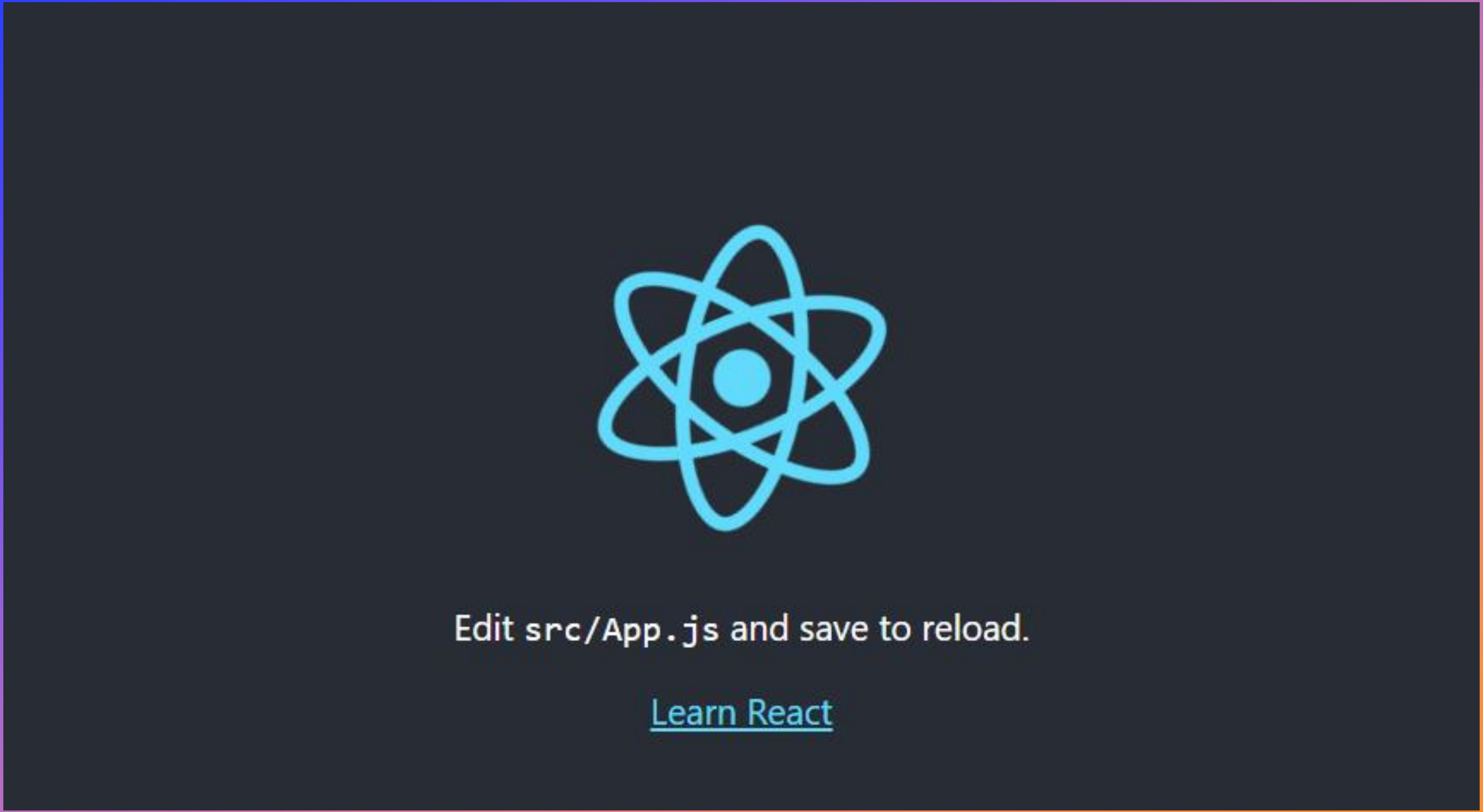
CREATE REACT APP WITH SASS

Open new powerdell or cmd

In Terminal write next comands:

- npm i create-react-app –g
- create-react-app client
- cd client
- npm start


Edit `src/App.js` and save to reload.

Learn React

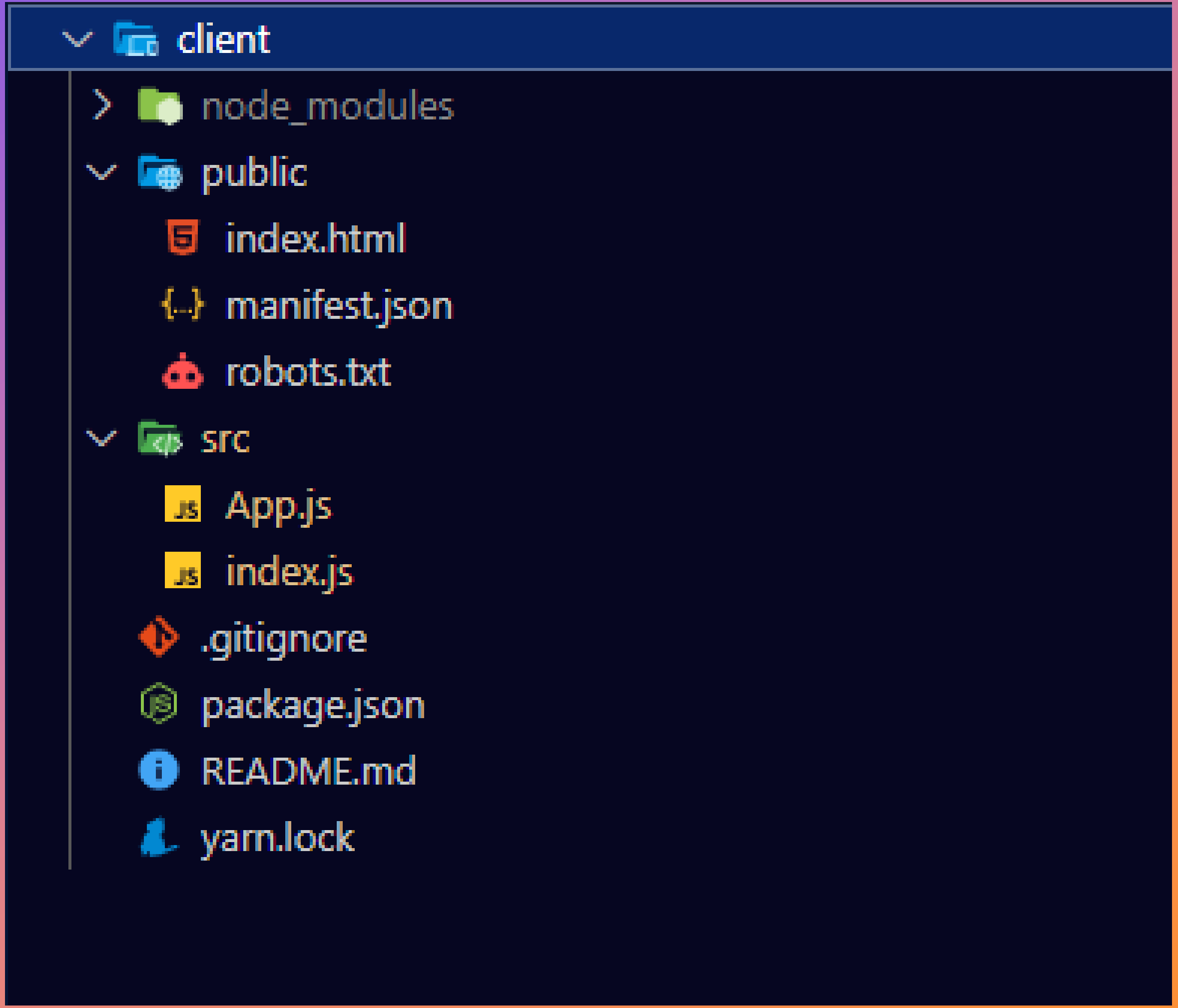
client/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';


ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

client/src/App.js

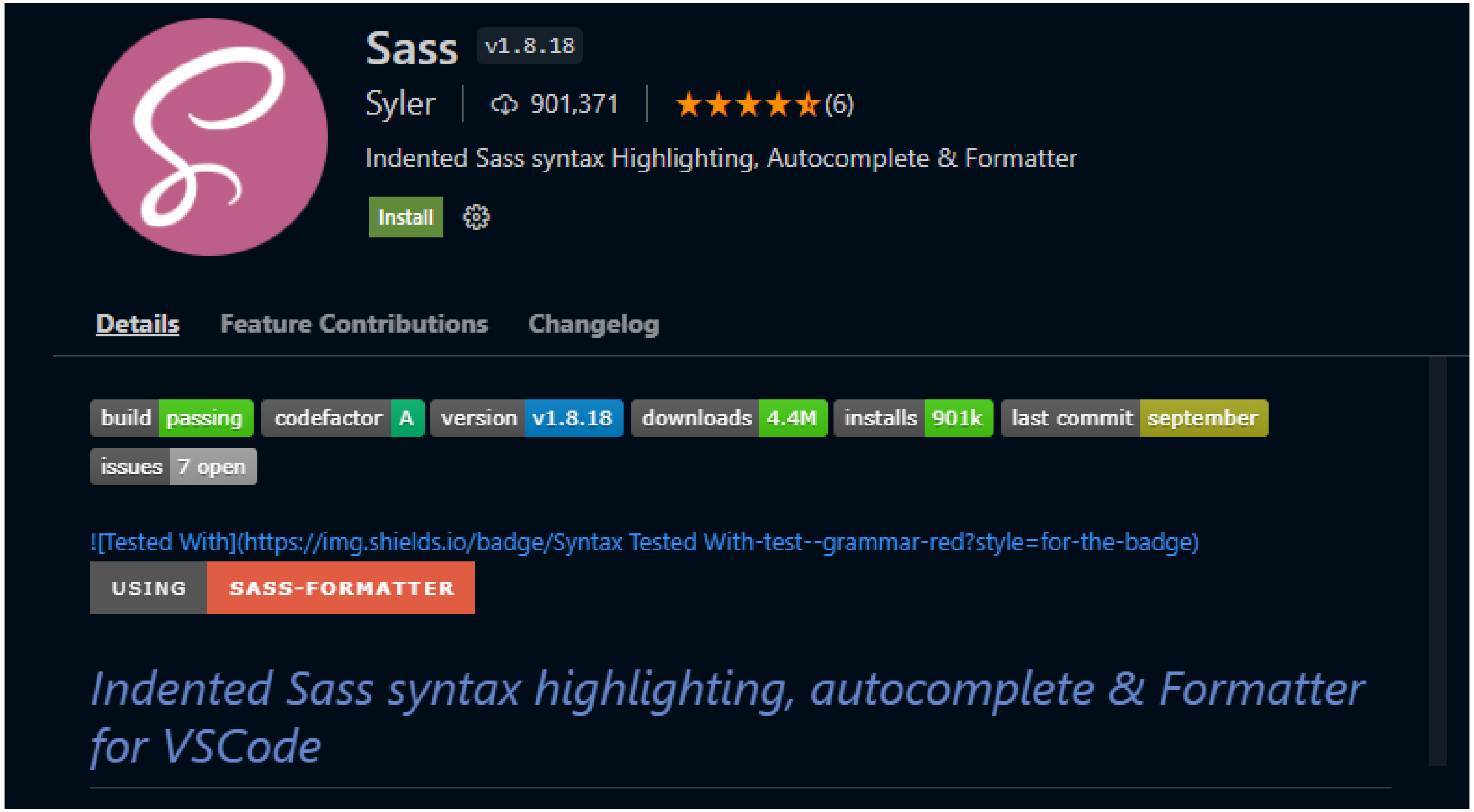
```
function App() {
  return (
    <div className="App">

    </div>
  );
}

export default App;
```

PROJECT STRUCTURE



```
client
  node_modules
  public
    index.html
    manifest.json
    robots.txt
  src
    App.js
    index.js
  .gitignore
  package.json
  README.md
  yarn.lock
```

Install Extentions


Sass v1.8.18

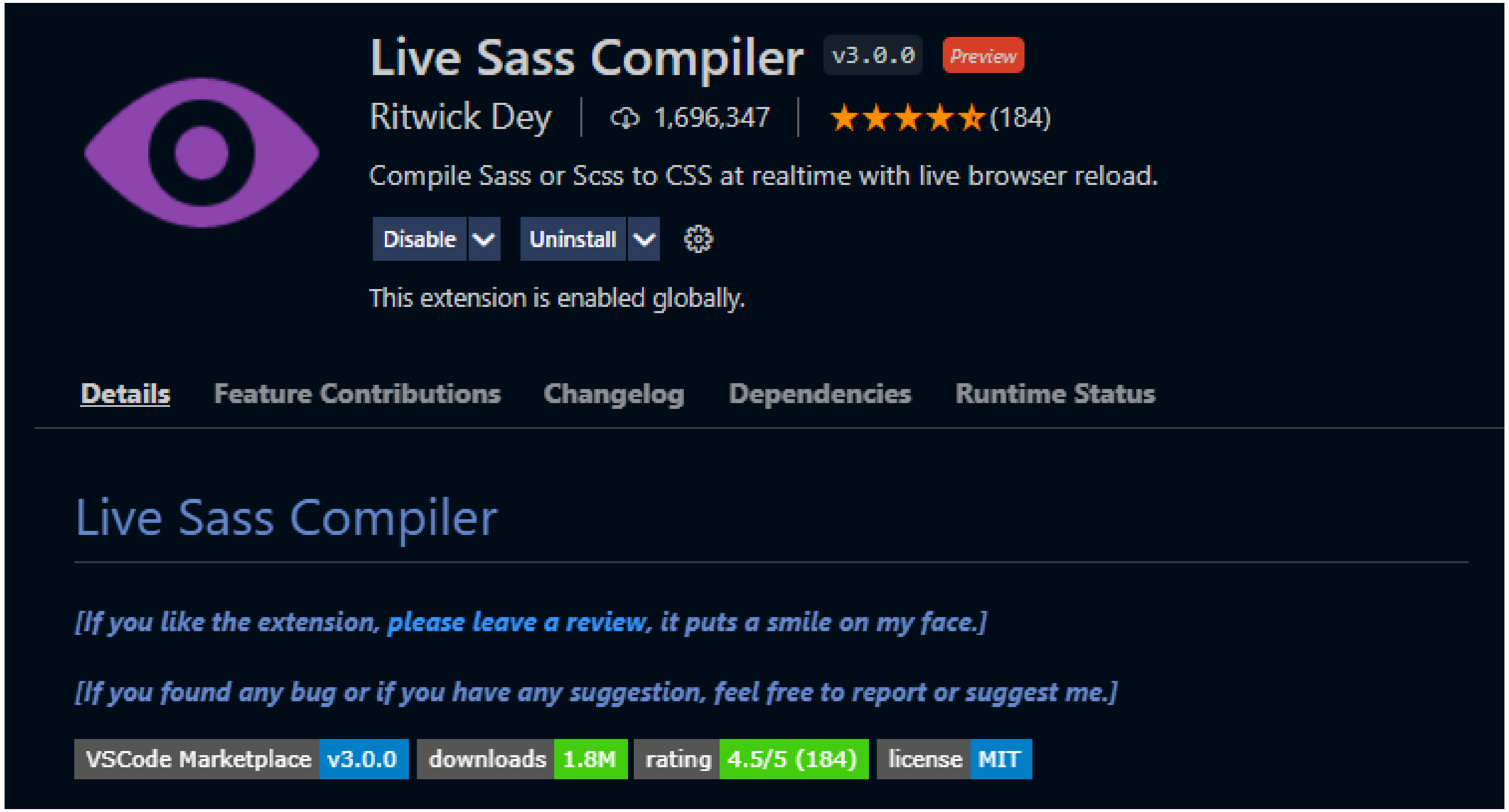
Syler | 901,371 | ★★★★★ (6)

Indented Sass syntax Highlighting, Autocomplete & Formatter

![Tested With](https://img.shields.io/badge/Syntax Tested With-test--grammar-red?style=for-the-badge)

Indented Sass syntax highlighting, autocomplete & Formatter for VSCode

Install Extentions



Live Sass Compiler v3.0.0 Preview

Ritwick Dey | 1,696,347 | ★★★★☆ (184)

Compile Sass or Scss to CSS at realtime with live browser reload.

This extension is enabled globally.

Live Sass Compiler

[If you like the extension, please leave a review, it puts a smile on my face.]

[If you found any bug or if you have any suggestion, feel free to report or suggest me.]

VSCode Marketplace v3.0.0 | downloads 1.8M | rating 4.5/5 (184) | license MIT

In to the folder **src** create folder **Components**
In to the folder **Components** create file **Navbar.js**

```
import React from "react";

function Navbar() {
    return (
      <div className="navbar">
        <ul>
            <li><a href="/home">Home</a></li>
            <li><a href="/about">About Us</a></li>
            <li><a href="/lessons">Lessons</a></li>
            <li><a href="/portfolio">Portfolio</a></li>
            <li><a href="/blog">Blog</a></li>
            <li><a href="/contact">Contact</a></li>
        </ul>
      </div>
    );
  }

  export default Navbar;
```

In to the folder **Components** create file **MainPage.js**

```
import React from "react";

function MainPage() {
    return (
        <div className="mainPage">

            <div className="lead-banner">
                <img src="img/banner-1.png" />
                <div className="banner-content">
                    <div className="wrapper">
                        <span className="title">Check our video</span>
                        <span className="sub-title">Learning in 6 weeks</span>
                    </div>
                </div>
            </div>

            <section className = "services">
                <div className="wrapper">
                    <h1>Services</h1>
                    <div >
                        <div className = "items">
                            <img src="img/icon-1.png" />
                            <a href="#">Recording<br />services</a>
                        </div>
                        <div className = "items">
                            <img src="img/icon-2.png" />
                            <a href="#">Get a printed<br />certificate</a>
                        </div>
                        <div className = "items">
                            <img src="img/icon-3.png" />
                            <a href="#">Lessons via<br />Skype</a>
                        </div>
                    </div>
                </div>
            </section>
```

```
           ...

            <div className="lessons-banner">
                <img src="img/banner-2.png" />
                <div className="banner-content">
                    <div className="wrapper">
                        <span className="title">Our Lessons</span>
                        <span className="sub-title">Are Easy</span>
                        <ul>
                            <li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
                            <li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
                            <li>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</li>
                        </ul>
                    </div>
                </div>
            </div>

            <section className="projects">
                <div className="wrapper">
                    <h1>Our Projects</h1>
                    <ul>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                        <li><img src="http://placehold.it/150x150" /></li>
                    </ul>
                </div>
            </section>
        </div>
    );
}

  export default MainPage;
```

Modify the file App.js

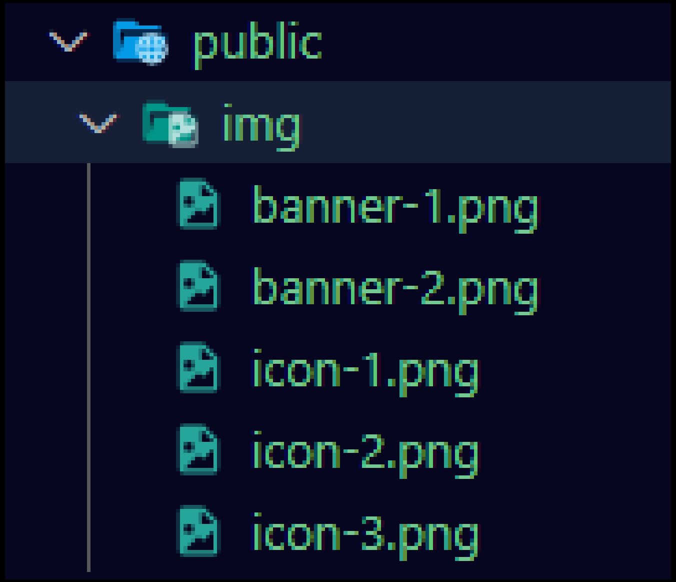
```
import Navbar from "./Components/Navbar.js"
import MainPage from "./Components/MainPage.js";


function App() {
  return (
    <div className="App">

      <Navbar />
      <MainPage />

    </div>
  );
}

export default App;
```

Add code

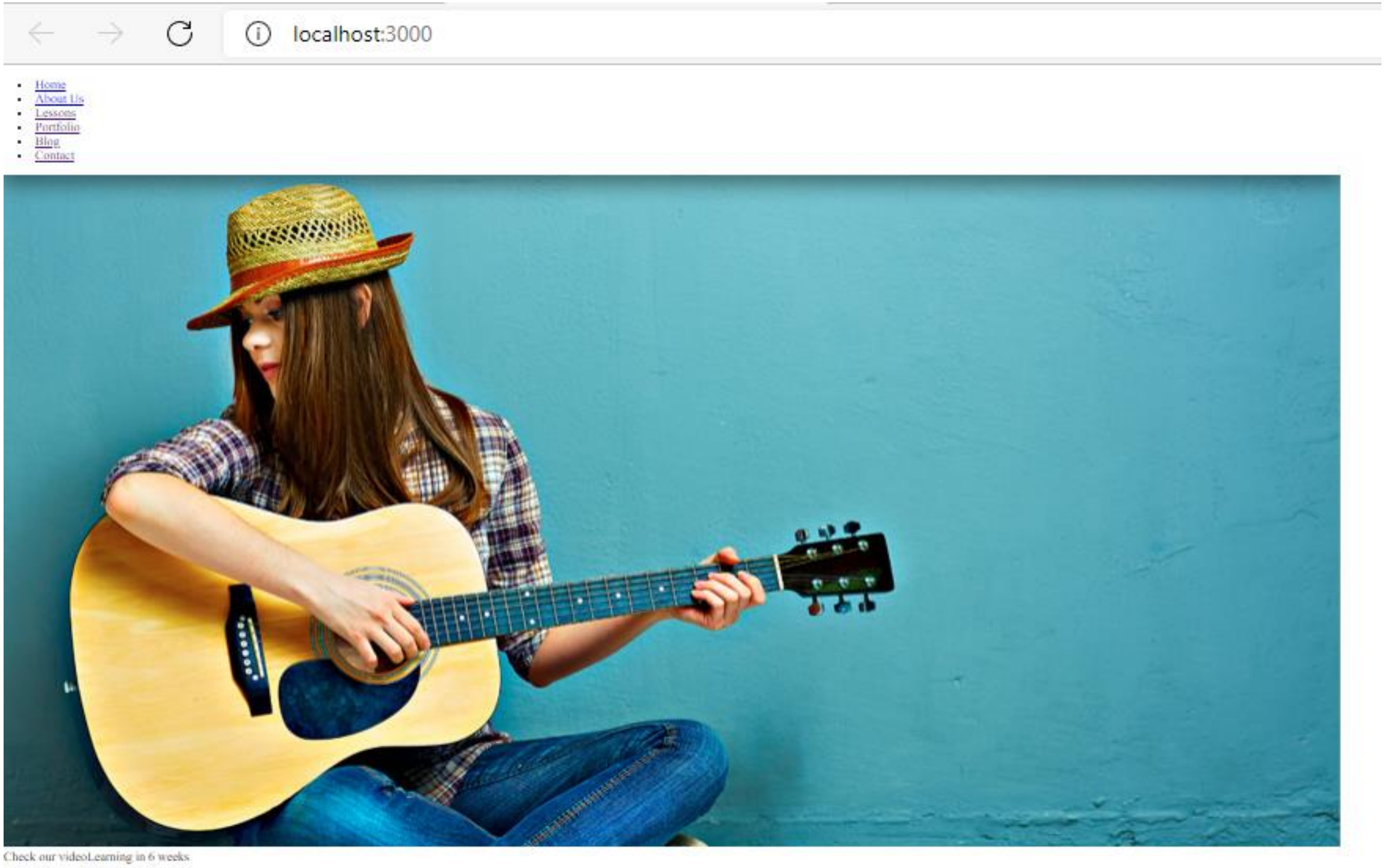
In to the directory **Components** create file **MainPage.js**

In to the directory **public** create directory **img**
Upload in to the directory img pictures from

https://github.com/MarizzaM/SASS_React/tree/01-Create-React-App/public/img



```
public
  img
    banner-1.png
    banner-2.png
    icon-1.png
    icon-2.png
    icon-3.png
```



localhost:3000

- Home
- About Us
- Lessons
- Portfolio
- Blog
- Contact

Check our videoLearning in 6 weeks

#02
SASS VARIABLES

in this SASS tutorial, we'll take a look at how we can create SASS variables to store information in. That information could be anything from hex codes for your colors, or font sizes, to font families.

In to the folder **src** create folder **Styles**
In to the folder **Styles** create file **Navbar.scss**

```scss
// Varibles
$deepBlue: #032f3e;
$sectionHeading: 24px;

.navbar {
    background: $deepBlue;
}
```

In to the directory **Styles** create file **MainPage.scss**

```scss
// Varibles
$deepBlue: #032f3e;
$sectionHeading: 24px;

.section h1 {
    color: $deepBlue;
    font-size: $sectionHeading;
}
```

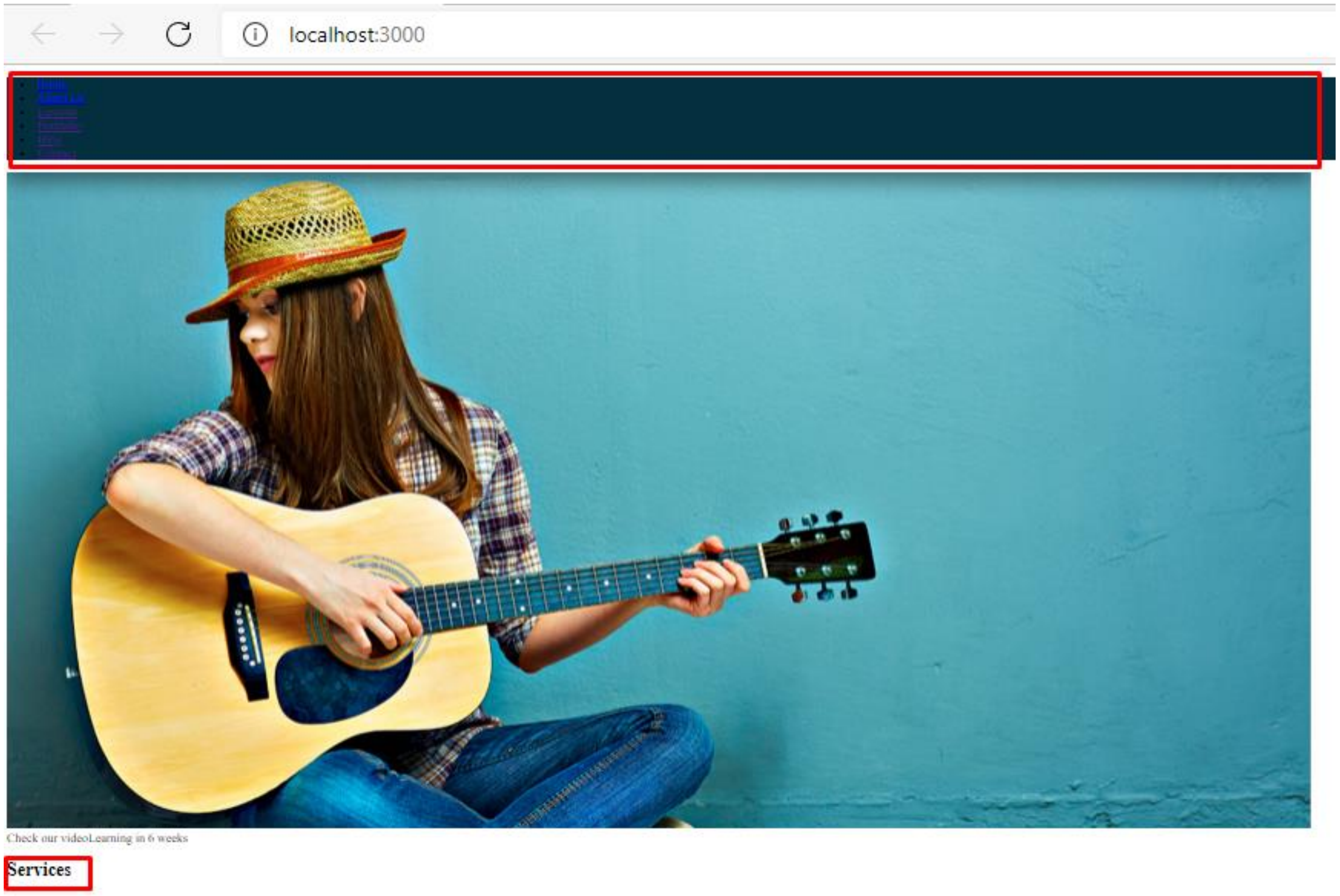
Modify file **Navbar.js**

```
import React from "react";

import '../Styles/Navbar.css';

function MainPage() { … }
```

Add code

Modify file **MainPage.js**

```
import React from "react";

import '../Styles/MainPage.css';

function Navbar() { … }
```



localhost:3000

- Home
- About Us
- [illegible]
- [illegible]
- [illegible]
- [illegible]

Check our videoLearning in 6 weeks

Services

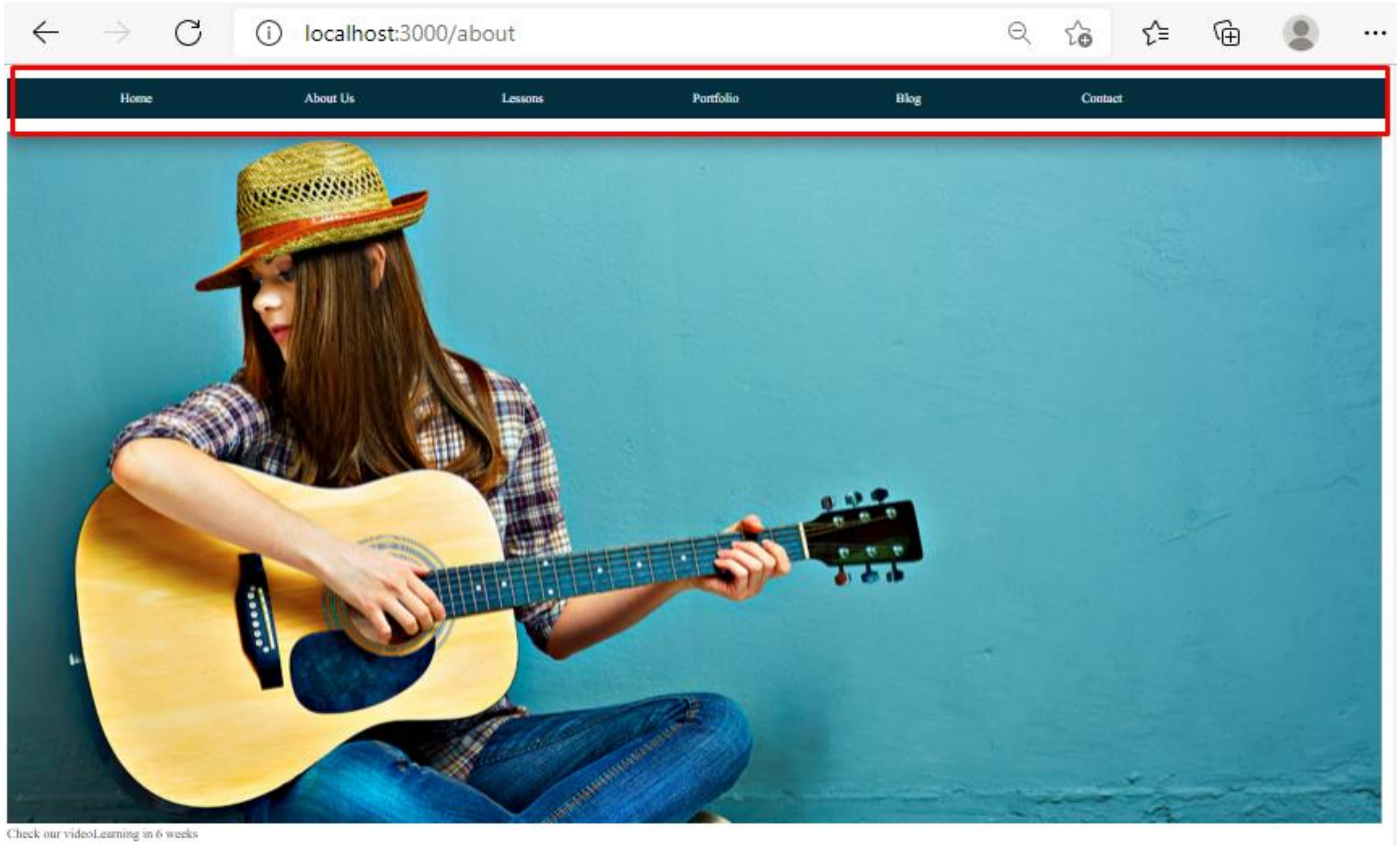
#03 NESTING STYLES

in this SASS tutorial I'll be showing you how we can nest styles within one another to help organise your CSS rules in a logical way. It also saves a little bit of time when writing out your code!

Modify file **Navbar.scss**

```scss
// Varibles

$offWhite: #f8f9fb;

$deepBlue: #032f3e;
$sectionHeading: 24px;

.navbar {
    background: $deepBlue;
    ul{
        width: 100%;
    }
    li{
        list-style-type : none;
        float: left;
        width: 14%;
    }
    a {
        color: $offWhite;
        text-decoration: none;
        padding: 16px 0;
        display: block;
        text-align: center;
    }
}

.navbar ul:after{
    content: "";
    display: block;
    clear: both;
}
```

Add code



Check our videoLearning in 6 weeks

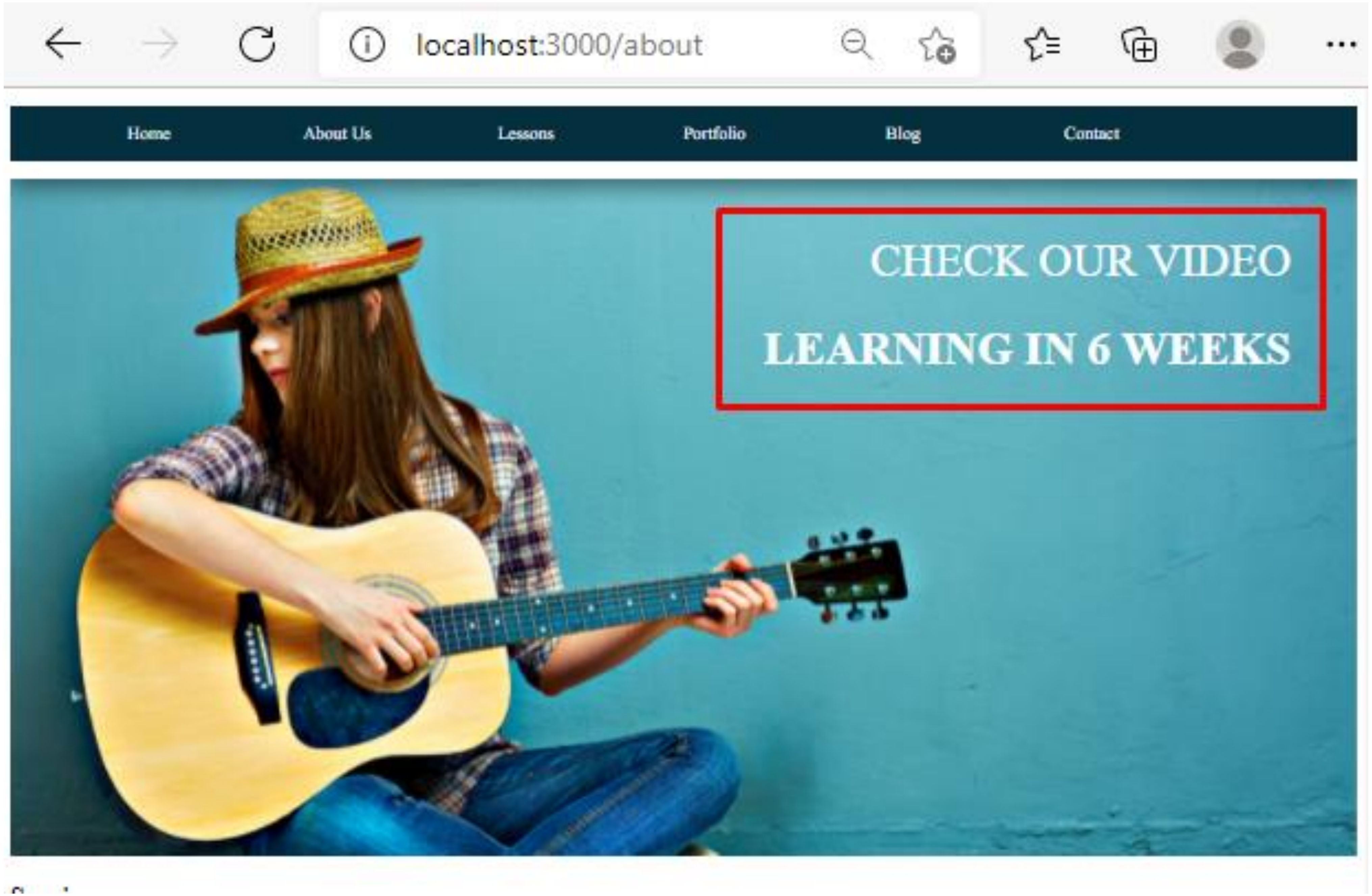
#04 MIXINS

in this SASS tutorial, I'll be showing you how we can use mixins to create re-usable chunks of CSS, whose behaviour can change based on variables we can pass through to them.

Modify file **MainPage.scss**

```scss
// Varibles
$deepBlue: #032f3e;
$sectionHeading: 24px;

$offWhite: #f8f9fb;
$bannerHeading: 42px;

.section h1 {
    color: $deepBlue;
    font-size: $sectionHeading;
}

@mixin banner{
    width: 100%;
    position: relative;
    color: white;
    .banner-content{
      position: absolute;
      top: 50px;
      width: 100%;
    }
    img{
      width: 100%;
    }
    span{
      font-size: $bannerHeading;
      display: block;
      text-transform: uppercase;
      font-weight: bold;
      padding: 0 60px;
    }
    span.title{
      font-weight: normal;
      margin-bottom: 30px;
    }
}

.lead-banner{
    @include banner;
    text-align: right;
}

.lessons-banner{
    @include banner;
    li{
        list-style-type : none;
        text-transform: uppercase;
        font-size: 20px;
        max-width: 1000px;
        margin: 60px 0;
    }
}
```

Add code



CHECK OUR VIDEO

LEARNING IN 6 WEEKS

#05 IMPORTING FILES

in this SASS tutorial, I'll be showing you how we can split our SCSS file into modules (or other .scss files) and then import them into our main file. This way, we can keep all of our SASS for certain things together.

In to the folder **src** create folder **Variables**
In to the folder **Variables** create file **variables.scss**

```scss
$offWhite: #f8f9fb;
$deepBlue: #032f3e;

$sectionHeading: 24px;
$bannerHeading: 42px;
```

In to the folder **src** create folder **Mixins**
In to the folder **Mixins**create file m**ixins.scss**

```scss
@import "../Variables/variables.scss";

@mixin banner{
    width: 100%;
    position: relative;
    color: white;
    .banner-content{
      position: absolute;
      top: 50px;
      width: 100%;
    }
    img{
      width: 100%;
    }
    span{
      font-size: $bannerHeading;
      display: block;
      text-transform: uppercase;
      font-weight: bold;
      padding: 0 60px;
    }
    span.title{
      font-weight: normal;
      margin-bottom: 30px;
    }
}
```

Modify Navbar.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.navbar {
    background: $deepBlue;
    ul{
        width: 100%;
    }
    li{
        list-style-type : none;
        float: left;
        width: 14%;
    }
    a {
        color: $offWhite;
        text-decoration: none;
        padding: 16px 0;
        display: block;
        text-align: center;
    }
}

.navbar ul:after{
    content: "";
    display: block;
    clear: both;
}
```

Modify MainPage.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.section h1 {
    color: $deepBlue;
    font-size: $sectionHeading;
}

.lead-banner{
    @include banner;
    text-align: right;
}

.lessons-banner{
    @include banner;
    li{
        list-style-type : none;
        text-transform: uppercase;
        font-size: 20px;
        max-width: 1000px;
        margin: 60px 0;
    }
}
```

#06 PSEUDO CLASSES

in this SASS tutorial I'll show you how we work with pseudo classes in SASS.

Modify mixins.scss

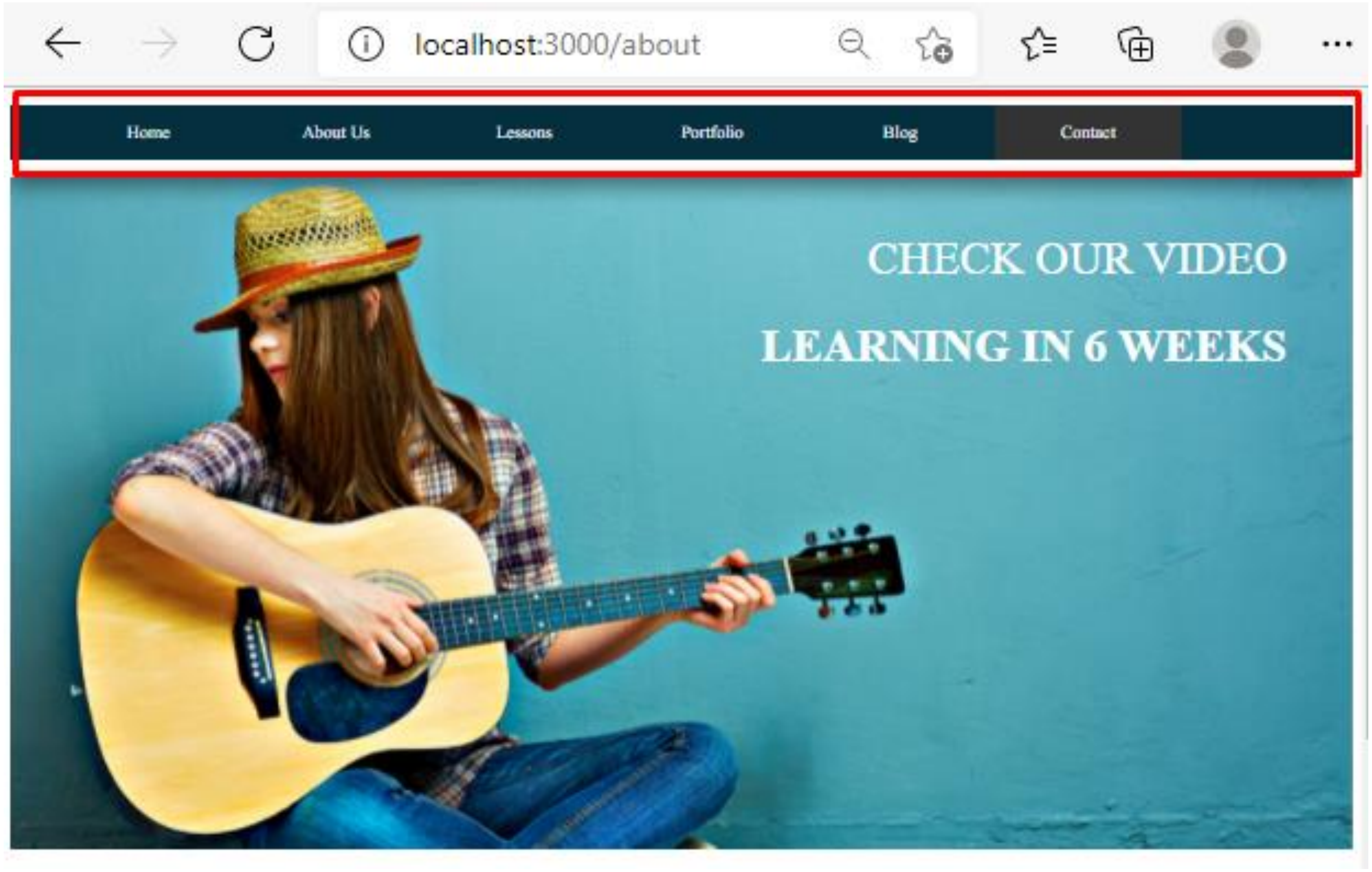
```scss
@mixin banner{
    width: 100%;
    position: relative;
    color: white;
    .banner-content{
      position: absolute;
      top: 50px;
      width: 100%;
    }
    img{
      width: 100%;
    }
    span{
      font-size: $bannerHeading;
      display: block;
      text-transform: uppercase;
      font-weight: bold;
      padding: 0 60px;
    }
    span.title{
      font-weight: normal;
      margin-bottom: 30px;
    }
}

@mixin clearFix{
    &:after{
      content: "";
      display: block;
      clear: both;
    }
  }
```

Add code

Modify Navbar.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.navbar {
    background: $deepBlue;
    ul{
        width: 100%;

        @include clearFix;
    }
    li{
        list-style-type : none;
        float: left;
        width: (100% / 6);
    }
    a {
        color: $offWhite;
        text-decoration: none;
        padding: 16px 0;
        display: block;
        text-align: center;

        &:hover{
            background: #333;
        }
    }
}

.navbar ul:after{
    content: "";
    display: block;
    clear: both;
}
```

Add code


localhost:3000/about
Home
About Us
Lessons
Portfolio
Blog
Contact
CHECK OUR VIDEO
LEARNING IN 6 WEEKS

#07 MATHEMATICAL OPERATORS

in this SASS tutorial I'll introduce you to mathematical operators, and how we can use them in SASS to perform calculations for us.

Modify Navbar.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.navbar {
    background: $deepBlue;
    ul{
        width: 100%;

        @include clearFix;
    }
    li{
        list-style-type : none;
        ~~float: left;~~
        width: (100% / 6);
    }
    a {
        color: $offWhite;
        text-decoration: none;
        padding: 16px 0;
        display: block;
        text-align: center;

        &:hover{
            background: #333;
        }
    }
}

.navbar ul:after{
    content: "";
    display: block;
    clear: both;
}
```

Change code

Modify MainPage.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.section h1 {...}

.lead-banner{...}

.lessons-banner{...}

.services{
    .items{
      float: left;
      width: (100% / 3);
      box-sizing: border-box;
      text-align: center;
    }
    img{
      width: 60%;
      margin: 20px 20%;
    }
    a{
      text-decoration: none;
      color: $deepBlue;
      font-weight: bold;
      &:hover{
        color: red;
      }
    }
    @include clearFix;
    margin-bottom: 60px;
  }
```

Add code

Services

Recording services

Get a printed certificate

Lessons via Skype

#08 CREATING A GRID WITH SASS MATH

in this SASS tutorial, I'll show you how we can create a flexible grid system using a little math inside SASS. It's super easy and very cool, and you'll learn about how we can pass variables to mixins along the way.

Modify MainPage.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.section h1 {...}

.lead-banner{...}

.lessons-banner{...}

.services{...  }

.projects
  li{
    list-style-type : none;
    float: left;
    width: 23%;
    margin-right: 2%;
    img{
      width: 100%;
    }
  }
```

Add code

Our Projects

Modify mixins.scss

```scss
@mixin banner{...}

@mixin clearFix{...}

@mixin grid($cols, $mgn){
  float: left;
  width: ((100% - (($cols - 1) * $mgn)) / $cols );
  margin-right: $mgn;
  margin-bottom: $mgn;
  &:nth-child(#{$cols}n){
    margin-right: 0;
  }
}
```

Add code

Modify MainPage.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.section h1 {...}

.lead-banner{...}

.lessons-banner{...}

.services{...  }

.projects
  li{
    list-style-type : none;
    @include grid(6, 2%);
    img{
      width: 100%;
    }
  }
```

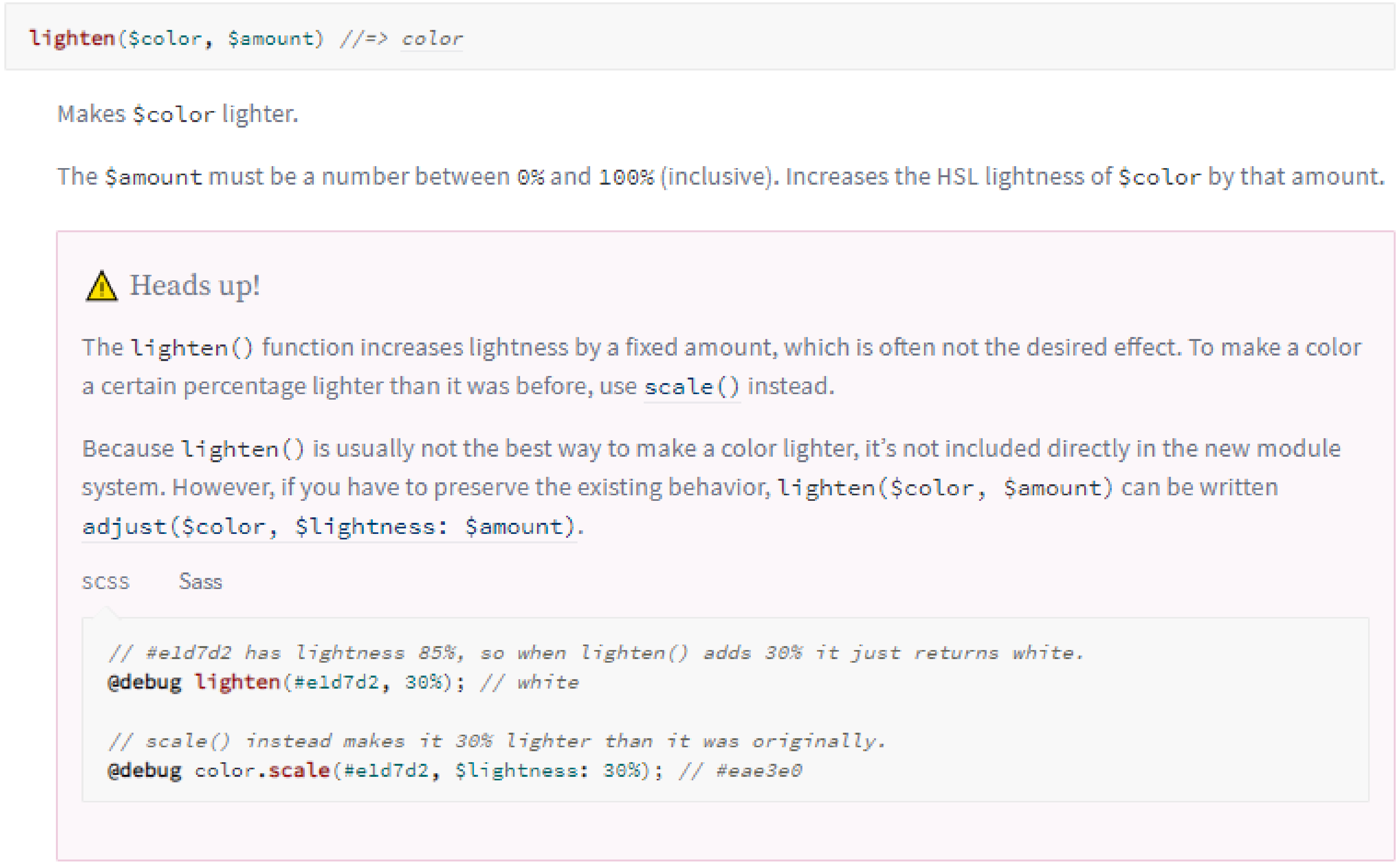
Change code

Our Projects

#09 COLOR FUNCTIONS

in this SASS tutorial, I'll be introducing you to some of the functions that come shipped with SASS. In particular, we'll be looking at a couple of cool color functions.

Go to https://sass-lang.com/documentation/modules/color



```
lighten($color, $amount) //=> color
```

Makes `$color` lighter.

The `$amount` must be a number between `0%` and `100%` (inclusive). Increases the HSL lightness of `$color` by that amount.

⚠ Heads up!

The `lighten()` function increases lightness by a fixed amount, which is often not the desired effect. To make a color a certain percentage lighter than it was before, use `scale()` instead.

Because `lighten()` is usually not the best way to make a color lighter, it's not included directly in the new module system. However, if you have to preserve the existing behavior, `lighten($color, $amount)` can be written `adjust($color, $lightness: $amount)`.

SCSS Sass

```
// #e1d7d2 has lightness 85%, so when lighten() adds 30% it just returns white.
@debug lighten(#e1d7d2, 30%); // white

// scale() instead makes it 30% lighter than it was originally.
@debug color.scale(#e1d7d2, $lightness: 30%); // #eae3e0
```

Modify Navbar.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.navbar {
    background: $deepBlue;
    ul{
        width: 100%;

        @include clearFix;
    }
    li{
        list-style-type : none;
        float: left;
        width: (100% / 6);
    }
    a {
        color: $offWhite;
        text-decoration: none;
        padding: 16px 0;
        display: block;
        text-align: center;

        &:hover{
            background: lighten($color: $deepBlue, $amount: 5);
        }
    }
}

.navbar ul:after{
    content: "";
    display: block;
    clear: both;
}
```

Change code

Modify MainPage.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.section h1 {...}

.lead-banner{...}

.lessons-banner{...}

.services{
    .items{
      float: left;
      width: (100% / 3);
      box-sizing: border-box;
      text-align: center;
    }
    img{
      width: 60%;
      margin: 20px 20%;
    }
    a{
      text-decoration: none;
      color: $deepBlue;
      font-weight: bold;

      &:hover{
        color: complement($deepBlue);
      }

    }
    @include clearFix;
    margin-bottom: 60px;
  }

.projects
  li{...  }
```

Change code

#10
THE @CONTENT KEYWORD

in this SASS tutorial, I'll show you how we can use the @content keyword to sub in our custom styles into mixins. The content keyword is pretty useful when creating mixins for media queries, so we'll take a look at making one.

Modify mixins.scss

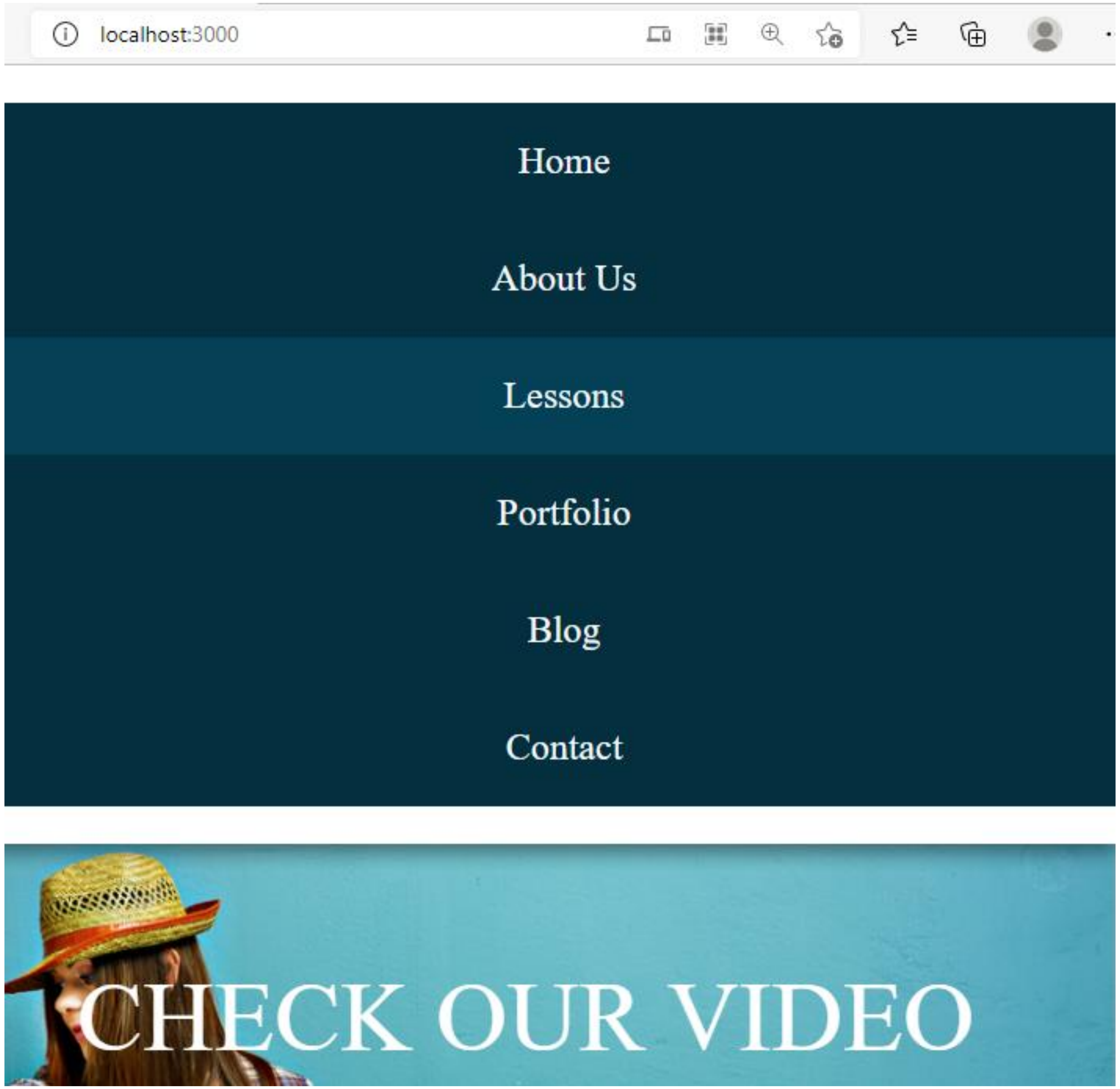
```
@mixin banner{...}

@mixin clearFix{...}

@mixin grid($cols, $mgn){...}

@mixin mQ($arg){
      @media screen and (max-width: $arg){
          @content;
  }
}
```

Add code

Modify Navbar.scss

```scss
@import "../Variables/variables.scss";
@import "../Mixins/mixins.scss";

.navbar {
    background: $deepBlue;
    ul{
        width: 100%;
        @include clearFix;
    }
    li{
        list-style-type : none;
        float: left;
        width: (100% / 6);

        @include mQ(600px){
            width: 100%;
        }
    }
    a {
        color: $offWhite;
        text-decoration: none;
        padding: 16px 0;
        display: block;
        text-align: center;

        &:hover{
            background: lighten($color: $deepBlue, $amount: 5);
        }
    }
}
```

Add code


localhost:3000

Home

About Us

Lessons

Portfolio

Blog

Contact

CHECK OUR VIDEO

#11
IF STATEMENTS

in this final SASS tutorial, I'll show you how we can use @if statements to control which parts of the CSS to use for particular elements.

Modify mixins.scss

```scss
@mixin banner{
...

  span{
    font-size: $bannerHeading;
    display: block;
    text-transform: uppercase;
    font-weight: bold;
    padding: 0 60px;

    @include mQ(3000px, 1200px){
      font-size: 68px;
    }
  }
  span.title{
    font-weight: normal;
    margin-bottom: 30px;
  }
}

@mixin clearFix{...}

@mixin grid($cols, $mgn){...}

@mixin mQ($arg...){
  @if length($arg) == 1{
      @media screen and (max-width: nth($arg, 1)){
      @content;
      }
  }
  @if length($arg) == 2{
      @media screen and (max-width: nth($arg, 1)) and (min-width: nth($arg 2)){
      @content;
      }
  }
}
```

Add code

Change code