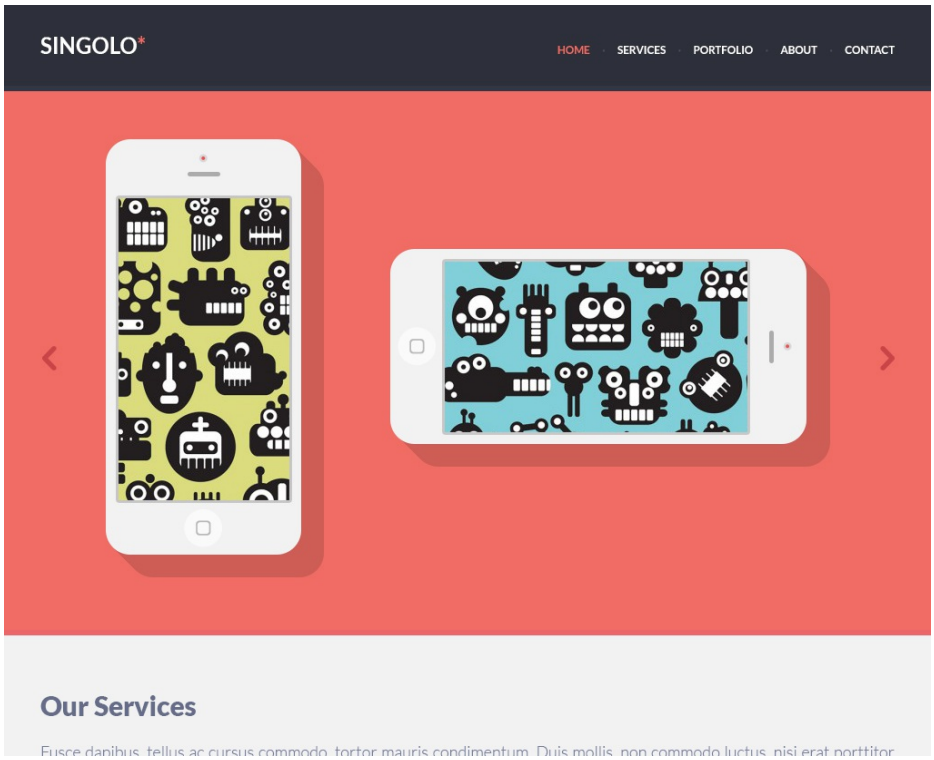
## Singolo Website

The *Singolo* design is a free template created by Marijan Petrovski. It's available on http://www.psdchat.com, but for your convenience, you can find it in the `289.101.05.files` download on Stream.




In this workshop, you will code an HTML website using the Singolo .psd (Photoshop) file. There are many ways to design websites (Photoshop, Gimp, Illustrator, Inkscape, XD, Sketch, etc.), but you'll need to convert designs in functioning websites using code.

### Document Setup

Locate and open the "singolo.psd" file. You will be using Photoshop throughout this task to take measurements, sample colours, and crop-out images.

Your lecturer/tutor will introduce the Photoshop Info panel (**Window > Info**) to help you sample colours and take measurements. If you miss the class or need more information, refer to Adobe's documentation:

https://helpx.adobe.com/photoshop/using/image-information.html

Create a new directory in your 289.101 repository named "singolo". Your website files will be contained within this. In Brackets, from the *Getting Started* drop-down, select **Open Folder...** and point it here.

Setup a scaffold -- an "index.html" and "styles.css" file -- for the website by creating an HTML5 document and linking it to a new CSS file:

*index.html*

```
<!DOCTYPE html>

<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Singolo</title>
  <meta name="description" content="A free template created by Marijan Petrovski" />
  <link rel="stylesheet" href="styles.css" />
</head>

<body>

</body>
</html>
```

**Note:** "index.html" will always be the file-name for your website's landing page.

*styles.css*

```
body {
  background-color: red;
}
```

If you preview the "index.html" page in your browser and find that the background is a bright red, you've managed to connect your style-sheet to your HTML document successfully. Now change the background colour to light grey, as per the design. You will also need to remove the default margin on the `body`, so as to avoid a gap appearing around your page contents:

```
body {
  background-color: #F2F2F2;
  margin-top: 0;
  margin-right: 0;
  margin-bottom: 0;
  margin-left: 0;
}
```

However, this code can be simplified because the margin property has a shorthand version (that accepts values in a clockwise sequence, starting at top):

```
body {
  background-color: #F2F2F2;
  margin: 0 0 0 0;
}
```

Moreover, as all of the margin sides are equal to 0, this can be taken one step further, using a single value to represent all sides:

```
body {
  background-color: #F2F2F2;
  margin: 0;
}
```

## Outlining Elements

It can be difficult to ascertain what is going on with your div's and other elements -- so applying an outline to everything using a wildcard selector can be helpful. To do this, add the following rule to the top of your CSS file:

```
* {
  outline: 1px dotted lime;
}

...
```

Remember to remove these outlines when you're done with the website. You can comment them out when no longer desired, for example:

```
/* {
  outline: 1px dotted lime;
}
*/

...
```

## Header Container

Inside the body element, create a div for your header.

*index.html*

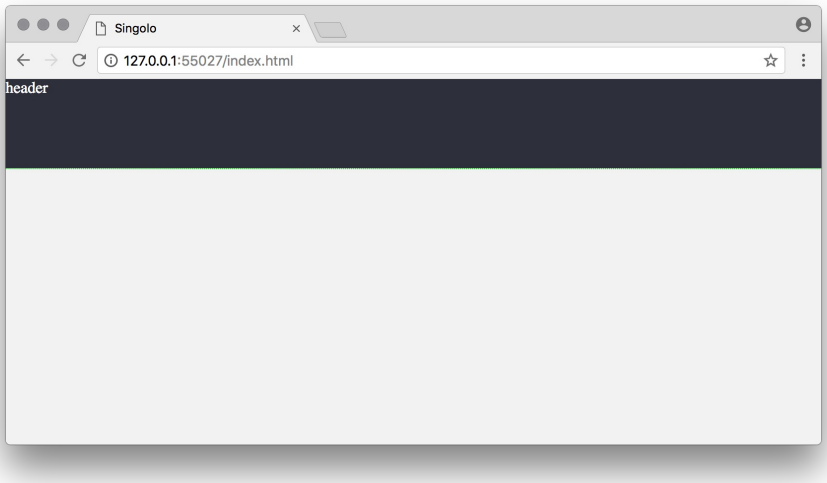
```
...

<body>

  <div class="header">
    header
  </div>

</body>

...
```

and the corresponding CSS:

*styles.css*

```
...

.header {
  background-color: #2D2F3B;
  color: white;
  height: 90px;
}
```

Notice that the header element is now 90px tall and, by default, is 100% wide.



The next step is to add a `div` within this to keep the content centred -- so that the blue-ish strip will continue to touch the sides of the browser, while the logo remains centred (or more correctly, to the left of the centred design):

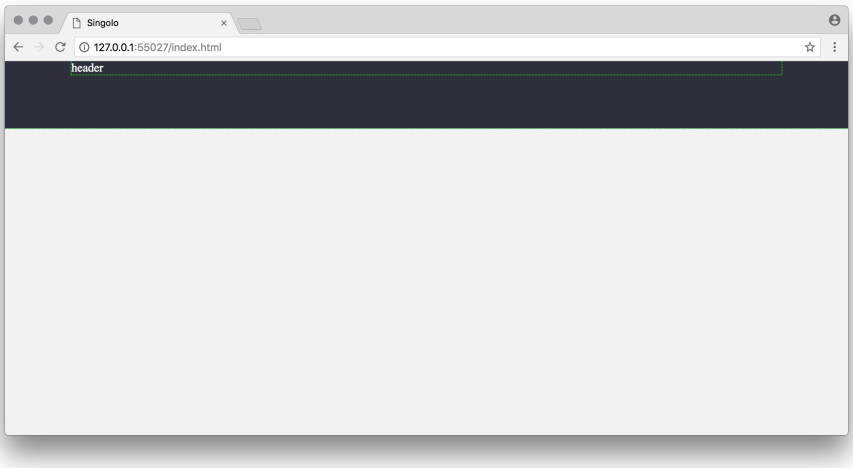
*index.html*

```
...

<body>

  <div class="header">
    <div class="nav">
      header
    </div>
  </div>

</body>

...
```

*styles.css*

```
...

.nav {
  display: grid;
  grid-template-columns: 200px auto;
  margin-left: auto;
  margin-right: auto;
  max-width: 950px;
}
```

Adjust the width of your browser window noting how the `.nav` element maxes-out at 950 pixels, thereafter centring itself.




## Header Contents

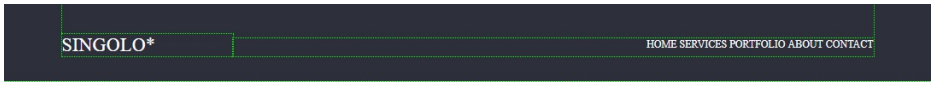
Now add the content to the header, namely: the logo and links.

*index.html*

```
...

<div class="header">
  <div class="nav">

    <div class="logo">
      SINGOLO*
    </div>

    <div class="links">
      HOME
      SERVICES
      PORTFOLIO
      ABOUT
      CONTACT
    </div>

  </div>
</div>

...
```

The logo and links `div`s sit alongside one another thanks to the grid properties in the parent (`.nav`) selector. Dial-in in the size and position properties using a new `logo` and `.links` selector:

```
...

.logo {
  margin-top: 35px;
  font-size: 1.4rem;
}

.links {
  margin-top: 40px;
  font-size: 0.75rem;
  text-align: right;
}
```

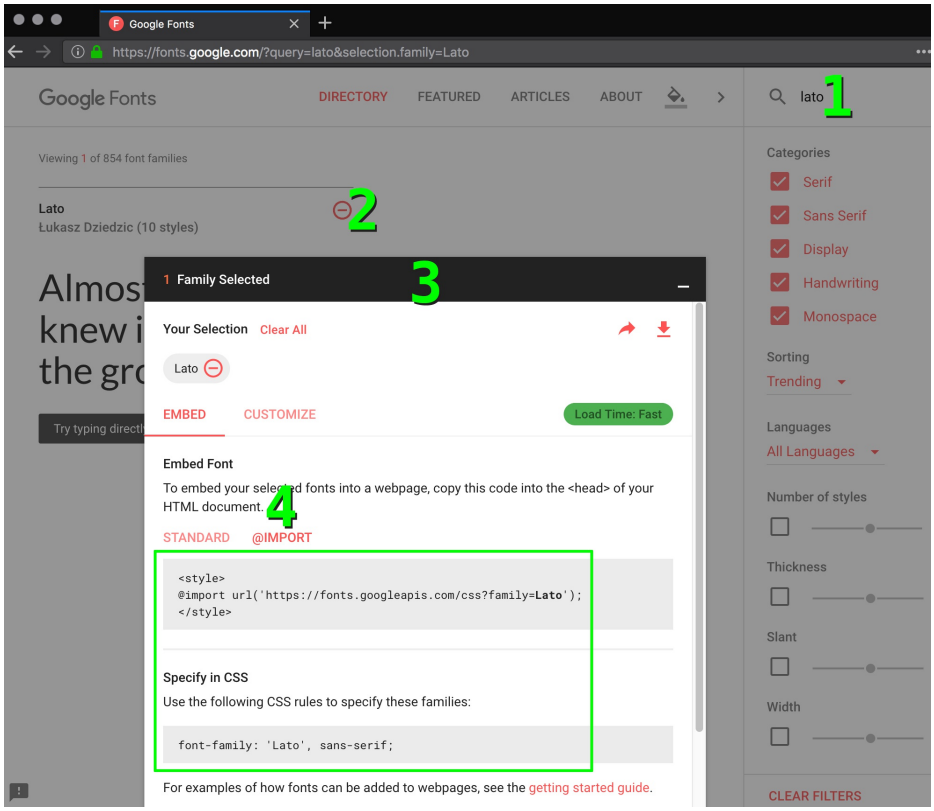



It's getting there ... but before proceeding, the font needs to be changed.

## Adding a Google Font

Using your web browser, head over to:
http://fonts.google.com

1. Search for the font named "lato";
2. click the + icon to add it to your selection;
3. raise the *Family Selected* tab on the bottom of the page;
4. then click `@import` to reveal the code to be copy/pasted into your working project.



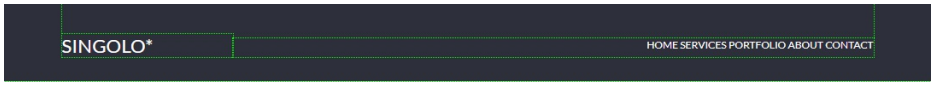

Before pasting-in any code, notice that default web-page text is black and rendered in a serif font. Whereas you could add a `font-family` property to every CSS selector you write, it's better to apply this at a page level -- as it is really the 'rule' rather than an 'exception':

*styles.css*

```
@import url('https://fonts.googleapis.com/css?family=Lato');

...

body {
  background-color: #F2F2F2;
  margin: 0;
  font-family: 'Lato', sans-serif;
}

...
```

You'll notice this code deviates slightly from the Google instructions, in favour of something more straightforward. Note that the `@import` line must always precede any references to the font it supplies; therefore, it's probably best to just place any `@imports` at the top of your CSS file.

You now have the Lato font applied across the whole web-page:



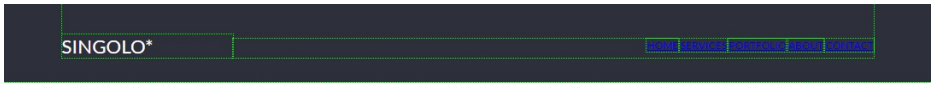

## Links

The links need anchor (a) tags so that they can be made click-able. For now, there is no need for the `href` to target anything, so just make this equal to a #:

*index.html*:

```
      ...

      <div class="links">
        <a href="#">HOME</a>
        <a href="#">SERVICES</a>
        <a href="#">PORTFOLIO</a>
        <a href="#">ABOUT</a>
        <a href="#">CONTACT</a>
      </div>

      ...
```

Of course, this makes the links blue and underlined:




To override the default display behaviour, you need to use CSS. A *descendant selector* will save you having to place a class selector on every a tag:

*index.html*:

```
...

.links a {
  color: white;
  padding-left: 10px;
  padding-right: 10px;
  text-decoration: none;
}
```

Your links are now white with no underline.

## Details

### Asterisk

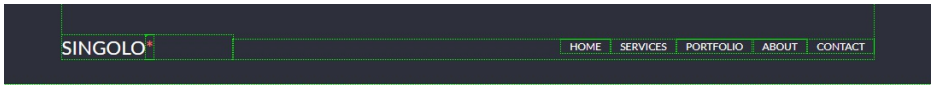
The red asterisk in the logo can be styled using a `span` element. You can think of `span`s as generic *inline-level* elements (the same way `div`s are generic block-level elements):

*index.html*

```
<div class="logo">
  SINGOLO<span class="salmon">*</span>
</div>
```

*styles.css*

```
.salmon {
  color: #F16C65;
}
```




### Bullets

The bullets between each link should be added using HTML entity codes. Your lecturer/tutor will explain more about reserved characters and why entities are necessary. You can find out more here:

https://developer.mozilla.org/en-US/docs/Glossary/Entity

For the bullets, use `&bull;`

*index.html*

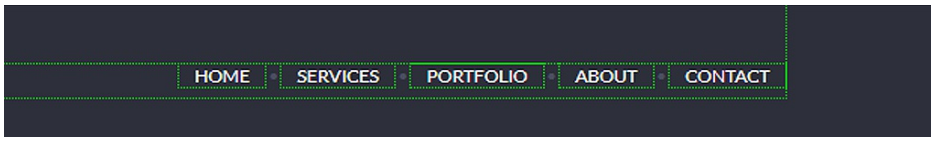
```
<div class="links">
  <a href="#">HOME</a>
  &bull;
  <a href="#">SERVICES</a>
  &bull;
  <a href="#">PORTFOLIO</a>
  &bull;
  <a href="#">ABOUT</a>
  &bull;
  <a href="#">CONTACT</a>
```

```
      </div>
```

To make the bullets pale blue-ish, add a `color` property to the `.links` selector. This takes advantage of the CSS cascade (the links will remain white because they are affected by a more specific selector):

*styles.css*

```
.links {
  margin-top: 40px;
  font-size: 0.75rem;
  text-align: right;
  color: #4B4E61;
}
```

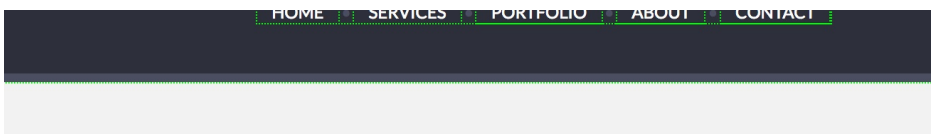



### Edge/Border

Along the bottom of the header is a (subtly) lighter border. Add this to the `.header` rule using a `border-bottom` property:

*styles.css*

```
...

.header {
  background-color: #2D2F3B;
  height: 90px;
  color: white;
  border-bottom: 6px solid #4B4E61;
}

...
```




### Banner

In Photoshop, crop the "singolo.psd" file so that you have just the banner image (the two phones and the arrows). Save it as a "banner.png" file in a new folder named "img". This images directory should sit alongside your "index.html" file.

Now undo the Photoshop crop step so that you can continue using the .psd file to prepare other elements.

As evidenced by the arrows in the banner image, this is intended to be a 'sliding' banner. However, to keep things simple (and avoid any JavaScript) you will use it as a single, static image; this also means it is not necessary to use a transparent background.

Create a new `div` *below* the header `div`. The image can be placed within it using an `img` tag:

*index.html*

```
  ...

  </div>

  <div class="banner">
    <img src="img/banner.png" />
  </div>

</body>
</html>
```

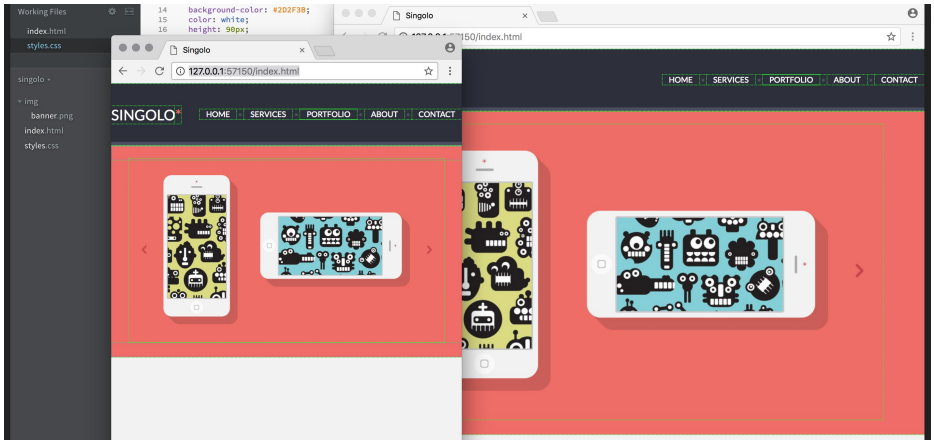
To complete the banner, add the necessary CSS. In this case, use a `max-width` to prevent the image from getting too large -- along with a `%`-based width value to respond to different screen sizes. Because images are inline-level elements, the banner can be centred using a `text-align`:

*styles.css*

```
...

.banner {
```

```
  background-color: #F16C65;
  padding: 20px 0;
  text-align: center;
}

.banner img {
  max-width: 780px;
  width: 90%;
}
```

You will now have a responsive banner image; test this by resizing your browser window:




**Note:** images will display at their exact pixel dimensions unless width and/or height properties are applied to them.

## Media Queries

Your lecturer will provide a brief introduction to media queries by applying this one to your web-page:

*styles.css*

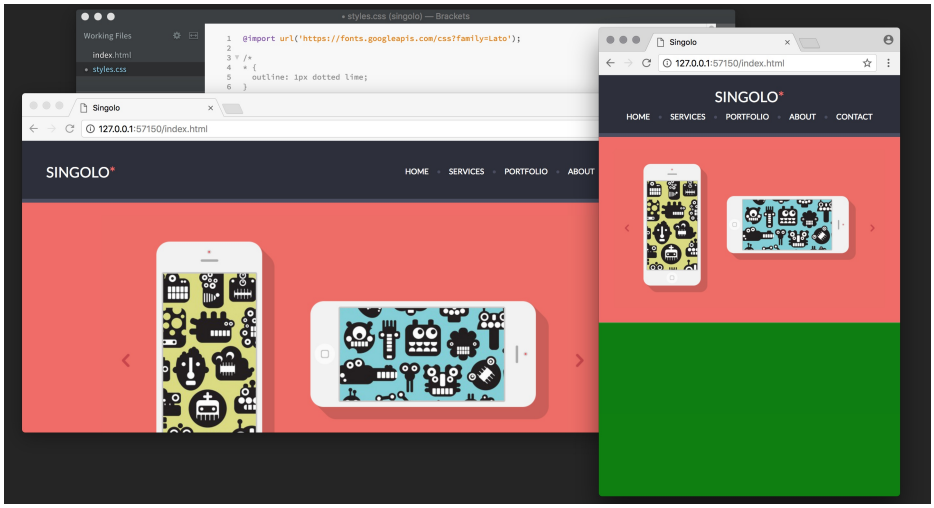
```
...

@media screen and (max-width: 480px) {
  body {
    background-color: green;
  }
}
```

When accommodating mobile devices, be sure to add this line to your `head` element to prevent the page zooming/scaling-everything-down to fit the device screen:

```
  ...
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
```

Test the result in your browser. If you scale the window down to a width of less than `480px`, the page background turns green. To have the navigation adapt to a centred arrangement for mobile, override the necessary `.logo` and `.links` properties by adding additional selectors to the media query:

```
...

@media screen and (max-width: 480px) {
  body {
    background-color: green;
  }

  .logo {
    grid-column: 1/3;
    margin-top: 20px;
    text-align: center;
  }

  .links {
    grid-column: 1/3;
    margin-top: 10px;
    text-align: center;
  }
}
```



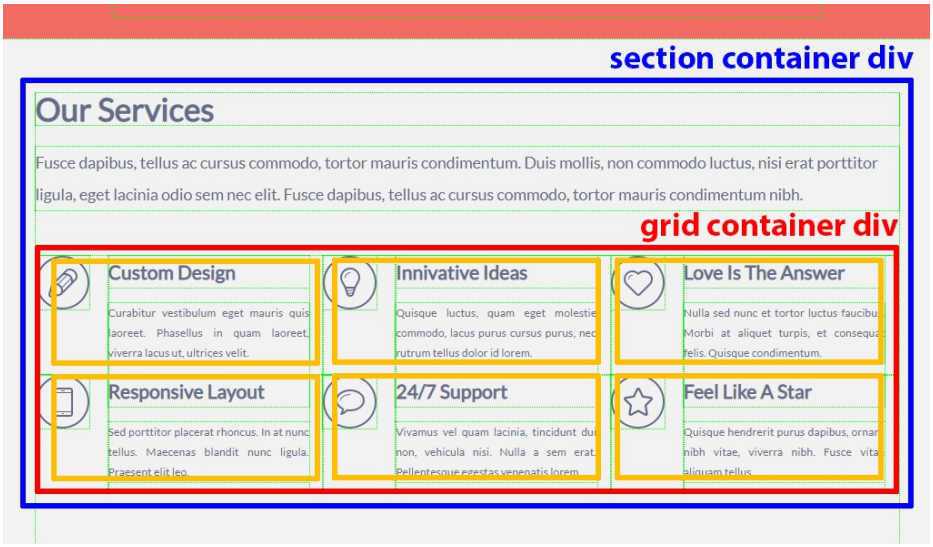
Usually, webpages are coded using a *mobile-first* approach -- meaning that the default CSS caters for mobile devices, and the media queries specify what happens for larger screens. In this case, we are working the other way around (default CSS targets desktop devices, and the media queries handle mobile). It's not important at this point in the course, though. **You will cover media queries and responsive coding in more detail in the latter part of this course. The purpose of mentioning them here is to provide advanced students with an extra challenge.**

If you wish to find out more on media queries, check-out Mozilla's developer documentation:

https://developer.mozilla.org/docs/Web/CSS/Media_Queries/Using_media_queries

## Finishing the Website

Complete the rest of the web-page. Note that it is best to begin by creating a grid for a given section, then refine and add content. Look at the first section and consider how you should tackle it in terms of `divs` and grids.




You will probably want to employ the `grid-template-rows` property in addition to `grid-template-columns`.

For the form elements in the bottom section of the page, refer to Mozilla's developer documentation:

https://developer.mozilla.org/Learn/HTML/Forms/Your_first_HTML_form

**Note** that any links need not actually link anywhere -- just use an `href="#"`. This includes the 'filter' buttons in the portfolio section (All, Web Design, Graphic Design, Artwork).

*end*