Full-Stack Web Apps with Meteor and React

Project 1 - Score Keep

- create the new project in your terminal

```
meteor create --release 1.4.2.1 score-keep

cd score-keep

meteor npm install

meteor run --release 1.4.2.1
```

- we are going to start our app by stripping down everything
- open up score keep on the text editor and go main.js in the server folder and delete everything there and type

```
console.log('Log from /server/ main.js');
```

- go to the client folder and delete everything in the main.js

```
console.log('Log from /client/main.js');
```

- create a new file in the root directory called app.js and type

```
console.log('Log from app.js');
```

- lets create a folder called imports in the root directory
- create another one called public on the root directory and a file called help.html and type

```
Help Page Here
```

- to see that text you need to go to http://localhost:3000/**help.html**
- **go to your main.html and delete everything but only the below**

```
<head>

  <title>Score Keep</title>

</head>
```

```
<body>
  <h1>Score Keep</h1>


</body>
```

Importing And Exporting

- delete app.js from imports
- create a file under imports called utils.js
- lets do a console log in that file so we now exaclty when that file is being executed

```
console.log('Log from /imports.utils.js');
```

- to actually see this log lets import it in the main.js server folder

```
import './../imports/utils'
```

- and lets do the same thing from the client folder file main.js

```
import './../imports/utils'
```

- now let's go to utils.js and let's do a functionality

```
console.log('Log from /imports.utils.js');


export let greetUser = function(){
    return 'hello user!';
};
```

- make sure to have export
- and now go server main.js and let's import this

```
import {greetUser} from './../imports/utils';


console.log('Log from /server/ main.js');
console.log(greetUser());
```

- and now we can do the same thing with client main.js

```
import {greetUser} from  './../imports/utils';


console.log('Log from /client/main.js');

console.log(greetUser());
```

- and the above will show on the console.log because it is located in the client folder
- now over in utils.js lets export a string

```
console.log('Log from /imports.utils.js');


export let greetUser = function(){

    return 'hello user!';

};


export let name = 'Jose Mendoza';
```

- **and now over at the main.js in the client directory we want to be able to access both**

```
import {greetUser, name} from  './../imports/utils';


console.log('Log from /client/main.js');

console.log(greetUser());

console.log(name);
```

- **now in the imports folder create a file called math.js and add the following**

```
export let add = function(a, b){

    return a + b;

}
```

- go over to main.js in the server folder and import it and call the function

```
import {greetUser} from './../imports/utils';

import{add} from './../imports/math';


console.log('Log from /server/ main.js');

console.log(add(2, 6));
```

Advanced Importing and Exporting

- so far we've only used name exports but now we are going to be using default exports
- go over to utils.js and type this

```
console.log('Log from /imports.utils.js');


export let greetUser = function(){

    return 'hello user!';

};


export let name = 'Jose Mendoza';


export default 'Default val';
```

- **go over to client main.js and import it**

```
import someDefault, {greetUser, name} from  './../imports/utils';


console.log('Log from /client/main.js');

console.log(greetUser());

console.log(someDefault);
```

- and now go over to the browser console and see Default val is showing
- over in math.js let's do another default export

```
let add = function(a, b){

    return a + b;

}


export default add;
```

- and let's import that default in our server main.js

```
// import './../imports/utils'

import {greetUser} from './../imports/utils';

import add from './../imports/math';
```

```
console.log('Log from /server/ main.js');

console.log(add(2, 6));
```

Installing React and Exploring JSX

- **go to your browser and google npm react**
- **go to the first link**
- **go to another tab and google npm react-dom**
- **now go to your terminal and check what version of npm you are usin**

```
meteor npm --version

meteor npm install react@15.3.2 react-dom@15.3.2
```

- make sure to use those versions because of the apps that we are going to running with this course
- now lets run the same line above but attach --save to it so it updates the json file in your folder that way you know which version we used to create this app

```
joses-MacBook-Pro:score-keep mendoza$ meteor npm install react@15.3.2 react-dom@15.3.2 --save
```

- **if by any chance your nodes_module gets deleted all you have to do is run this line to regenerate this folder**

```
meteor npm install
```

- make sure everything you install shows up in package.json
- now let's restart the app by typing

```
meteor run --release 1.4.2.1
```

- go over to client main.js and delete everything from there
- and now over in the main.html create a div

```
<head>
  <title>Score Keep</title>
</head>


<body>
<div id='app'></div>


</body>
```

- now go to client main.js and import some modules and also want a Meteor module

```
import React from 'react';

import ReactDom from 'react-dom';

import {Meteor} from 'meteor/meteor';
```

- now we have to wait for the dom to render the way we do this is by using this code

```
import React from 'react';

import ReactDom from 'react-dom';

import {Meteor} from 'meteor/meteor';


Meteor.startup(function(){


});
```

- **once the dom is ready the code inside the the Meteor.startup will be executed**

```
Meteor.startup(function(){

  let jsx = <p>This is from the main.js</p>

});
```

- and now we are going to render it to the screen by sing the following code
- the render() takes two parameter the first in the element you want to render the second one is where you want that element to render to, in our case is 'app'

```
import React from 'react';

import ReactDOM from 'react-dom';

import {Meteor} from 'meteor/meteor';


Meteor.startup(function(){

  let jsx = <p>This is from the main.js</p>

  ReactDOM.render(jsx, document.getElementById('app'))

});
```

- we can also use javascript in our jsx like so... by using curly braces

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';

Meteor.startup(function(){
  let name = 'Jose';
  let jsx = <p>{name}</p>
  ReactDOM.render(jsx, document.getElementById('app'))
});
```

- to verify that are jsx code works go over to https://babeljs.io/repl/

```
 let name = 'Jose';
  let jsx = <p>{name}</p>
```

- and this is what it is converting to... just plain javascript

```
'use strict';

var name = 'Jose';
var jsx = React.createElement(
  'p',
  null,
  name
);
```

Render Complex HTML with JSX

- one thing to remember jsx will throw an error you are trying to render two or more elements like so...

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';

Meteor.startup(function(){
  let name = 'Jose';
  let jsx = <p>Hi my name is {name}!</p><p>This is my second p</p>;
  ReactDOM.render(jsx, document.getElementById('app'))
});
```

- this can work if wrap it like so

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';

Meteor.startup(function(){
  let name = 'Jose';
  let jsx = <div><p>Hi my name is {name}!</p><p>This is my second p</p></div>;
  ReactDOM.render(jsx, document.getElementById('app'))
});
```

- we can also wrapp everything in parenthesis and it won't cause an error like so...

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';

Meteor.startup(function(){
  let name = 'Jose';
  let jsx = (
  <div>
    <p>Hi my name is {name}!</p>
    <p>This is my second p</p>
  </div>
  );
  ReactDOM.render(jsx, document.getElementById('app'))
});
```

- and this will just make the code look better
- let's create a variable and injected in our div element

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';
```

```
Meteor.startup(function(){
  let title = 'Account Settings';
  let name = 'Jose';
  let jsx = (
  <div>
    <h1>{title}</h1>
    <p>Hi my name is {name}!</p>
    <p>This is my second p</p>
  </div>
  );
  ReactDOM.render(jsx, document.getElementById('app'))
});
```

DB Planning and Rendering Static Data

- let's create an array of objects in the client main.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import {Meteor} from 'meteor/meteor';

const players = [{
  _id: '1',
    name: 'Lauren',
    score: 99
},{
  _id: '2',
  name: 'Cory',
  score: -1
},{
  _id: '3',
  name: 'Jose',
  score: -12
}];
```

- we then want to show that we can render something dynamic in our render so let's try an example

```
Meteor.startup(function(){
  let title = 'Score Keep';
  let name = 'Jose';
  let jsx = (
  <div>
    <h1>{title}</h1>
    <p>Hi my name is {name}!</p>
    <p>This is my second p</p>
    {[<p>1</p>, <p>2</p>,<p>3</p>]}
  </div>
  );
```

- so this will work but in the console it'll throw an error about a key, so each element needs it's own unique key

```
Meteor.startup(function(){
  let title = 'Score Keep';
  let name = 'Jose';
  let jsx = (
  <div>
    <h1>{title}</h1>
    <p>Hi my name is {name}!</p>
    <p>This is my second p</p>
    {[<p key='1'>1</p>, <p key='2'>2</p>,<p key='3'>3</p>]}
  </div>
  );
```

- now let's figure out how we are going to render our players list
- let's first create a function that will take a n array and that i'll return jsx code
- lets cut and paste what we have down below

```
const renderPlayers = function(){
  return [<p key='1'>1</p>, <p key='2'>2</p>,<p key='3'>3</p>];
};

Meteor.startup(function(){
  let title = 'Score Keep';
  let name = 'Jose';
  let jsx = (
  <div>
    <h1>{title}</h1>
    <p>Hi my name is {name}!</p>
    <p>This is my second p</p>
    {renderPlayers()}
  </div>
```

- now lets' pass an argument to our renderPlayers

```
const renderPlayers = function(playerList){
  return [<p key='1'>1</p>, <p key='2'>2</p>,<p key='3'>3</p>];
};


Meteor.startup(function(){
  let title = 'Score Keep';
  let name = 'Jose';
  let jsx = (
  <div>
    <h1>{title}</h1>
    <p>Hi my name is {name}!</p>
    <p>This is my second p</p>
    {renderPlayers(players)}
  </div>
  );
```

- we next are going to be using the array map function

```
const players = [{
  _id: '1',
    name: 'Lauren',
    score: 99
},{
  _id: '2',
  name: 'Cory',
  score: -1
},{
  _id: '3',
    name: 'Jose',
    score: -12
  }];


  const renderPlayers = function(playerList){
    return playerList.map(function(player){
      return <p key={player._id}>{player.name} has {player.score} point(s)</p>
    });
  };
```

What is NoSQL?

- With SQL you have a predifined table with various columns and all of your records have a value for that file
- with MongoDB/NoSQL we have a collection of documents
- SQLs have tables and NoSQL have a collection of documents
- check the image 'MySQL-postgreSQL vs MongDB-NoSQL'

Creating a MongoDB and MIniMongo Collection

-
-
-
-
-
-
-
-
-
-
-