Mockup to HTML/CSS

In this handout, you will learn how to build an HTML page with CSS styling from a Figma mockup.

You will learn how to:

- figure out the sizes of page components (fonts, sections, headers, etc)
- find the spacing between elements
- build out a CSS layout based on a mockup
- get colour hex codes from a Figma mockup
- use your HTML/CSS knowledge to pull together a webpage

We'll be working off of the following mockup:
https://www.figma.com/file/s9Ulk4AWqhMSEMiFpEEOus/Week-6-mockup---full?node-id=0%3A1

The actual .fig file is uploaded to Blackboard as well. It's better to import the .fig file so that you get full access to everything.

For the initial page setup, just have an HTML file with blank boilerplate (i.e. the HTML basic stuff with nothing in <body>), an empty CSS file, and an ***/images*** folder. This is already available in the skeleton files on Blackboard.

Main techniques

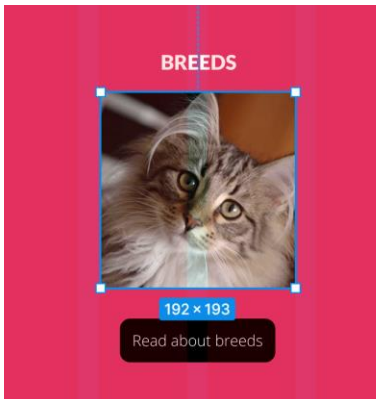
There are a few important techniques we'll be using, mainly:

- extracting images from a mockup
- determining the spacing between elements
- getting font sizes and other font styling
- extracting colour codes

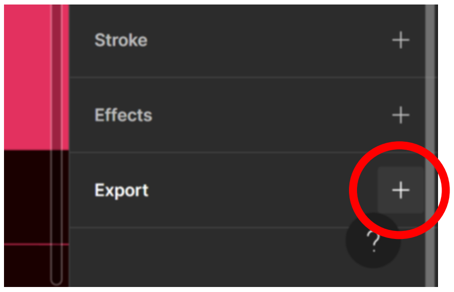
Extracting images from a Figma mockup

To extract images, you need to select the image layer then do an export.

1. Click on the image layer in the ***Layers*** panel **or**—with the ***Design*** panel on the right selected—click on an image in the mockup design. (If your image layer is in a group and you're clicking in the mockup design, you may need to double-click the image to select.)



2. With the image selected, scroll down to the bottom of the ***Design*** panel and click **+** next to ***Export***.

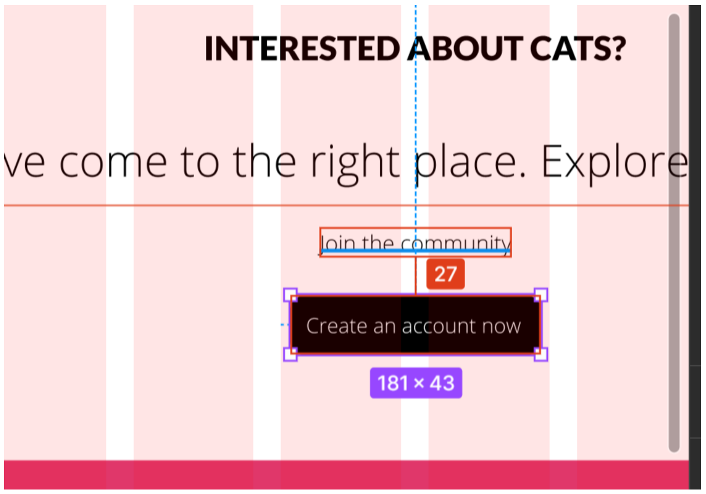


3. Click on the button to export the image. Note that you can make multiple sizes (e.g. 1x, 2x, etc.). This is for a responsive page and is not needed in our case because we haven't discussed responsive web design yet.

Determining spacing between elements

There is a very simple way to figure out how large the gaps are between elements in the design.

1. Select an item in the design (or select a layer).
2. With the ***Inspect*** panel selected, move your cursor to different elements in the design. You should see orange lines with the spacing/distance in pixels appear as you move over different elements.



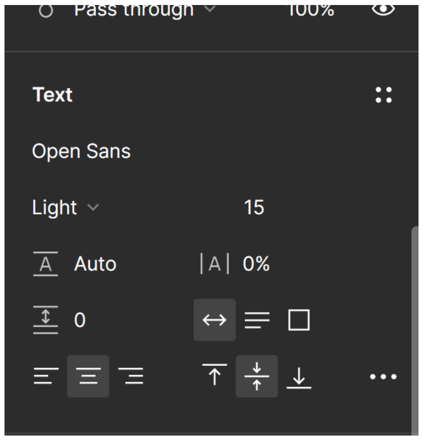
The keyboard shortcut (you **<u>don't</u>** need to select the ***Inspect*** panel first when using this shortcut):

1. Select an item (or layer) in the design.
2. While holding down the **<Alt>** key (Windows) or **<Option>** key (Mac), move the cursor over other items in the page. The orange lines with distance in pixels should appear.

Determining font settings

This is a simple task which you can do using the ***Design*** > ***Text*** panel.

1. Select the text you want to inspect. If the text is within a layer group or component, you may need to double-click twice to select the text. Alternatively, select the text layer in question in the ***Layers*** panel.
2. Scroll down to the ***Text*** section within the ***Design*** panel to view the font info.



Pass through 100%

Text

Open Sans

Light 15

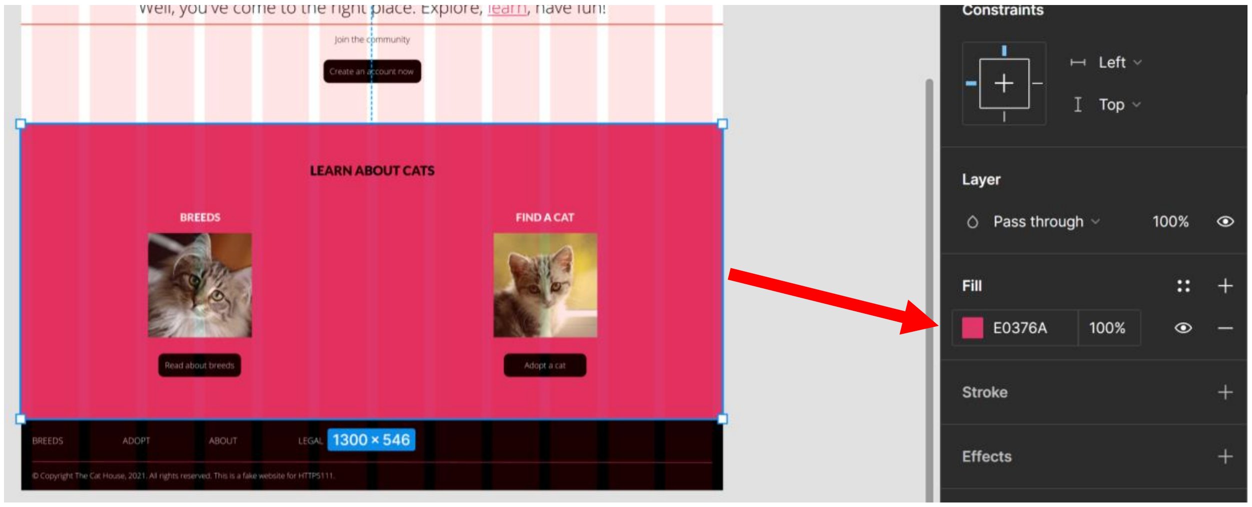
Auto 0%

0

Extracting colours

To get shape colours or text colours:

2. Select the item in the mockup design, or alternatively, select the layer in the ***Layers*** panel.
3. With the item selected, scroll down to the ***Fill*** section in the ***Design*** panel. The fill colour is provided as a hex number.



To get a border colour:

Borders are found under the ***Stroke*** section. If a shape has a border, the border settings will be found under ***Stroke***. For a line shape, the colour will also be found under ***Stroke***.

Identify the layout

The first thing to do before coding is to identify the layout parts. This is to help you organize your thoughts and approach.

- Header
  - Logo and site name
  - Navigation menu
- Main content area
  - Large banner/hero image and text
  - *"Interested about cats"* section
  - *"Learn about cats"* section
- Footer
  - Footer navigation
  - Copyright text

Coding the HTML/CSS

Typically, I would code the HTML first before the CSS so that the focus is purely on the semantics. In the steps below, we're taking it a section at a time so that you can see each part of the page added as we go.

1. Get the images from the mockup and save it in the ***/images*** folder (see steps on pages 1-2). These will be used as we code our page. You can give them the names:
   - *hero-black-cat.png*

   - *norwegian-cat.png*
   - *kitten.png*
2. Add the header with containers for the site name and menu.

```
<header id="header">
  <h2 id="logo-and-site-name">

  </h2>
  <nav id="main-navigation" aria-label="Main navigation">


  </nav>
</header>
```

3. Inspecting the mockup, the logo and site name are text, so we can type this directly in HTML. You can further wrap the logo to make it easier for different colouring. Note that the logo and site name are wrapped in a link.

```
<h2 id="logo-and-site-name">
  <a href="./">
    <span id="logo">(^._.^)</span>
    The Cat House
  </a>
</h2>
```

Note

Alternatively, if you don't want the symbols for the logo read aloud by screen readers, you can use an empty <span> and add in the CSS:

```
#logo::before {
  content:"(^._.^)";
}
```

4. For the <nav>, just structure like we've done before for menus.

```
<nav id="main-navigation" aria-label="Main navigation">
  <ul class="menu">
    <li><a href="./">Home</a></li>
    <li><a href="#">Cats</a></li>
    <li><a href="#">Shop</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```

5. Before we get to the styling for the header, let's take a look at some initial styling we need to set up. Take a look at the fonts used in the mockup. There look to be two fonts used: Lato black and Open Sans light. Take a look on Google Fonts to check if these fonts exist there. (They do.) Select Lato (Black style) and Open Sans (Light style) and copy the import statement into your CSS file at the top.

```
@import
url('https://fonts.googleapis.com/css2?family=Lato:wght@900&family=Open+Sa
ns:wght@300&display=swap');
```

   Notice that the "black" style for Lato corresponds to a weight number of 900 and the Open Sans light style corresponds to a weight of 300.
6. Set the box-sizing property to help prevent sizing issues when adding paddings and borders. In the CSS, add:

```
html {
  box-sizing:border-box;
}
*, *::before, *::after {
  box-sizing:inherit;
}
```

7. Looking at the overall design, it looks like the regular text is Open Sans, so this is the font to use as the default font-family set on html. Zooming in to 100%, the regular font size looks to be 16px (or, 1em). We can set this in the html rule:

```
html {
  box-sizing:border-box;
  font:300 16px "Open Sans",sans-serif;
}
```

8. Remove the default margin on body to remove the default gap.

```
body {
  margin:0;
}
```

9. Now, back to the header: the logo and site name as well as the main navigation are all Lato black (900 weight) and all uppercase. (See top of page 2 for steps for how to find text settings in Figma.)

```
#logo-and-site-name, #main-navigation {
  font-family:"Lato",sans-serif;
  font-weight:900;
  text-transform:uppercase;
}
```

Note

In the above CSS, the selectors were specific in that the #logo-and-site-name and #main-navigation elements were specifically selected. As these are the only things in #header, you could just use #header as the selector. If you anticipate there being other content added to the header which could possibly use Open Sans instead, stick with the selectors above.

10. Select the logo and site name text in Figma. Inspecting the ***Design*** > ***Text*** panel, the font sizes for the logo and site name are 32px. To convert to em, divide by the default font size, so 32px / 16px = 2em.

```
#logo-and-site-name {
  font-size:2em;
}
```

11. The #main-navigation text size 24px. Convert to em: 24px / 16px = 1.5em.

```
#main-navigation {
  font-size:1.5em;
}
```

12. Both menus are inline so we can use flexbox to just set .menu to display inline by default. We can remove default margin/padding (from the ul) and remove the bullet points as well.

```
.menu {
  display:flex;
  padding:0;
  margin:0;
  list-style:none;
}
```

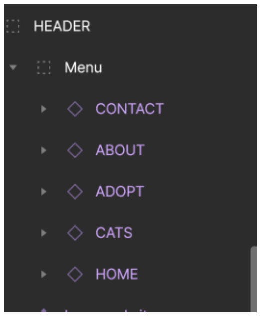
13. Using the technique on page 2 for determining spacing, the mockup shows a distance of 38.75px between menu links. Converting to em, this is 1.6em (38.75px / 24px) because the default font size for #main-navigation was changed to 24px in step 10 above.

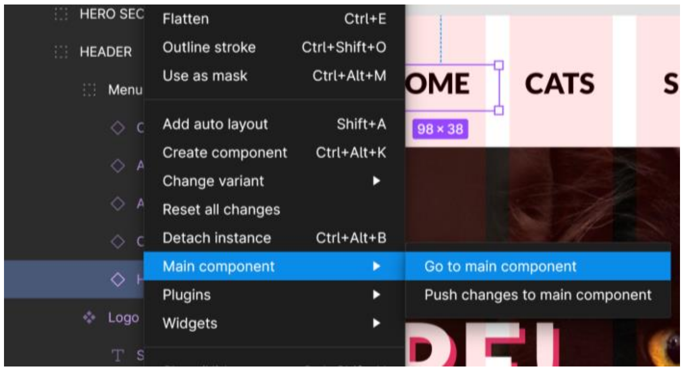
```
.menu {
  display:flex;
  padding:0;
  margin:0;
  list-style:none;
  gap:1.6em; /* or 38.75px */

}
```

14. Now for the link colours. Default link colours in the header are black with no underline.

```
#header a {
  color:#000;
  text-decoration:none;
}
```

15. Get the hover colours. In Figma, in the ***Layers*** panel, any instance of a component has a diamond symbol next to the layer name. You'll see that each menu link is an instance. To see if there's a variant, you can right-click and **Go to main component**. This will jump to the original component. From there you can get the hover style directly from the variant.





16. Set the hover colour for the menu links to #E0376A.

```
#main-navigation a:hover {
  color:#E0376A;
}
```

17. Now set the #logo to always be pink.

```
#logo {
  color:#E0376A;
}
```

18. Now we can get the site name and menu side-by-side. We can use flexbox by making #header the flex container.

```
#header {
  display:flex;
}
```

19. To align the flex items vertically, you can use the align-items property for the flex container and set it to center. We can also use the justify-content property to make the logo and site name and menu go left and right.

```
#header {
  display:flex;
  align-items:center;
  justify-content:space-between;
}
```

20. Now adjust the padding above the site name (and below to keep it even). In the mockup, the gap above the site name is 34px. To make calculations a little easier, remove the default margin on the site name and then set the padding (top and bottom) on the header. Since header has not had the font-size changed, we can determine padding using 34 / 16 = 2.125em.

```
#header {
  display:flex;
  justify-content:space-between;
  align-items:center;
  padding:2.125em 0;
}
…
#logo-and-site-name {
  font-size:2em;
  margin:0;
}
```

21. Now for the spacing. If the Figma mockup has a layout grid, that should aid your flex item sizing. If not, you can add one or come up with one (usually, desktop has 12 columns). To help figure out gutters, if there's no layout grid, look at the gaps on the sides of the page. If there's a consistent gap, set guides there and measure the distance. In this case, there's a gap of 20px on either side. (Usually, you can extrapolate from this and assume gutters of 20px for consistency.) The gaps on either side can help you determine a content wrapper (i.e. don't let the content exceed this width). I'm making this assumption because the black cat image has a set size so I don't want my content to exceed 1300px. According to the mockup with guides set at 20px from the left and right, all content falls within this range, so that gives a max-width of 1260px (1280px – 20px).

```
.content-wrapper {
  max-width:1260px;
  margin:0 auto;
}
```

22. Add the .content-wrapper class to #header to restrict content to not exceed 1260px width.

23. Before adding the hero text/banner stuff, notice that all headings are using Lato black and are all uppercase. We can add default styling for headings.

```
h1, h2, h3, h4, h5, h6 {
  font-family:"Lato",sans-serif;
  font-weight:900;
  text-transform:uppercase;
}
```

24. Add the HTML for the hero text/banner. If you consider the image as purely decorative, you can add the image as a background image on .hero-text. Use the .content-wrapper on the h1 to restrict the width of the h1.

```
<div class="hero-text">
  <div class="content-wrapper">
    <h1>Hello, there!</h1>
  </div>
</div>
```

> Alternative
>
> If the image *is* important content, then you'll need to include the image but overlay the text using absolute positioning.

25. Now, set the image as a background image, centered.

```
.hero-text {
  background:url("../images/hero-black-cat.png") center center;
}
```

26. Adjust the text size and color to white. To adjust to 96px, divide by 16px (96px / 16px = 6em).

```
.hero-text h1 {
  font-size:6em;
  color:#fff;
}
```

27. Adjust the padding for the h1 and add a text-shadow on the text to add a drop shadow. (The padding numbers are based on 96px, the font-size for h1. So a gap of 123px on the top is 123px / 96px = 1.28125em.)

```
.hero-text h1 {
  font-size:6em;
  color:#fff;
  padding:1.28125em 0.11458em;
  text-shadow:5px 5px 0 #E0376A;
}
```

28. Add the content for the join us section.

```
<section id="join-us-section">
  <h2>Interested about cats?</h2>
  <p class="feature-text">
    Well, you've come to the right place. Explore,
    <a href="#">learn</a>, have fun!
  </p>
  <p>Join the community</p>
  <a href="#">Create an account now</a>
</section>
```

29. Center all content in the join us section and adjust the font size for the feature text.

```
#join-us-section {
  text-align:center;
}
.feature-text {
  font-size:2em;
}
```

30. Finally, we need to style the links. For the link, running the prototype shows the link as static, but examining the components outside the frame, there is a component for regular links. Set rules for this. Since there's no specific styling for the "learn" link, we can assume it follows the default link styling.

```
a {
  color:#E0376A;
}
a:hover {
  text-decoration:none;
}
```

31. For the button link, using settings from the Figma mockup:

```
a.button, button {
  background-color:#000;
  color:#fff;
  font-size:0.9375em;
  padding:0.8em;
  text-decoration:none;
  border-radius:10px;
  margin:0.7333em 0; /* top padding is 27px - 16px margin from paragraph
*/
  min-width:153px;
}
```

32. Add the .button class to the link.
33. Now add the hover styling.

```
a.button:hover,
button:hover {
  background-color:#E0376A;
}
```

36. Adjust sections for consistent padding.
37. Now see if you can finish off the rest. For the two-column area, the content is centered in each column. The button hover styling is a little different from the hover styling for buttons on a light background. The white headings in the columns are also links.