Client Server and Web development

Chapter 2: Structure an entire page and spice your content with CSS

I. Introduction to HTML and CSS

In this chapter, you'll see the structure of a web page in HTML with some added color via CSS. Creating the overall structure of a page will give more context to the HTML elements you've already learned in part 2: text, lists, images, and more.

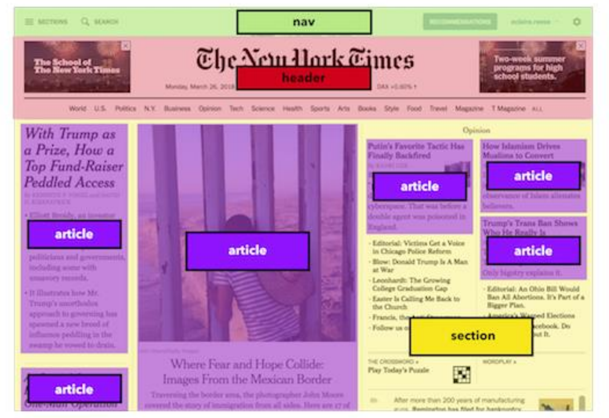
When you build a house, you start by building a structure first, and then filling in the smaller elements, right? The same is true when building a web page!

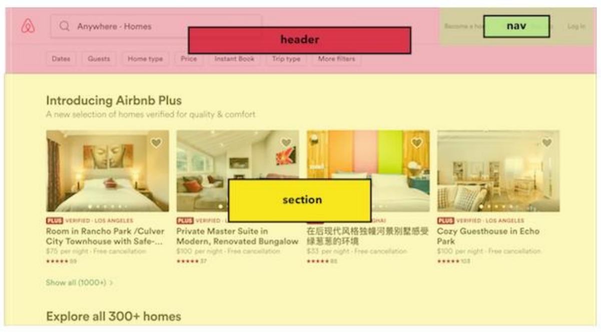
Web page structure

What do the New York Times, Wikipedia, Airbnb, and Lady Gaga's website have in common? *They can all be broken down into predictable blocks.*

I could actually name almost any website, and you could break down the site into remarkably predictable sections. *Developers write HTML in a way that permits consistent content structure across the entire web.* This reduces cognitive work for users and makes different sites display reliably across browsers, screen readers, and search engines.

Check out the following examples that I've annotated to show their different sections:



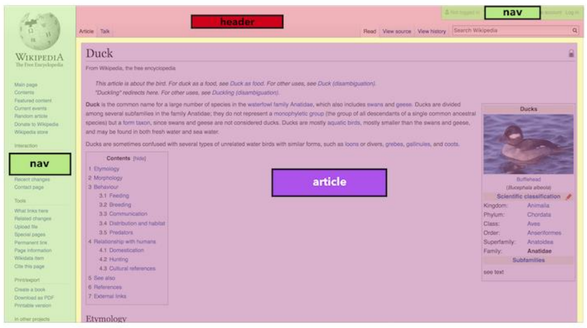


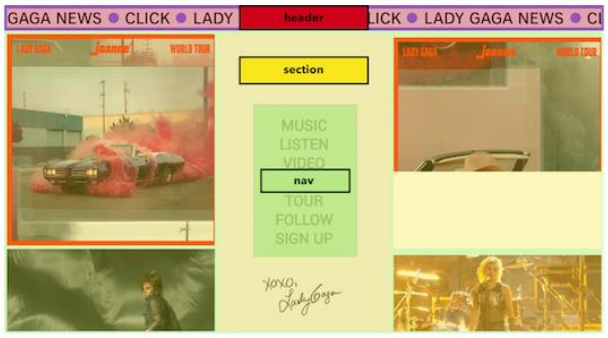
Airbnb page structure

New York Times page structure

We have four wildly different sites, but they all use the same structural elements, or zones: headers, navigation areas, sections, and articles.

For example, if you look at the New York Times, they use the nav for browsing sections, searching for an article, or accessing membership options and account settings. On Airbnb, the nav allows you to become a host or access your account settings. The navigation bars are suited to each site's content but are not fundamentally different.





Lady Gaga web page structure

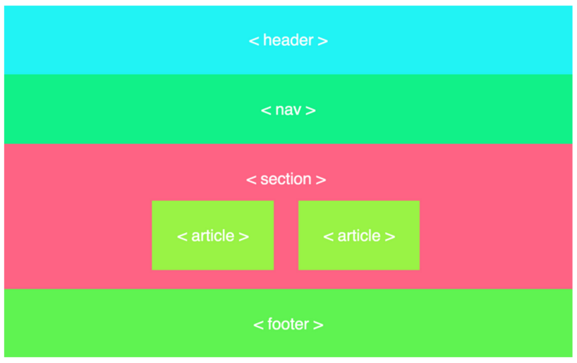
Wikipedia page structure

Additionally, websites often rearrange their content blocks when viewed on mobile devices and tablets. You'll often notice articles or smaller content pieces stacking vertically on smaller screen sizes.

Structural elements in HTML include:

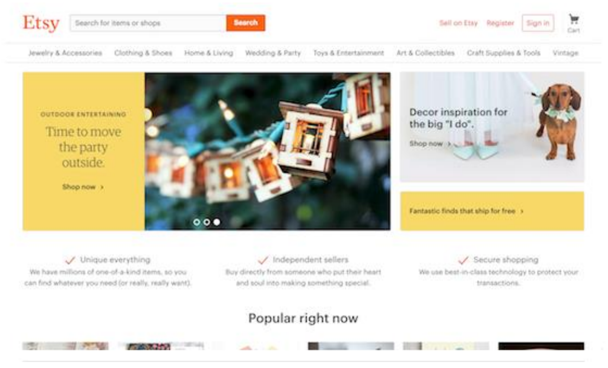
- <header>: the section at the top of a web page that often includes a logo and sometimes a nav
- <nav>: a set of menu items that allow a user to navigate to different pages on a site
- <section>: a general section of related content
- <article>: a piece of content that can be independently shared, like a blog post or newspaper article (even if you don't display the full article text, like you just show a preview, you can still use an article tag to delineate this content)
- <footer>: the section at the bottom of a page that often has additional links and perhaps social sharing icons
- <aside>: content that is complementary but not crucial to the page's main content (this could be an informative side section on a related subject)
- <figure>: a grouped image and caption that create an informative visual of some kind

Let's look at a *very* generic desktop page structure first:



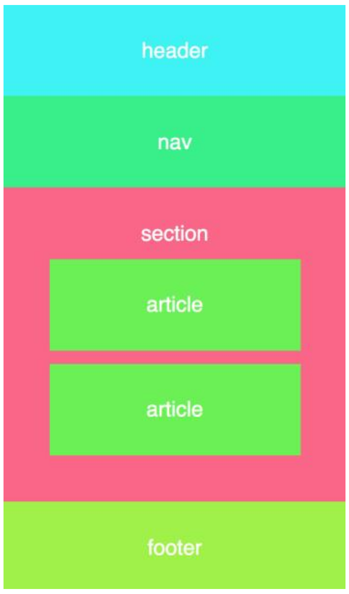
Possible desktop site structure

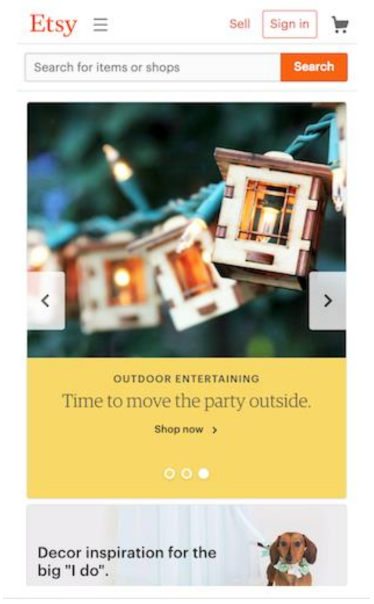
The page is pretty wide. Now let's look at a real-life example. You can see that on the website Etsy, content does indeed have a layout that spreads out from left to right:



Desktop view of Etsy (elements are laid out widely)

On the other hand, mobile views often "stack" elements vertically so they take up less horizontal space.





Possible mobile site structure

Mobile view of Etsy (elements are laid out widely)

This is the core of semantic HTML: your code should match your content! The sooner you start thinking in terms of content zones as presented above, the better your HTML will be and the more you can adapt your CSS to make your site look differently on desktop, mobile, and more.

Structural HTML tags

Earlier in this course, you saw that to create an HTML element (like a heading), you use a set of opening and closing tags. For example, the following tags will create a heading ("h1"):

<h1>Camping essentials</h1>

The format is exactly the same for creating structural elements like sections or footers. The following tags create the structural elements in the website examples above:

```
<header>
   <!--content-->
</header>
<nav>
   <!--content-->
</nav>
<section>
   <!--content-->
</section>
<article>
   <!--content-->
</article>
<footer>
   <!--content-->
</footer>
```

II. Understand block-level and inline elements

HTML elements have default behavior, via CSS behind the scenes, that controls their positioning.

Look at the behavior of the paragraph elements versus the image elements in this example:

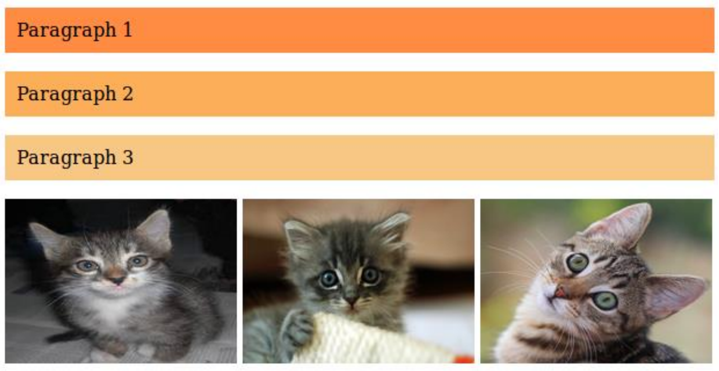
```
<html>

  <head>

    <link rel="stylesheet" href="css/style.css" type="text/css">

  </head>

  <body>

    <p id="first-paragraph">Paragraph 1</p>

    <p id="second-paragraph">Paragraph 2</p>

    <p id="third-paragraph">Paragraph 3</p>

<img src="images/kitten1.jpg" alt="Cute Lynx kitten" title="Lynx kitten" width="200px" height="142px">

<img src="images/kitten2.jpg" alt="Cute spotted kitten" title="Spotted kitten" width="200px" height="142px">
```

```
<img src="images/kitten3.jpg" alt="Cute Siamese kitten" title="Siamese kitten" width="200px"
height="142px">

  </body>

</html>
```

I added the paragraph colors via CSS, which can be done using the following code

```
#first-paragraph{
  background-color: #FF8C42;
  padding: 10px;
}
#second-paragraph{
  background-color: #FCAF58;
  padding: 10px;
}
#third-paragraph{
  background-color: #F9C784;
  padding: 10px;
}
```

Study the code and its result below. Can you identify which elements start on new lines by default (and which do not)?



Paragraph 1

Paragraph 2

Paragraph 3

Each paragraph expands to *take up the full width of the page on its own line* (greedy paragraphs)! However, the images all *appear in the same line*. That also explains why, earlier in this course, the two images of Millie Hughes-Fulford stacked right next to each other.

Every HTML element has a default block-level or inline behavior. Paragraphs are block-level elements, which means that they block off a whole line for themselves, and images are inline elements, which means they will automatically be placed next to one another on the same line.

Block-level elements that you've seen so far include:

- Headings
- Paragraphs (p)
- Lists and list items (ul, ol, li)
- Structuring elements (header, nav, section, article, aside, figure, footer)

Inline elements that you've seen so far include:

- Images (img)
- Emphasized text (em)
- Strong text (strong)
- Links (a)

As you code more and more, you'll memorize which HTML elements are block-level or inline. You'll also easily see their behavior every time you create them.

This concept is important in the next chapter on divs and spans, the ultimate generalist HTML elements (one of which is block-level and the other inline)!

III. Group content with divs and spans

Just a few years ago, you would have covered divs and spans much earlier in any course on HTML and CSS. Why? They are the most general, vanilla way to group content together — and used to be the only way to do so.

The difference between divs and spans is simple: one is block-level, and the other is inline.

Quick recap: block-level elements appear on their own lines by default, causing line breaks. Inline elements appear next to each other and do not cause line breaks.

```
<html>

  <head>

    <link rel="stylesheet" href="css/style.css" type="text/css">

  </head>

  <body>

    <p id="first-paragraph">Paragraph 1</p>

    <p id="second-paragraph">Paragraph 2</p>

    <p id="third-paragraph">Paragraph 3</p>

<img src="images/kitten1.jpg" alt="Cute Lynx kitten" title="Lynx kitten" width="200px" height="142px">

<img src="images/kitten2.jpg" alt="Cute spotted kitten" title="Spotted kitten" width="200px" height="142px">
```

```
<img src="images/kitten3.jpg" alt="Cute Siamese kitten" title="Siamese kitten" width="200px"
height="142px">

  </body>

</html>
```

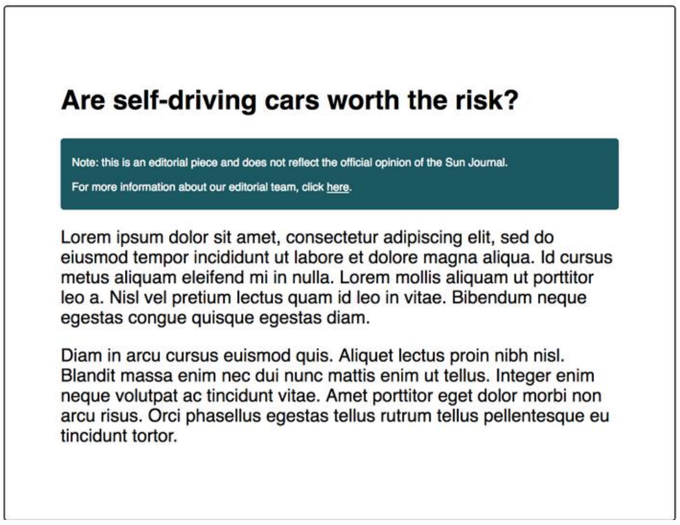
Paragraphs are block-level elements: they take up the full-width of the page on their own line. Images are inline elements: they line up next to each other and do not appear on their own lines.

Divs

Divs are a block-level element used to group content together, not dissimilar to <header>, <nav>, <section>, <article>, <footer>, <figure>, and <aside> which you've already seen in this course. There's one key difference, though.

Most of the time, when you want to group content together, you'll be able to do it using an HTML5 semantic element like header, nav, section, article, footer, figure, or aside. However, sometimes your content doesn't really fit any of these categories! In that case, you're encouraged to use divs to create block-level content groups.

Take the following web page screenshot:



Are self-driving cars worth the risk?

Note: this is an editorial piece and does not reflect the official opinion of the Sun Journal.

For more information about our editorial team, click here.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Id cursus metus aliquam eleifend mi in nulla. Lorem mollis aliquam ut porttitor leo a. Nisl vel pretium lectus quam id leo in vitae. Bibendum neque egestas congue quisque egestas diam.

Diam in arcu cursus euismod quis. Aliquet lectus proin nibh nisl. Blandit massa enim nec dui nunc mattis enim ut tellus. Integer enim neque volutpat ac tincidunt vitae. Amet porttitor eget dolor morbi non arcu risus. Orci phasellus egestas tellus rutrum tellus pellentesque eu tincidunt tortor.

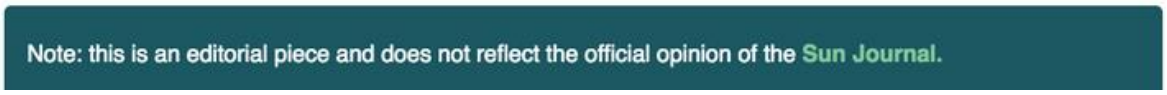
One heading and four paragraphs, two of which have a dark green background and small text

The dark green disclaimer block isn't really a full section, nor is it an article, footer, nav, etc. Nonetheless, its two paragraphs need to be grouped together so the block's appearance can be changed to have a dark green background and be visually different using CSS (you'll see this later in the course).

This is a legitimate reason to use a div to group the two paragraphs together.

Here's the code:

```
<article>

   <h1>Are self-driving cars worth the risk?</h1>

   <div>

      <p>Note: this is an editorial piece and does not reflect the official opinion of the Sun Journal.</p>

      <p>For more information about our editorial team, click <a href="#">here</a>.</p>

   </div>

   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Id cursus metus aliquam eleifend mi in nulla. Lorem mollis aliquam ut porttitor leo a. Nisl vel pretium lectus quam id leo in vitae. Bibendum neque egestas congue quisque egestas diam.</p>

   <p>Diam in arcu cursus euismod quis. Aliquet lectus proin nibh nisl. Blandit massa enim nec dui nunc mattis enim ut tellus. Integer enim neque volutpat ac tincidunt vitae. Amet porttitor eget dolor morbi non arcu risus. Orci phasellus egestas tellus rutrum tellus pellentesque eu tincidunt tortor.</p>

</article>

<p>Note: this is an editorial piece and does not reflect the official opinion of the <span>Sun
```


Note: this is an editorial piece and does not reflect the official opinion of the Sun Journal.

```
Journal.</span></p>
```

results

Do you see where we isolated the section we wanted using a <div>?

Spans

Spans are similar to divs, except they are inline elements. You might use them to group words together, since words appear inline and not each on their own line.

Imagine the newspaper above always prints its name in bolded light green. You need to grab those two words — "Sun Journal" — to be able to apply CSS to them. Why not wrap them in a <span>?

IV. Add classes and ids to elements

Class and id tags will revolutionize your HTML. They're absolutely necessary when your pages start becoming more complex.

Let's say you're working on a news website, which is likely to contain paragraphs and paragraphs of text. Some paragraphs will be within news articles, others might be disclaimer text at the bottom of the page, others will be short article blurbs; in sum, you have no way to tell these paragraphs apart from one another. Since they're all just <p> tags, you won't be able to apply CSS (appearance) changes to some paragraphs but not others.

This is where classes and ids enter the scene.

Classes and **ids** are custom attributes you can add to your elements in order to distinguish them from one another.

Classes apply to groups of elements, and ids apply to only one single element on an entire page. Let's check out classes first.

These examples will include basic CSS to help you visualize how ids and classes are used, but you're not expected to know CSS yet! Don't stress about it.

Classes

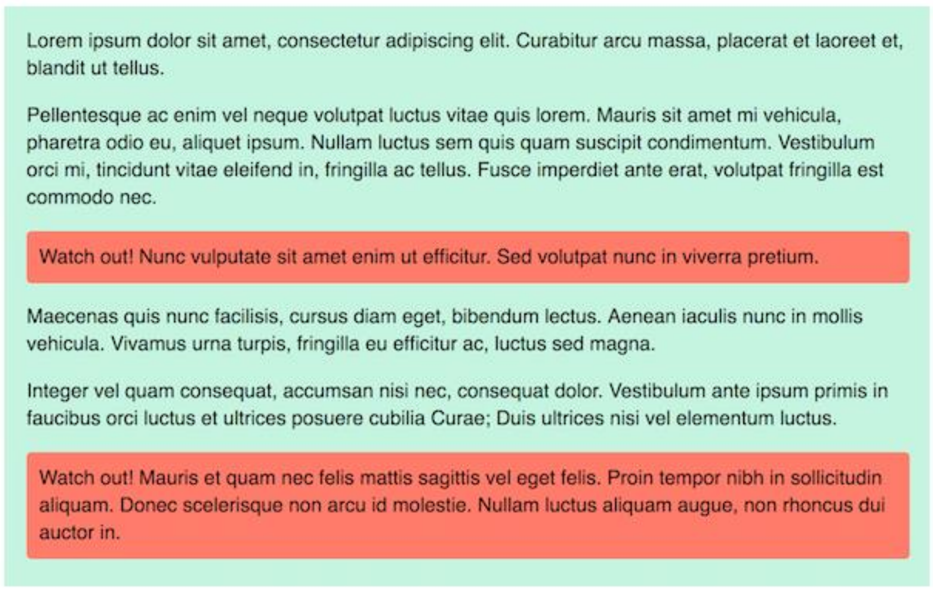
Setting a class attribute is as simple as choosing the class name and adding the attribute to your element, like:

```
<h1 class="breaking-news">Your heading here</h1>

<p class="warning">Your paragraph text here</p>

<p>More paragraph text is <span class="highlight">coming your way.</span></p>
```

In this very generic code example,

- The heading has a class of "breaking-news".
- The paragraph has a class of "warning".
- The span has a class of "highlight".

Classes like this allow you to apply custom CSS to certain elements but not others. Take this web page screenshot with filler text:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur arcu massa, placerat et laoreet et, blandit ut tellus.

Pellentesque ac enim vel neque volutpat luctus vitae quis lorem. Mauris sit amet mi vehicula, pharetra odio eu, aliquet ipsum. Nullam luctus sem quis quam suscipit condimentum. Vestibulum orci mi, tincidunt vitae eleifend in, fringilla ac tellus. Fusce imperdiet ante erat, volutpat fringilla est commodo nec.

Watch out! Nunc vulputate sit amet enim ut efficitur. Sed volutpat nunc in viverra pretium.

Maecenas quis nunc facilisis, cursus diam eget, bibendum lectus. Aenean iaculis nunc in mollis vehicula. Vivamus urna turpis, fringilla eu efficitur ac, luctus sed magna.

Integer vel quam consequat, accumsan nisi nec, consequat dolor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis ultrices nisi vel elementum luctus.

Watch out! Mauris et quam nec felis mattis sagittis vel eget felis. Proin tempor nibh in sollicitudin aliquam. Donec scelerisque non arcu id molestie. Nullam luctus aliquam augue, non rhoncus dui auctor in.

Two "warning" style paragraphs have a different background color than the others

The code behind it has 6 different paragraph tags, but 2 of them are orange to indicate a warning, or something the reader should watch out for. The way to accomplish this is to add a class of "warning" to the appropriate paragraphs:

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur arcu massa, placerat et laoreet et, blandit ut tellus.</p>

<p>Pellentesque ac enim vel neque volutpat luctus vitae quis lorem. Mauris sit amet mi vehicula, pharetra odio eu, aliquet ipsum. Nullam luctus sem quis quam suscipit condimentum. Vestibulum orci mi, tincidunt vitae eleifend in, fringilla ac tellus. Fusce imperdiet ante erat, volutpat fringilla est commodo nec. </p>

<p class="warning">Watch out! Nunc vulputate sit amet enim ut efficitur. Sed volutpat nunc in viverra pretium.</p>

<p>Maecenas quis nunc facilisis, cursus diam eget, bibendum lectus. Aenean iaculis nunc in mollis vehicula. Vivamus urna turpis, fringilla eu efficitur ac, luctus sed magna.</p>

<p>Integer vel quam consequat, accumsan nisi nec, consequat dolor. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Duis ultrices nisi vel elementum luctus.</p>

<p class="warning">Watch out! Mauris et quam nec felis mattis sagittis vel eget felis. Proin tempor nibh in sollicitudin aliquam. Donec scelerisque non arcu id molestie. Nullam luctus aliquam augue, non rhoncus dui auctor in.</p>

By adding class="warning" as an attribute to certain paragraph tags, you can grab the relevant paragraphs and update the background color to orange for specific paragraphs!

This scheme of adapting paragraph background colors might be familiar to you as a behavior on OpenClassrooms. You see how some paragraphs appear with blue backgrounds (like this one) or yellow backgrounds? That's because they have special classes!

It bears repeating one more time that classes can be applied to multiple elements on a page, unlike ids. Let's check them out.

ids

Ids are just like classes — they are custom attributes that can be added to elements — except they'll only apply to **one element per page.**

There were multiple warning paragraphs. On the same page, let's say we want to have one single key takeaway (added to the HTML above):

<p id="key-takeaway">Therefore, this is the one thing you should remember from the whole page.</p>

There will only be one key takeaway per page, so we're safe to use an id on this element instead of a class!

Now in my CSS, I can adapt that one element's appearance!

Naming classes and ids

I named the class warning and the id key-takeaway myself, but I could've given them ridiculous names like grizzly-bear-jazz-trio and taco-tuesday and they would still work, as long as I referenced the same names (grizzly-bear-jazz-trio and taco-tuesday) in my CSS.

Nonetheless, it's good to name your classes and ids understandably. Name them based on context or function, not on appearance.

For example, a class called "warning" could apply to scenarios where the content should have a warning function or context.

A really bad class name for the same example would be "orange-background-paragraph" because it describes the content's pure appearance and not its function or context. Besides, what if you decide warning text should be yellow someday instead of orange? You'd have to change your class name. Not cool.

Lesson: don't name classes and ids to describe an element's appearance. Name them to describe an element's function or context.

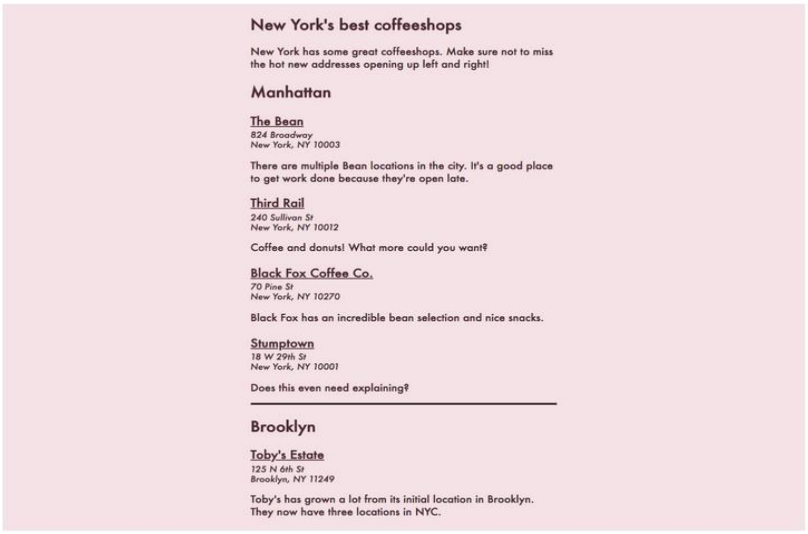
V.Add breaks and lines to your content

With classes and ids, we saw that it's possible to customize the appearance of certain elements.

There are easy ways to break up the structure of your page in order to separate themes or ideas without creating classes or ids.

You can use line breaks or horizontal rules (i.e., lines) to do so!

Let's say you want to code an article about New York's best coffeeshops by borough, and you want it to be structured as follows:



New York's best coffeeshops

New York has some great coffeeshops. Make sure not to miss the hot new addresses opening up left and right!

Manhattan

The Bean
824 Broadway
New York, NY 10003

There are multiple Bean locations in the city. It's a good place to get work done because they're open late.

Third Rail
240 Sullivan St
New York, NY 10012

Coffee and donuts! What more could you want?

Black Fox Coffee Co.
70 Pine St
New York, NY 10270

Black Fox has an incredible bean selection and nice snacks.

Stumptown
18 W 29th St
New York, NY 10001

Does this even need explaining?

Brooklyn

Toby's Estate
125 N 6th St
Brooklyn, NY 11249

Toby's has grown a lot from its initial location in Brooklyn. They now have three locations in NYC.

Web article about New York's coffeeshops

Pay attention to two things:

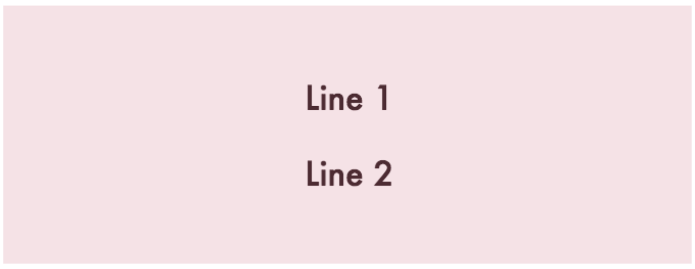
- The addresses under each coffeeshop name
- The line separating Manhattan listings from Brooklyn listings

This is what we'll cover in this chapter.

Line breaks

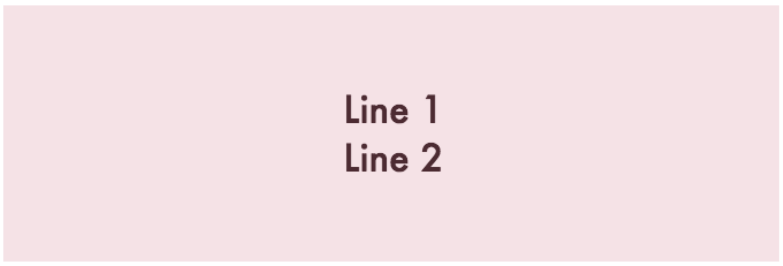
Paragraphs are block-level elements, meaning they take up the width of their containing element by default. They also have spacing above and below them. It's more than a simple line break.

See the space between "Line 1" and "Line 2" in the following image? That space is added automatically to the HTML element (by CSS):



Default line space that appears between paragraphs

Sometimes, you don't want that extra space, though. You'll want less space between each line:



Less space between two different lines

This is true especially for addresses and poems. You don't need an entirely new paragraph for each line in an address! A simple, and narrower, line break will do.

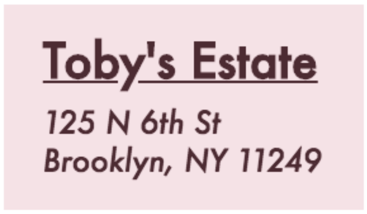
To create line breaks in HTML, use the <br> tag. There is no closing tag necessary.

```
<h3>Toby's Estate</h3>

<p class="address">

  125 N 6th St

  <br>
```

Brooklyn, NY 11249

</p>

In the code above, there will be a line break between "125 N 6th St" and "Brooklyn, NY 11249" that won't have the outrageous amount of space that appears between two paragraph elements. It'll just be a nice line break!

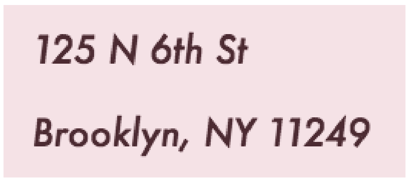


Toby's Estate

125 N 6th St
Brooklyn, NY 11249

Addresses with breaks to separate lines

This is what the address would look like as two separate <p> elements instead of one paragraph containing a line break



125 N 6th St

Brooklyn, NY 11249

Address with lines as separate paragraphs: weird appearance

It's a weird amount of space, right? It also doesn't make sense semantically because an address is one unit, so to separate it as multiple paragraphs isn't appropriate.

Don't add <br> tags everywhere just to create additional space on the page. Use them if a line break between two lines of text makes sense for your content (the lines of an address, for example, are always bunched closer together. Breaking them apart would make it harder for readers to understand). Otherwise, if the extra space you want is purely visual, create it using CSS. Remember that HTML tags are for your content, not its appearance.

Horizontal rules

You may have different content themes in one page that are related but are worth separating for clarity. In the above example, coffeeshops in Manhattan are separated from coffeeshops in Brooklyn because it helps group different content within the article.

To create a line, or a "horizontal rule" in HTML, simply use an <hr> tag.

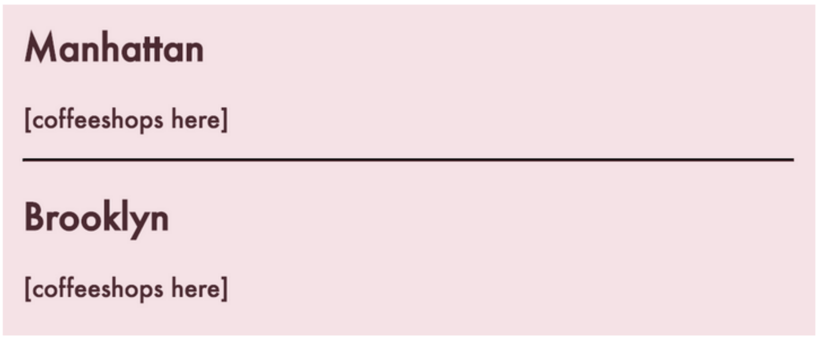
<h3>Manhattan</h3>

...

<hr>

...

```
<h3>Brooklyn</h3>
```



Manhattan

[coffeeshops here]

Brooklyn

[coffeeshops here]

Horizontal rule

Here's the full HTML code for the coffeeshop article (find those <br> and <hr> tags!):

```
<article>
  <h1>New York's best coffeeshops</h1>
  <p>New York has some great coffeeshops. Make sure not to miss the hot new addresses opening up left and right!</p>
  <h2>Manhattan</h2>
  <h3>The Bean</h3>
  <p class="address">
    824 Broadway<br>
    New York, NY 10003
  </p>
  <p>There are multiple Bean locations in the city. It's a good place to get work done because they're open late.</p>
  <h3>Third Rail</h3>
  <p class="address">
    240 Sullivan St<br>
    New York, NY 10012
  </p>
  <p>Coffee and donuts! What more could you want?</p>
  <h3>Black Fox Coffee Co.</h3>
  <p class="address">
    70 Pine St<br>
    New York, NY 10270
  </p>
  <p>Black Fox has an incredible bean selection and nice snacks.</p>
  <h3>Stumptown</h3>
  <p class="address">
    18 W 29th St<br>
    New York, NY 10001
  </p>
  <p>Does this even need explaining?</p>
  <hr>
  <h2>Brooklyn</h2>
  <h3>Toby's Estate</h3>
  <p class="address">
    125 N 6th St<br>
    Brooklyn, NY 11249
  </p>
  <p>Toby's has grown a lot from its initial location in Brooklyn. They now have three locations in NYC.</p>
</article>
```

VI. Trick out your text

In addition to the spacing options introduced in the previous chapter, there are other text-related properties to change your text's appearance.

We'll go through properties to make your text bold, italicized, underlined, or all in uppercase or lowercase.

font-weight

Adapting the weight of your font via CSS will cause your text to appear bolder or lighter than normal.

Hey! In part 2 of this course, didn't we learn that wrapping content in HTML <strong> tags is enough to make text bold?

True! However, remember that **HTML is semantic marku**p for your content, and **CSS controls its appearance**. <strong> elements are bold by default but should only be used when the content you're writing is emphatic. Sometimes, you'll want a heavier font weight just for the sake of appearances — not for emphatic content. Use the font-weight property instead!

In this chapter, we'll use a navigation bar as our example.

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">About</a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">Contact</a></li>
  </ul>
</nav>
```

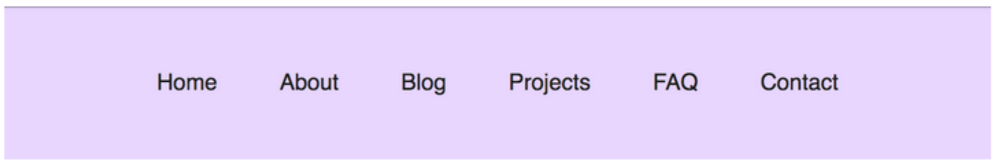
Navigation bars are sometimes coded as lists as HTML, sometimes not. A quick Google search will reveal many methods for creating navs.

font-weight can be set in several different ways. It can be:

- a word like "normal" or "bold"
- a numerical value like 400 (for normal weight), 700 (for bold), or a number in between

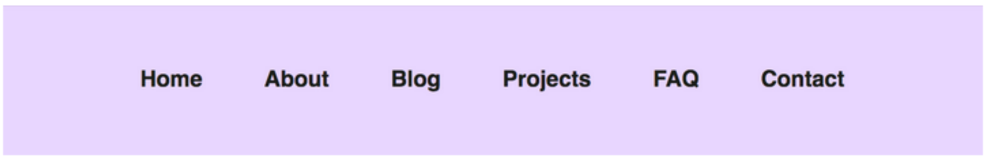
By setting a font-weight of normal, the font will appear at its default heaviness.

```
a {
  font-weight: normal;
}
```



By setting a font-weight of bold, the font will appear at its default boldness:

```
a {
  font-weight: bold;
}
```



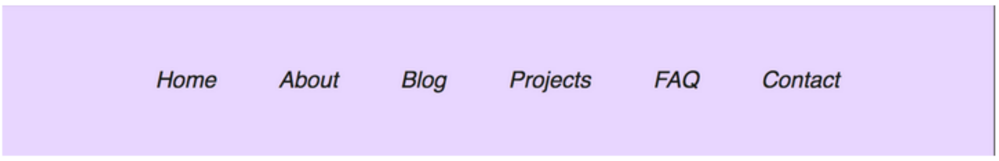
As mentioned above, you can also set a numerical value for your font weight (where 400=normal and 700=bold). You can also set a font-weight of less than 400, which often gives a nice result but depends on the font you're using:

```
a {
    font-weight: 200;
}
```

font-style

We can't talk about bold text without talking about italics! To italicize text, you'll use the font-style property. It can also be combined with a font-weight property if you like:

```
a {
   font-style: italic;
   font-weight: 200;
}
```



The font-style property has two other possible values: normal and oblique. Oblique text vaguely resembles italic text but takes up more horizontal space. You'll rarely use it.

text-decoration

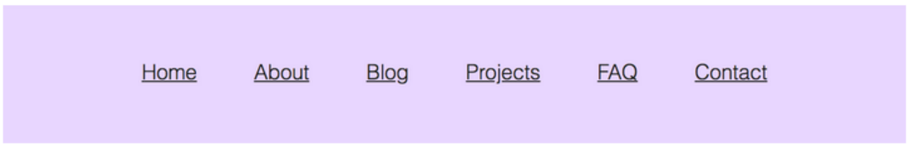
We've gone this whole course without talking about how to underline text. The <strong> HTML tag produces bold text by default, the <em> HTML tag produces italicized text by default, but no HTML element produces underlined text by default.

You must use CSS to accomplish this, no matter what. The text-decoration property will help you out.

It can take multiple values:

- underline
- none
- line-through

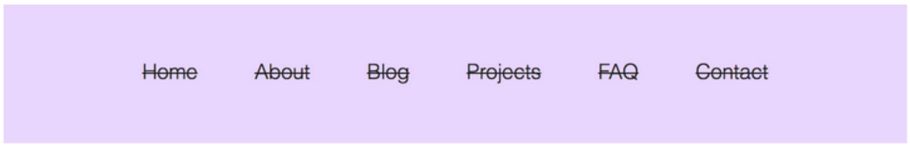
```
a {
  text-decoration: underline;
}
```



<a> elements are underlined by default, so you'll often find yourself setting text-decoration: none on them to get rid of an unwanted underline.

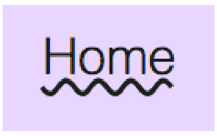
Draw a line through text by using a property called line-through (fancy that!):

```
a {
  text-decoration: line-through;
}
```



You can also add wavy or dashed effects to lines (text-decoration: underline wavy or underline dashed), but they look a little goofy. Use sparingly!

```
a {
  text-decoration: wavy underline;
}
```



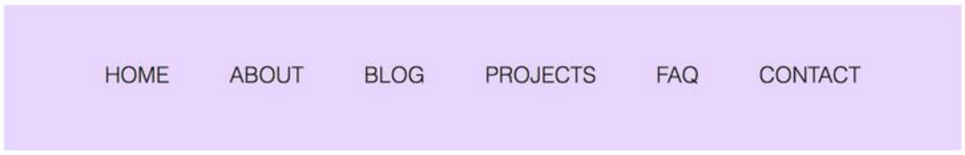
Another useful visual trick is to change all your letters to uppercase or lowercase. This can be accomplished without changing your HTML. Handle it in CSS using the text-transform property.

When used, the text-transform property will most often take one of the following values:

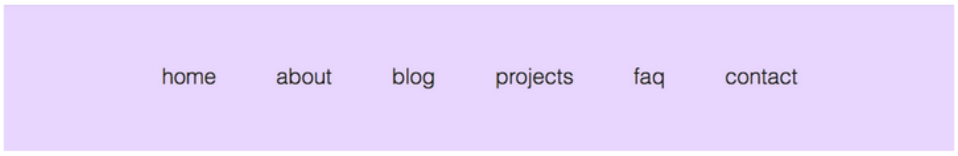
- capitalize
- uppercase
- lowercase

Creating letter-case harmony in navigation bars can be a pleasing visual effect:

```
a {
   font-weight: 200;
   font-family: Helvetica;
   text-decoration: none;
   text-transform: uppercase;
   color: #151814;
}
```



```
a {
   font-weight: 200;
   font-family: Helvetica;
   text-decoration: none;
   text-transform: lowercase;
   color: #151814;
}
```



Styles based on status

You'll sometimes want to apply styles when an element has a particular status, like when a link has been visited or when the user is hovering over an element with their mouse.

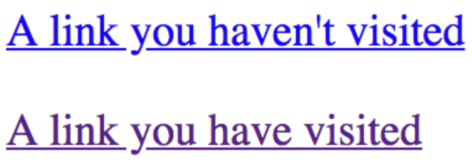
Enter CSS **pseudo-classes**. Despite the sci-fi-sounding name, their syntax is simple! Just add a colon and the name of the pseudo-class onto your selected element.

We'll cover only a few pseudo-classes in this chapter that are commonly applied to links:

- :visited
- :hover
- :active

visited

Web links are blue by default and purple when they've visited. You've probably seen these colors before:



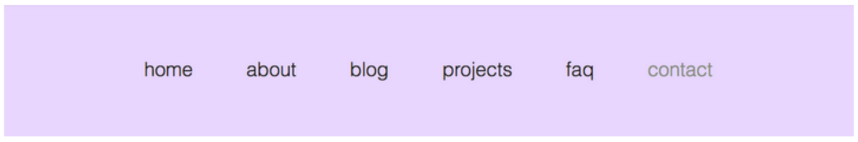
A link you haven't visited

A link you have visited

The generic blue and purple are not very awe-inspiring, and you'll often want to change them. You can change the blue by changing the color property on the a tag directly, but what about that bummer purple?

Let's apply a new visited link color to the navigation example above, where a user has already clicked on the "contact" link:

```
a:visited {
  color: #858C7B;
}
```



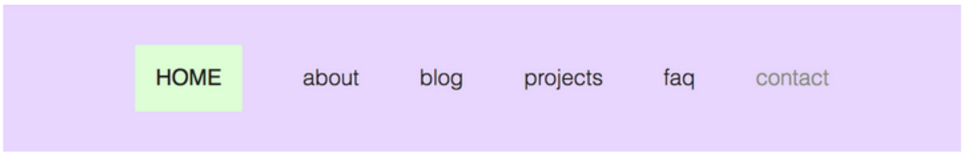
This isn't a dramatic visual effect but nonetheless helps users orient themselves around your site by making it clear which links they haven't clicked yet.

:hover

When a user hovers over elements like buttons or links, it can be helpful to have a visual change take place on the element. Use the pseudo-class "hover" to apply this change to any element!

In the following navigation bar example, when a user hovers over a link, its background color will change, and its text will be uppercase (the screenshot is from when I was hovering over the "Home" button):

```
a:hover {
   color: #151814;
   background-color: #DFFFD6;
   text-transform: uppercase;
   font-weight: normal;
}
```

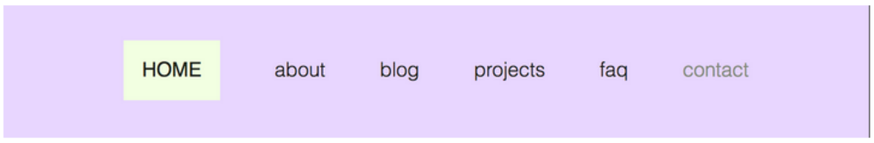


:active

When an element is clicked, it's nice to show a little visual effect to increase the sensation of interactivity. A light visual effect can be pleasant when an element is clicked. In this case, the clicked element is an "active" element only while the mouse button is pressing down on it.

When a link is clicked in our navigation example, I've made it so the background color changes from the light green in its hovered state to a light yellow in its active state:

```
a:active {
   background-color: #F3FFE1;
}
```



CSS rule: if you're using them, you should specify pseudo-classes in a particular order. That order is :visited, :hover, and then :active.