

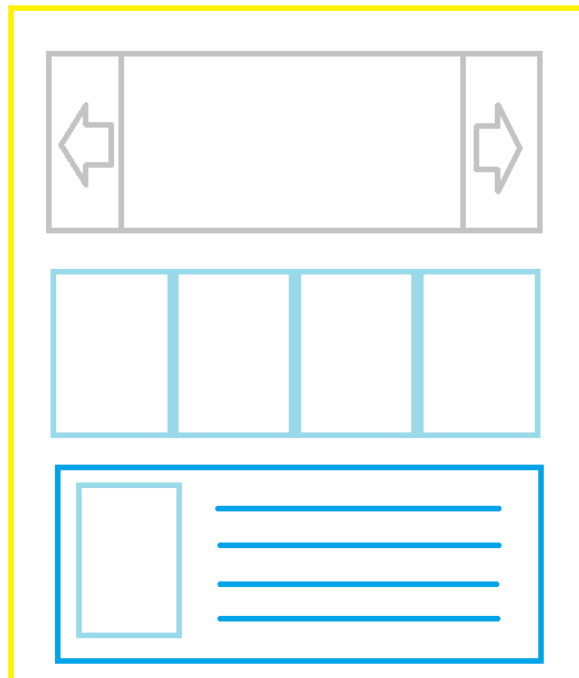
Task Requirements

Create page with three sections

The first section (the gray one) should display by default the result of request to '/episodePreview/0', upon clicking on the buttons, left or right, the component should reload with new image, displaying another episode.

The second section should display images, of all the character within the '/roster' response.

The third section should display detailed data, about a character, after the user has clicked on image from the second section.



1. Create react app

Using the React generator, create React app in a new folder.

If you don't have the generator installation, type the following command in your command prompt

```
npm
PS C:\Users\User\Desktop\reactTest> npm install -g create-react-app
C:\Users\User\AppData\Roaming\npm\create-react-app -> C:\Users\User\AppData\Roaming\npm\node_modules\create-react-app\index.js
+ create-react-app@3.4.1
added 10 packages from 6 contributors, removed 3 packages and updated 47 packages in 24.163s
```

After you successfully have installed the generator or if you already have it type the following line in the command prompt, where 'my-app' represents the name of the app that will be generated

```
npm
PS C:\Users\User\Desktop\reactTest> create-react-app my-app
Creating a new React app in C:\Users\User\Desktop\reactTest\my-app.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
[.....] \ extract:caniuse-lite: sill caniuse-lite@1.0.30001111 extracted to C:\Users\User\Desktop\reactTest\my-app\node_modul
```

1.1 Run the server

Place the provided server in appropriate folder and install all needed dependencies.

Run the server via terminal, it will display the port on which it is listening, if by any chances the port is already occupied, change the port value in the 'index.js' file.

1.2 Test the server

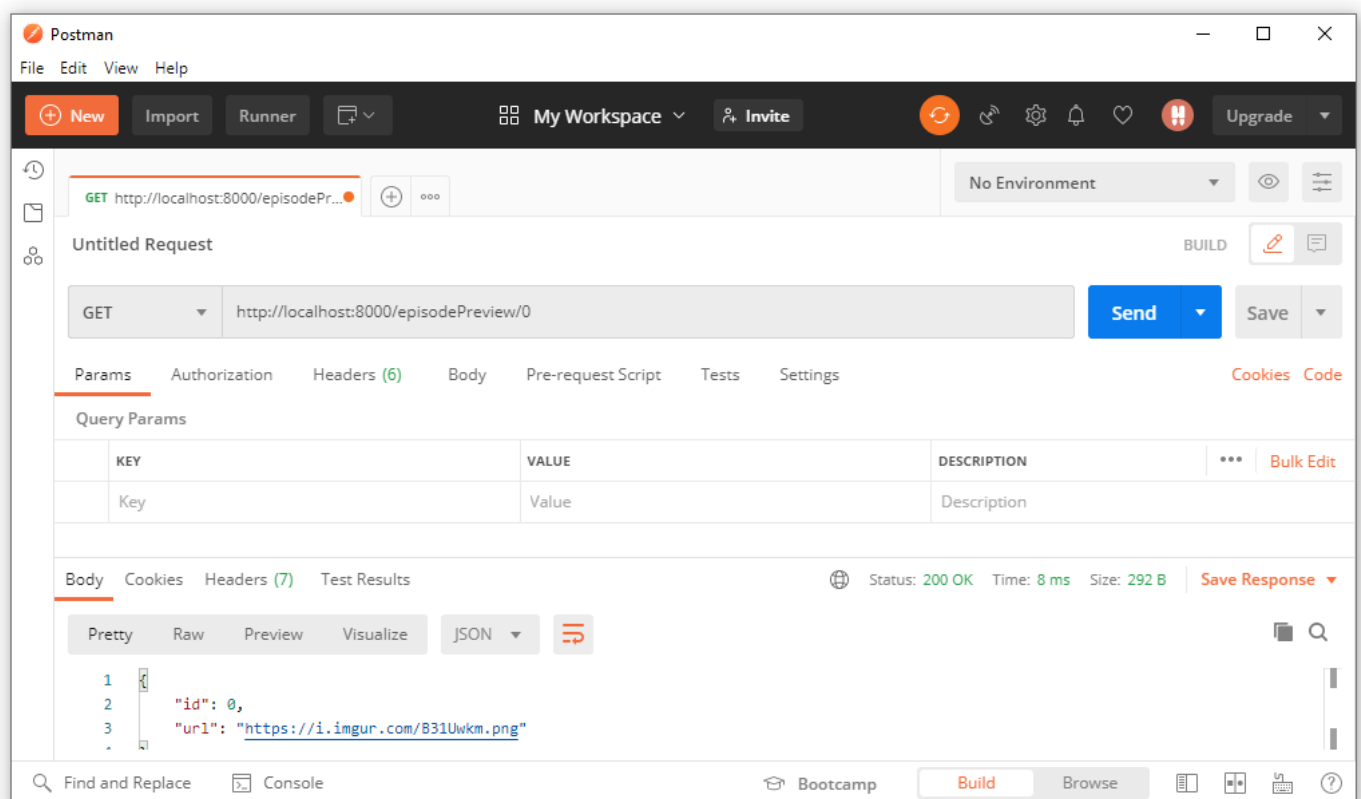
Using Postman make get requests to the following routes:

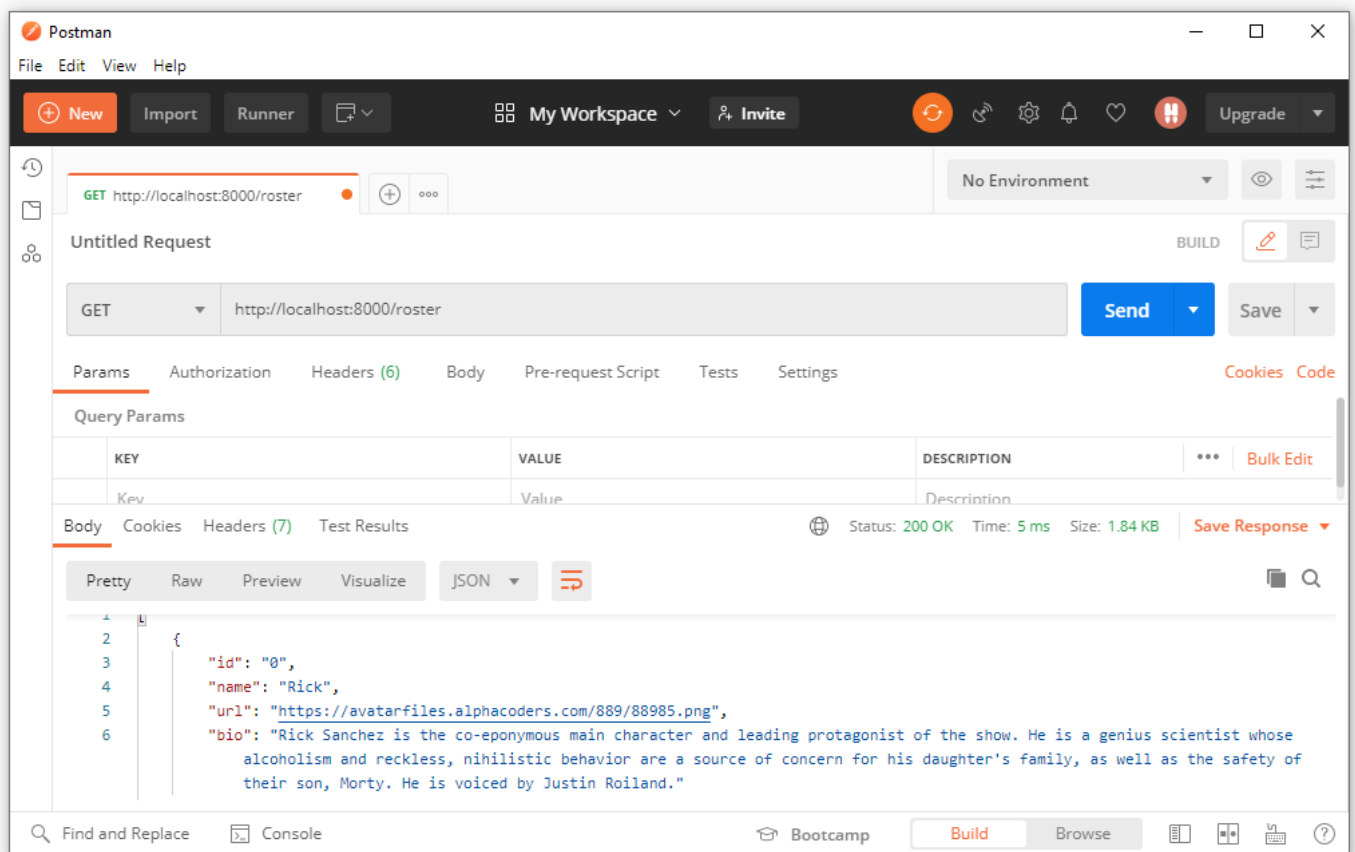
The first request will return a single episode, with id and img url:

`http://localhost:[yourPortHere]/episodePreview/[idOfEpisode]`

The second request will return list of all characters, that are in our database:

`http://localhost:[yourPortHere]/roster`

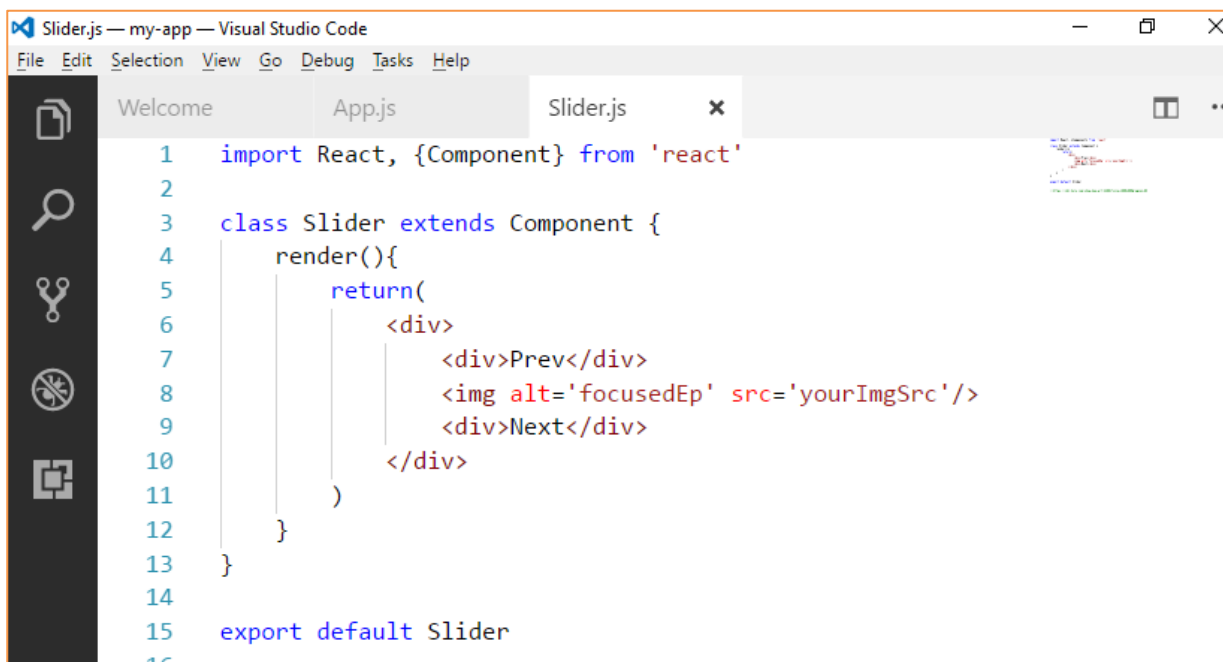




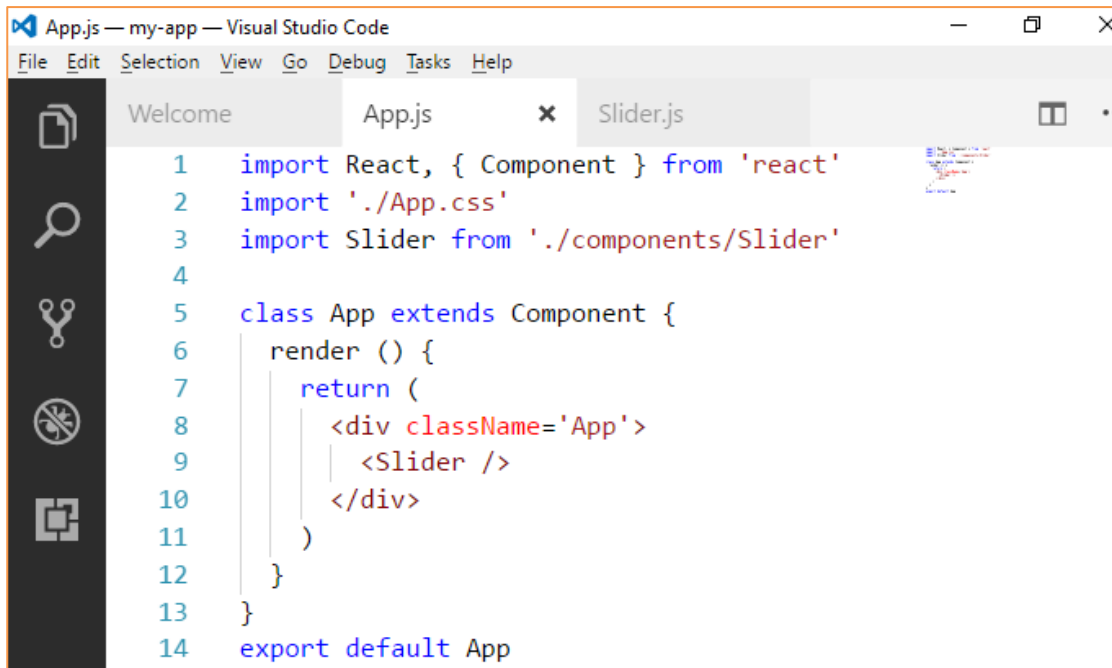
2. Slider Section

2.1 Create New Component

Create new component in a different file and export it, it might look like the example below.



Import the component in 'App.js' and try to run your app



```
App.js — my-app — Visual Studio Code
File Edit Selection View Go Debug Tasks Help

Welcome App.js Slider.js

1 import React, { Component } from 'react'
2 import './App.css'
3 import Slider from './components/Slider'
4
5 class App extends Component {
6   render () {
7     return (
8       <div className='App'>
9         <Slider />
10      </div>
11    )
12  }
13 }
14 export default App
```

2.2 Extend the Main Component

```
import React, { Component } from 'react'
import './App.css'
import Slider from './components/Slider'

class App extends Component {
  constructor(){
    super()

    this.state = {
      epOnFocus : 0
    }

    this.changeEp = (ep)=>{
      this.setState({epOnFocus:ep})
    }
  }
  render () {
    return (
      <div className='App'>
        <Slider updateFunc={this.changeEp} focusedEp={this.state.epOnFocus}/>
      </div>
    )
  }
}
export default App
```

2.3 Extend Slider Component

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

```

import left from '../resources/left.png'
import right from '../resources/right.png'

class Slider extends Component {
  constructor () {
    super()
    this.state = {
      seletedImg: ''
    }
    this.promisfyState = obj => {
      return new Promise(res => {
        this.setState(obj, res)
      }).catch(e => {
        console.log(e)
      })
    }
  }
  componentWillMount () {
    fetch('http://localhost:9999/episodePreview/' + this.props.focusedEp)
      .then(data => {
        return data.json()
      })
      .then(parseData => {
        this.promisfyState({ seletedImg: parseData.url }).then(() => {
          console.log('loaded new state')
        })
      })
  }
  componentDidMount () {
    fetch('http://localhost:9999/episodePreview/' + this.props.focusedEp)
      .then(data => {
        return data.json()
      })
      .then(parseData => {
        this.promisfyState({ seletedImg: parseData.url }).then(() => {
          console.log('mount')
        })
      })
  }
  render () {
    return (
      <div>
        <div className='warper'>
          <img
            alt='nope'
            src={left}
            className='slider-elem slider-button case-left'
            onClick={() =>
              this.props.updateFunc(
                Number(this.props.focusedEp) - 1 < 0

```

```

        ? 0
        : Number(this.props.focusedEp) - 1
    )}
  />
  <img
    className='sliderImg slider-elem'
    alt='focusedEp'
    src={this.state.seletedImg}
  />
  <img
    alt='nope'
    src={right}
    className='slider-elem slider-button case-right'
    onClick={() =>
      this.props.updateFunc(
        Number(this.props.focusedEp) + 1 > 2
          ? 2
          : Number(this.props.focusedEp) + 1
        )
    }
  />
</div>
</div>
)
}
}
export default Slider

```

2.4 Add some CSS

```

.warper{
  display: inline-block;
  position: relative;
}

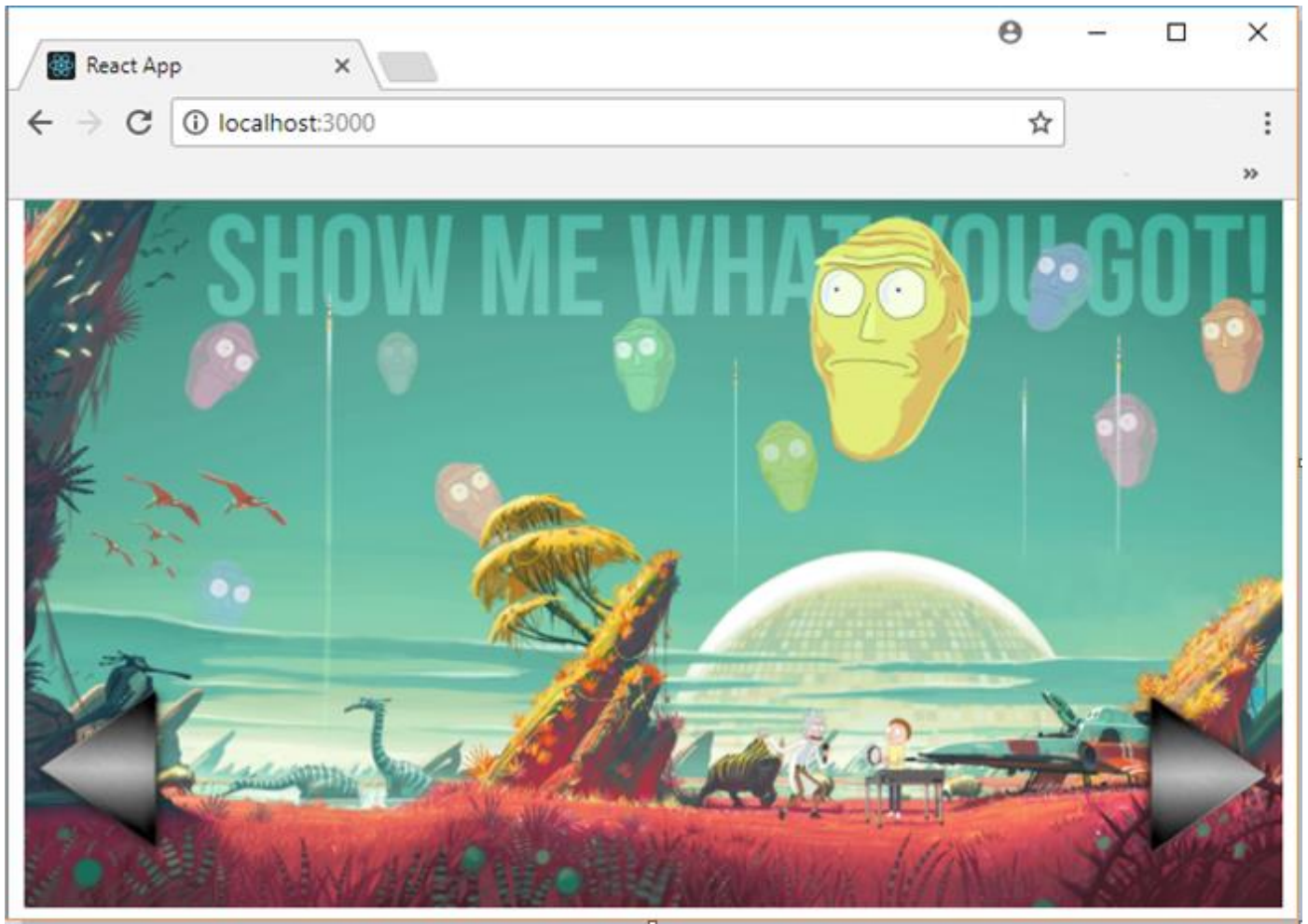
.warper .sliderImg{
  max-height: 400px;
}

.warper .slider-button{
  position: absolute;
  top:50%;
  transform:translateY(-50%);
  width: 150px;
  height: 150px;
}

.warper .case-left {
  left: 0;
}

```

```
.warper .case-right {  
  right: 0;  
}
```



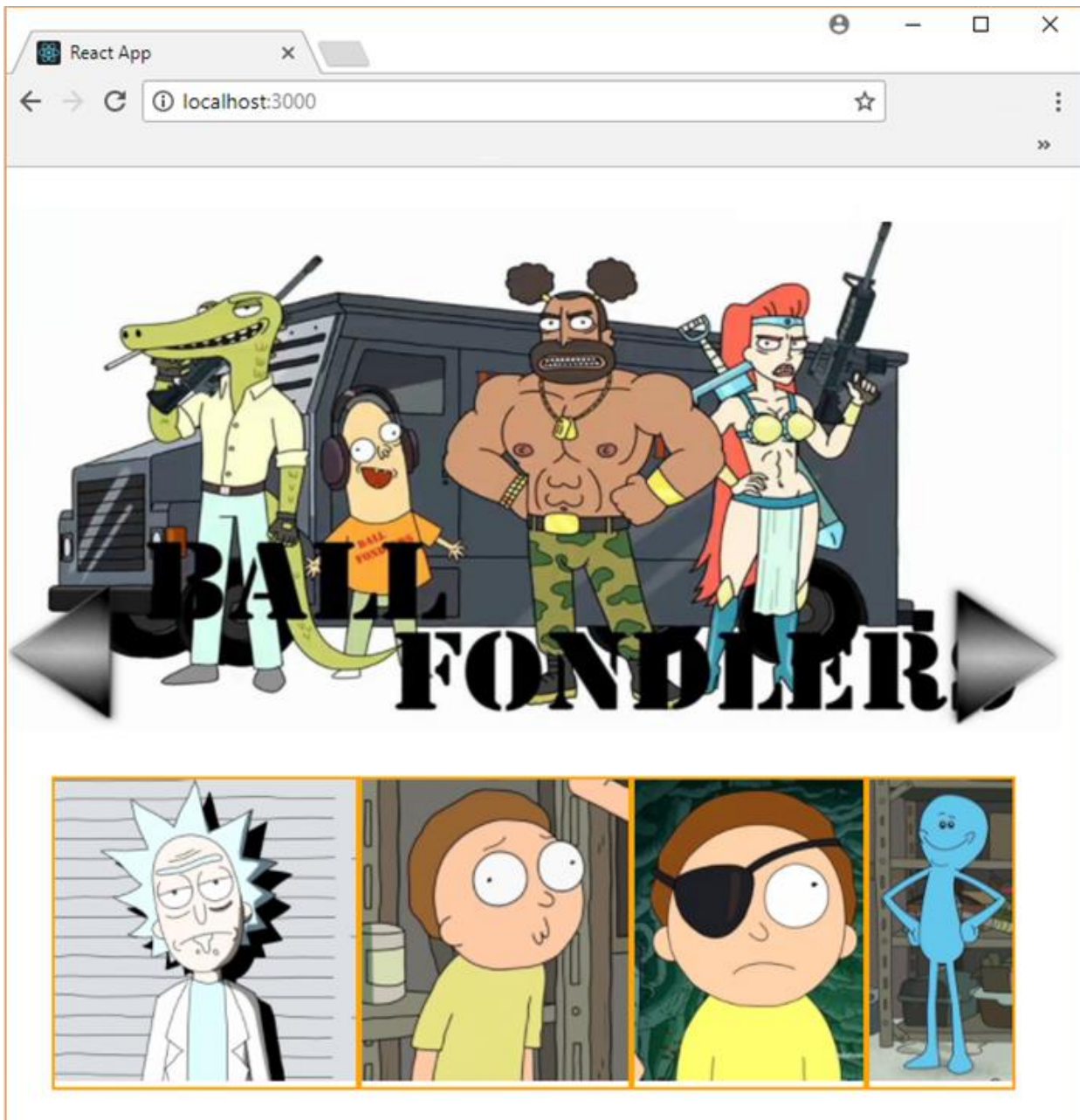
3. Roster Section

3.1 Create Image Component

Create functional component, that will receive as props link to a specific image and visualize it.

3.2 Fill the Roster

Create component, that will fetch, all characters from the server and for each character single character, will create 'Image Component'



4. Focused Character Section

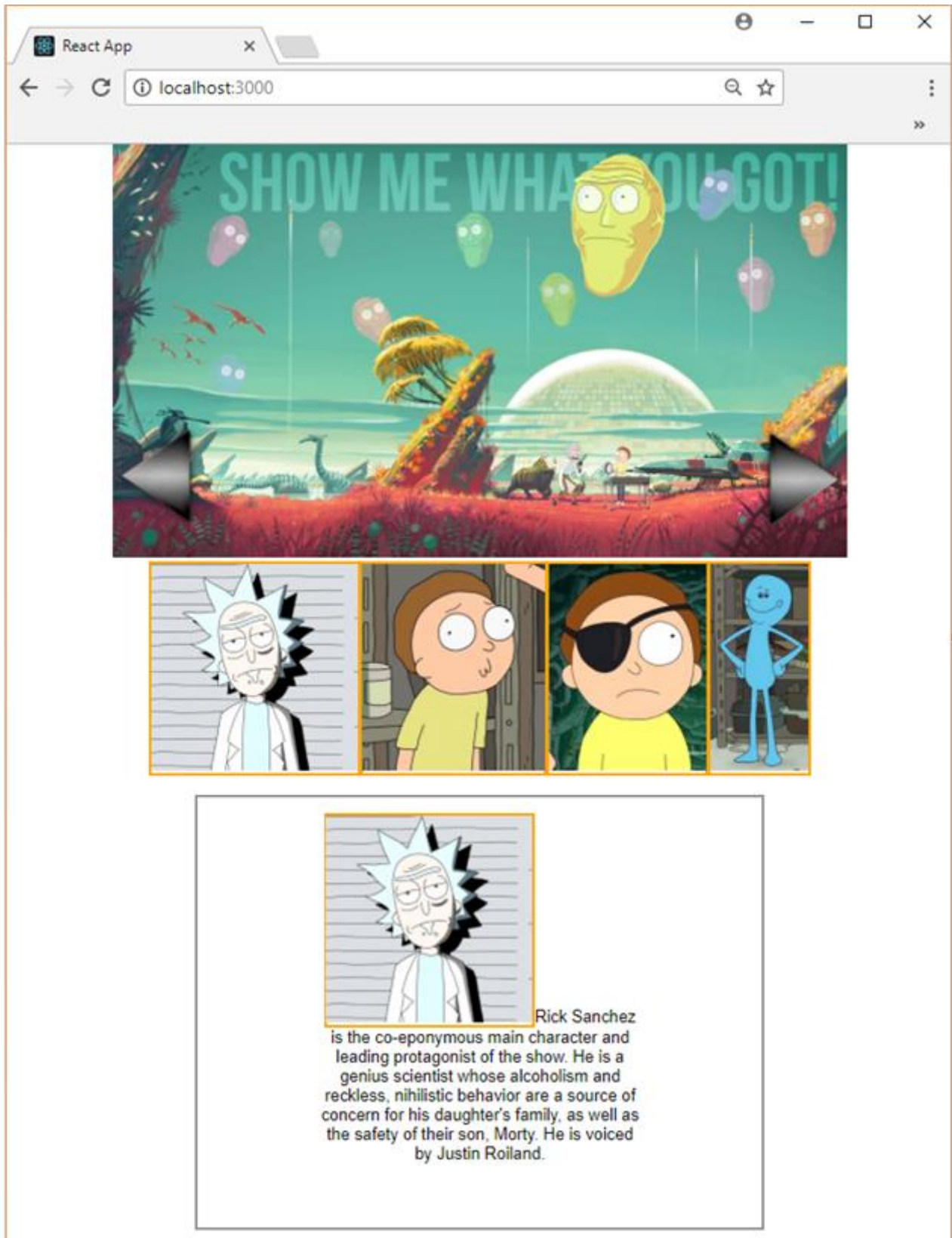
4.1 Create details component

Create functional component, that will receive as props a character data and will visualize details about the that character

Hint: Use the image component that we already created

4.2 Extend details

Create logic, that will visualize in the 'Focused Section' the character that was last clicked form the roster field



5. Optimize React Structure *

Upon completing all tasks, review your code and try to extract all repeating code, in appropriate component, if possible, so that it can be reused.