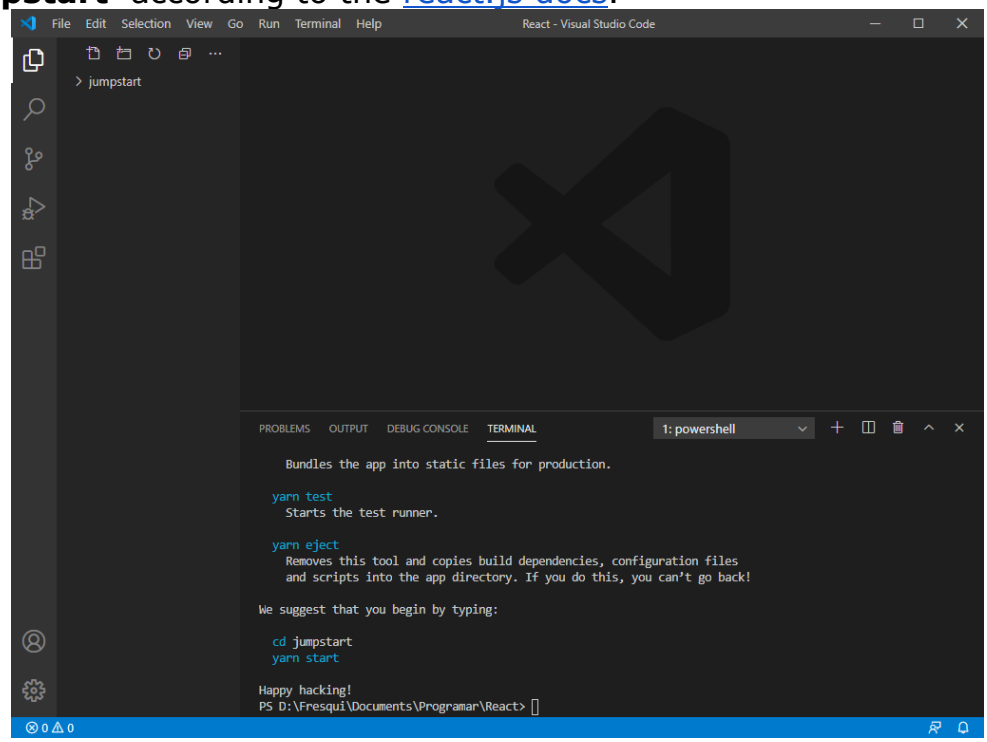
Hello Jumpstart, I am Lucas Manuel Faner, an applicant for the Front-End Engineer (REMOTE) job.
I tried to keep the guide as comprehensible as possible, while adding shortcuts for a better experience.

If I can provide you with any further information, please let me know.

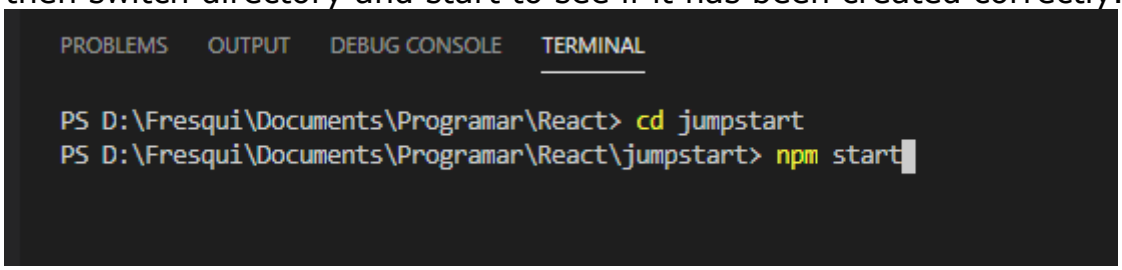
Table of Contents:

1. Introduction

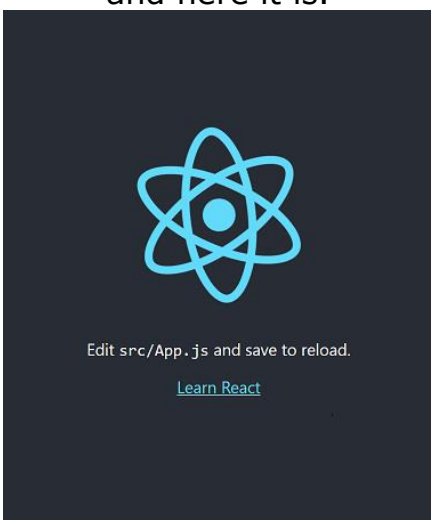
I began creating a React project with Visual Studio Code's integrated terminal, using the command **npx create-react-app jumpstart** according to the react.js docs.



then switch directory and start to see if it has been created correctly.

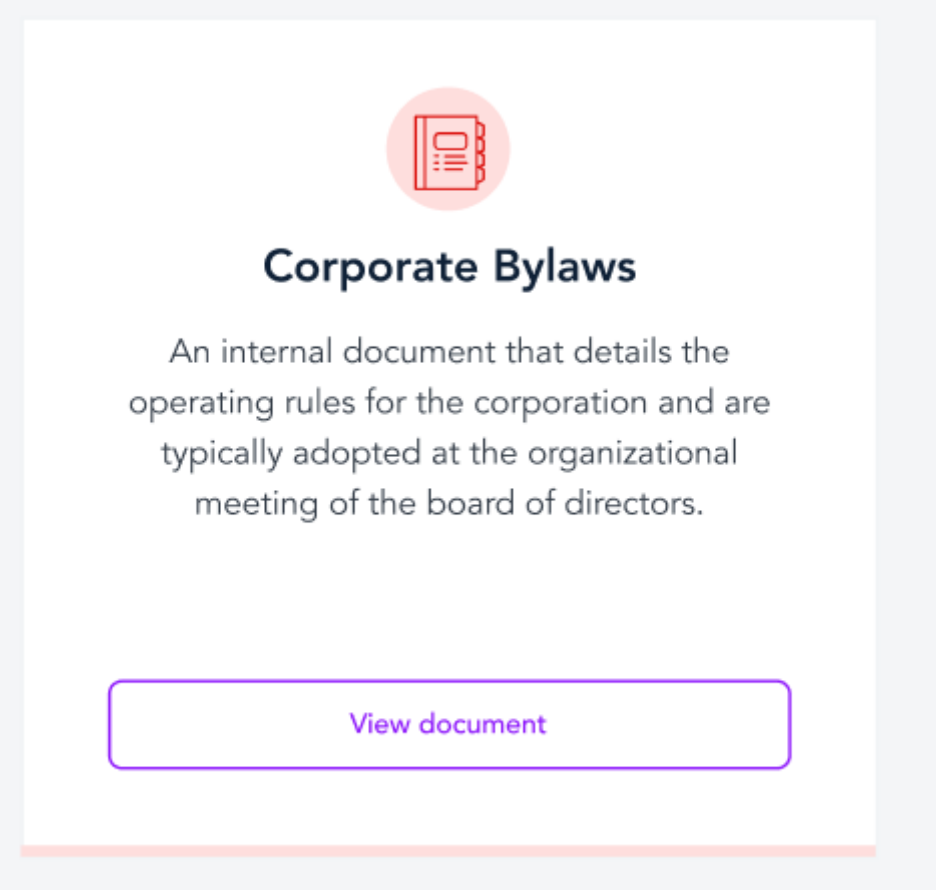


and here it is.

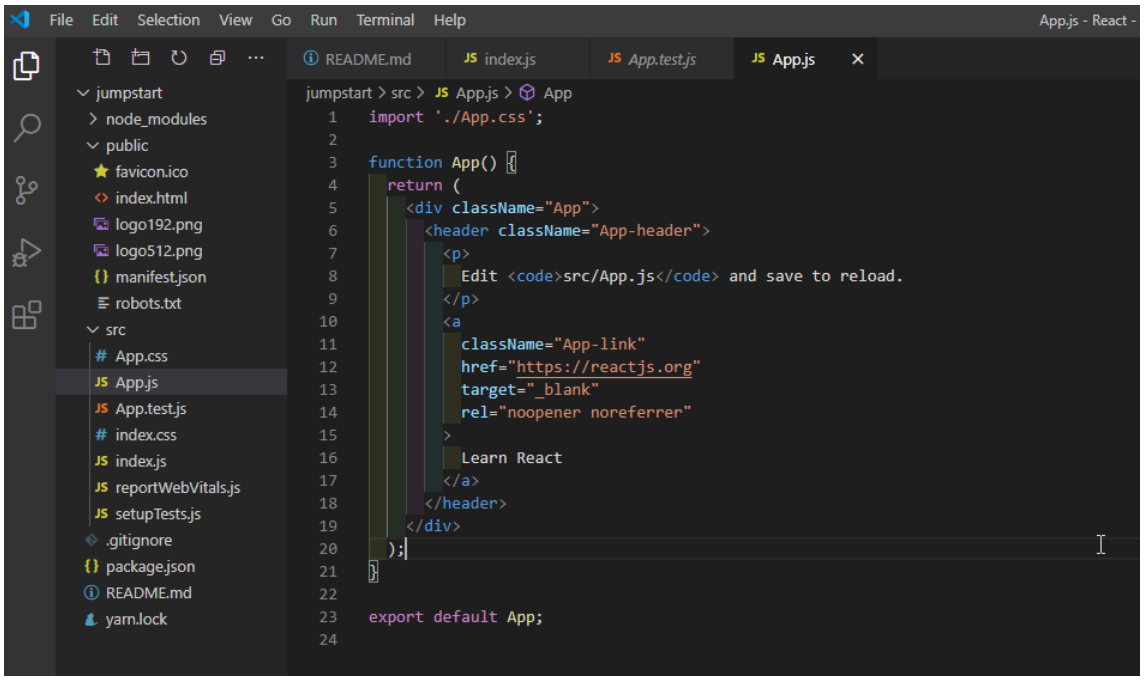


2. Creating the Main Rectangle

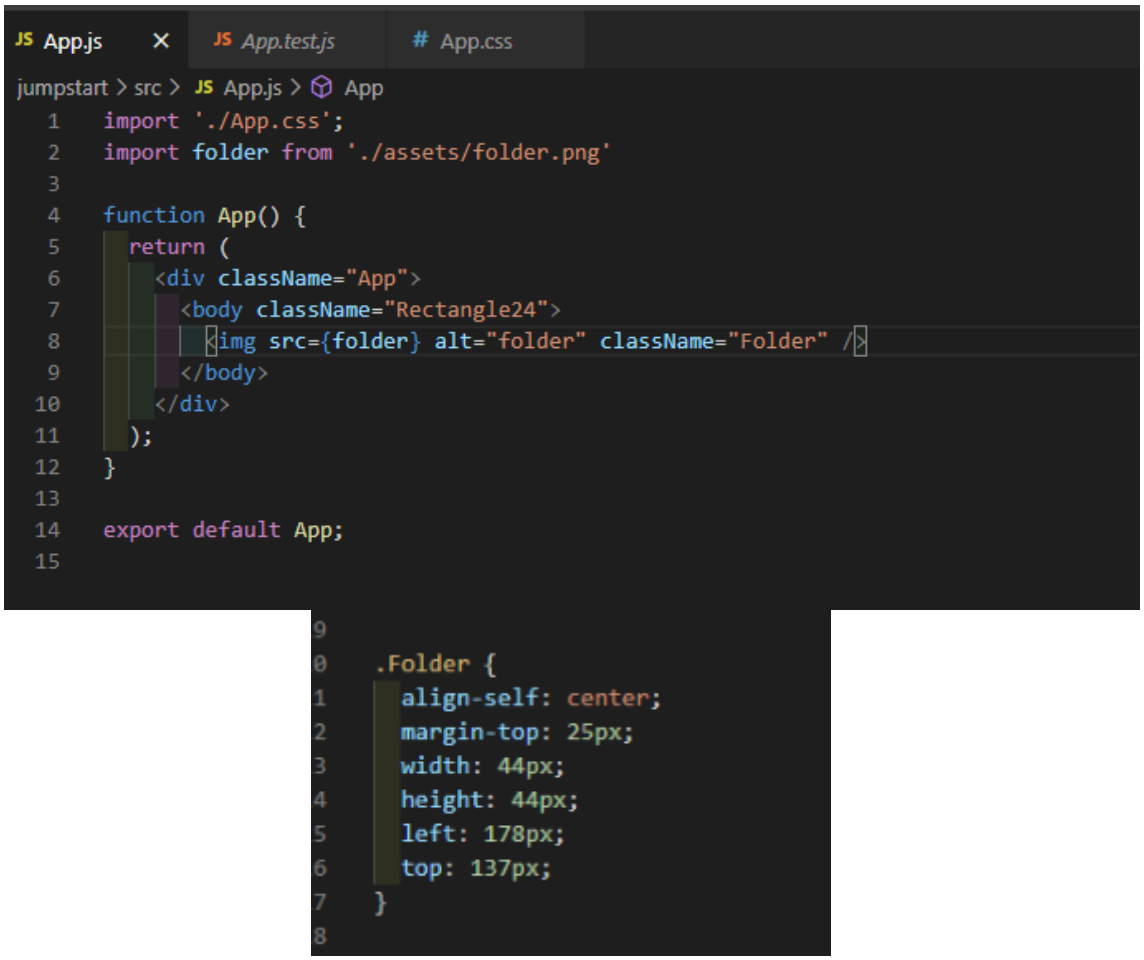
Following the design provided,

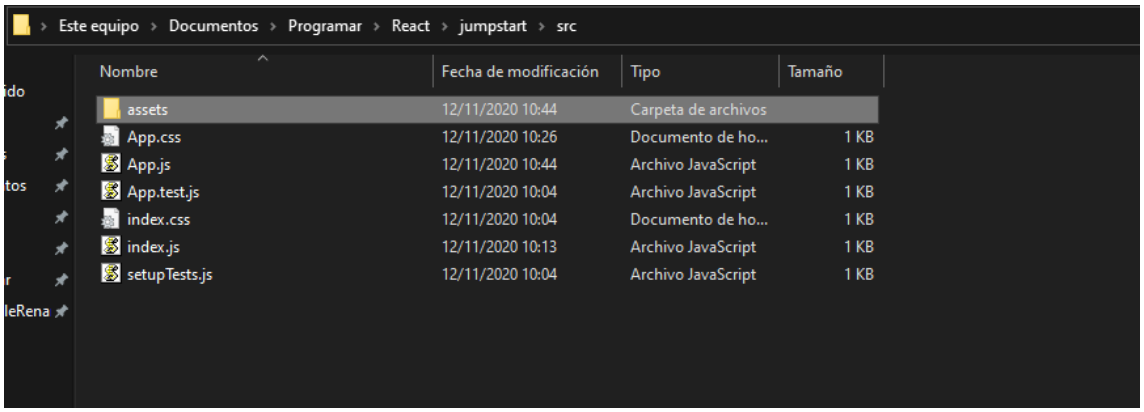


I feel like I could start from creating the main rectangle to contain the **elements** (icon, title text, body text and button). To do so, I had to erase unused code, such as the default template for a fresh started react app, while creating a new "**Rectangle24"** CSS class for the mentioned rectangle. (It will be renamed eventually)

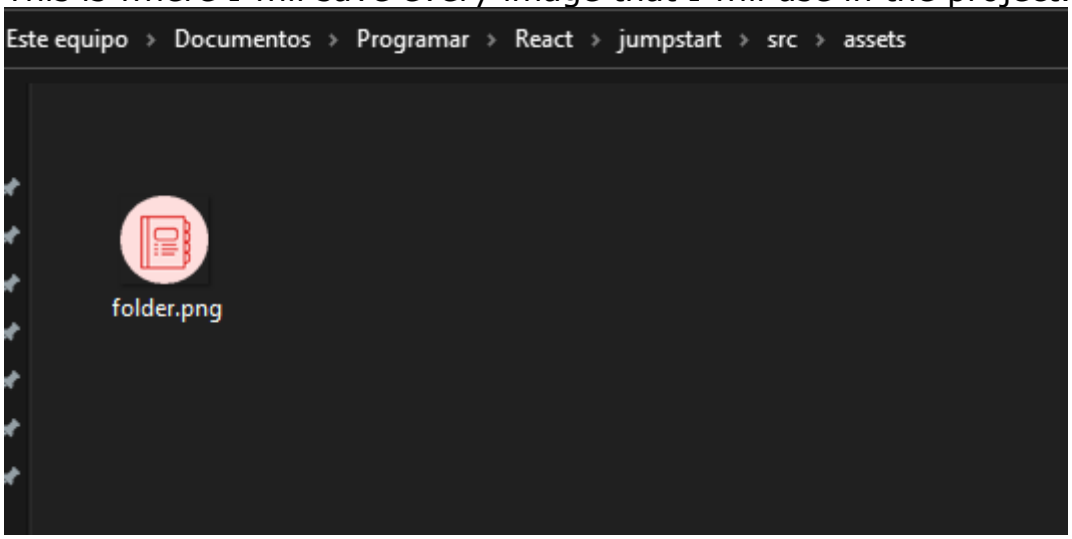


Now I go to the new app directory and create an **assets** folder within the "**src"** folder of the project.

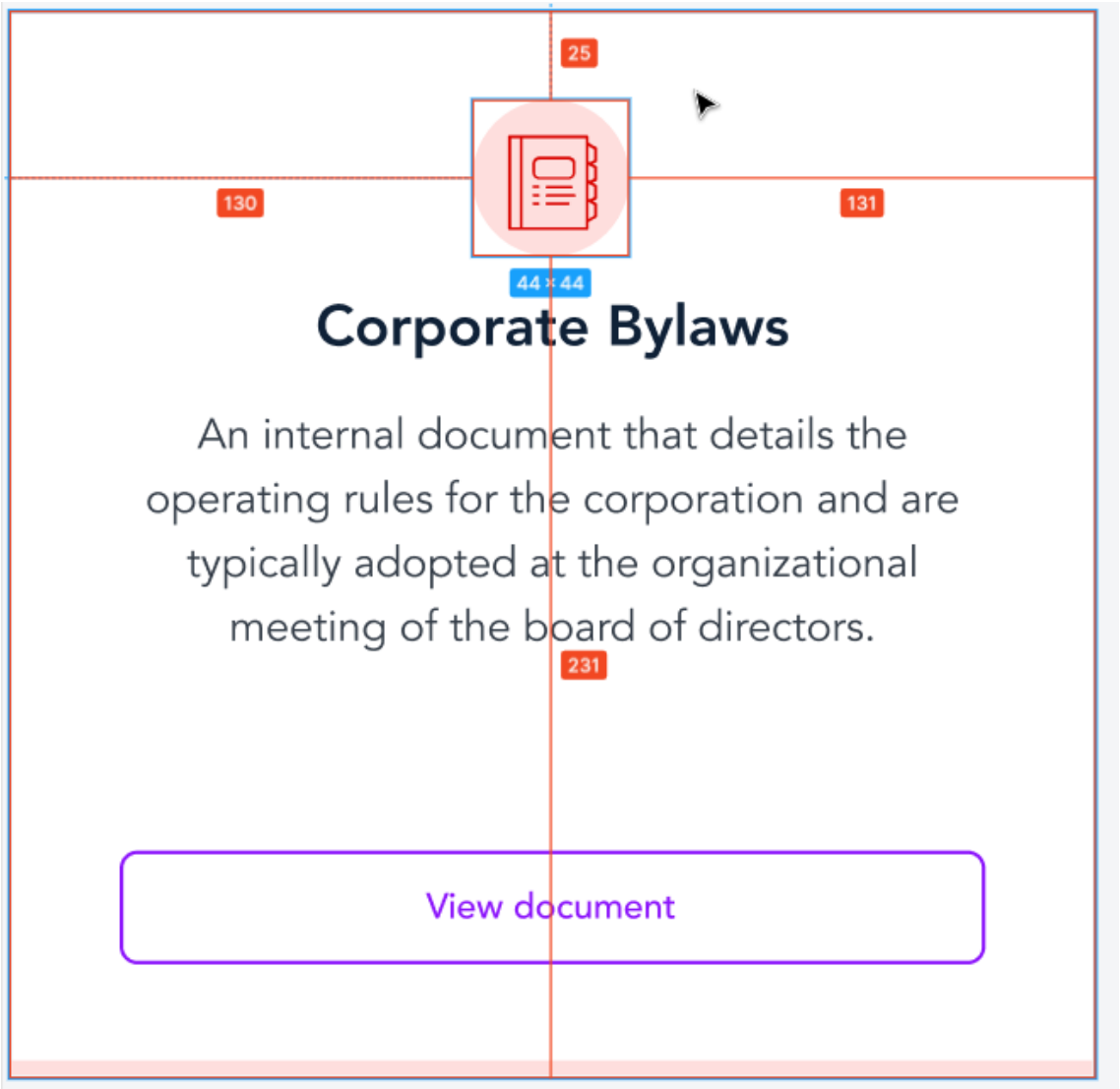




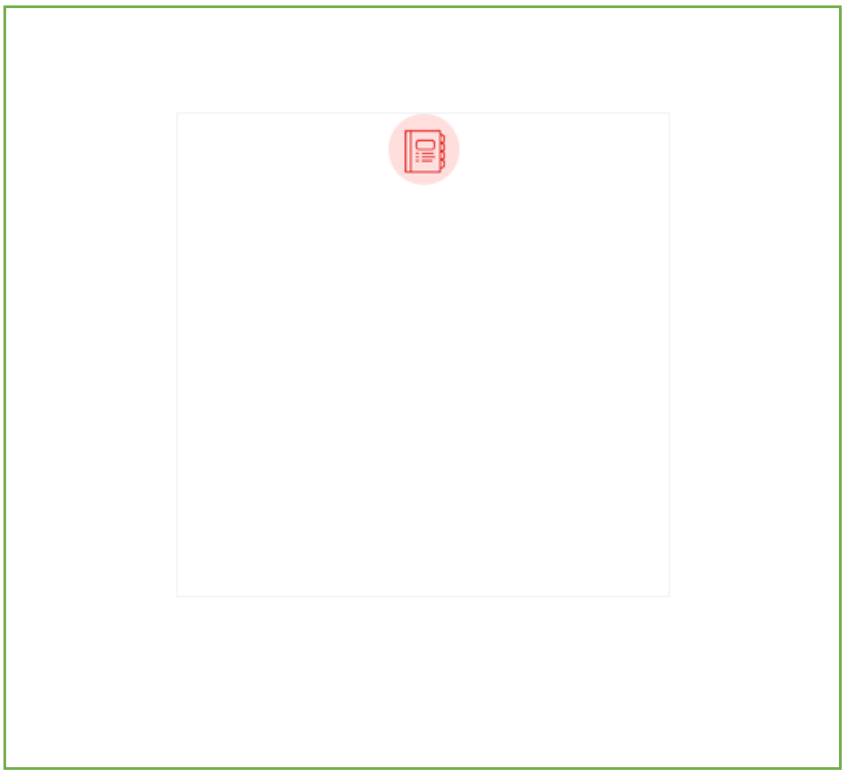
This is where I will save every image that I will use in the project.



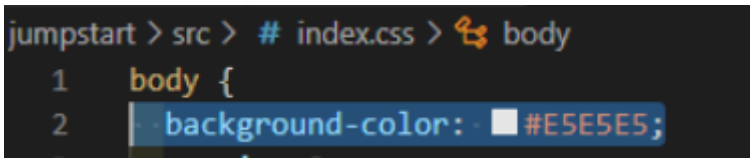
For guidance, I followed the dimensions that Figma had provided me in px. Sometimes I did not have the exact measures, but I found a workaround to solve it, which I mention in additional notes.



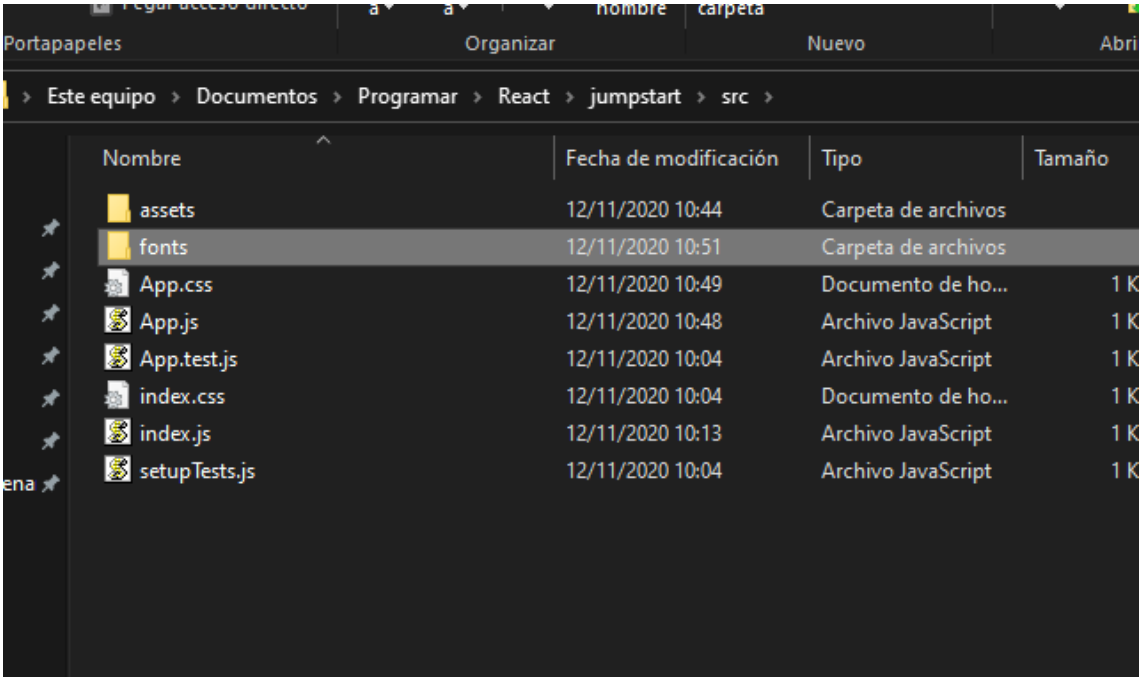
The result is as follows:



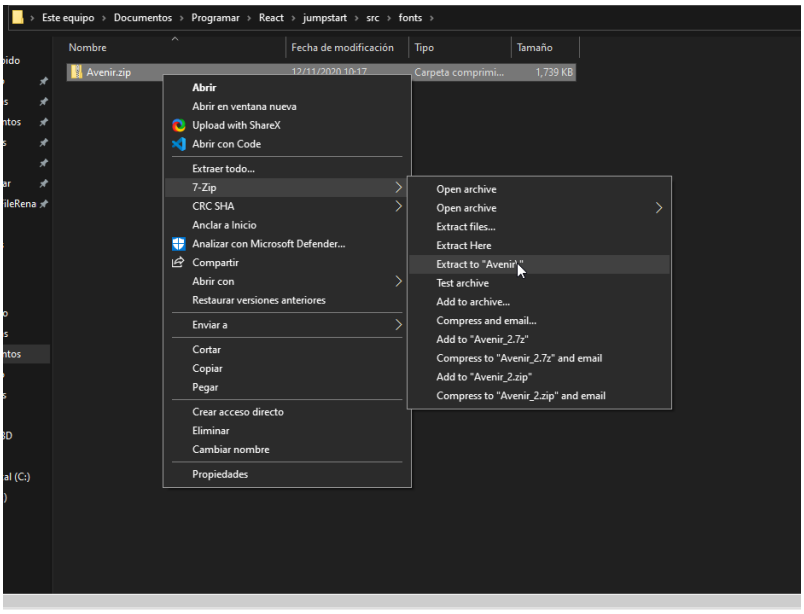
A white box with a grey border in a white background. It looks messy so I will add the background color to work better.
It must be added within the **index.css** file.

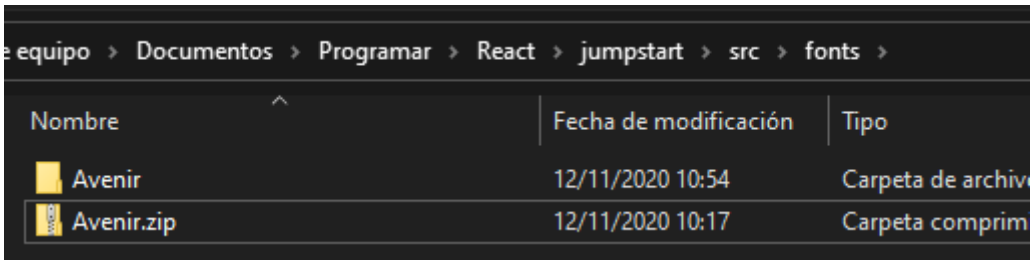


Okay, now that the Icon works, I will go back to the **src** folder and create a “fonts” folder within it.

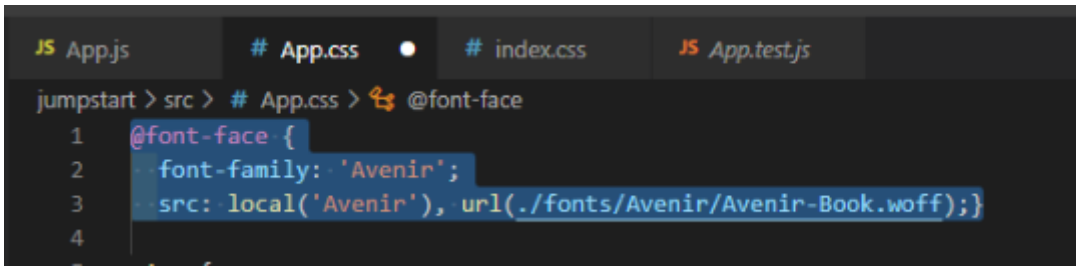


Download the font provided by [Jumpstart] and extract it in a folder inside “fonts” folder

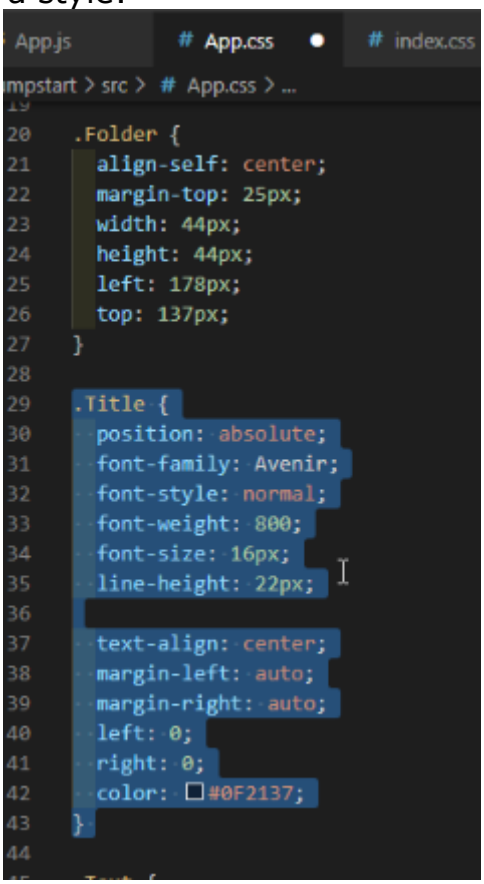




Within **app.css** we will install the new font, following the directory in which it was extracted.



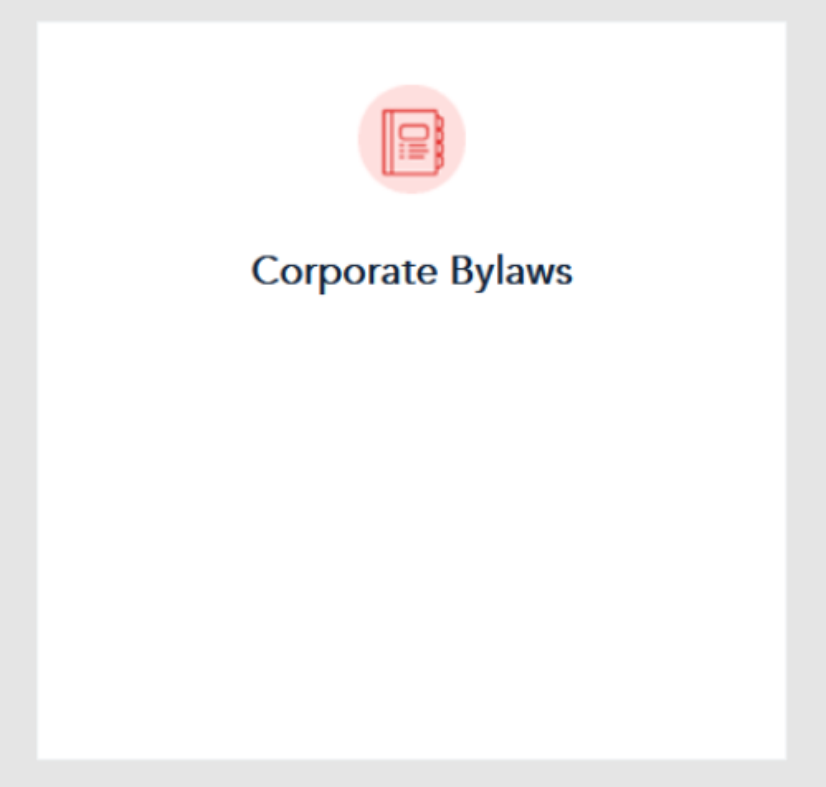
and then insert it to a style:



Now we can add the **.Title** class to a HTML paragraph.

```
function App() {
  return (
    <div className="App">
      <body className="Rectangle24">
        <img src={folder} alt="folder" className="Folder" />
        <p className="Title">Corporate Bylaws</p>

      </body>

    </div>
  );
}
```

So far:



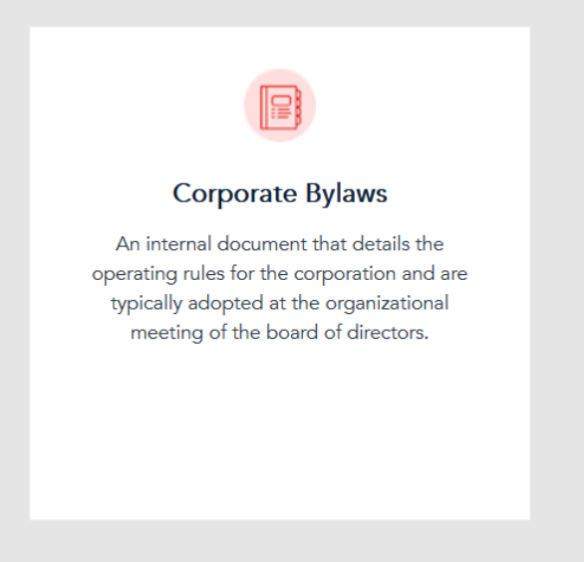
Now it is time for the text body:

```
function App() {
  return (
    <div className="App">
      <body className="Rectangle24">
        <img src={folder} alt="folder" className="Folder" />
        <p className="Title">Corporate Bylaws</p>
        <p className="Text">An internal document that details the operating rules for the corporation and are typically adopted at the organizational meeting of the board of directors.</p>
      </body>

    </div>
  );
}
```

And its class:

```css
.Text {
  position: absolute;
  width: 243px;
  height: 72px;
  left: 79px;
  top: 110px;

  font-family: Avenir;
  font-style: normal;
  font-weight: normal;
  font-size: 12px;
  line-height: 150%;

  margin-left: auto;
  margin-right: auto;
  left: 0;
  right: 0;

  text-align: center;

  color: #343D48;
}
```

Result:



Now for the View Document button, I will create a paragraph with a class:

```jsx
function App() {
  return (
    <div className="App">
      <body className="Rectangle24">
        <img src={folder} alt="folder" className="Folder" />
        <p className="Title">Corporate Bylaws</p>
        <p className="Text">An internal document that details
        <div style={{marginTop: 176}}>
          <p className="ViewDocument">View document</p>
        </div>
      </body>

    </div>
  );
```

```css
.ViewDocument {
font-family: Avenir;
font-style: normal;
font-weight: 500;
font-size: 10px;
line-height: 14px;

color: #8C14FC;
}
```

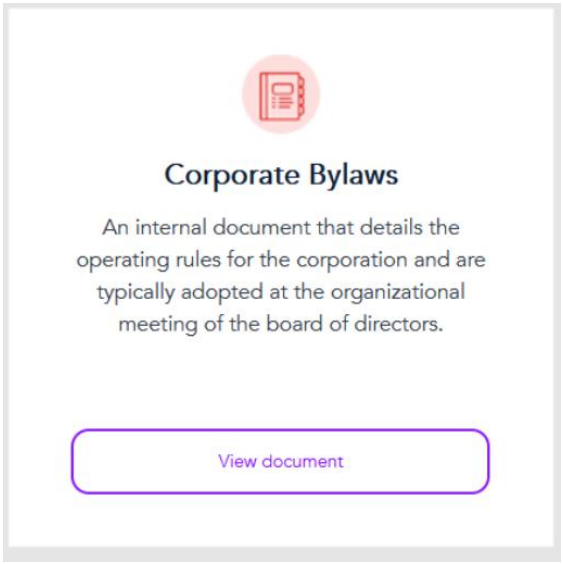
Result:

Now for the border of the button:

```
.Border {
  border-style: solid;
  border-color: #8c14fc;
  color: #8c14fc;
  border-width: 1.5px;
  margin: 35px;
  border-radius: 10px;
}
```

```
<div style={{marginTop: 176}}>
  <div className="Border">
  <p className="ViewDocument">View document</p>
  </div>
</div>
```

And here it is:

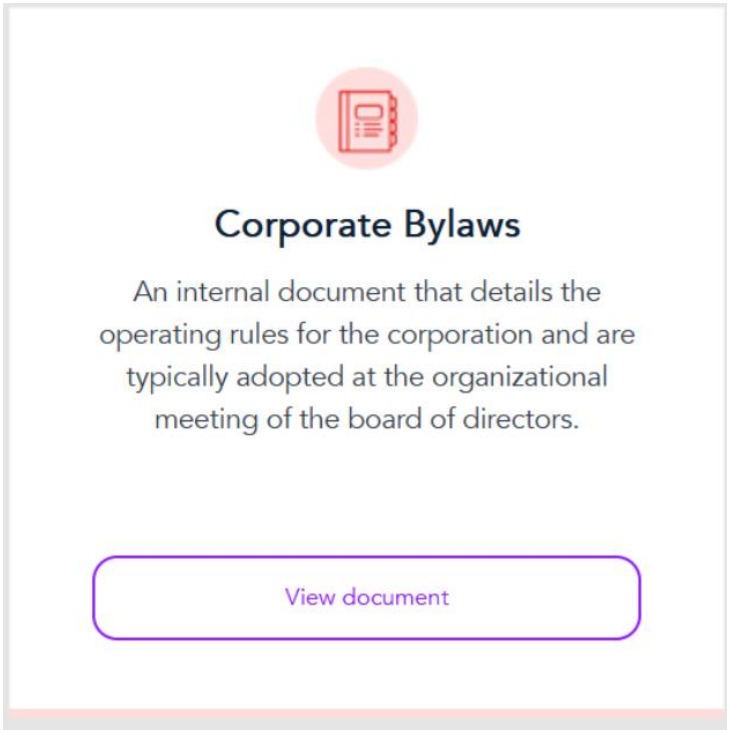


For the rectangle below the *Main Rectangle*, I will create a rectangle with the same width as the main one, and a height of 4px according to Figma once again.

```
.MainRectangleShadow {
  width: 305px;
  height: 4px;
  background: #FEDEDD;
  margin-top: 320px;
  margin-left: 20px
}
```

And put the code below the </body> of the main rectangle.

```
</body>
  <div className="MainRectangleShadow"></div>
```

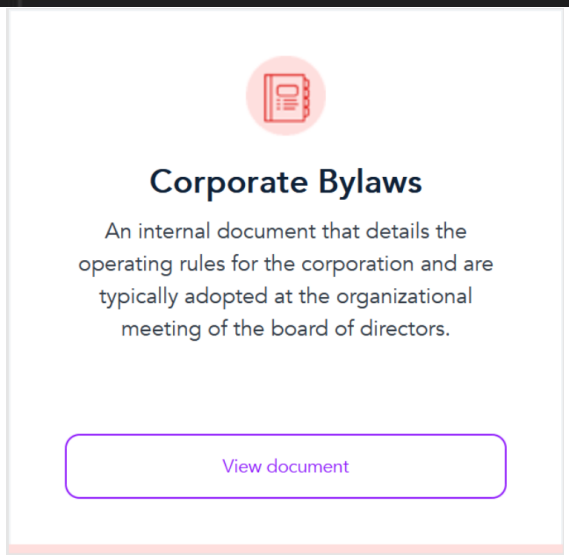


Title was looking a little messy, so I added a new font:

```
@font-face {
  font-family: 'Avenir-medium';
  src: local('Avenir-medium'), url(./fonts/Avenir/Avenir-Medium.woff);
}
```

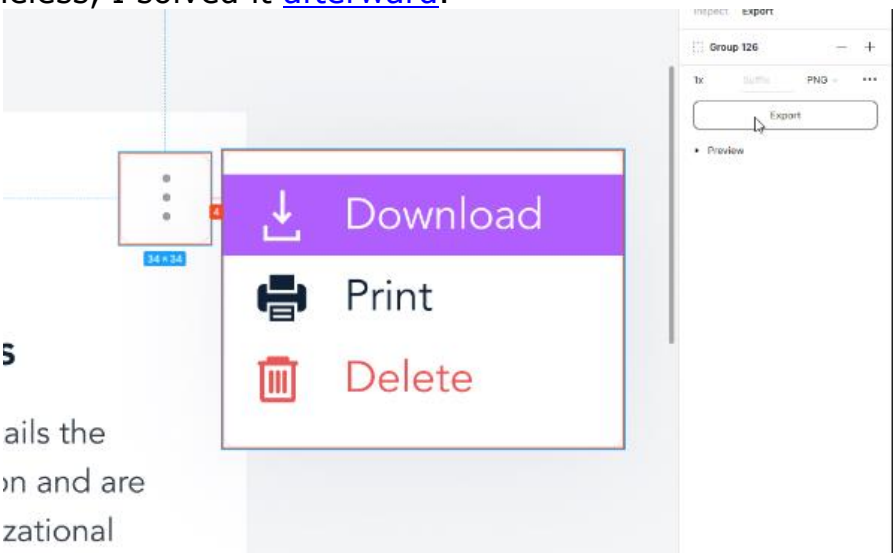
And I added it to the class:

```
.Title {
  position: absolute;
  font-family: Avenir-medium;
  font-weight: 800;
  font-size: 18px;
  line-height: 22px;
  top: 65px;
```

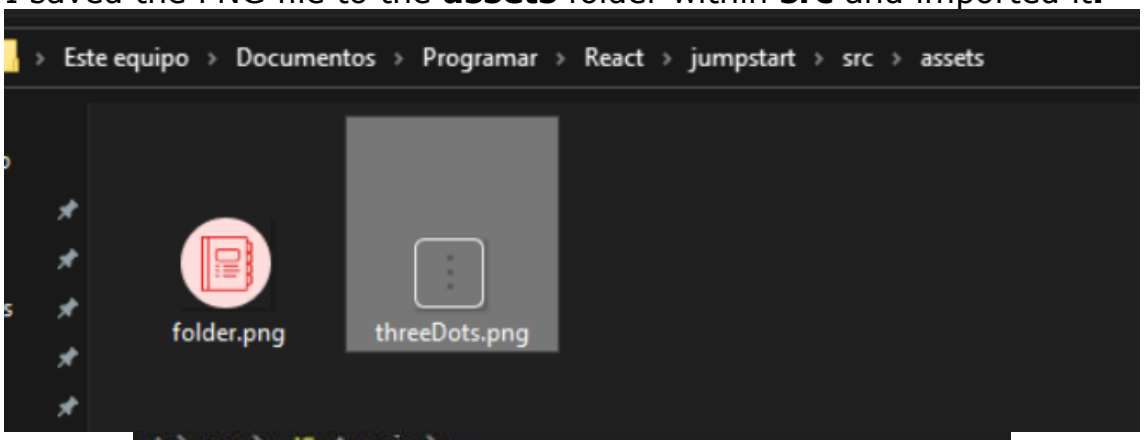


3. Interactive Object

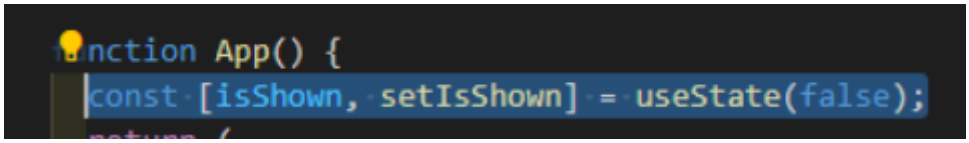
Now for the *three dots button* (…) I discovered a feature Figma had to export as PNG. Sadly, it becomes pixelated.
Nevertheless, I solved it [afterward](#).



I saved the PNG file to the **assets** folder within **src** and imported it**.**



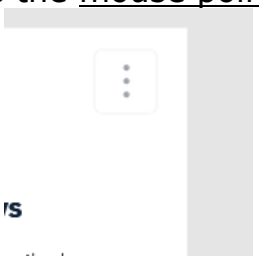
To make the *three dots button* show when the mouse was hovered on the *main rectangle*, I had to add **states** to my function, and the **CSS** class.



```
<body className="Rectangle24" onMouseEnter={() => setIsShown(true)} onMouseLeave={() => setIsShown(false)}>
  {isShown && (
  <div>
    <img src={threeDots} alt="threeDots" className="threeDots"/>
  </div>
)}
```

```
.threeDots {
  position: fixed;
  left: 276px;
  top: 31px;
  height: 34px;
  width: 34px;
}
```

It now shows as soon as the mouse pointer is on the rectangle



Now I needed to add an **onClick** event for the image to bring the new rectangle

```
  <img src={threeDots} alt="threeDots" className="threeDots" onClick={() => setDotsMenu(true)} />
</div>
```

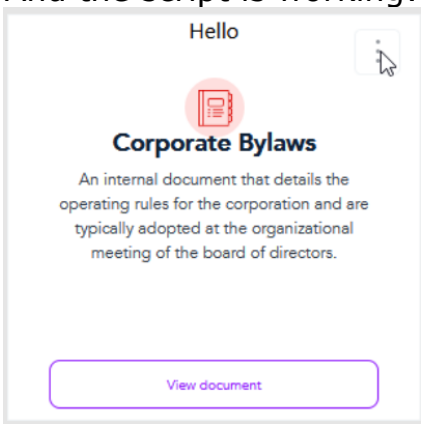
For the program to work correctly, I must add a new state to the function

```
function App() {
  const [isShown, setIsShown] = useState(false);
  const [DotsMenu, setDotsMenu] = useState(false);
  return (
```

Before I continue, I must check if everything is in order so far. To do so, I add a Hello paragraph with the **if (&&) conditioner** in React.

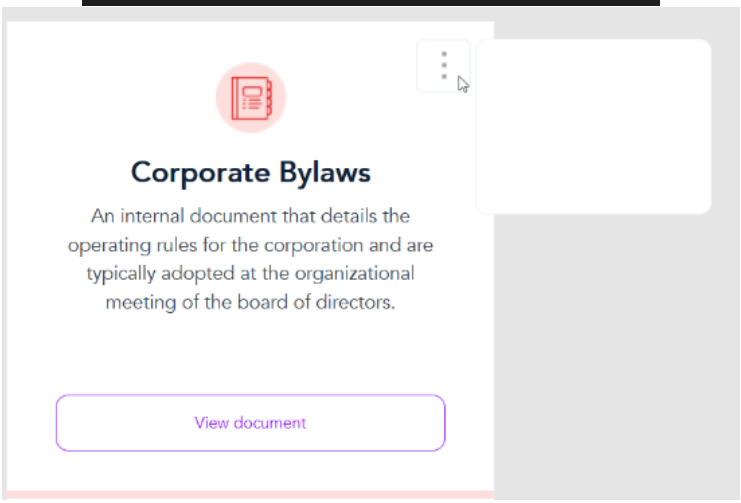
```
function App() {
  const [isShown, setIsShown] = useState(false);
  const [DotsMenu, setDotsMenu] = useState(false);
  return (
    <div className="App">
      <body className="Rectangle24" onMouseEnter={() => setIsShown(true)} onMouseLeave={() => setIsShown(false)}>
        {isShown && (
        <div>
          <img src={threeDots} alt="threeDots" className="threeDots" onClick={() => setDotsMenu(true)}/>
          {DotsMenu && (<div>Hello</div>)}
        </div>
```

And the script is working:



Great! Now I will create the new rectangle, being smaller than the main one, following the dimensions provided.

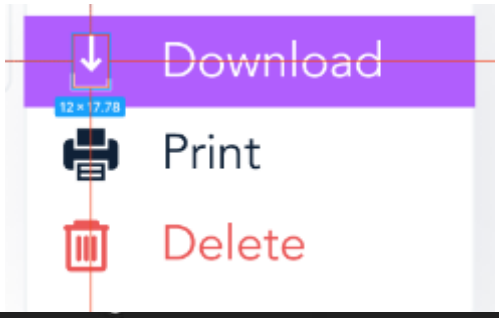
```css
.RectangleDots {
  position: fixed;
  width: 148px;
  height: 110px;
  left: 313px;
  top: 31px;
  background: #FFFFFF;
  border: 1px solid #F1F4F6;
  border-radius: 8px;
  box-sizing: border-box;
}
```

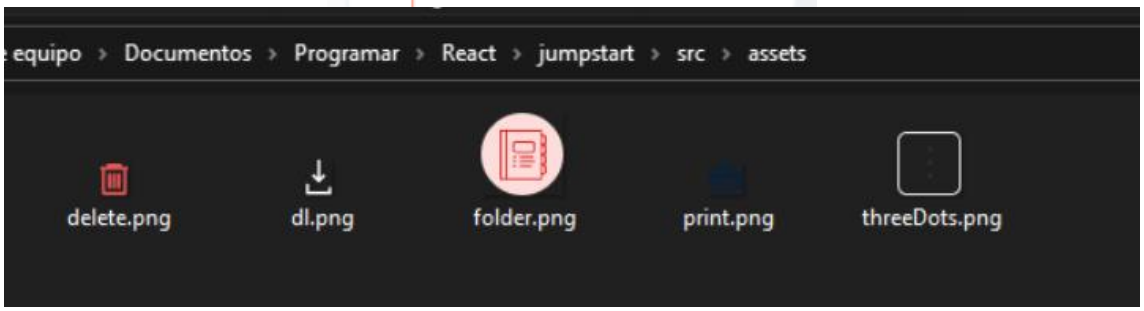


I will now make the menu close when the mouse pointer leaves both the *main rectangle* and the small new rectangle.

```
{isShown && (<div>
  <img src={threeDots} alt="threeDots" className="MainRectangleThreeDotsButton" onClick={() => setDotsMenu(true)}/>
  {DotsMenu && (<div className="DotsMenu" onMouseLeave={() => setDotsMenu(false)}>
```

Repeating the process used before to download a PNG image from Figma, I downloaded the small rectangle icons, saving them to the **assets** folder once again.





And of course, imported them.

```
import dl from './assets/dl.png'
import del from './assets/delete.png'
import print from './assets/print.png'
```

Now the classes:

```
.DotsMenuImage {
    float: left;
    margin-top: 6px;
    margin-top: 6.22px;
    margin-left: 16px;
}
.DotsMenuPurpleBackground {
    height: 30px;
    background: #B05EFD;
    box-sizing: border-box;
}
.DotsMenuPurpleBackgroundText {
    font-family: Avenir;
    font-style: normal;
    font-weight: normal;
    font-size: 16px;
    line-height: 30px;
    color: #ffffff;
    display: flex;
    align-items: center;
    margin-left: 45px;
    margin-top: 10px;
}
```

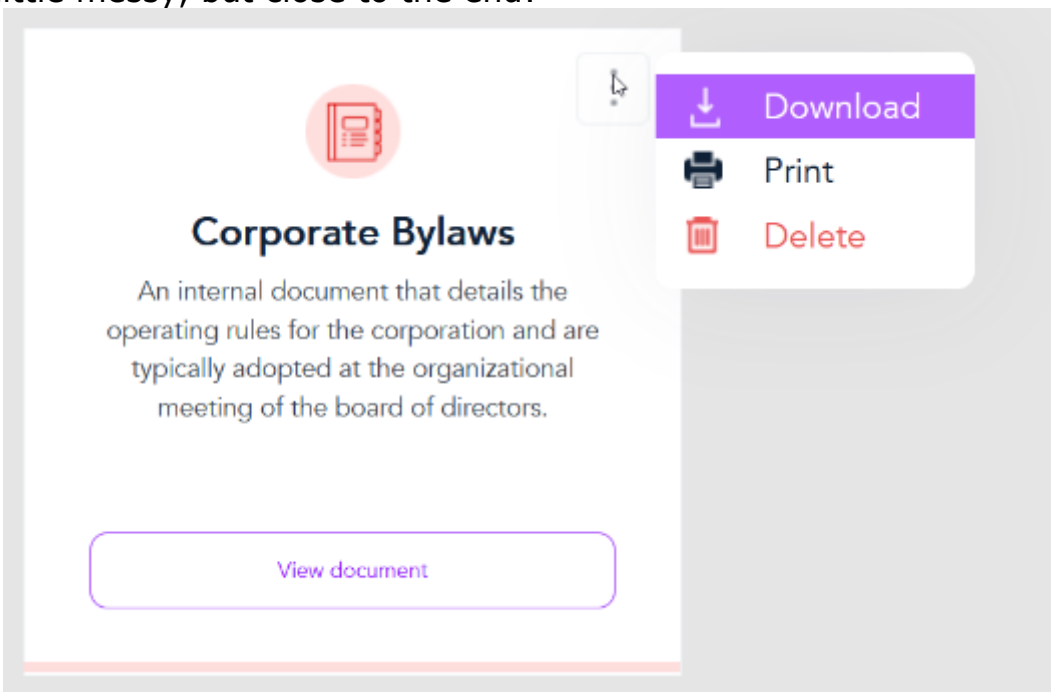
```
.DotsMenuPrintText {
    font-family: Avenir;
    font-style: normal;
    font-weight: normal;
    font-size: 16px;
    line-height: 30px;
    color: #0F2137;
    display: flex;
    align-items: center;
    margin-left: 45px;
    margin-top: 10px;
}

.DotsMenuDeleteText {
    font-family: Avenir;
    font-style: normal;
    font-weight: normal;
    font-size: 16px;
    line-height: 30px;
    color: #EB5757;
    display: flex;
    align-items: center;
    margin-left: 45px;
    margin-top: 10px;
}
```

```
.DotsMenuDownloadImage {
    float: left;
    margin-top: 6px;
    margin-top: 6.22px;
    margin-left: 16px;
}

.DotsMenuPrintImage {
    float: left;
    margin-top: 6px;
    margin-top: 6.22px;
    margin-left: 16px;
}
.DotsMenuDeleteImage {
    float: left;
    margin-top: 6px;
    margin-top: 6.22px;
    margin-left: 16px;
}
```

And the HTML code:

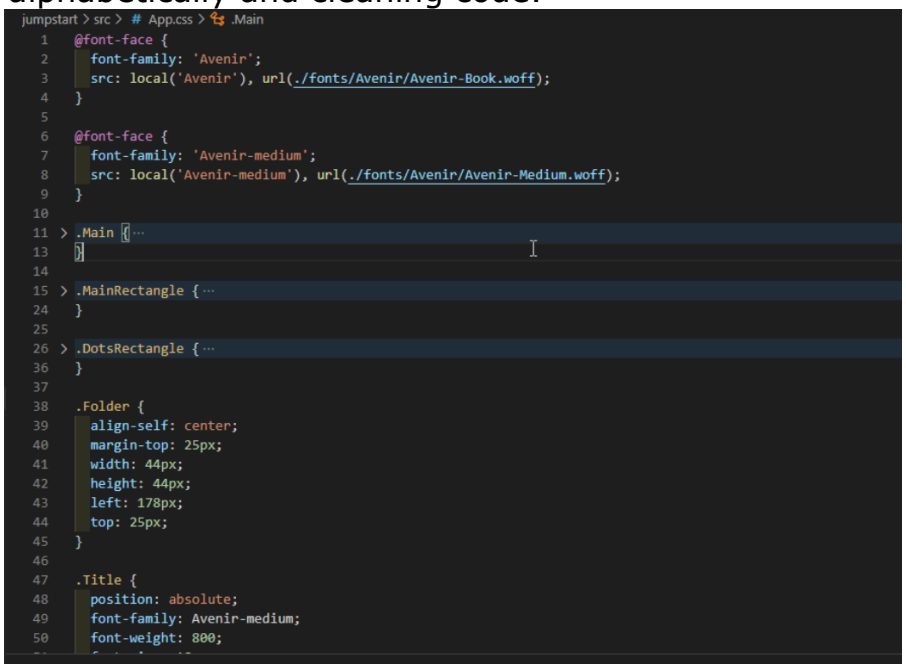
```
<div className="DotsMenuPurpleBackground">
    <img src={dl} alt="dl" className="DotsMenuDownloadImage"/><div>
    <p className="DotsMenuPurpleBackgroundText">Download</p></div>
</div>

<div className="DotsMenuNoBackground">
    <img src={print} alt="print" className="DotsMenuPrintImage"/><div>
    <p className="DotsMenuPrintText">Print</p></div>
</div>

<div className="DotsMenuNoBackground">
    <img src={del} alt="del" className="DotsMenuDeleteImage"/><div>
    <p className="DotsMenuDeleteText">Delete</p></div>
</div>
```

A little messy, but close to the end:

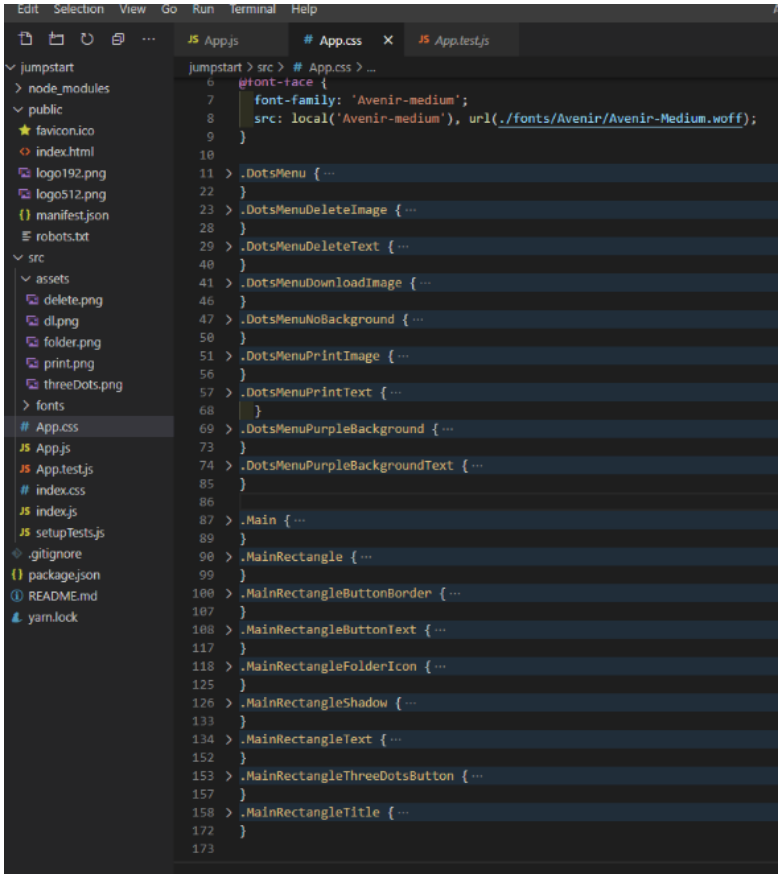


4. Cleaning Code

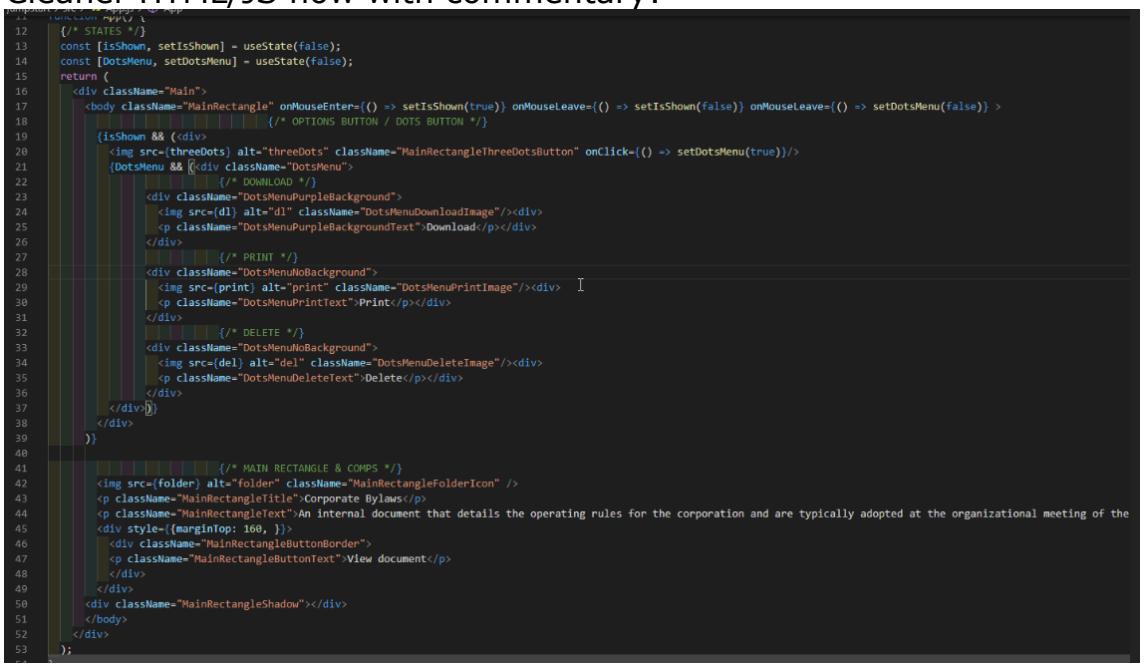
Classes were disorganized as well as code. It was time to begin sorting alphabetically and cleaning code.



Sorted:



Cleaner HTML/JS now with commentary:

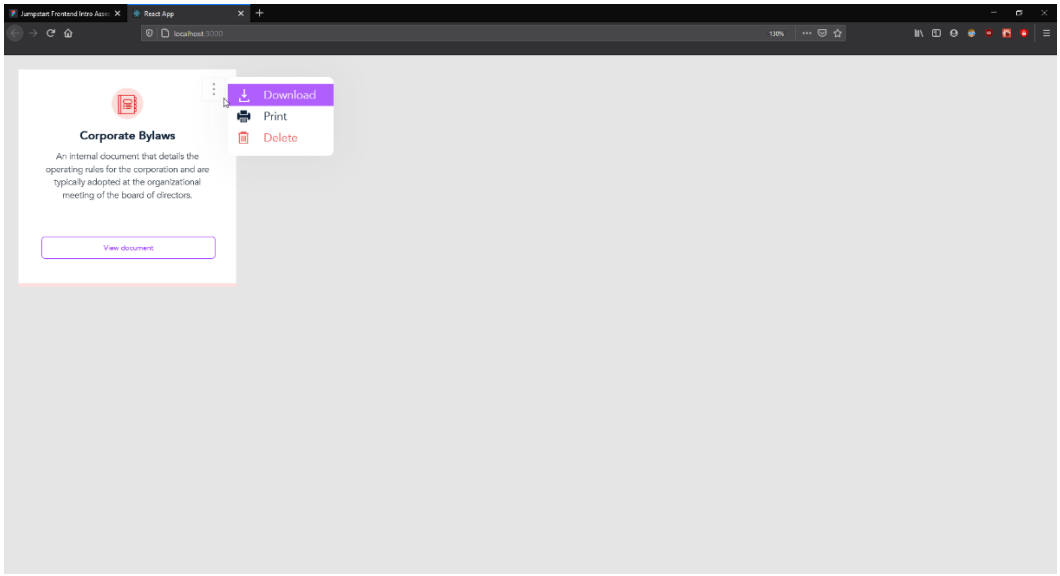


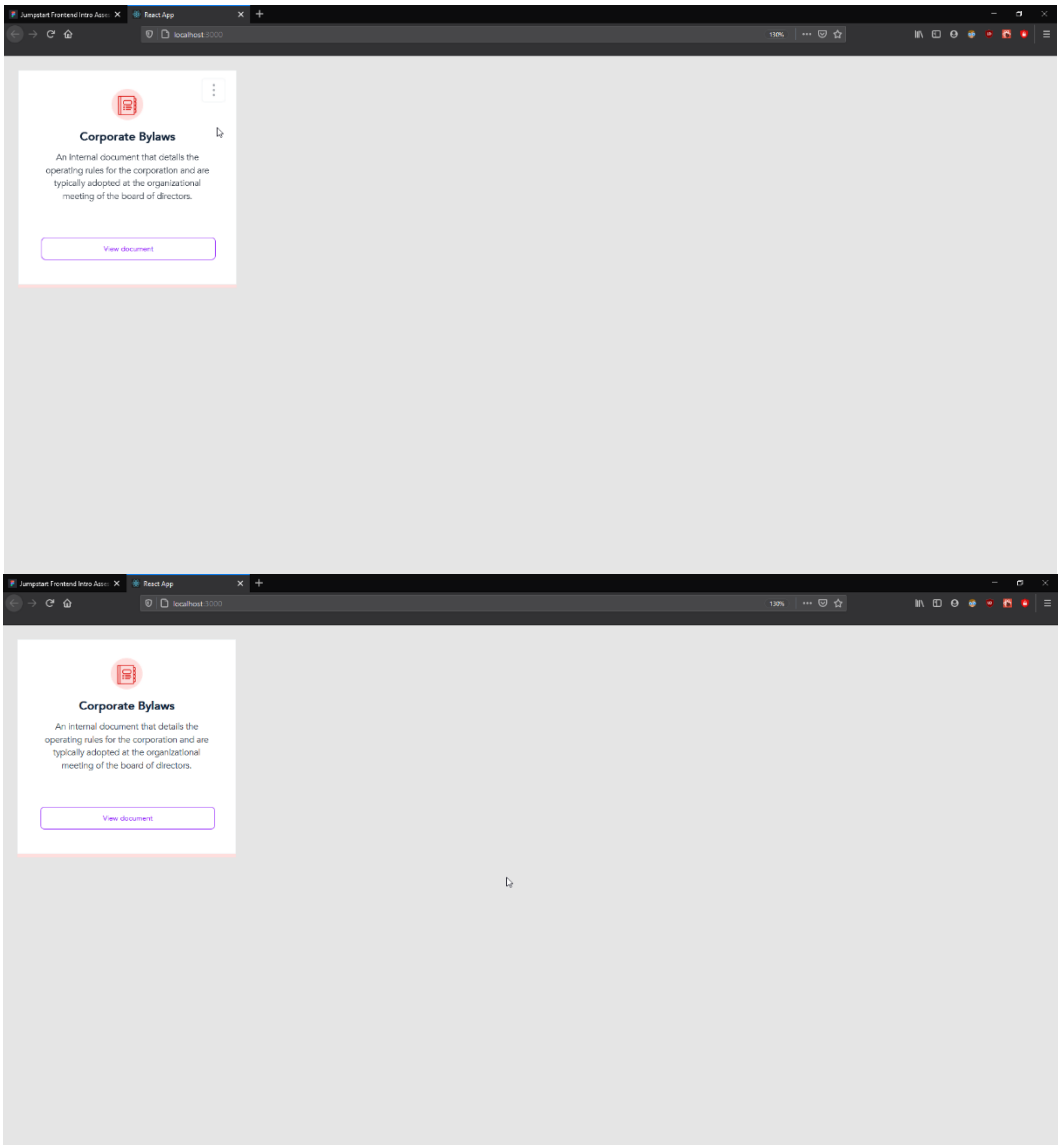
```
  {/* STATES */}
  const [isShown, setIsShown] = useState(false);
  const [DotsMenu, setDotsMenu] = useState(false);
  return (
    <div className="Main">
      <body className="MainRectangle" onMouseEnter={() => setIsShown(true)} onMouseLeave={() => setIsShown(false)} onMouseLeave={() => setDotsMenu(false)} >
                                {/* OPTIONS BUTTON / DOTS BUTTON */}
      {isShown && (<div>
        <img src={threeDots} alt="threeDots" className="MainRectangleThreeDotsButton" onClick={() => setDotsMenu(true)}/>
        {DotsMenu && (<div className="DotsMenu">
                        {/* DOWNLOAD */}
                <div className="DotsMenuPurpleBackground">
                  <img src={dl} alt="dl" className="DotsMenuDownloadImage"/><div>
                  <p className="DotsMenuPurpleBackgroundText">Download</p></div>
                </div>
                        {/* PRINT */}
                <div className="DotsMenuNoBackground">
                  <img src={print} alt="print" className="DotsMenuPrintImage"/><div>
                  <p className="DotsMenuPrintText">Print</p></div>
                </div>
                        {/* DELETE */}
                <div className="DotsMenuNoBackground">
                  <img src={del} alt="del" className="DotsMenuDeleteImage"/><div>
                  <p className="DotsMenuDeleteText">Delete</p></div>
                </div>
              </div>)}
        </div>
      )}

                        {/* MAIN RECTANGLE & COMPS */}
        <img src={folder} alt="folder" className="MainRectangleFolderIcon" />
        <p className="MainRectangleTitle">Corporate Bylaws</p>
        <p className="MainRectangleText">An internal document that details the operating rules for the corporation and are typically adopted at the organizational meeting of the
        <div style={{marginTop: 160, }}>
          <div className="MainRectangleButtonBorder">
          <p className="MainRectangleButtonText">View document</p>
          </div>
        </div>
      <div className="MainRectangleShadow"></div>
      </body>
    </div>
  );
```

Followed by a couple CSS code tweaks to improve visuals and formatting, deleted some extra HTML/JS code as well as useless files.

5. Result

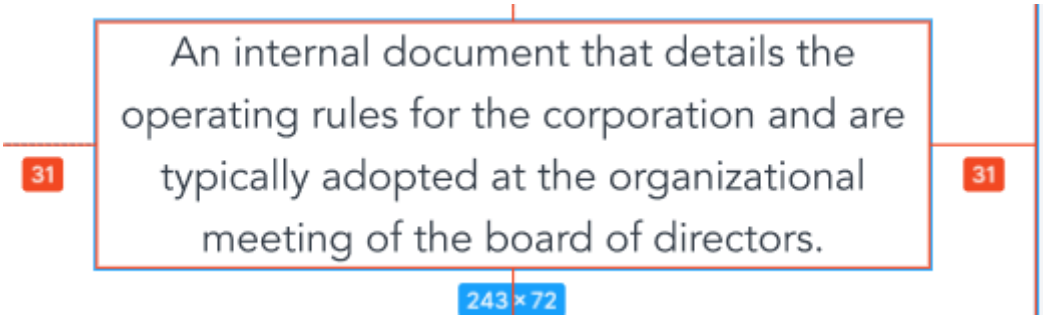
Here is the definitive **result:**



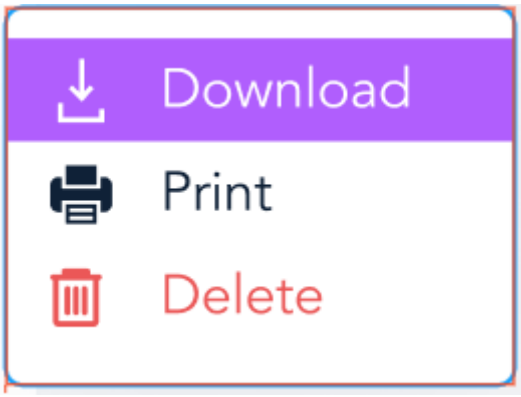

Corporate Bylaws
An internal document that details the operating rules for the corporation and are typically adopted at the organizational meeting of the board of directors.
View document
Corporate Bylaws
An internal document that details the operating rules for the corporation and are typically adopted at the organizational meeting of the board of directors.
View document

ADITIONAL NOTES

Sometimes I had to think outside of the box, because the representations were not as accurate as expected (e.g. the *View Document border* didn't specify its size or margins, so I assumed it would follow the same line as the *body text container* had.



This happened to me as well when I saw the *three dot options menu icons* (2) I calculated it by downloading the image at 100% size and using the ruler provided by **Adobe Photoshop**, to estimate the px value.



To fix the pixelated icons I asked my girlfriend, who is a Graphic Designer, to re-draw those icons for exclusive use within this exercise. I then resized them within the code to sharpen the way they look.

```
.DotsMenuDeleteImage {
  float: left;
  margin-top: 6px;
  margin-bottom: 6.22px;
  margin-left: 14px;
  width: 15px;
  height: 17px;
}
.DotsMenuDeleteText { ···
}
.DotsMenuDownloadImage {
  float: left;
  margin-top: 6px;
  margin-top: 6.22px;
  margin-left: 16px;
  width: 14px;
  height: 19px;
}
.DotsMenuNoBackground { ···
}
.DotsMenuPrintImage {
  float: left;
  margin-top: 6px;
  margin-top: 6.22px;
  margin-left: 13px;
  width: 18px;
  height: 17px;
}
```