">
    class App extends React.Component {
```

***
Root component
***

```
      state = {
        name: 'Ryu',
        age: 30
      }
      handleChange = (e) => {
        this.setState({
          name: e.target.value
        })
      }
```

Takes in event object ***e***

Arrow function within component so that can reference *this* or modify it

```
      handleSubmit = (e) => {
        e.preventDefault();
```

Prevent default behaviour of button: refreshes

```
        console.log('form submitted', this.state);
      }
      render(){
        return(
```

Generally return only **one** root element ***<div></div>*** (there can be many nested within)

```
          <div className="app-content">
```

Use ***className*** since ***class*** is a keyword reserved for JavaScript

```
            <h1>My name is {this.state.name}</h1>
            <form onSubmit={this.handleSubmit}>
              <input type="text" onChange={this.handleChange} />
              <button>Submit</button>
            </form>
          </div>
        )
      }
    }

    ReactDOM.render(<App />, document.getElementById('app'));
  </script>
</body>
</html>
```

Triggered when *Enter* pressed or button

Creates a text field for user to input: fires everytime there is a change in input

```
render(){
  return(
    <div className="app-content">
      <h1>Hello, ninjas!</h1>
      <p>My name is: { this.state.name } and I am { this.state.age }</p>
      <button onClick={this.handleClick}>Click me</button>
      <button onMouseOver={this.handleMouseOver}>Hover me</button>
      <p onCopy={this.handleCopy}>What we think, we become</p>
    </div>
  )
}
```

Use curly braces ***{ }*** to output dynamic data/JavaScript

When clicked or hovered over, function will be triggered

When text selected and copied, function triggered

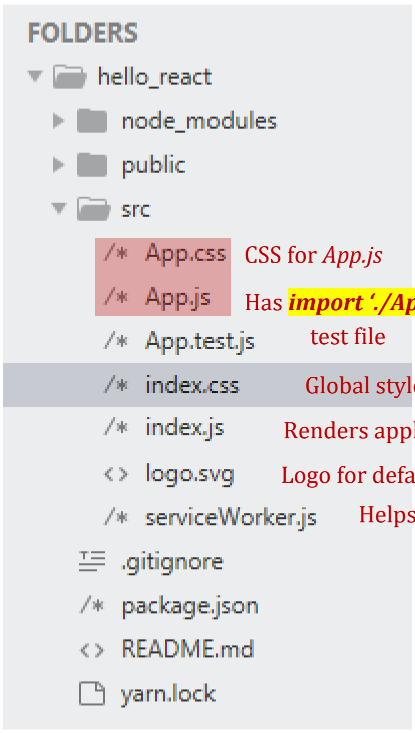
Don't use ***={this.handleCopy()}*** with parentheses ***( )*** because will invoke function straight away when page loads

```
handleMouseOver(e){
  console.log(e.target, e.pageX);
}
```

***e*.pageX:** records x-coord of mouse position on page

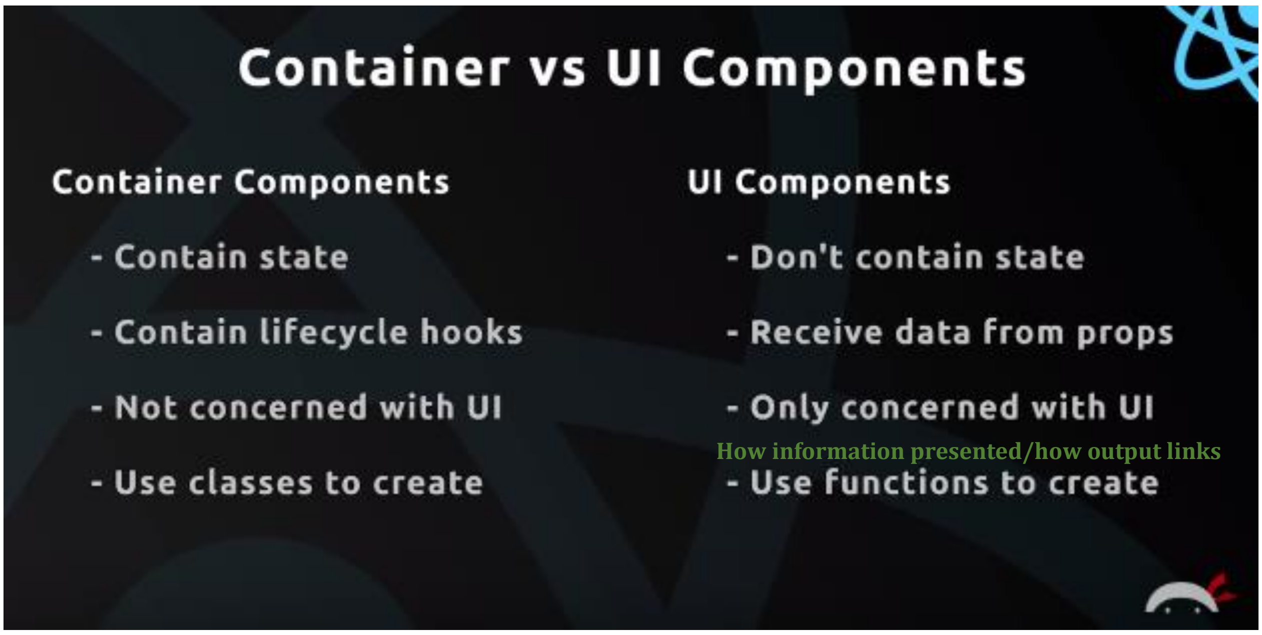
event.target: what user clicked to cause event/where event originally fired from

Basic create-react-app files



FOLDERS

- hello_react
  - node_modules
  - public
  - src
    - App.css — CSS for *App.js*
    - App.js — Has ***import './App.css';*** at the beginning and ***export default App;*** at the end
    - App.test.js — test file
    - index.css — Global style sheet for application (applies everywhere)
    - index.js — Renders application to DOM
    - logo.svg — Logo for default template app created initially
    - serviceWorker.js — Helps with caching assets and files for better UX
  - .gitignore
  - package.json
  - README.md
  - yarn.lock



Container vs UI Components

Container Components	UI Components
- Contain state	- Don't contain state
- Contain lifecycle hooks	- Receive data from props
- Not concerned with UI	- Only concerned with UI
- Use classes to create	- Use functions to create

How information presented/how output links

```
import React, { Component } from 'react';
import Ninjas from './Ninjas'
```

From the default template:
import logo from './logo.svg'; (for images)
import './App.css'; (for CSS)

```
class App extends Component {
```
"Parent component"

```
  state = {
    ninjas: [
      { name: 'Ryu', age: 30, belt: 'black', id: 1 },
      { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
    ]
  }
```

Pass 3 props (i.e. "properties") with values into Ninjas component

Instead of writing so many lines of Ninjas components, cycle through a list of ninjas

```
  render() {
    return (
      <div className="App">
        <h1>My first React app</h1>
        <Ninjas ninjas={this.state.ninjas}/>
      </div>
    );
  }
}

export default App;
```

Nest Ninja component within (i.e. "child component")

```
import React, { Component } from 'react'
```

Whenever create a class-based component, need to import

```
class Ninjas extends Component{
  render(){
    const { ninjas } = this.props;
```

To destructure to get properties directly, ***const {name, age, belt} = this.props;***

this refers to the component, ***props*** refers to properties

```
    const ninjaList = ninjas.map(ninja => {
      return (
        <div className="ninja" key={ninja.id}>
          <div>Name: { ninja.name }</div>
          <div>Age: { ninja.age }</div>
          <div>Belt: { ninja.belt }</div>
        </div>
      )
    });
```

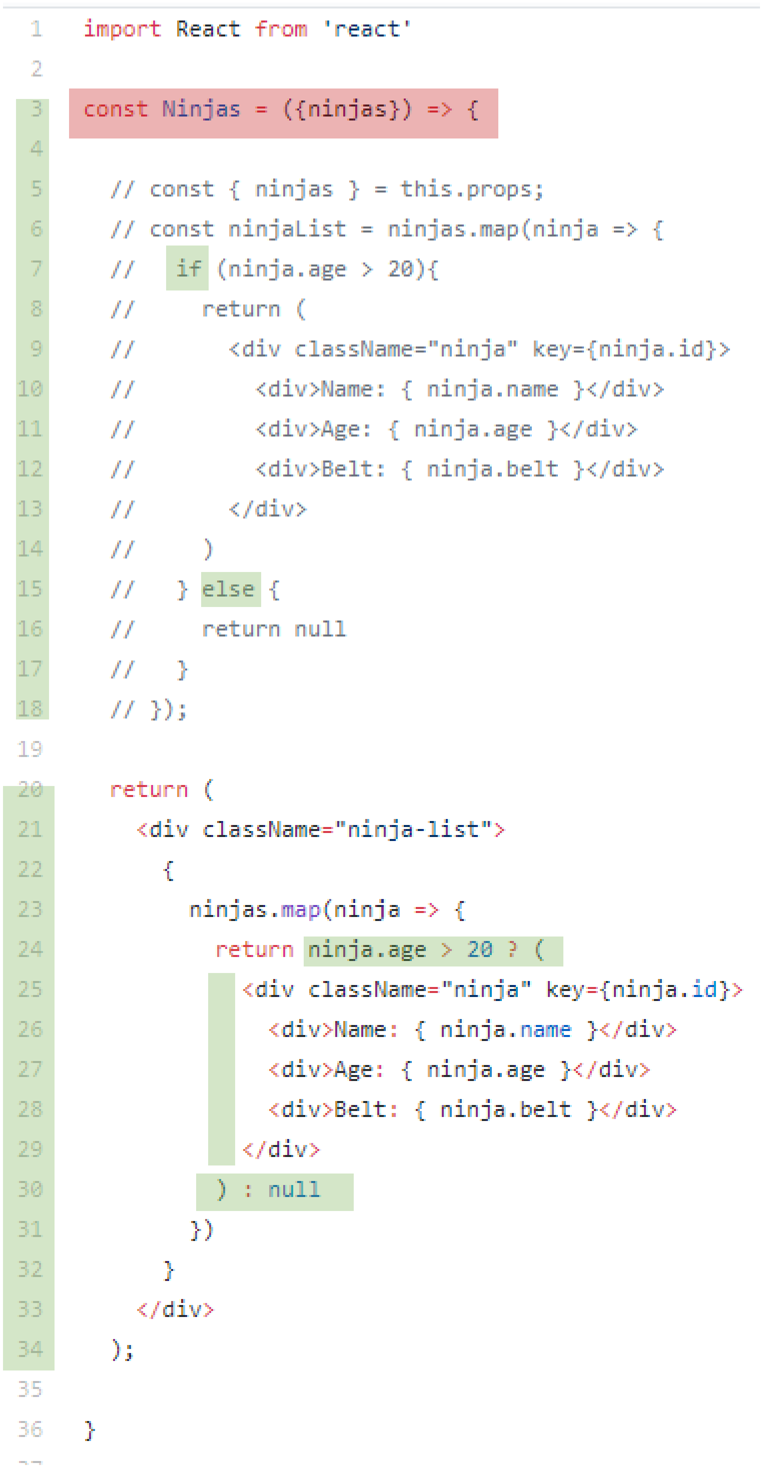
Maps each element in array to new structure by applying function, loops through data and outputs an array

ninjas.map(ninja => {...}) where ***{return(...)}***
- React expects a key when using the map function

Set a unique key for each element so that React can identify for easier addition/removal

```
    return (
      <div className="ninja-list">
        { ninjaList }
      </div>
    )
  }
}

export default Ninjas
```

Export so can import in ***App.js***



```
import React from 'react'

const Ninjas = ({ninjas}) => {

  // const { ninjas } = this.props;
  // const ninjaList = ninjas.map(ninja => {
  //   if (ninja.age > 20){
  //     return (
  //       <div className="ninja" key={ninja.id}>
  //         <div>Name: { ninja.name }</div>
  //         <div>Age: { ninja.age }</div>
  //         <div>Belt: { ninja.belt }</div>
  //       </div>
  //     )
  //   } else {
  //     return null
  //   }
  // });

  return (
    <div className="ninja-list">
      {
        ninjas.map(ninja => {
          return ninja.age > 20 ? (
            <div className="ninja" key={ninja.id}>
              <div>Name: { ninja.name }</div>
              <div>Age: { ninja.age }</div>
              <div>Belt: { ninja.belt }</div>
            </div>
          ) : null
        })
      }
    </div>
  );

}
```

Access props as parameters now (as functional components)

Method 1: conditional output

Method 2:

condition? (value if true) : (value if false)

```
import React, { Component } from 'react';
import Ninjas from './Ninjas'
import AddNinja from './AddNinja'

class App extends Component {
  state = {
    ninjas: [
      { name: 'Ryu', age: 30, belt: 'black', id: 1 },
      { name: 'Yoshi', age: 20, belt: 'green', id: 2 },
      { name: 'Crystal', age: 25, belt: 'pink', id: 3 }
    ]
  }
  addNinja = (ninja) => {
    ninja.id = Math.random();
    let ninjas = [...this.state.ninjas, ninja];
    this.setState({
      ninjas: ninjas
    });
  }
```

Adds a new property to ***ninja***
(similar to adding a new column to a dataframe with df$new_col)

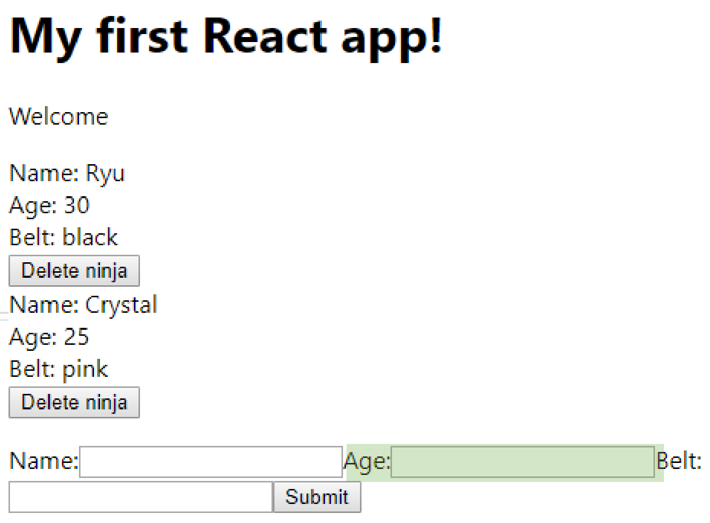
Spread operator: creates a copy of an array
- Takes individual elements of array as elements of new array

```
  deleteNinja = (id) => {
    // console.log(id);
    let ninjas = this.state.ninjas.filter(ninja => {
      return ninja.id !== id
    });
    this.setState({
      ninjas: ninjas
    });
  }
```

Filter function: returns new arrat
- Filters out elements whose conditional statement evaluates to **false**

```
  render() {
    return (
      <div className="App">
        <h1>My first React app</h1>
        <Ninjas ninjas={this.state.ninjas} deleteNinja={this.deleteNinja} />
        <AddNinja addNinja={this.addNinja} />
      </div>
    );
  }
}

export default App;
```

```
import React from 'react'

const Ninjas = ({ninjas, deleteNinja}) => {
  return (
    <div className="ninja-list">
      {
        ninjas.map(ninja => {
          return (
            <div className="ninja" key={ninja.id}>
              <div>Name: { ninja.name }</div>
              <div>Age: { ninja.age }</div>
              <div>Belt: { ninja.belt }</div>
              <button onClick={() => {deleteNinja(ninja.id)}}>Delete ninja</button>
              <hr />
            </div>
          )
        })
      }
    </div>
  );
}

export default Ninjas
```



My first React app!

Welcome

Name: Ryu
Age: 30
Belt: black
Delete ninja
Name: Crystal
Age: 25
Belt: pink
Delete ninja

Name: Age: Belt:
Submit

```
import React, { Component } from 'react'

class AddNinja extends Component {
  state = {
    name: null,
    age: null,
    belt: null
  }
  handleChange = (e) => {
    this.setState({
      [e.target.id]: e.target.value
    });
  }
  handleSubmit = (e) => {
    e.preventDefault();
    this.props.addNinja(this.state);
  }
  render() {
    return (
      <div>
        <form onSubmit={this.handleSubmit}>
          <label htmlFor="name">Name:</label>
          <input type="text" id="name" onChange={this.handleChange} />
          <label htmlFor="age">Age:</label>
          <input type="text" id="age" onChange={this.handleChange} />
          <label htmlFor="belt">Belt:</label>
          <input type="text"id="belt" onChange={this.handleChange} />
          <button>Submit</button>
        </form>
      </div>
    )
  }
}

export default AddNinja
```

htmlFor is like ***className*** but for HTML

```
componentDidMount(){
  console.log('component mounted');
}
componentDidUpdate(prevProps, prevState, snapshot){
  console.log('component updated');
  console.log(prevProps, prevState);
}
render() {
```

component only mounts once (until page refreshes)

fires when there is a change in state/props

```
import React, { Component } from 'react';
import Todos from './Todos'
import AddTodo from './AddTodo'

class App extends Component {
  state = {
    todos: [
      {id: 1, content: 'buy some milk'},
      {id: 2, content: 'play mario kart'}
    ]
  }
  deleteTodo = (id) => {
    const todos = this.state.todos.filter(todo => {
      return todo.id !== id
    });
    this.setState({
      todos
    });
  }
  addTodo = (todo) => {
    todo.id = Math.random();
    let todos = [...this.state.todos, todo];
    this.setState({
      todos
    });
  }
  render() {
    return (
      <div className="todo-app container">
        <h1 className="center blue-text">Todo's</h1>
        <Todos todos={this.state.todos} deleteTodo={this.deleteTodo} />
        <AddTodo addTodo={this.addTodo} />
      </div>
    );
  }
}

export default App;
```

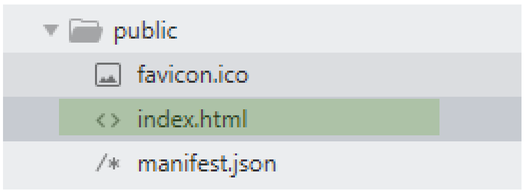
Function

Reduce the container from spanning across the entire page to a smaller one

```jsx
import React from 'react';

const Todos = ({todos, deleteTodo}) => {

  const todoList = todos.length ? (
    todos.map(todo => {
      return (
        <div className="collection-item" key={todo.id}>
          <span onClick={() => {deleteTodo(todo.id)}}>{todo.content}</span>
        </div>
      )
    })
  ) : (
    <p className="center">You have no todo's left, yay!</p>
  );

  return (
    <div className="todos collection">
      {todoList}
    </div>
  )
}

export default Todos;
```

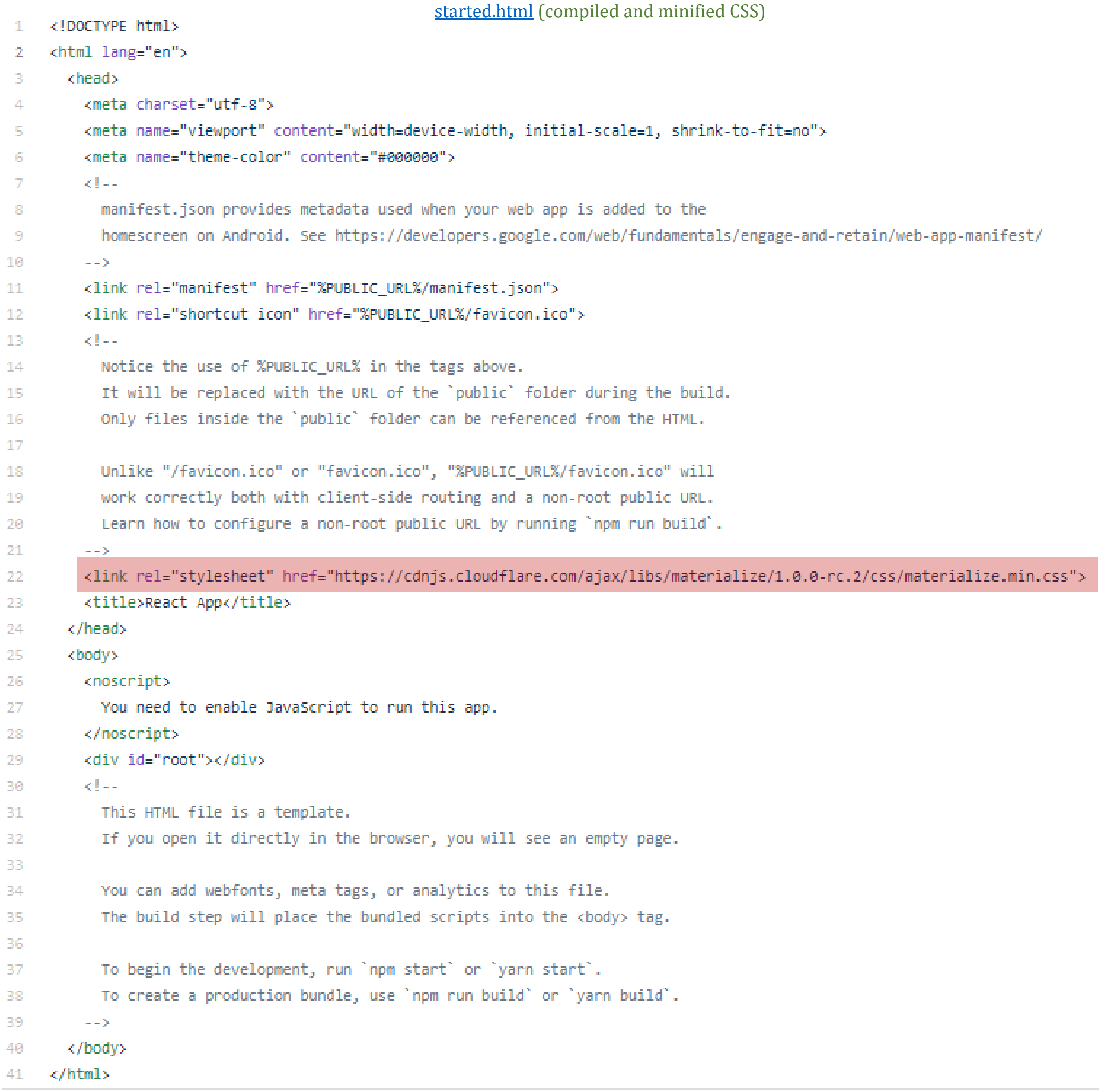
```jsx
import React, { Component } from 'react'

class AddTodo extends Component {
  state = {
    content: ''
  }
  handleChange = (e) => {
    this.setState({
      content: e.target.value
    });
  }
  handleSubmit = (e) => {
    e.preventDefault();
    // call function to add a todo
    this.props.addTodo(this.state);
    this.setState({
      content: ''
    })
  }
  render() {
    return (
      <div>
        <form onSubmit={this.handleSubmit}>
          <label>Add a new todo:</label>
          <input type="text" onChange={this.handleChange} value={this.state.content} />
        </form>
      </div>
    )
  }
}

export default AddTodo
```

Input field updates to blank when form submitted

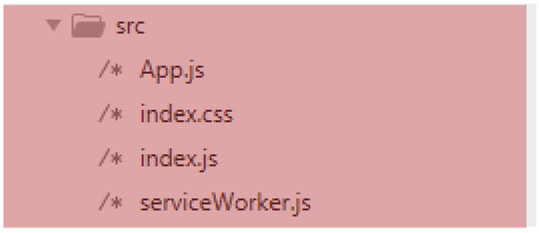


```
public
  favicon.ico
  index.html
  manifest.json
```

Add in link to basic style sheet
- Taken from https://materializecss.com/getting-started.html (compiled and minified CSS)

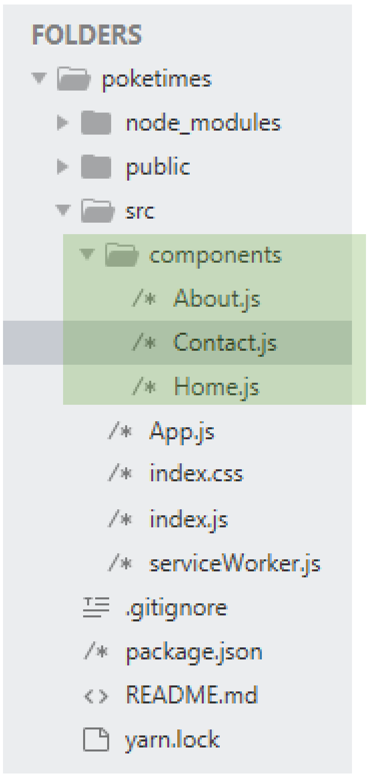


```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <!--
      manifest.json provides metadata used when your web app is added to the
      homescreen on Android. See https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0-rc.2/css/materialize.min.css">
    <title>React App</title>
  </head>
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```



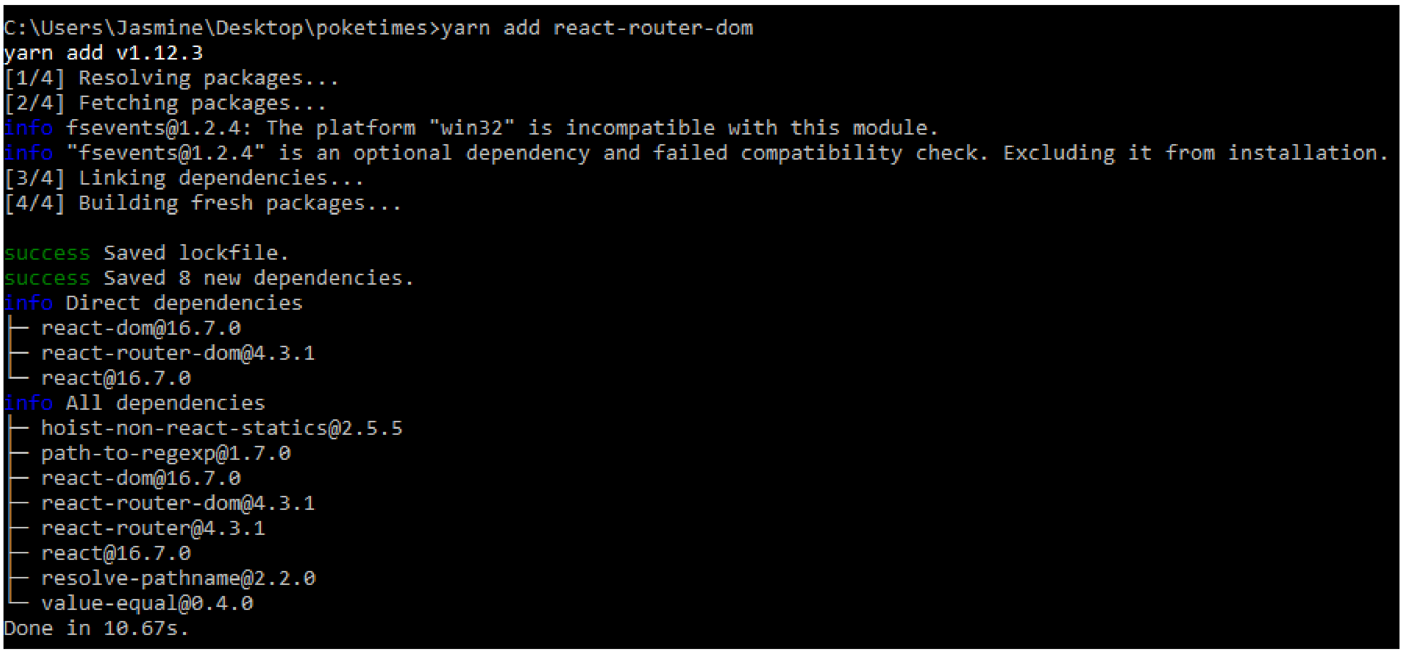
```
src
  App.js
  index.css
  index.js
  serviceWorker.js
```

Files to keep
- some imports removed



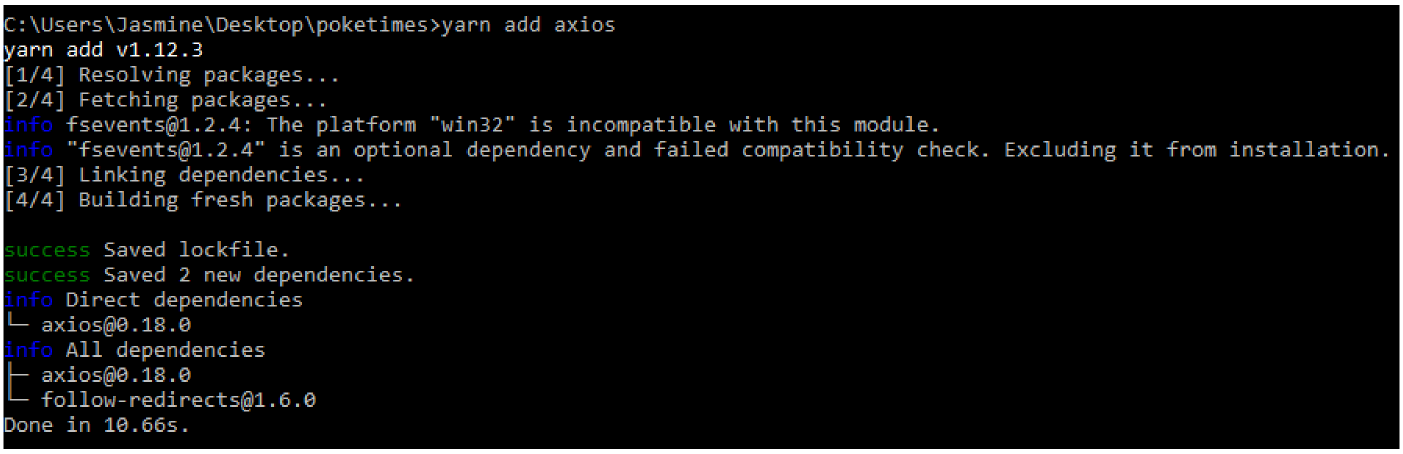
Sort of represents the 3 pages in the application
- E.g. .../About
  .../Contact
  .../Home

Install package that allows set-up of router in application:

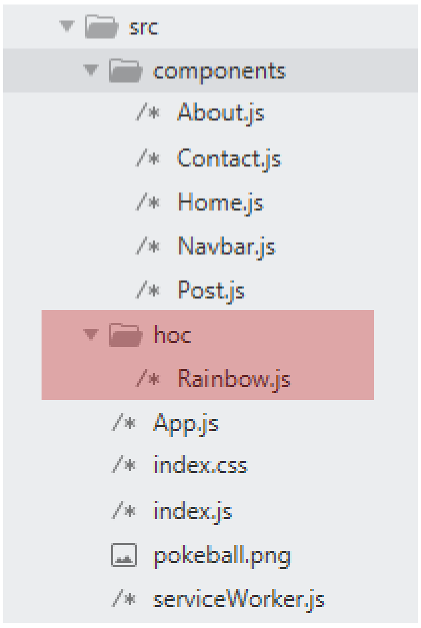


```
C:\Users\Jasmine\Desktop\poketimes>yarn add react-router-dom
yarn add v1.12.3
[1/4] Resolving packages...
[2/4] Fetching packages...
info fsevents@1.2.4: The platform "win32" is incompatible with this module.
info "fsevents@1.2.4" is an optional dependency and failed compatibility check. Excluding it from installation.
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Saved lockfile.
success Saved 8 new dependencies.
info Direct dependencies
├─ react-dom@16.7.0
├─ react-router-dom@4.3.1
└─ react@16.7.0
info All dependencies
├─ hoist-non-react-statics@2.5.5
├─ path-to-regexp@1.7.0
├─ react-dom@16.7.0
├─ react-router-dom@4.3.1
├─ react-router@4.3.1
├─ react@16.7.0
├─ resolve-pathname@2.2.0
└─ value-equal@0.4.0
Done in 10.67s.
```

Install HTTP request library, fetches data from an external source



```
C:\Users\Jasmine\Desktop\poketimes>yarn add axios
yarn add v1.12.3
[1/4] Resolving packages...
[2/4] Fetching packages...
info fsevents@1.2.4: The platform "win32" is incompatible with this module.
info "fsevents@1.2.4" is an optional dependency and failed compatibility check. Excluding it from installation.
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Saved lockfile.
success Saved 2 new dependencies.
info Direct dependencies
└─ axios@0.18.0
info All dependencies
├─ axios@0.18.0
└─ follow-redirects@1.6.0
Done in 10.66s.
```



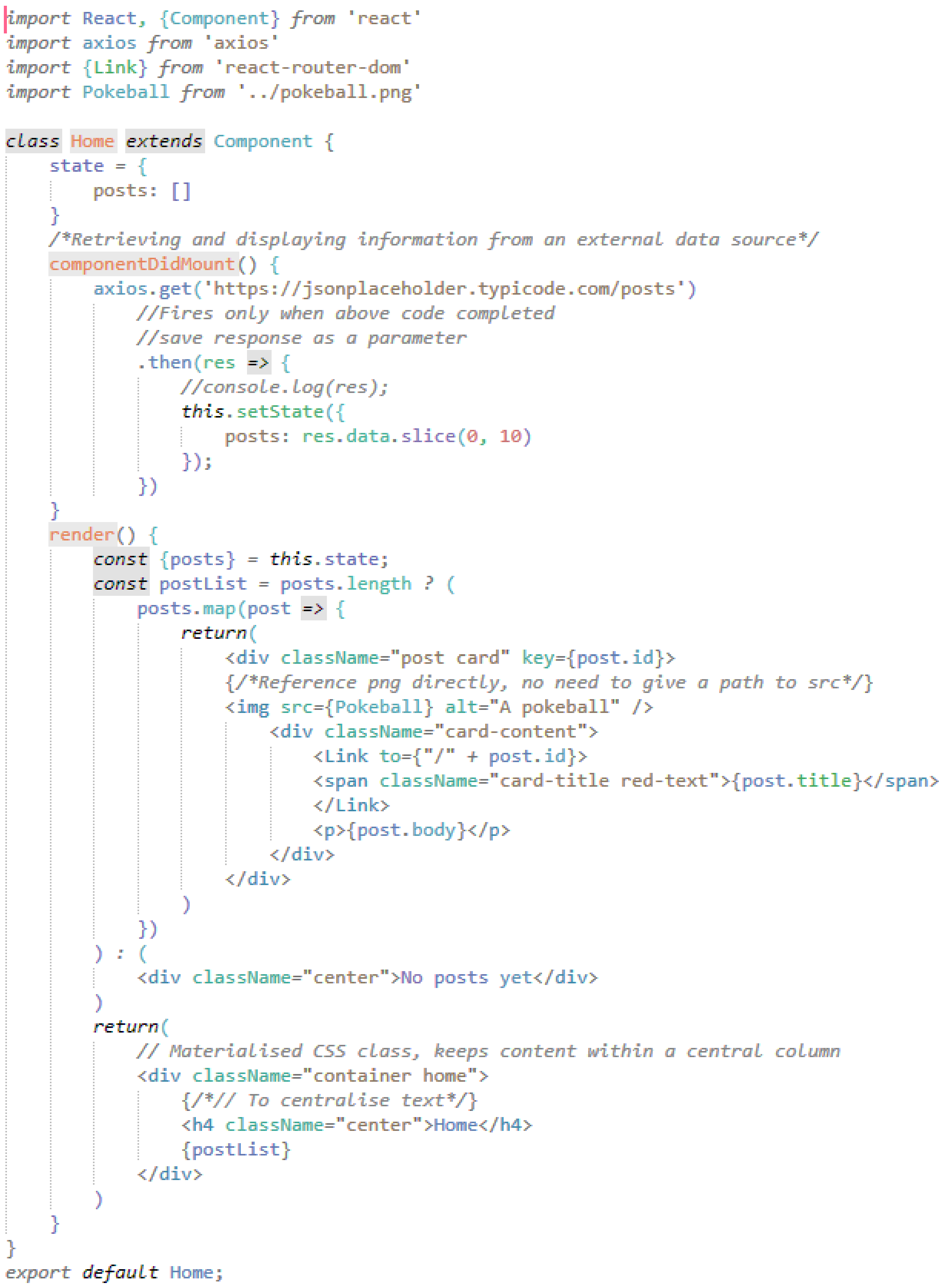
```
src
  components
    About.js
    Contact.js
    Home.js
    Navbar.js
    Post.js
  hoc
    Rainbow.js
  App.js
  index.css
  index.js
  pokeball.png
  serviceWorker.js
```

higher order component

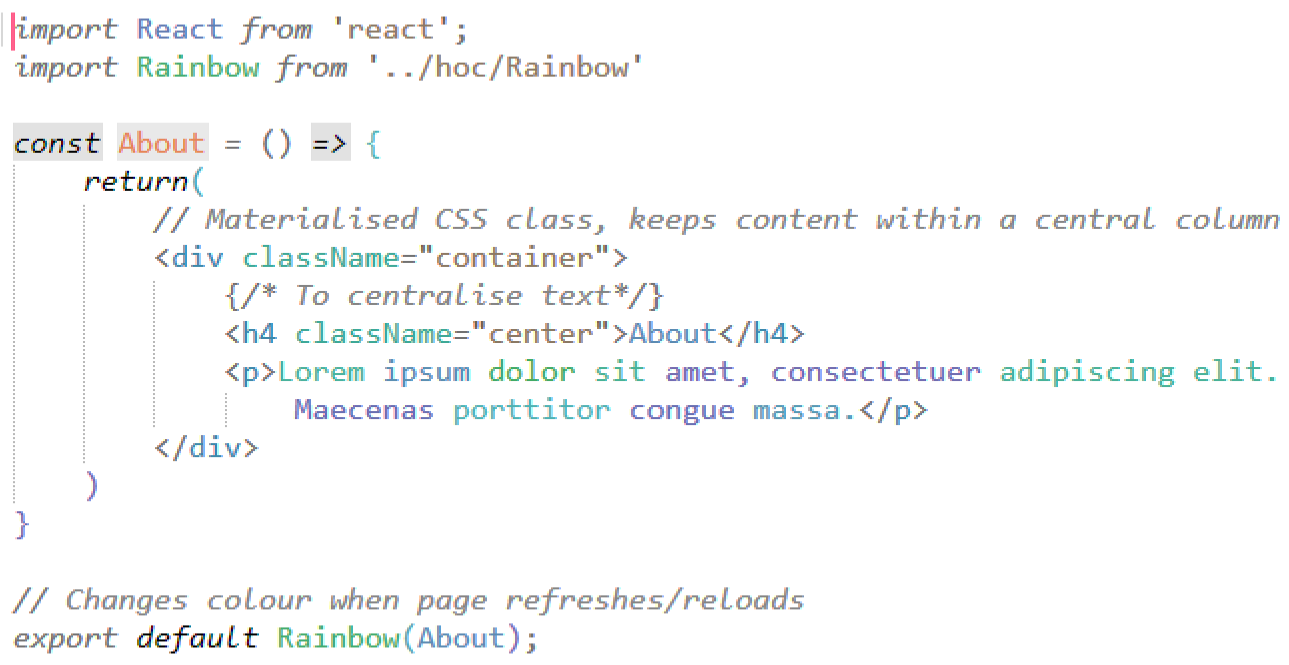
```
import React, { Component } from 'react';
import Navbar from './components/Navbar';
// Surrounding entire application in JSX with BrowserRouter tag
import {BrowserRouter, Route, Switch} from 'react-router-dom';
import Home from './components/Home';
import About from './components/About';
import Contact from './components/Contact';
import Post from './components/Post'

class App extends Component {
  render() {
    return (
      // Add properties to props for components nested within it
      <BrowserRouter>
        <div className="App">
          <Navbar />
          {/*Indicate that want to match ONLY ONE route at a time, not multiple
          - searches for it from routes top to bottom,
          - if found, will stop searching*/}
          <Switch>
            {/* Load routes here */}
            {/* Whenever user goes to this route/URL, load in component at this position */}
            {/*<Route path="/home" component={Home} />*/}
            {/*Need to add in 'exact' because else if will load in as well when About/Contact
              are clicked due to them having URLs that are a subset of (/)*/}
            <Route exact path="/" component={Home} />
            <Route path="/about" component={About} />
            <Route path="/contact" component={Contact} />
            {/*When click on a post, will go to its own page
            - similar URLs, but different extensions
            - need to put exact path since it treats 'Contact' and 'About' matching to'post_id'*/}
            <Route path="/:post_id" component={Post} />
          </Switch>
        </div>
      </BrowserRouter>
    );
  }
}

export default App;
```

```
import React from 'react'
// NOTE
import {Link, /*NavLink,*/ withRouter} from 'react-router-dom';

const Navbar = (props) => {
    //*Redirect to the About page no matter go which page*/}
    /*setTimeout(() => {
        props.history.push("/")
    }, 2000)*/
    return (
        <nav className="nav-wrapper red darken-3">
            <div className="container">
                <a href="google.com" className="brand-logo left">PokeTimes</a>
                {/* Display a series of links */}
                <ul className="right">
                    {/* Set the URLs for the different components*/}
                    {/*<li><a href="/home">Home</a></li>*/}
                    {/* Anchor tags
                    <li><a href="/">Home</a></li>
                    <li><a href="/about">About</a></li>
                    <li><a href="/contact">Contact</a></li>*/}
                    {/*Change to 'Link' tags
                    - 'to' decides where link goes
                    - prevents default action from occuring (page reloads everytime,
                                                            sends additional requests to server)
                    - if change to 'NavLink', get an active class for the link */}
                    <li><Link to="/">Home</Link></li>
                    <li><Link to="/about">About</Link></li>
                    <li><Link to="/contact">Contact</Link></li>
                </ul>
            </div>
        </nav>
    )
}
/*'Higher-order component
- add props to the props object in the Navbar component
- s.t. can have information about router in super-charged component*/
export default withRouter(Navbar);
```

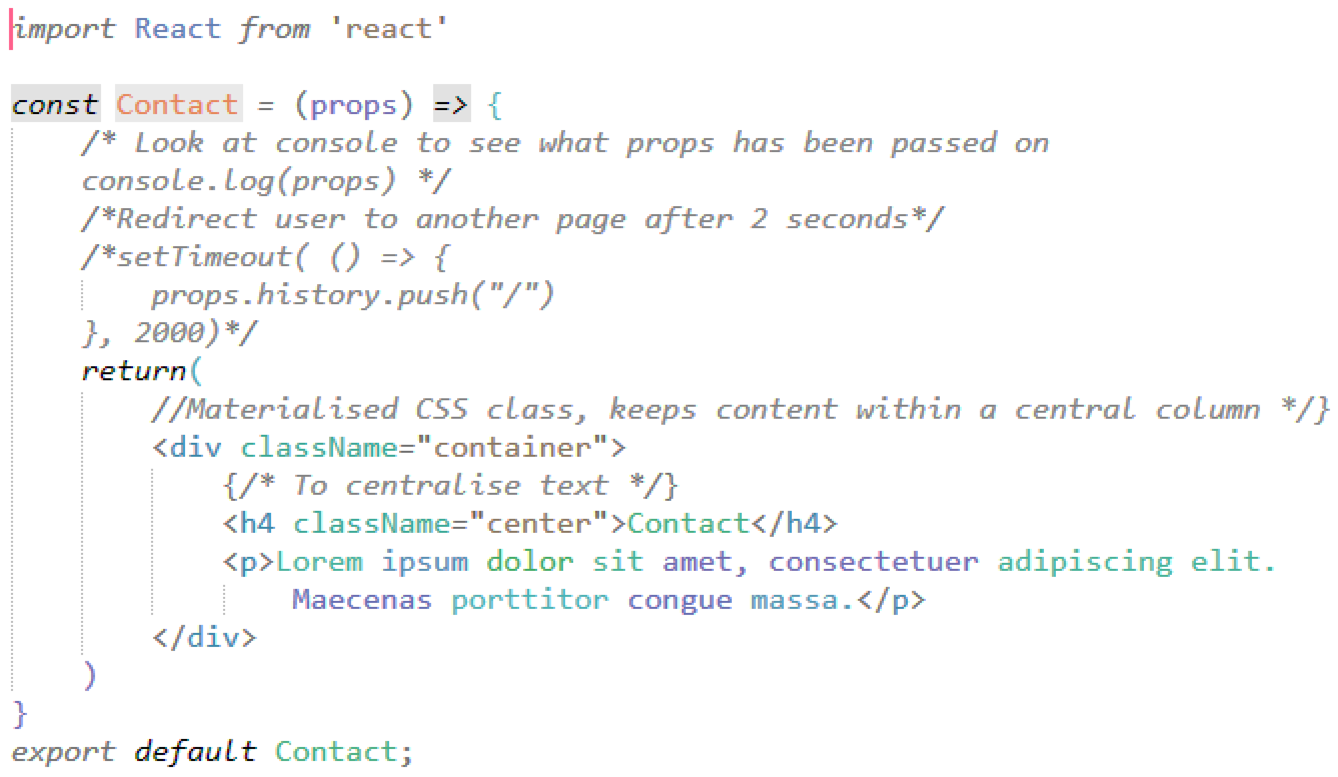
```
import React, {Component} from 'react'
import axios from 'axios'

class Post extends Component {
    state = {
        post: null
    }
    componentDidMount() {
        // post_id taken from App.js
        let id = this.props.match.params.post_id;
        // Retrieves that individual post
        axios.get('https://jsonplaceholder.typicode.com/posts/' + id)
            .then(res => {
                this.setState({
                    post: res.data
                });
                console.log(res);
            });
    }
    // render() method generates a template
    render() {

        const post = this.state.post ? (
            <div className="post">
                <h4 className = "center">{this.state.post.title}</h4>
            </div>
        ) : (
            <div className="center">Loading post...</div>
        )
        return(
            <div className="container">
                <h4>{post}</h4>
            </div>
        )
    }
}

export default Post
```



```
import React, {Component} from 'react'
import axios from 'axios'
import {Link} from 'react-router-dom'
import Pokeball from '../pokeball.png'

class Home extends Component {
    state = {
        posts: []
    }
    /*Retrieving and displaying information from an external data source*/
    componentDidMount() {
        axios.get('https://jsonplaceholder.typicode.com/posts')
            //Fires only when above code completed
            //save response as a parameter
            .then(res => {
                //console.log(res);
                this.setState({
                    posts: res.data.slice(0, 10)
                });
            })
    }
    render() {
        const {posts} = this.state;
        const postList = posts.length ? (
            posts.map(post => {
                return(
                    <div className="post card" key={post.id}>
                    {/*Reference png directly, no need to give a path to src*/}
                    <img src={Pokeball} alt="A pokeball" />
                        <div className="card-content">
                            <Link to={"/" + post.id}>
                            <span className="card-title red-text">{post.title}</span>
                            </Link>
                            <p>{post.body}</p>
                        </div>
                    </div>
                )
            })
        ) : (
            <div className="center">No posts yet</div>
        )
        return(
            // Materialised CSS class, keeps content within a central column
            <div className="container home">
                {/*// To centralise text*/}
                <h4 className="center">Home</h4>
                {postList}
            </div>
        )
    }
}
export default Home;
```



```
import React from 'react';
import Rainbow from '../hoc/Rainbow'

const About = () => {
    return(
        // Materialised CSS class, keeps content within a central column
        <div className="container">
            {/* To centralise text*/}
            <h4 className="center">About</h4>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
                Maecenas porttitor congue massa.</p>
        </div>
    )
}

// Changes colour when page refreshes/reloads
export default Rainbow(About);
```



```
import React from 'react'

const Contact = (props) => {
    /* Look at console to see what props has been passed on
    console.log(props) */
    /*Redirect user to another page after 2 seconds*/
    /*setTimeout( () => {
        props.history.push("/")
    }, 2000)*/
    return(
        //Materialised CSS class, keeps content within a central column */}
        <div className="container">
            {/* To centralise text */}
            <h4 className="center">Contact</h4>
            <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
                Maecenas porttitor congue massa.</p>
        </div>
    )
}
export default Contact;
```

```
import React from 'react'

/*Wraps a component and supercharges it*/
const Rainbow = (WrappedComponent) => {
    const colours = ["red", "pink", "orange", "blue", "green", "yellow"];
    const randomColour = colours[Math.floor(Math.random()*5)]
    const className = randomColour + "-text"
    return(props) => {
        return(
            <div className={className}>
                <WrappedComponent {...props}/>
            </div>
        )
    }
}
export default Rainbow;
```

```
body {
  margin: 0;
  padding: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
    monospace;
}

{/*Look at Home.js, where img is placed*/}
.home .post img {
  position: absolute;
  top: 20px;
  left: -100px;
  opacity: 0.6;
}

.home .post {
  overflow: hidden;
  /* Shifts text in container to right to make space for pokeball */
  padding-left: 80px;
}
```

bind() – creates a new function that when called, has its *this* keyword set to the provided value, with a given sequence of arguments preceding any provided when new function is called