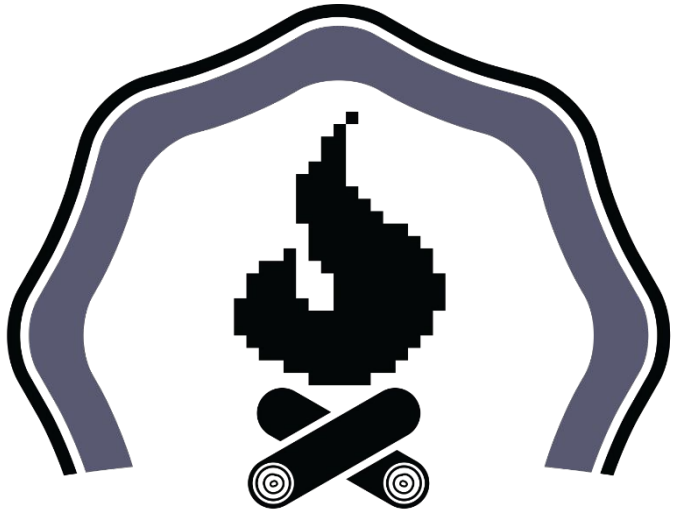


·CAMP TECH·

HTML & CSS
FOR BEGINNERS

CAMPTECH.CA

WHILE YOU'RE GETTING ORGANIZED…

1. Go to camptech.ca/html and download the PDF of slides and the .zip file. Save the .zip file to your desktop.
2. If you're on a Mac, double-click the .zip on your desktop to extract it. If you're on a Windows computer, right-click the .zip and then click Extract All. Leave the default options and click the Extract button.
3. Make sure you have Sublime Text installed. If not, you can download it at sublimetext.com.

HELLO MY NAME IS

Linn Øyen Farley

Web designer & developer
Hand-coding HTML & CSS since 2005

Enthusiastic about cats, IKEA, bacon-related cooking

@LinnOyenFarley
drollic.ca

THE PLAN FOR TODAY

1. What is a website? What is HTML & CSS?
2. How HTML & CSS work together
3. HTML in more detail
4. Build a one-page website (HTML)

   – lunch –

5. CSS in more detail
6. Build a one-page website (CSS)

Extra credit:

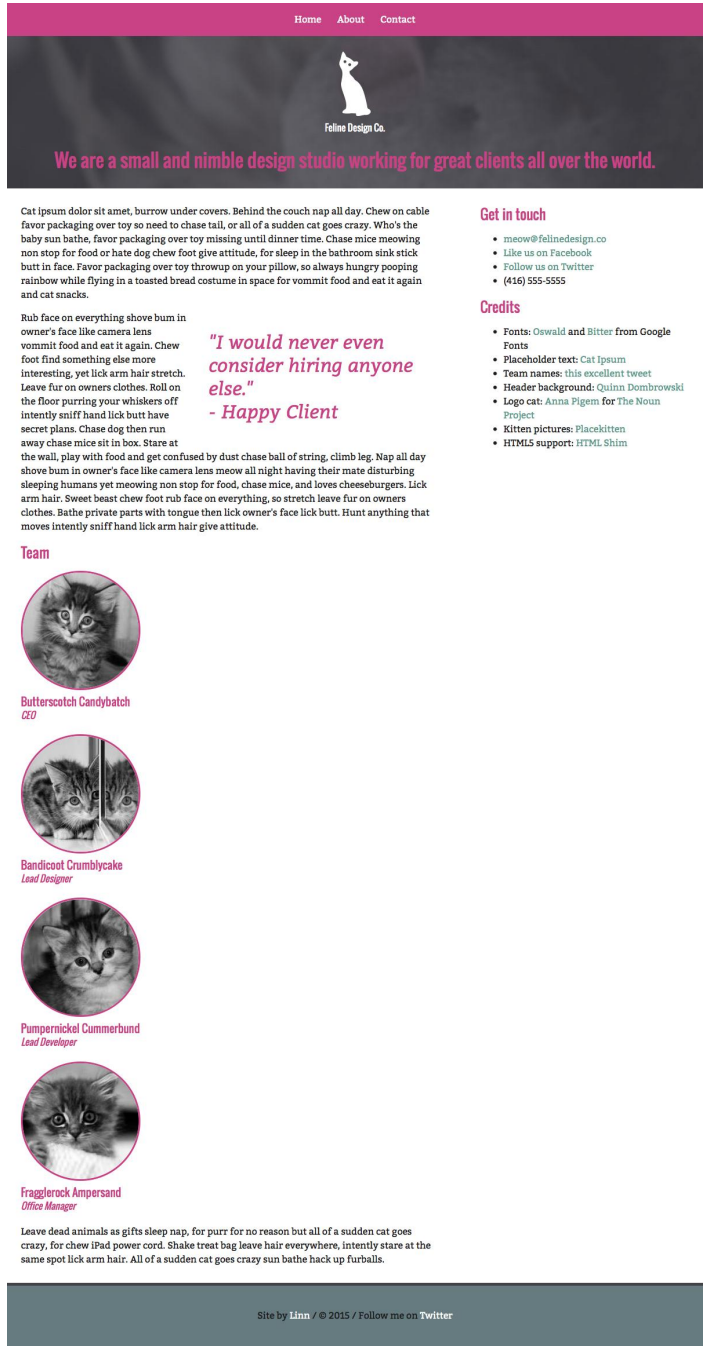
- Multiple pages
- Responsive design
- Supporting older browsers

WHAT WE'LL BE BUILDING

- A one-page website with:
  - A horizontal navigation menu
  - A header with a logo and tagline
  - A two-column content area
  - A footer

We're starting with a design and content that's ready to go.

Don't hesitate to ask for help if you're behind, or (equally valid) if you're jumping ahead!



THE ESSENTIALS

What is a website?

What is HTML & CSS?

WHAT IS A WEBSITE?

A website is a folder on a computer somewhere that contains website-specific files.

A one-page website can consist of a single file. A complex website can consist of lots of subfolders and hundreds of files.

WHAT IS HTML & CSS?

HTML and CSS are languages. They work together to present the majority of what you see on the internet.

To create HTML and CSS files, you just need a text editor. When you save the file, make sure it ends in either .html or .css to indicate which language it's written in. That's it!

Developers use more advanced code editors to make our jobs easier, because they have helpful features like colour coding and code completion.

Today we'll be using Sublime Text (sublimetext.com). Other options include:

- **Mac:**
  - TextEdit (in Plain Text mode)
  - Coda (panic.com/coda/)
- **Windows:**
  - Notepad (with word wrap turned off)
  - Notepad++ (notepad-plus-plus.org)

WHAT IS HTML & CSS?

HTML stands for HyperText Markup Language, because we use it to “mark up” content so the browser knows how to display it.

For example, you use HTML to tell the browser which parts of your content are paragraphs, which parts are lists, which parts are headings, and so on.

WHAT IS HTML & CSS?

CSS stands for Cascading Style Sheets, because we use it to apply styles to our HTML.

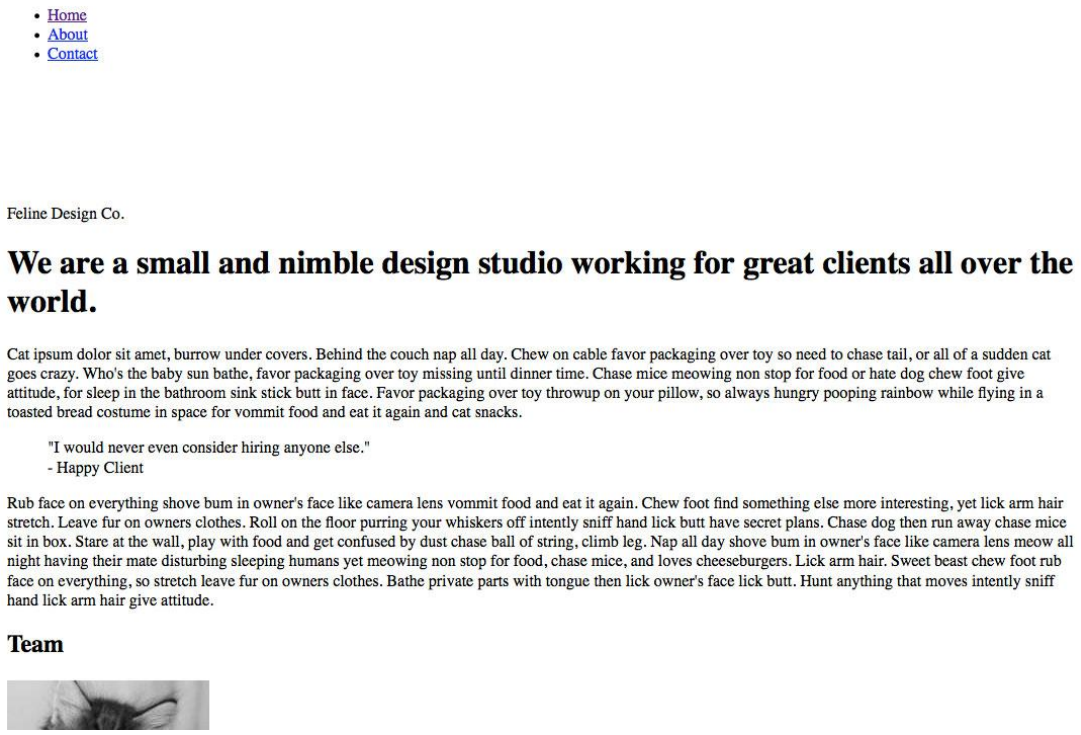
Without CSS, each browser applies its own (very basic) styles to various elements. For example, paragraph text is usually black Times New Roman at 16px size. And headings are just larger, bolded Times New Roman.

Using CSS, we can tell the browser to ignore these defaults, and instead show paragraphs (for example) in dark grey Georgia at 18px, and headings in purple Verdana.

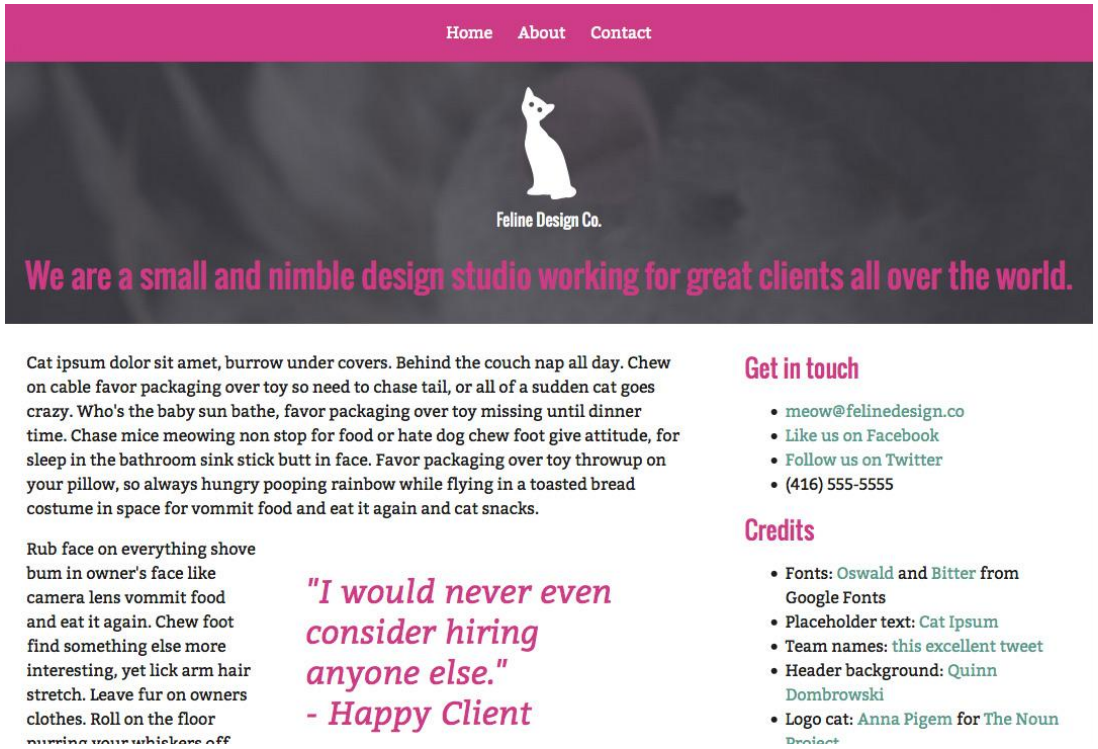
WHAT IS HTML & CSS?

You can have HTML without CSS, but you can't have CSS without HTML – it would just be a list of styles that aren't being applied to anything.

HTML (content) only:



HTML (content) and CSS (style):



RECAP:

A website is a folder of files.

HTML is for content.
CSS is for design and layout.

HOW HTML & CSS WORK TOGETHER

HTML & CSS basics

HTML BASICS

HTML looks like this:

```
<p>This is a paragraph.</p>
```

This is a tag pair. A tag is a specific keyword* between two angle/pointy brackets.

*Examples of keywords: `p` for paragraph, `img` for image, `h1` for heading level 1

HTML BASICS

Usually there is a matching closing tag (indicated by the forward slash before the keyword). Together, the opening and closing tags make a tag pair.

```
<p>This is a paragraph.</p>
```

This particular tag pair is for the HTML element paragraph. It tells the browser that everything between the opening `<p>` and closing `</p>` is a paragraph.

- **HTML tag**:
  - The pointy brackets and the keyword inside them
  - E.g. `<p>` or `</p>`
- **HTML element**:
  - The whole thing, content and all
  - E.g. `<p>This is a paragraph.</p>`

HTML BASICS

If there's no closing tag, the element is a standalone element. One example of a standalone element is a line break, which closes itself:

```
<br />
```

You don't put content inside a line break – you just want to go to a new line – so there's no need for a closing tag here.

HTML BASICS

There are a few special tags that need to be in every HTML document.

```
<!DOCTYPE html>
<html lang="en">
    <head>
    </head>
    <body>
    </body>
</html>
```

← tells the browser which version of HTML we're using

← opening tag (`lang="en"` sets language for screen readers)

← closing tag (everything between opening tag & closing tag is HTML)

HTML BASICS

```
<head>
</head>
```

Information *about* the page that you don't want to display *on* the page goes inside the `head` tags, such as:

- Search engine information
- Where to find CSS (we'll get to this soon!)
- The favicon (the little icon that appears in bookmarks and tabs)
- Scripts
- Analytics code

HTML BASICS

```
<body>
</body>
```

The content you want to *show* to your visitors goes inside the `body` tags.

HTML BASICS

You can nest HTML tags inside each other. Indenting helps keep track of matching opening and closing tags.

Code editors like Sublime Text will automatically indent nested tags for you.

This is easy to follow:

```
<!DOCTYPE html>
<html lang="en">
    <head>
    </head>
    <body>
        <p>Hello!</p>
    </body>
</html>
```

This is a little trickier:

```
<!DOCTYPE html>
<html lang="en">
<head>
</head>
<body>
<p>Hello!</p>
</body>
</html>
```

EXERCISE: BUILD A BASIC WEB PAGE

From now on, we'll be treating the (currently empty) folder called "Website" inside the "Camp-Tech-HTML-CSS" folder on your desktop as if it's the main folder on your hosting company's server.

To put the website you create today online, you'd need to upload the contents of this folder to the main folder of your hosting account.

EXERCISE: BUILD A BASIC WEB PAGE

Fire up Sublime Text and create a new file (File → New File).

Add the required tags and tag pairs:

```
<!DOCTYPE html>
<html lang="en">
    <head>
    </head>
    <body>
    </body>
</html>
```

EXERCISE: BUILD A BASIC WEB PAGE

Add two more pieces of info to your `head` section:

```
<meta charset="utf-8" />
```
← tells the browser which character set you're using

This doesn't need to display on the page, which is why it's in the `head` section.

```
<title>My 1st Website</title>
```
← symbols, spaces, uppercase and lowercase are all allowed

This appears at the top of your browser window, in search results, and in bookmarks.

Your `head` should look something like this:

```
<head>
   <meta charset="utf-8" />
   <title>My 1st Website</title>
</head>
```

EXERCISE: BUILD A BASIC WEB PAGE

After the opening `<body>` tag and before the closing `</body>` tag, type some headings, line breaks, and a few paragraphs of text:

```
Feline Design Co.
We are a small and nimble design studio working for great clients all over
the world.
Cat ipsum dolor sit amet, burrow under covers. Behind the couch nap all day.
Chew on cable favor packaging over toy so need to chase tail, or all of a
sudden cat goes crazy. Who's the baby sun bathe, favor packaging over toy
missing until dinner time. Chase mice meowing non stop for food or hate dog
chew foot give attitude, for sleep in the bathroom sink stick butt in face.
Favor packaging over toy throwup on your pillow, so always hungry pooping
rainbow while flying in a toasted bread costume in space for vomit food and
eat it again and cat snacks.
"I would never even consider hiring anyone else."
- Happy Client
```

EXERCISE: BUILD A BASIC WEB PAGE

Save the file (File → Save). *Remember to save it in the "Website" folder on your desktop*.

```
Save As: index.html
```

Rules for file names:

- Lowercase
- No spaces
- No special characters *except* hyphens and underscores
- End with a period and the type of file you're creating, e.g. `.html` or `.css`
- The homepage has to be named `index.html`

Once you've saved your `index.html` file, find it on your computer and open it in a browser.

EXERCISE: BUILD A BASIC WEB PAGE

It works! Kind of.

A couple things went wrong:

1. Our line breaks disappeared.
2. There's no way to tell what part is the title and what part is the main text.

Let's mark up our text a bit more with HTML tags.

EXERCISE: BUILD A BASIC WEB PAGE

```
<p>Feline Design Co.</p>
<h1>We are a small and nimble design studio working for great clients
all over the world.</h1>
<p>Cat ipsum dolor sit amet, burrow under covers. Behind the couch
nap all day. Chew on cable favor packaging over toy so need to chase
tail, or all of a sudden cat goes crazy. Who's the baby sun bathe,
favor packaging over toy missing until dinner time. Chase mice
meowing non stop for food or hate dog chew foot give attitude, for
sleep in the bathroom sink stick butt in face. Favor packaging over
toy throwup on your pillow, so always hungry pooping rainbow while
flying in a toasted bread costume in space for vomit food and eat it
again and cat snacks.</p>
<p>"I would never even consider hiring anyone else."<br />
- Happy Client</p>
```

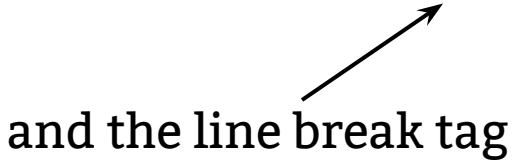
EXERCISE: BUILD A BASIC WEB PAGE

```
<p>Feline Design Co.</p>
<h1>We are a small and nimble design studio working for great clients
all over the world.</h1>
<p>Cat ipsum dolor sit amet, burrow under covers. Behind the couch
nap all day. Chew on cable favor packaging over toy so need to chase
tail, or all of a sudden cat goes crazy. Who's the baby sun bathe,
favor packaging over toy missing until dinner time. Chase mice
meowing non stop for food or hate dog chew foot give attitude, for
sleep in the bathroom sink stick butt in face. Favor packaging over
toy throwup on your pillow, so always hungry pooping rainbow while
flying in a toasted bread costume in space for vomit food and eat it
again and cat snacks.</p>
<p>"I would never even consider hiring anyone else."<br />
- Happy Client</p>
```

You're already familiar with the paragraph tag

EXERCISE: BUILD A BASIC WEB PAGE

```
<p>Feline Design Co.</p>
<h1>We are a small and nimble design studio working for great clients
all over the world.</h1>
<p>Cat ipsum dolor sit amet, burrow under covers. Behind the couch
nap all day. Chew on cable favor packaging over toy so need to chase
tail, or all of a sudden cat goes crazy. Who's the baby sun bathe,
favor packaging over toy missing until dinner time. Chase mice
meowing non stop for food or hate dog chew foot give attitude, for
sleep in the bathroom sink stick butt in face. Favor packaging over
toy throwup on your pillow, so always hungry pooping rainbow while
flying in a toasted bread costume in space for vomit food and eat it
again and cat snacks.</p>
<p>"I would never even consider hiring anyone else."<br />
- Happy Client</p>
```



EXERCISE: BUILD A BASIC WEB PAGE

```
<p>Feline Design Co.</p>
<h1>We are a small and nimble design studio working for great clients
all over the world.</h1>
<p>Cat ipsum dolor sit amet, burrow under covers. Behind the couch
nap all day. Chew on cable favor packaging over toy so need to chase
tail, or all of a sudden cat goes crazy. Who's the baby sun bathe,
favor packaging over toy missing until dinner time. Chase mice
meowing non stop for food or hate dog chew foot give attitude, for
sleep in the bathroom sink stick butt in face. Favor packaging over
toy throwup on your pillow, so always hungry pooping rainbow while
flying in a toasted bread costume in space for vomit food and eat it
again and cat snacks.</p>
<p>"I would never even consider hiring anyone else."<br />
- Happy Client</p>
```

This stands for Heading Level 1

There are 6 levels of headings in total: h1 is the most important, and h6 is the least important

EXERCISE: BUILD A BASIC WEB PAGE

Save your file and refresh it in your browser. That's a little better!

Now you're seeing the browser's default Times New Roman in action. Let's apply some CSS to make it our own.

CSS BASICS

CSS looks like this:

```
p { color: gray; }
```

The first part is a selector, which *selects* a specific HTML element to style – in this case, the paragraph.

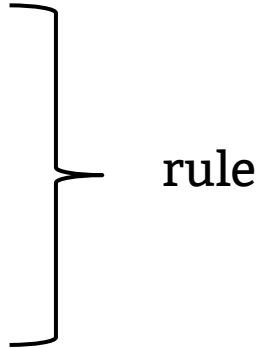
Then, inside the two curly braces, you add styles to apply to the selector. These styles are called declarations, or property/value pairs.

In the above example, the property is `color`, and the value is `gray`. This tells the browser to display all paragraphs in the colour grey (note the American spelling).

CSS BASICS

You can have several declarations applied to a single selector. The whole thing is called a rule. Rules are easier to read if you put each declaration on its own line:

```
p {                          ← selector and opening brace
    color: gray;             ← property + value = declaration
    font-family: Georgia;    ← property + value = declaration
    font-size: 16px;         ← property + value = declaration
}                            ← closing brace
```



EXERCISE: APPLY CSS TO YOUR WEB PAGE

Before we get into more properties, let's apply these paragraph styles to our page.

Create a new file in Sublime Text (File → New File). Unlike HTML, you don't need to add any default tags here – you can get started writing your CSS right away.

Type in these paragraph styles:

```
p {
    color: gray;
    font-family: Georgia;
    font-size: 16px;
}
```

Then save your file in the "Website" folder on your desktop.

```
Save As: styles.css
```

EXERCISE: APPLY CSS TO YOUR WEB PAGE

The final step is telling our HTML document where to find our new styles.

In `index.html`, after the opening `<head>` tag and before the closing `</head>` tag, type:

```
<link rel="stylesheet" href="styles.css" />
```

This is a self-closing HTML tag. It's slightly more complicated than the HTML tags we've seen so far, but it has the same basic structure:

1. Opening pointy bracket
2. Keyword (in this case it's `link`)
3. Some additional info
4. Closing slash and pointy bracket

EXERCISE: APPLY CSS TO YOUR WEB PAGE

```
<link rel="stylesheet" href="styles.css" />
```

↑ the *relationship* between this file & the one you're linking to

↑ *hypertext reference*/location of the file*

*Hrefs usually have to start with `http://`, but because `index.html` and `styles.css` are in the same folder, it's sufficient to just specify the file name. (We'll talk more about links later on.)

The `link` tag is going in the `head` section because we don't actually want our visitors to see it on the page. We just want the browser to know where to go to get the styles.

EXERCISE: APPLY CSS TO YOUR WEB PAGE

Your `head` section should now look something like this:

```
<head>
    <meta charset="utf-8" />
    <title>My 1st Website</title>
    <link rel="stylesheet" href="styles.css" />
</head>
```

EXERCISE: APPLY CSS TO YOUR WEB PAGE

Save your `index.html` file and refresh it in your browser.

Your paragraphs should be grey, slightly larger than they were before, and in Georgia instead of Times New Roman.

EXERCISE: APPLY CSS TO YOUR WEB PAGE

Add one more rule to your stylesheet:

```
h1, h2, h3, h4, h5, h6 {
    color: purple;
    font-family: Verdana;
}
```

EXERCISE: APPLY CSS TO YOUR WEB PAGE

```
h1, h2, h3, h4, h5, h6 {     ← select multiple things at once
   color: purple;
   font-family: Verdana;
}
```

To apply the same styles to multiple selectors without repeating yourself, select them all in a single rule and separate the selectors with commas.

Note that using `h1 h2 h3 h4 h5 h6` (without commas) as the selector here would not work, because it would make the browser look for a heading level 6 *inside* a heading level 5 *inside* a heading level 4 *inside*... you get the idea.

Add this rule to your CSS file, save it, and refresh your `index.html` page in your browser. Your heading(s) should now be purple Verdana.

CONGRATULATIONS!

You just made a website. Time to give yourself a high five!

RECAP:

< HTML = pointy brackets >

{ CSS = curly brackets }

HTML IN MORE DETAIL

HTML elements for content

ATTRIBUTES & VALUES

As we dive a little deeper into different types of HTML elements, keep in mind that CSS styles can be applied to any of these elements. We'll come back to CSS later on.

Basic HTML elements like `<p></p>` and `<br />` just need to give the browser a keyword. Some need to give more info, which is where attributes and values come in.

HTML elements can have several attributes and values, but their syntax is always:

[attribute] + [equal sign] + [double quotes] + [value] + [double quotes]

e.g.

```
src="http://camptech.ca"

alt="Cup of coffee"
  ↑           ↑
attribute    value
```

IMAGES

The image element is self-closing, and it always has at least two attributes:

```
<img src="images/headshot.jpg" alt="Photo of the author" />
```

↑ the *source* or location of this image

↑ the *alternate text* describing what this image depicts

↑ self-closing tag

The alternate text will appear if the image can't be displayed, *or* if your site is being accessed by a device that can't "see" (assistive technology reading your site aloud, or a search engine robot).

IMAGES

Create a new folder called “images” inside your “Website” folder. Then copy the image called `01.jpg` in the “Assets” folder, and paste it into “images”.

Now you can reference the image in `index.html` like this:

```
<img src="images/01.jpg" alt=" Butterscotch Candybatch" />
```

IMAGES

Because the file name in the `src` attribute has to be *exactly* the same as the file name on your computer (including whether the letters are upper or lowercase), it's a good idea to save images for your website using only lowercase letters in the file name, and no spaces or special characters.

The file type also has to be exactly the same as your image file. For example, these are all considered *different* file types by HTML:

- `.jpg`
- `.JPG`
- `.jpeg`
- `.JPEG`

If the image isn't showing up, double-check your file name and file type.

LINKS

A link always has opening and closing tags, and at least one attribute (`href`):

```
<a href="http://camptech.ca">Visit Camp Tech</a>
```

- `<a` — opening tag
- `href="http://camptech.ca"` — where to go when the link is clicked
- `Visit Camp Tech` — text that should appear *on* the page, as a link
- `</a>` — closing tag

LINKS

The `href` value should always start with http:// or https://, *unless* the link is pointing to somewhere else on your website:

```
<a href="myotherpage.html">My Other Page</a>
```

or if it's an email address link:

```
<a href="mailto:my@email.com">my@email.com</a>
```

When clicked, a `mailto` link opens the default mail program on your computer with the address pre-filled. This isn't always ideal for your visitor, so use them with care.

LINKS

If you need the link to open in a new window, include the `target` attribute and give it the value `_blank`:

```
<a href="http://drollic.ca" target="_blank">Linn</a>
```

Use this very sparingly, if at all. It's not recommended for accessibility reasons, because a visitor accessing your website using a screen reader or other assistive technology could get "lost" if a link unexpectedly opens in a new window and disables the Back button.

Also, most people know how to open a link in a new tab or window if they want that behaviour, but using `target="_blank"` forces the decision for them (there's no way to *stop* a link with that attribute and value from opening in a new window).

ORDERED (NUMBERED) LISTS

Ordered lists consist of a single `<ol>` tag pair with a series of `<li>` tag pairs inside:

HTML:

```
<p>Steps to success:</p>
<ol>
   <li>Get blanket</li>
   <li>Add sleeves</li>
   <li>???</li>
   <li>PROFIT</li>
</ol>
```

Result:

Steps to success:

1. Get blanket
2. Add sleeves
3. ???
4. PROFIT

UNORDERED (BULLETED) LISTS

Unordered lists consist of a single `<ul>` tag pair with a series of `<li>` tag pairs inside:

HTML:

```
<p>Things I like:</p>
<ul>
    <li>Cats</li>
    <li>Coffee</li>
    <li>To-do lists</li>
    <li>Walking</li>
</ul>
```

Result:

Things I like:

- Cats
- Coffee
- To-do lists
- Walking

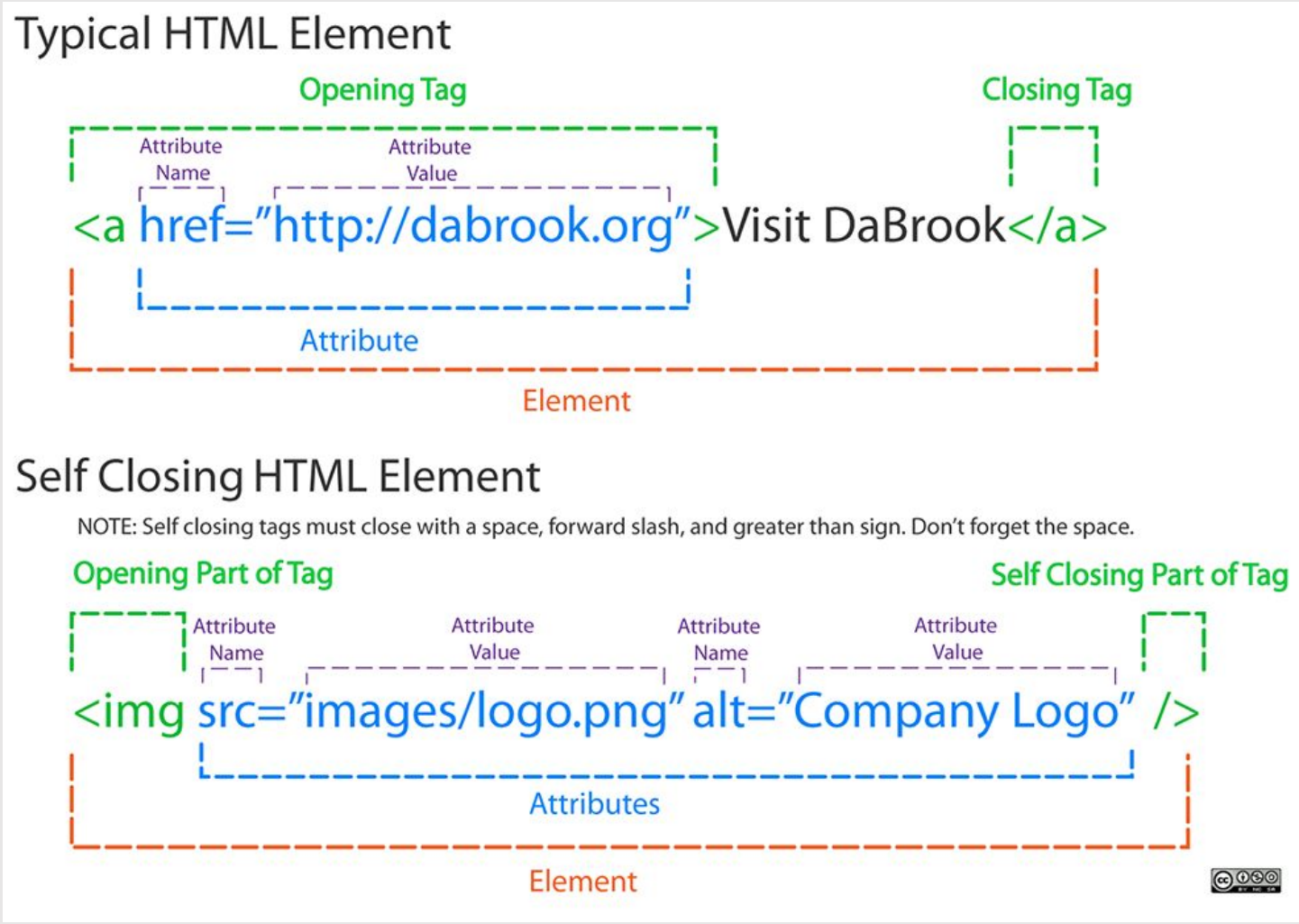
UNORDERED (BULLETED) LISTS

Unordered lists can also be used for navigation menus: as vertical menus in sidebars, and as horizontal menus with no bullets in sight (removed with CSS).

This afternoon we'll create a horizontal navigation menu using an unordered list, but for now, let's create a regular vertical list in `index.html`:

```
<ul>
    <li>meow@felinedesign.co</li>
    <li>Like us on Facebook</li>
    <li>Follow us on Twitter</li>
    <li>(416) 555-5555</li>
</ul>
```

RECAP: HTML SYNTAX



Source: dabrook.org/resources/posters/

HTML IN MORE DETAIL

Getting more specific with classes

CLASSES

What if you want to style one *specific* paragraph differently with CSS?

Give that paragraph a class! A class is an additional attribute we add to our HTML elements, which we can later select in our CSS.

CLASSES

If this is our HTML:

```
<p>I'm a paragraph!</p>
<p>Me too!</p>
```

And this is our CSS:

```
p {
    color: green;
}
```

The output on the page will look like this:

I'm a paragraph!
Me too!

CLASSES

But if we give one of our paragraphs a class:

```
<p class="special">I'm a paragraph!</p>
<p>Me too!</p>
```

And change our CSS to select that class instead:

```
.special {
   color: green;
}
```

Then our page will look like this:

I'm a paragraph!

Me too!

NAMING CLASSES

1. Only use letters, numbers, hyphens, and underscores (no spaces or other symbols)
2. Avoid random names that will confuse you when you see them in your CSS
3. Name your classes based on their *function*, not the *style* you're planning to use

Good examples:

- `class="profile-picture"` (for styling profile pictures differently)
- `class="social-icon"` (for giving social icons a special background or border)
- `class="pull-quote"` (for setting pull quotes apart from normal text)

Bad examples:

- `class="section2"` (unclear what this is used for)
- `class="blue"` (what if you later change the style to purple?)

CLASSES vs IDs

You might come across IDs in addition to, or instead of, classes.

They're used in almost exactly the same way, except IDs are referenced in your CSS with a pound sign (#) instead of a period (.). Also, IDs are only used once per page, whereas classes can be used multiple times on a page.

HTML:

```
<p id="special">A paragraph</p>
<p>Another paragraph</p>
```

CSS:

```
#special {
    color: green;
}
```

Result:

A paragraph

Another paragraph

We'll only be using classes today.

ADD A CLASS

Target an element in your `index.html` file, and style it using your `styles.css` file.

HTML:

```
<p class="pull-quote">Nice things said about me<br />
- Happy Client</p>
<p>My totally normal paragraph.</p>
```

CSS:

```
.pull-quote {
   color: purple;
}
```

Result:

Nice things said about me
- Happy Client
My totally normal paragraph.

RECAP:

Only add classes to your HTML if you're going to use CSS.
Name your classes so they'll make sense to you in future.

HTML IN MORE DETAIL

HTML elements for structure

HTML ELEMENTS FOR STRUCTURE

The HTML we've covered so far is great for marking up our content so we can style it with CSS, but where HTML and CSS really shine is for website layouts.

On our current page:

- everything is 100% width in the browser
- each piece of content shows up below the one that came before it
- we have to select one element at a time with classes, or all elements at once

We can use HTML and CSS to:

- add columns
- create areas with different borders and background colours/images
- group together and style complete sections at once

HTML ELEMENTS FOR STRUCTURE

We achieve this by wrapping each section of the page in a special HTML element in preparation for styling. Today we'll be using these elements:

- `<header></header>` *
- `<nav></nav>`
- `<section></section>`
- `<aside></aside>`
- `<footer></footer>`

You can also add classes to these elements if you need to get more specific.

As a bonus, they're helpful for screen readers and search engines that are trying to find things on your site.

*__Potential confusion alert__: `<header>` (the top section of your page) is *not* the same as `<head>` (the hidden area that contains info about your website)!

HTML ELEMENTS FOR STRUCTURE

On their own, these structural elements do nothing visually on your page.

If we're not going to use CSS, we don't need to add structural elements. We're *only* putting them in our HTML now so we can target them later with CSS.

HTML ELEMENTS FOR STRUCTURE

Here's one example of why you would use a structural element.

The area at the top of your site contains your logo (`img`) and business name (`p`). Say you want the background to be blue. You could use CSS to separately style both elements:

...but that doesn't look so good. The logo and the business name belong together, so the whole area around *and* between them should have the same style. That's where a structural element like `header` comes in.

HTML ELEMENTS FOR STRUCTURE

By wrapping the logo *and* the business name in `<header>`:

we can style the whole area at once, instead of each element individually:

RECAP:

Structural elements surround sections of content for styling.
They also help robots understand the structure of your page.

SIDENOTE: COMMENTS

In both HTML and CSS you can leave a note for yourself, or for future developers working with your code. Browsers will ignore anything inside a comment, so these will only be visible when viewing the code itself; they won't show up on the page at all.

In HTML:

```
<!-- comment text goes here -->
```

In CSS:

```
/* comment text goes here */
```

Examples of HTML comments:

```
<!-- end header area -->
<!-- Analytics code -->
<!-- remove this later! -->
```

Examples of CSS comments:

```
/* Start typography styles */
/* to-do: fix this part */
/* Blue: rgb(204,229,234) */
```

BUILD A ONE-PAGE WEBSITE

HTML

ADD YOUR CONTENT

Let's turn our basic website into a fully functional single-page website, starting with the HTML.

Make sure to include all of the following HTML elements:

- `<header>` containing your logo and tagline
- `<nav>` containing a `<ul>` with navigation items as `<li>`s
- A `<section>` with a class of `main` for your primary content, containing a `<p>` with a class of `pull-quote`
- `<aside>` for a sidebar, containing `<ul>`s with contact info and site credits
- `<footer>` containing social media links and copyright info
- A few headings (`<h1>`, `<h2>` etc), `<p>`s, and `<img>`s

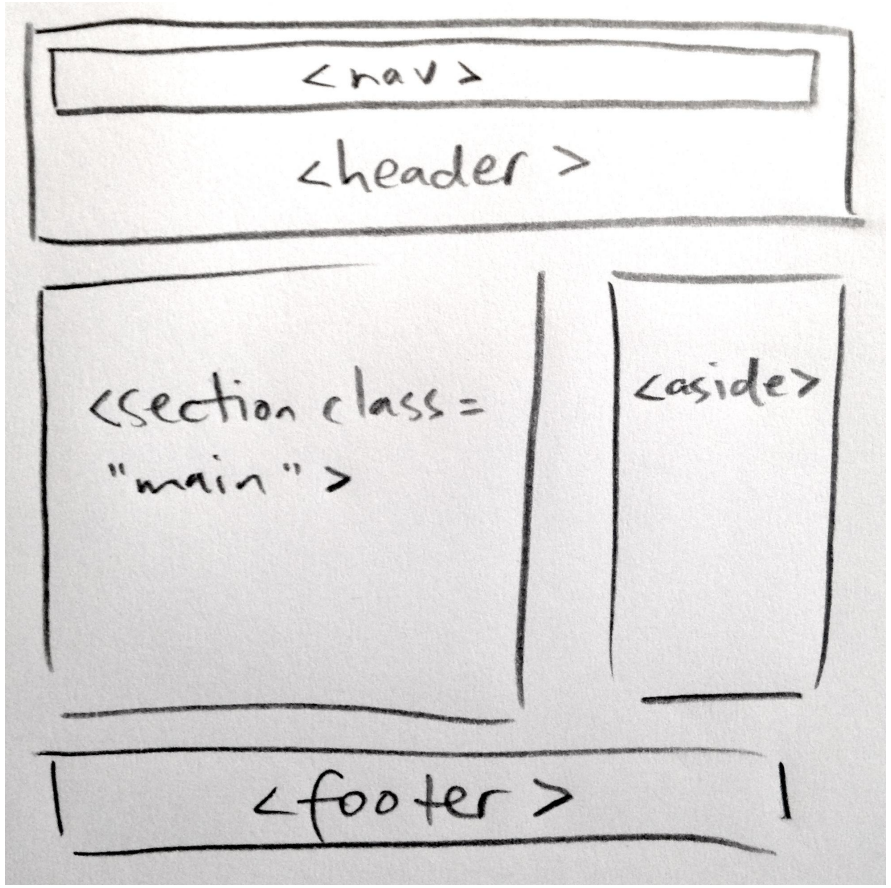
See the HTML reference guide for more elements you can use.

ADD YOUR CONTENT

This is the stage of a project where web designers often draw everything out on paper.

Here's an example →

It's not going to win any design awards, but having a sketch like this handy while you're coding a site can be super helpful.



ADD YOUR CONTENT

Note that all of the structural elements you're adding (`header`, `nav`, `aside`, `footer`) won't visibly change your page at all; we'll need to use CSS to create a layout with columns and different styles for different sections. Right now we're just marking up our content so we can target it in our CSS later on.

Keep your `index.html` file open in a browser, and refresh it as you work to see your HTML in action. It looks pretty plain, but we'll be adding lots of style with CSS this afternoon.

Make sure to save your file as you go!

STARTER CONTENT

If you need a jumpstart, you can copy the code from the `starter-content.html` file. Make sure to open it in Sublime Text (not in your browser).

CSS IN MORE DETAIL

Selectors & common properties

CSS SELECTORS

Here's a refresher on what CSS looks like:

```
p { color: gray; }
```

The selector can be:

- An element, e.g. `a`, `p`, `h2`, `img`, `header`
- A class name preceded by a period, e.g. `.profile-picture`
- A child element, e.g. `p a { color: yellow; }`

This selector will only apply to `<a>` tags that are *nested inside* `<p>` tags, like this:

```
<p>It's a <a href="http://camptech.ca">link</a>!</p>
```

CSS SELECTORS

In your `styles.css` file, write a child selector. To do this, you'll first have to have nested HTML tags in your `index.html` file.

For example:

```
header p {
    color: red;
}
```

Any styles inside these curly brackets will only apply to paragraphs (`p` elements) that are *inside* the `header` element. Paragraphs elsewhere on the page won't be affected.

CSS HIERARCHY

They're called *Cascading* Style Sheets because rules further down in your CSS will override rules that came before.

One way to think of this is that "the last thing you heard is the one you'll remember".

For example, if your CSS looks like this:

```
p { color: red; }
p { color: blue; }
p { color: green; }
```

your **paragraphs** will be **green**, because that was the last rule in the sheet.

If a rule doesn't seem to be working, look further down in your stylesheet for an existing rule that's overriding the one you just wrote.

CSS COLOUR

You can define colour in a few different formats:

- Colour name: e.g. `red`, `black`, `coral`, `lime` ... the full list is at w3schools.com/cssref/css_colornames.asp but there are only 140 (mostly weird) colours
- Hex value: e.g. `#FF1493`
- RGB value: e.g. `rgb(30,144,255)`
- RGBA value: e.g. `rgba(30,144,255,0.5)` where the final value is opacity

You can get these values from Photoshop, or one of these websites:

- Colour Lovers (palettes and trends): colourlovers.com
- HTML Color Codes (a fun palette and colour picker tool): htmlcolorcodes.com
- Adobe Color (interactive colour wheel): color.adobe.com

Related tool: Contrast Check (for accessibility): snook.ca/technical/colour_contrast/colour.html

COMMON CSS PROPERTIES

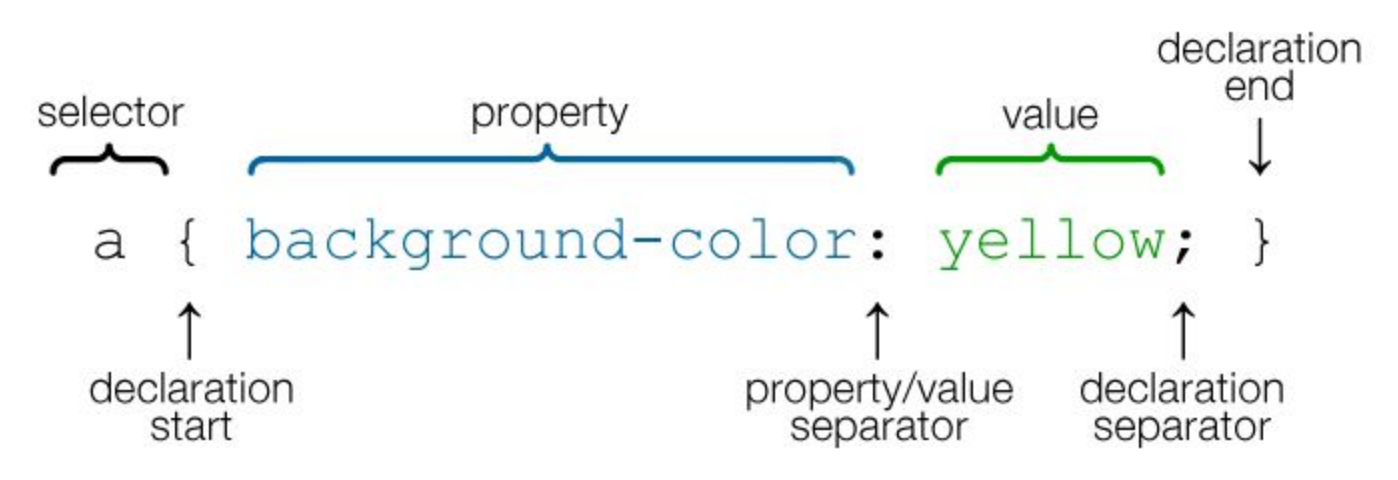
These are the CSS properties I use most often:

```
background
border
color
font-family, font-size, font-weight
line-height
text-align, text-decoration, text-transform
margin
padding
height, width
```

We'll try out each of these properties together as we style our page.

For reference, the full list of CSS properties is here: w3schools.com/cssref/default.asp

RECAP: CSS SYNTAX



BUILD A ONE-PAGE WEBSITE

CSS

RESET

To start with a blank slate, first “reset” all of the browser’s default styles.

Copy and paste the following at the very top of your `styles.css` file:

```
/* CSS Reset */
html, body, a, em, img, strong, header, nav, section, aside, footer {
    background: transparent;
    border: 0;
    margin: 0;
    padding: 0;
}
```

When you write your own styles below this rule, they will override it.

This is a short version of a standard reset; some are much longer. It selects almost every HTML element we might use, and makes sure the browser doesn’t apply any backgrounds, borders, margins, or padding to them.

COLOUR PALETTE

It's a good idea to have your colour palette ready before you write any code, so you're not making up values as you go, or constantly referring back to Photoshop.

For this website design, I pulled colours from the original version of the header image.

Add the palette to the top of your CSS file as a comment, so you can refer to it later:

```
/* Colour palette:
    Dark grey (regular text) - rgb(30,30,30)
    Teal (inactive links) - rgb(77,128,116)
    Grey (active links, footer background) - rgb(99,121,125)
    Medium grey (header background, footer border) - rgb(68,67,73)
    Fuchsia (nav background, borders, headings) - rgb(204,63,134)
*/
```

CSS BACKGROUND PROPERTIES

Property: `background-color`
Value: hex, RGB, or RGBa colour, e.g.

- `#47455E` **(hex)**
- `rgb(71,69,94)` **(RGB)**
- `rgba(71,69,94,0.5)` **(RGB with 50% transparency)**

Property: `background-image`
Value: URL to image file, either local or external, e.g.

- `url(/path/to/images/my-image.jpg)` **(local)**
- `url(http://mywebsite.com/my-image.jpg)` **(external)**

Even if you set a background image, you should also set a background colour that's similar, just in case the image loads slowly or not at all.

CSS BACKGROUND PROPERTIES

Add a background colour and image to your `header` element.

First, copy the file called `bg.jpg` from the "Assets" folder and put it in your "images" folder, so you can reference it using this URL: `images/bg.jpg`

Then add this rule to your stylesheet:

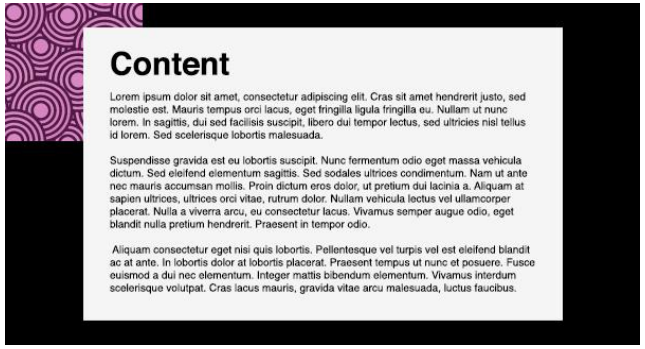
```
header {                                    ← selects the entire header area
    background-color: rgb(68,67,73);        ← makes the background grey
    background-image: url(images/bg.jpg);   ← adds the background image from the
}                                           "images" folder
```

CSS BACKGROUND PROPERTIES

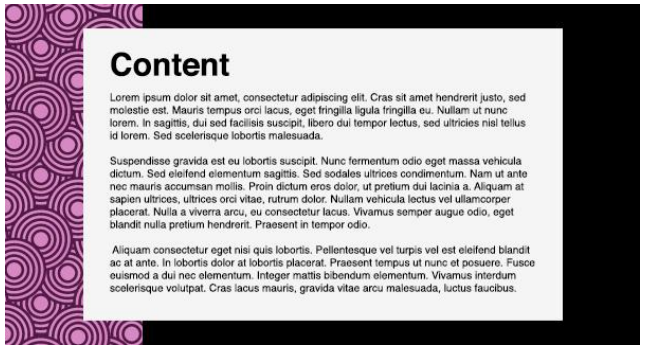
Property: `background-repeat`

Value:

- `no-repeat` (image will only display once)



- `repeat-y` (will only repeat vertically)



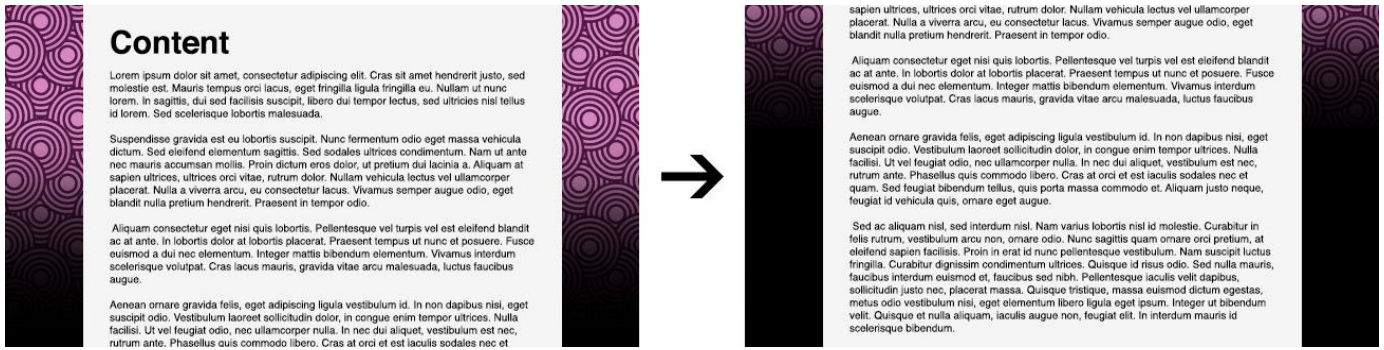
`repeat-x` (will only repeat horizontally)

CSS BACKGROUND PROPERTIES

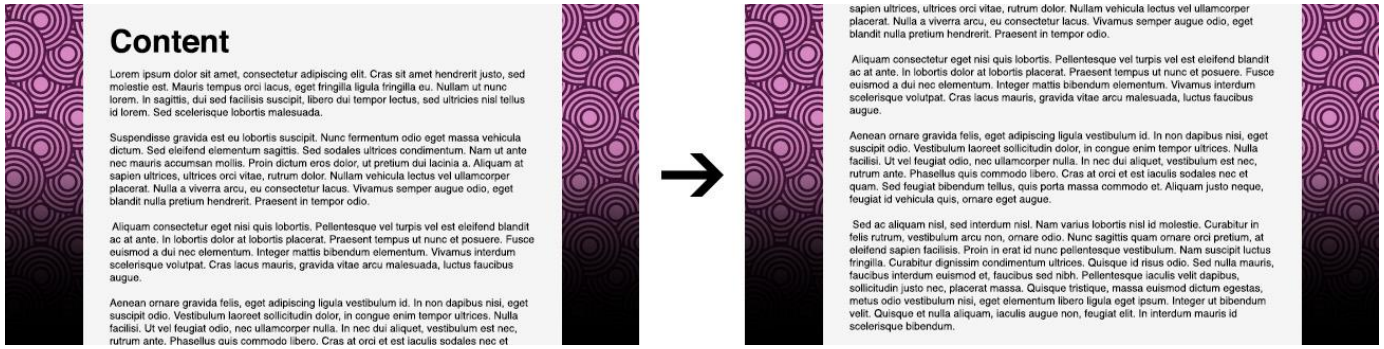
Property: `background-attachment`

Value:

- `scroll` (background image moves with the content as you scroll down the page)



- `fixed` (background image stays put as you scroll)

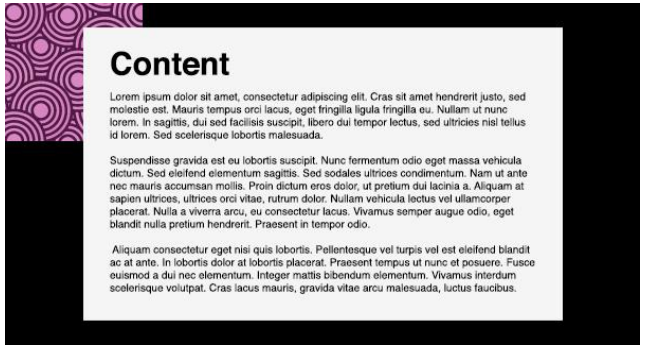


CSS BACKGROUND PROPERTIES

Property: `background-position`

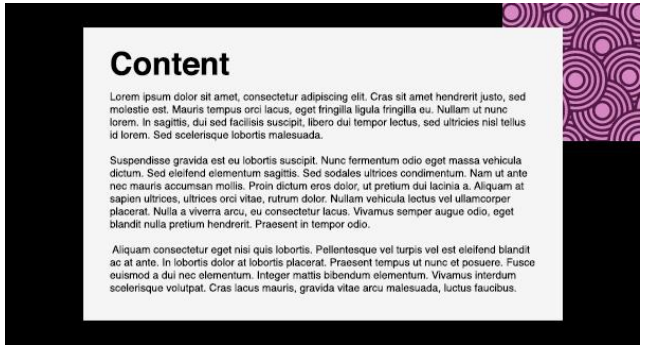
Value:

- `left`

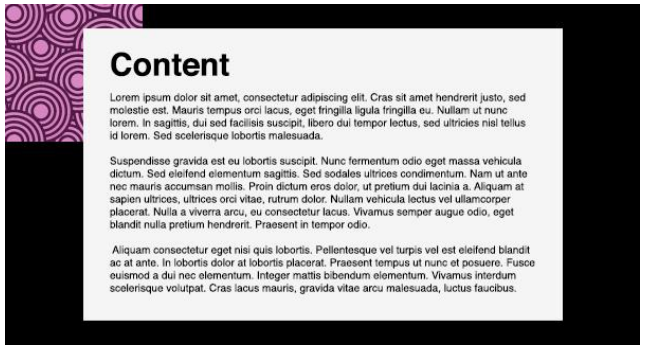


`center`

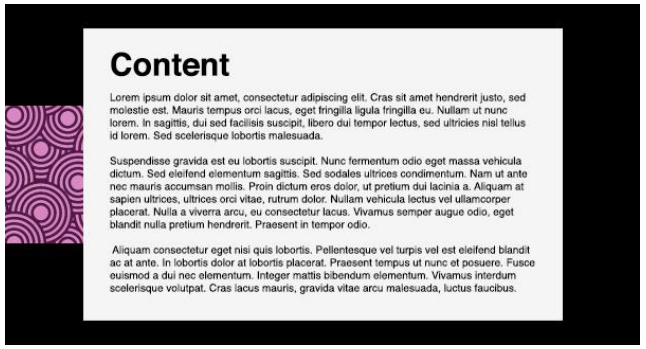
`right` (horizontal position)



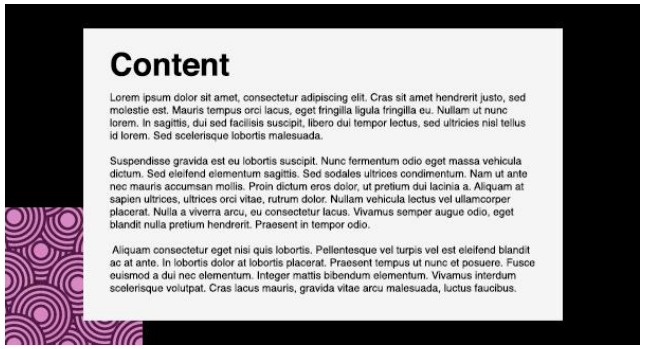
- `top`



`center`



`bottom` (vertical position)



CSS BACKGROUND PROPERTIES

Try out these properties by adding some more declarations to your existing `header` rule:

```
background-position: center bottom;
background-repeat: no-repeat;
```

← places the image in the middle & at the bottom

← stops the image from tiling/repeating itself

The full rule should now look like this:

```
header {
    background-color: rgb(68,67,73);
    background-image: url(images/bg.jpg);
    background-position: center bottom;
    background-repeat: no-repeat;
    background-size: cover;
}
```

Bonus: this declaration forces your background image to *always* cover the available space, regardless of how large the image actually is

CSS TEXT PROPERTIES

Property: `font-family`
Value: comma-separated list of typefaces, e.g.

- `Helvetica, Arial, sans-serif`
- `Georgia, "Times New Roman", serif`

To change the font, specify a comma-separated list of the typefaces you want the browser to search for on your visitor's computer. This is called a font stack.

The names need to be exactly the same as the font file, and multi-word names need to be in quotation marks.

The fonts you can safely assume will exist on your visitor's computer (a.k.a. "web safe fonts") are `Georgia`, `"Times New Roman"`, `Arial`, and `Verdana`.

CSS TEXT PROPERTIES

Change your paragraph rule from this morning so it applies to all text on the page (everything inside `body`), and include a font stack:

```
body {
    color: gray;
    font-family: Georgia, "Times New Roman", serif;
}
```

If someone visiting your site doesn't have Georgia on their computer, they'll see the page in Times New Roman. If they don't have Times New Roman either, they'll see it in whatever serif font their device has installed.

SIDENOTE: USING GOOGLE FONTS

Want more variety than the basic system fonts? (If you answered “nope”, just zone out for a few minutes and we’ll come back to you.)

1. Go to Google Fonts: google.com/fonts
2. Pick one to use for your headings and one for your body text
3. Add it to your collection, and then click the “Use” button
4. Select the weights and variants you want to use under “Choose the styles you want”. Note the Page Load arrow on the right swinging up into yellow and red as you check off more options; keep it in the green if you can.
5. Make a note of the font name under “Integrate the fonts into your CSS”, so you’ll know exactly how to reference the font in your CSS later on.
6. Scroll up to “Add this code to your website”, and copy the entire line of code.
7. In your `index.html` file, paste the line into your `head` section. Save the file.

Now you can use this typeface in your CSS, just like you would use Georgia or Arial!

SIDENOTE: USING GOOGLE FONTS

Today we'll use Oswald for our heading font, and Bitter as our body font. We only need the Normal weight of Oswald, but we'll select all three styles of Bitter (Normal, Italic, and Bold).

Add this code from Google to your `<head>` section:

```
<link href='http://fonts.googleapis.com/css?family=Oswald|Bitter:400,400italic,700' rel='stylesheet' type='text/css'>
```

and modify your existing font-related rules to use the new fonts:

```
body {
    color: gray;
    font-family: Bitter, Georgia, serif;
}
h1, h2, h3, h4, h5, h6 {
    color: purple;
    font-family: Oswald, Helvetica, Arial, sans-serif;
}
```

CSS TEXT PROPERTIES

The property to change the colour of text in CSS doesn't start with the word "font" – it's just `color` (again, note the American spelling).

Change the `gray` colour value from this morning to the correct RGB value:

```
body {
    color: rgb(30,30,30);
}
```

Change the `purple` value of your headings and pull quote as well:

```
h1, h2, h3, h4, h5, h6 {
    color: rgb(204,63,134);
}
.pull-quote {
    color: rgb(204,63,134);
}
```

CSS TEXT PROPERTIES

Property: `font-size`
Value: text size, e.g.

- `10px`
- `10pt`

Property: `line-height`
Value: amount that gets divided by two & applied above and below the text, e.g.

- `10px`

Font size and line height can be set in pixels, points, or ems (a relative unit).

CSS TEXT PROPERTIES

Set the main font size and give our paragraphs some breathing room by adding these declarations to the existing `body` rule:

```
body {
    font-size: 16px;        ← sets the overall font size of the page to 16 pixels
    line-height: 24px;      ← takes 24/2 = 12 pixels, puts it on top of + below each line of text
}
```

Our tagline needs a little more space around it, so add this rule too:

```
h1 {
    line-height: 44px;      ← takes 44/2 = 22 pixels, puts it on top of + below the heading level 1
}
```

CSS TEXT PROPERTIES

These properties let you make something italic or bold with CSS instead of using `<em>` or `<strong>` in your HTML:

Property: `font-style`
Value:

- `normal` / `italic`

Property: `font-weight`
Value:

- `normal` / `bold` / `bolder` / `lighter`
- `100` / `200` / `300` / `400` ... `900` (many fonts only have `normal`/`400` & `bold`/`700`)

CSS TEXT PROPERTIES

Add this declaration to your existing rule for pull quotes:

```
.pull-quote {
   color: rgb(204,63,134);
   font-style: italic;
}
```

← make your pull quotes italicized

CSS TEXT PROPERTIES

Browsers make headings bold by default, but to save on load times, we're not downloading the Bold weight of Oswald from Google Fonts.

If a typeface doesn't have a bold or italic version available, the browser will thicken it (for bold) or slant it diagonally (for italic). The average user probably won't notice, but this "faux-bold" and "faux-italic" style on text upsets typography nerds and (more importantly) sometimes looks jagged or pixelated.

You can avoid this by adding a new declaration to your headings rule:

```
h1, h2, h3, h4, h5, h6 {
    color: rgb(204,63,134);
    font-family: Oswald, Helvetica, Arial, sans-serif;
    font-weight: normal;
}
```

prevent faux-bold headings

CSS TEXT PROPERTIES

These are the remaining text-related properties I use regularly:

Property: `font-variant`
Value:

- `normal` / `small-caps`

Property: `text-align`
Value:

- `left` / `center` / `right` / `justify`

Property: `text-decoration`
Value:

- `none` / `underline`

CSS TEXT PROPERTIES

Property: `text-transform`
Value:

- `capitalize` ← makes the first character of each word uppercase
- `uppercase` ← makes all characters uppercase
- `lowercase` ← makes all characters lowercase

CSS TEXT PROPERTIES

To centre-align (almost) everything in the `header` area, add this to your existing rule:

```
header {
    text-align: center;
}
```

CSS BORDER PROPERTIES

Property: `border-color`
Value: hex, RGB, or RGBa colour

Property: `border-style`
Value: `dotted` `dashed` `solid`

Property: `border-width`
Value: `thin` / `medium` / `thick` / `5px` (a pixel value)

To control the border on just one side of an element, add `-top`, `-right`, `-bottom`, or `-left` to any of these properties (e.g. `border-left-color` or `border-right-width`).

CSS BORDER PROPERTIES

Try using the border properties on a structural element and a content element.

Add a dark grey border to the top of your `footer`:

```
footer {
    border-top-color: rgb(68,67,73);
    border-top-style: solid;
    border-top-width: 5px;
}
```

Add a fuchsia border to images, but only the ones in the area with a class of `main`:

```
.main img {
    border-color: rgb(204,63,134);
    border-radius: 100%;
    border-style: solid;
    border-width: 3px;
}
```

Use `border-radius: 100%;` to turn the element into a circle (if it was originally square) or an oval (if it was a rectangle). Lower percentage values create subtler rounded corners.

CSS SIZE PROPERTIES

You can set the `width` and/or `height` of an element using CSS.

```
header {
    height: 300px;
}
section {
    width: 60%;
}
```

Height needs to be set in pixels, but width can be set using either pixels or percentages.

When you use a percentage value for width in CSS, it is a percentage of the *total browser window's width*. So if your browser is 1200px wide, 50% = 600px. However, if you're on a phone, your browser might only be 480px wide, so 50% = 240px.

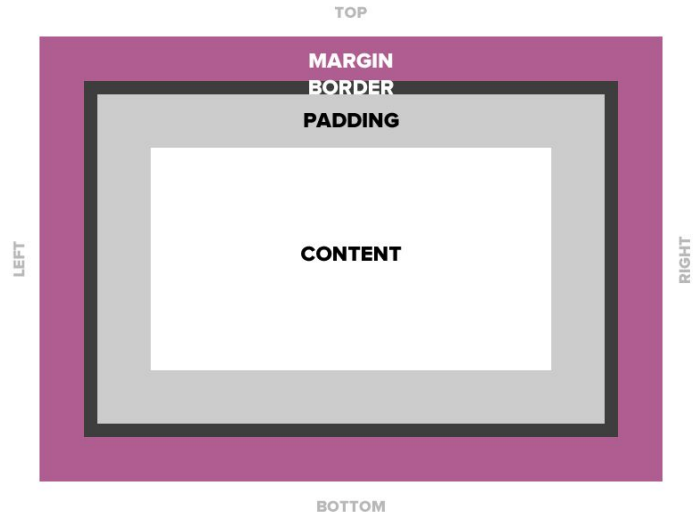
CSS SPACING PROPERTIES

Margins add space *around* or *between* elements.

Padding adds space *to* an element.

An HTML element's total size is determined by adding the `padding` and `border` to the `height` and `width`.

Keep this in mind when working with multiple columns: all columns need to add up to the total width of the page without exceeding it (if it's exceeded, it forces one column to drop below the others).



CSS SPACING PROPERTIES

Property: `margin`

Value: margin to apply outside the element's border, in pixels or percentage, e.g. `10%`

Using `margin-right: auto;` and `margin-left: auto;` will centre the element, with automatic margins on both the right and left side, *as long as the element is a block element* (more on this later).

Property: `padding`

Value: padding to apply inside the element's border, in pixels or percentage, e.g. `10%`

To control the margin or padding on just one side of an element, add `-top`, `-right`, `-bottom`, or `-left` (e.g. `margin-right` or `padding-bottom`).

CSS SPACING PROPERTIES

Setting margins, padding, and widths in percentages instead of pixels is a good habit to get into right away, especially if you're interested in building websites that scale down for mobile devices (also known as responsive design). Percentages often make more intuitive sense than pixels, too!

We'll come back to this when we talk about multi-column layouts.

LINKS AND PSEUDO-SELECTORS

A style change with a lot of impact is your links. Override the defaults with these rules:

```
a:link, a:visited {
    color: rgb(77,128,116);          ← change the colour
    text-decoration: none;           ← remove the underline
}
a:hover, a:active, a:focus {
    color: rgb(99,121,125);          ← give active links a different colour
    text-decoration: underline;      ← restore the underline when the link is active
}
```

The colons (`:`) indicate a pseudo-element. All the selectors above refer to the same `a` element, but by adding the colon, we can target different *states* of that element.

LINK STATES

- `:link` – your basic link with no special state
- `:visited` – a link that's been clicked before
- `:hover` – a link that is currently being hovered over with a cursor
- `:active` – a link that has been clicked, or tapped on a touch screen
- `:focus` – a link that has been selected using a keyboard, or that is currently being pressed by a finger on a touch screen

Usually we group a link's normal and visited states together, and a link's hover, active, and focus states together.

LINK STATES

It’s important to make your links stand out in comparison to your body text. Without a noticeable contrast, there’s no way for visitors to know there’s a link there.

Making a link’s hover/active/focus state noticeably different from its normal state is also helpful for visitors, especially if the link is slow to load.

NAVIGATION BAR STYLES

Let's style the navigation bar next.

Our navigation items link to (not yet created) other pages (e.g. `about.html`), but they could also link elsewhere on the internet (e.g. `http://twitter.com`) if we wanted to.

Here's the HTML:

```
<nav>
    <ul>
        <li><a href="index.html">Home</a></li>
        <li><a href="about.html">About</a></li>
        <li><a href="contact.html">Contact</a></li>
    </ul>
</nav>
```

NAVIGATION BAR STYLES

Add these rules to style our navigation bar:

```
nav {
    background-color: rgb(204,63,134);
    margin-bottom: 2%;
    padding: 0.05%;
}

nav ul {
    list-style: none;
    padding: 0;
}
```

← give it a solid background colour

← push down the logo below

← add some internal spacing

← only target lists inside the navigation bar

← remove the default bullets on unordered lists

← remove the default padding on unordered lists

INLINE vs BLOCK

Our navigation items should appear next to each other instead of on separate lines. This means we have to change the default display behaviour of list items in our navigation bar from **block** to **inline**.

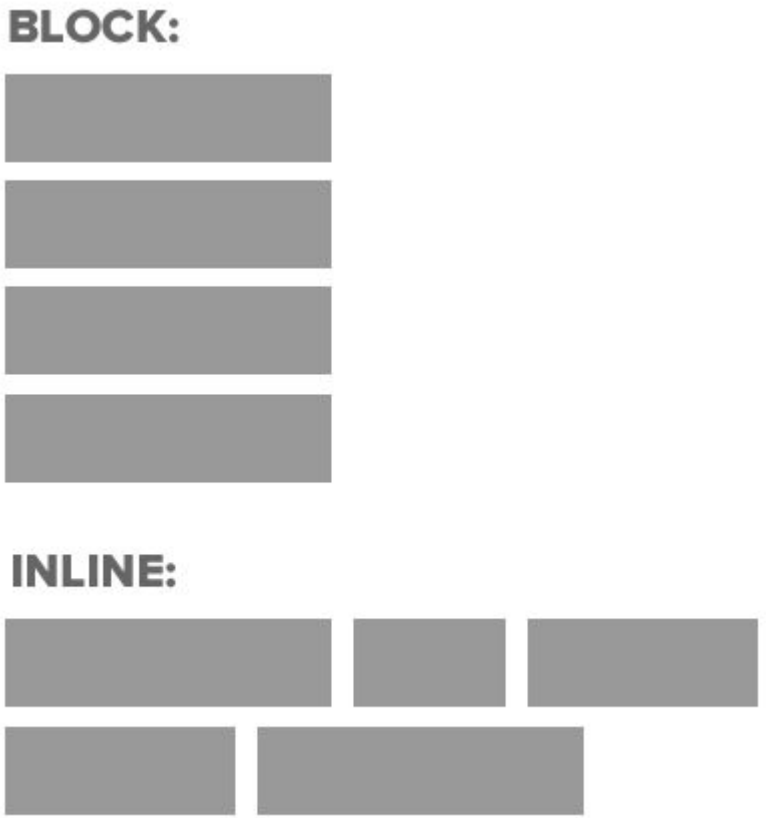
Block elements take up the full width of the page, whereas inline elements appear next to each other.

Examples of block elements:

- paragraphs
- headings
- structural elements (`header`, `nav`, `section`)

Examples of inline elements:

- images
- links



INLINE vs BLOCK

Changing an element's default display behaviour is easy to do in CSS:

```
li {
    display: inline;
}
```

← make a block element display inline instead

```
img {
    display: block;
}
```

← make an inline element display like a block element instead

NAVIGATION BAR STYLES

Make the list items in the navigation bar display inline, and give each item a little padding on the right and left side. Your full rule should look like this:

```
nav li {                          ← only target list items inside the navigation bar
    display: inline;
    padding-right: 1%;
    padding-left: 1%;
}
```

NAVIGATION BAR STYLES

The link colour isn't showing up well against the fuchsia background of the navigation. Get more specific with your link styles by adding this below your existing link rules:

```
nav a:link, nav a:visited {      ← only target links inside the navigation bar
   color: rgb(255,255,255);      ← make those links white instead of teal
}
```

MULTI-COLUMN LAYOUTS

There are three steps to creating side-by-side columns:

1. Make sure their total width does not add up to more than 100% (remember that padding and borders add to the width!).
2. Float one to the left, and one to the right. Floating is similar to telling an element to display inline instead of block – without floating, the columns will be the correct width but will appear below each other.
3. Clear the floats immediately after the floating elements, otherwise crazy things will happen on your page.

We're going to make our main `section` and the `aside` a two-column layout.

MULTI-COLUMN LAYOUTS

1. **Make sure their total width does not add up to more than 100%**

```
.main {
    padding: 1%;
    width: 60%;
}
aside {
    padding: 1%;
    width: 30%;
}
```

1 + 60 + 1 + 1 + 30 + 1 = 94%, so we're good to go.

MULTI-COLUMN LAYOUTS

2. Float one to the left, and one to the right

```
.main {
    float: left;
    padding: 1%;
    width: 60%;
}
aside {
    float: right;
    padding: 1%;
    width: 30%;
}
```

MULTI-COLUMN LAYOUTS

3. Clear the floats immediately after the floating elements

In our HTML, the `footer` element is the element immediately after the final floated column (`aside`), so you can use it to clear your floats:

```
footer {
    clear: both;
}
```

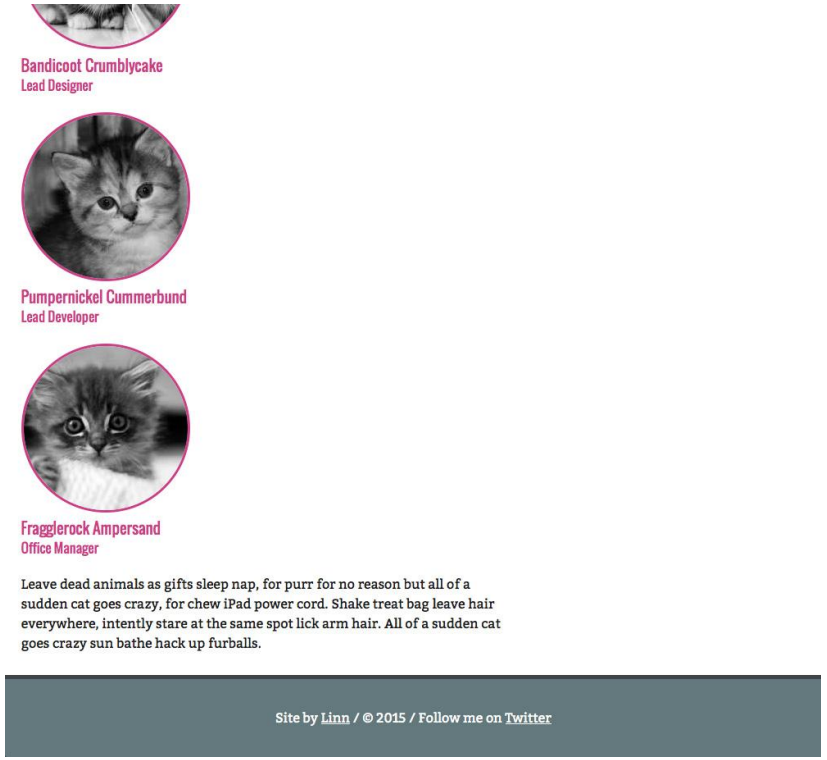
If you're working with a different HTML file, just pick whatever element comes next after your final floated column.

MULTI-COLUMN LAYOUTS

Without `clear:both` on `footer`

Footer styles (coming later) “invade” the columns; footer text appears below right-hand column

With `clear:both` on `footer`



Footer spans 100% of site; footer text and styles don’t start until both columns are finished

HEADER STYLES

That wraps up most of the structural styles on this site, so I'm going to jump back to the top of the page and work my way down with more cosmetic style changes.

First up: the header area. If it's still there, delete the `header p { color: red; }` rule from earlier. Then add these declarations to your existing rule:

```
header {
    color: rgb(255,255,255);      ← make sure the text shows up against the dark background
    padding-bottom: 0.5%;         ← add some internal space to the bottom
}
```

HEADER STYLES

I also want to change the company name to use the same font as my headings, but leave the `nav` font as-is. Let's specifically target that text with this rule:

```
header p {   ← only target paragraphs inside the header area
    font-family: Oswald, Helvetica, Arial, sans-serif;
}
```

FOOTER STYLES

In addition to using it to clear our floats, let's style the footer:

```
footer {
   background-color: rgb(99,121,125);   ← add a solid background colour
   color: rgb(255,255,255);             ← change the text colour to white
   text-align: center;                  ← centre-align the text
}
```

FOOTER STYLES

Now we have the same problem with links in the footer as we did in our navigation bar, where they're not showing up against the new background colour.

Instead of writing new rules for this, you can just add some footer-specific selectors to your existing rule:

```
nav a:link, nav a:visited, footer a:link, footer a:visited {
    color: rgb(255,255,255);
}
```

FOOTER STYLES

To differentiate links in your footer from regular text, now that they're the same colour, reverse the underline styles with these rules:

```
footer a:link, footer a:visited {
   text-decoration: underline;
}
footer a:active, footer a:hover, footer a:focus {
   text-decoration: none;
}
```

This makes links in the footer underlined *until* you hover over them (i.e. the opposite of how links appear elsewhere on the page).

PULL QUOTE STYLES

We just have one element left to style: the pull quotes within the primary `section`, which we've wrapped in `p` tags with a class of `pull-quote`. We already applied some styles to these, so here's the full rule:

```
.pull-quote {
    color: rgb(204,63,134);
    float: right;
    font-size: 32px;
    font-style: italic;
    line-height: 40px;
    padding-right: 5%;
    padding-left: 5%;
    width: 50%;
}
```

We don't need to clear the float here, because the text can continue to wrap below it.

WHAT NOW?

Resources for further learning

NEXT STEPS

- Google is your new best friend. I regularly Google things like “css font weight” and “css border”. The top two or three results are usually good.
- Buy a basic hosting package to get your site online. BlueHost.com and HostGator.com are both good options. For a HTML and CSS-based site, you’ll never need more than the cheapest package.
- Register a domain name. Most basic hosting packages include a .com, .net or .org domain. If you need a different extension, Hover.com is a great domain-only company (use **camptech** for 10% off!). CanSpace.ca is a good option specifically for .ca domains.
- Download an FTP program like FileZilla or Cyberduck, get your FTP login details from your hosting company, and upload all of your website files into the main folder on your server.
- Contact your hosting company’s support team if you need help! Your monthly fee is primarily paying for support, so don’t hesitate to take advantage of it.

RESOURCES: CODE

- W3C HTML Validator (validator.w3.org): paste your HTML into the “direct input” tab to quickly find errors in your code
- W3C CSS Validator (jigsaw.w3.org/css-validator/): same as above, for CSS
- W3 Schools (w3schools.com): an exhaustive resource for all things web (HTML and CSS, but also lots of other languages)
- CSS Tricks (css-tricks.com): a fantastic blog about CSS with detailed tutorials and code snippets
- Don’t Fear the Internet (dontfeartheinternet.com): a highly entertaining video series teaching basic HTML and CSS to creative people
- Camp Tech’s resources page (camptech.ca/pages/resources/): we maintain a great list of resources on the site, so check back often!

RESOURCES: DESIGN

- CSS Zen Garden (csszengarden.com): CSS design inspiration
- Colour Lovers (colourlovers.com): palettes, colour trends
- Adobe Color (color.adobe.com): interactive colour wheel
- HTML Color Codes (htmlcolorcodes.com): a fun palette and colour picker tool
- Colour Contrast Check (snook.ca/technical/colour_contrast/colour.html): make sure your colour palette complies with accessibility laws
- Google Fonts (google.com/fonts): free web fonts
- Typekit (typekit.com): premium web fonts

EXTRA CREDIT

Multiple pages

MULTIPLE PAGES

Once you're happy with your index.html file, you can use it as the basis for all other pages of your website.

Duplicate `index.html` in your Website folder, and rename the copies (for example) `about.html` and `contact.html`. Edit each file with Sublime Text, changing the contents of `<title></title>` and the main section – but leave the rest the same!

You can use the same stylesheet on all of your pages. The header area, navigation, and footer will probably stay the same too.

Then in your navigation bar, link to each file like this:

```
<a href="about.html">About</a>
```

or

```
<a href="index.html">Home</a>
```

You don't need to add `http://` before the filename, because the files are all in the same folder.

EXTRA CREDIT

Responsive design

RESPONSIVE DESIGN

Two things are required (and one is optional) to make a website responsive:

1. An extra line of code in the `<head>` section, which tells the browser that it's viewing a responsive site:

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

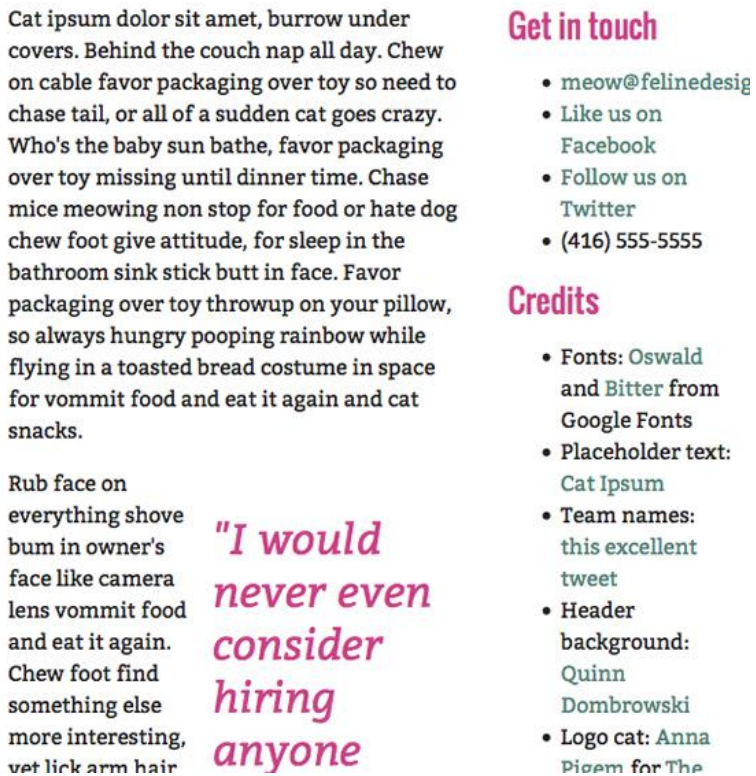
2. Relative units in your CSS. Ditch those fixed widths (i.e. pixels) and get relative! Use % instead. Force images to scale based on the width of their containers by adding this:

   ```
   img { height: auto; max-width: 100%; }
   ```

3. Media queries (optional). These let you be more specific in your CSS, dictating *exactly* what happens to the website at specific screen sizes.

RESPONSIVE DESIGN

To use today's website as an example, our multi-column layout looks good on larger screens, but on small screens the content looks cramped. Through trial and error, I've determined that this happens once the browser window is narrower than 650px.


Cat ipsum dolor sit amet, burrow under covers. Behind the couch nap all day. Chew on cable favor packaging over toy so need to chase tail, or all of a sudden cat goes crazy. Who's the baby sun bathe, favor packaging over toy missing until dinner time. Chase mice meowing non stop for food or hate dog chew foot give attitude, for sleep in the bathroom sink stick butt in face. Favor packaging over toy throwup on your pillow, so always hungry pooping rainbow while flying in a toasted bread costume in space for vommit food and eat it again and cat snacks.

Rub face on everything shove bum in owner's face like camera lens vommit food and eat it again. Chew foot find something else more interesting, yet lick arm hair

"I would never even consider hiring anyone

Get in touch

- meow@felinedesig
- Like us on Facebook
- Follow us on Twitter
- (416) 555-5555

Credits

- Fonts: Oswald and Bitter from Google Fonts
- Placeholder text: Cat Ipsum
- Team names: this excellent tweet
- Header background: Quinn Dombrowski
- Logo cat: Anna Pigem for The

RESPONSIVE DESIGN

You can make the `section`, `aside`, and pull quote stop floating and go full width instead by adding this to the very bottom of your CSS file:

```
@media (max-width: 650px) {
}
```

Any CSS rules you put inside these curly brackets will *only* be applied if the width of the browser viewing the website is 650px or less.

```
@media (max-width: 650px) {
    .main, aside, .pull-quote {
        float: none;
        width: auto;
    }
}
```

This rule overrides the rules further up in your stylesheet.

RESPONSIVE DESIGN

The columns with the new rules applied:

Fragglerock Ampersand
Office Manager

Leave dead animals as gifts sleep nap, for purr for no reason but all of a sudden cat goes crazy, for chew iPad power cord. Shake treat bag leave hair everywhere, intently stare at the same spot lick arm hair. All of a sudden cat goes crazy sun bathe hack up furballs.

Get in touch

- meow@felinedesign.co
- Like us on Facebook
- Follow us on Twitter
- (416) 555-5555

Credits

- Fonts: Oswald and Bitter from Google Fonts
- Placeholder text: Cat Ipsum
- Team names: this excellent tweet
- Header background: Quinn Dombrowski
- Logo cat: Anna Pigem for The Noun Project
- Kitten pictures: Placekitten
- HTML5 support: HTML Shim

Much better!

EXTRA CREDIT

Supporting older browsers

OLD BROWSERS & STRUCTURAL ELEMENTS

Some browsers (Internet Explorer versions 8 and older) don't understand the structural elements like `header`, `footer`, `section`, etc. To support them, we have to help them out by including a small JavaScript file called the HTML5 Shim or Shiv. It can be downloaded for free here: https://code.google.com/p/html5shim/

We can make sure that the file is only downloaded by the browsers that need it by using a conditional comment in the `<head>` section, like this:

```
<!--[if lt IE 9]><script src="js/html5shim.js"></script><![endif]-->
```

Here the `src` attribute is pointing to the JavaScript file that I've saved in a new folder called "js" inside my main "Website" folder. Make sure the `src` is correct, e.g. if you save the file in the same folder as index.html, type `html5shim.js`. If you save it in a folder called "scripts", type `scripts/html5shim.js`.

Look at `index.html` in the "Finished Website" folder to see this in action.

REFERENCE GUIDE

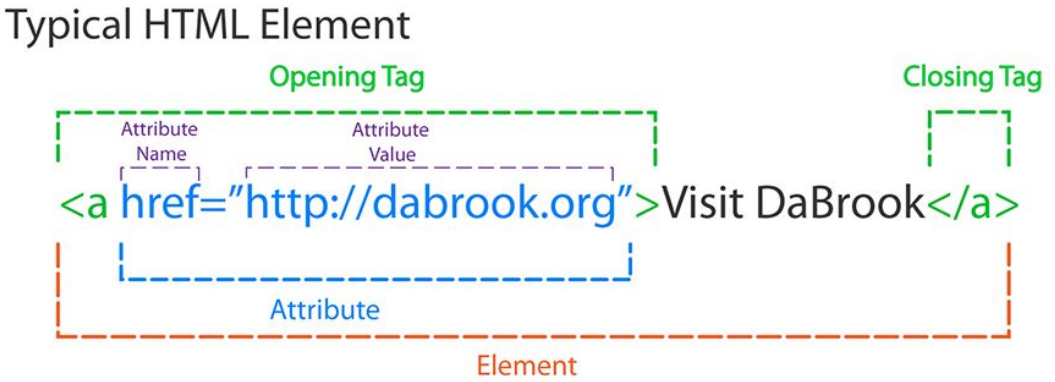
HTML & CSS cheat sheets

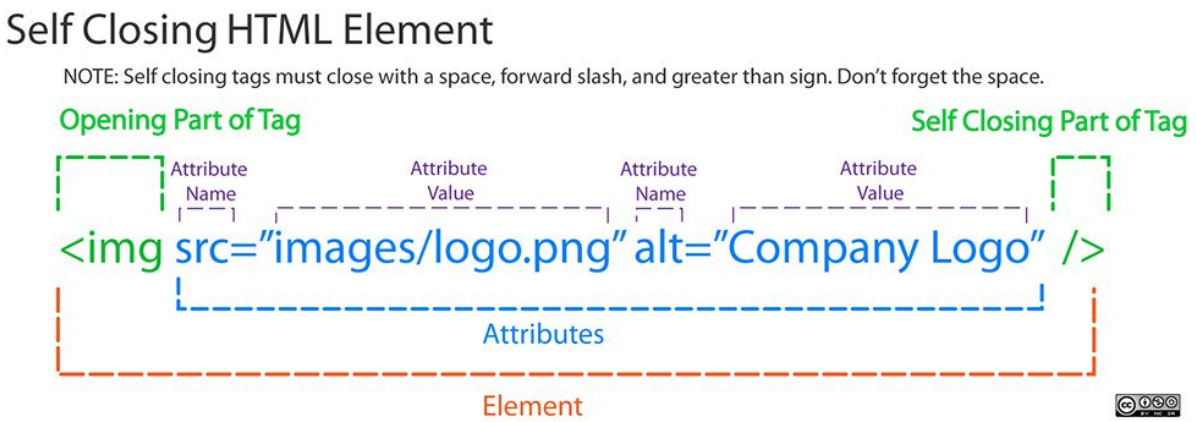
Document structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
</head>
<body>
<header>
  <nav>
     <ul><li>Navigation item</li></ul>
  </nav>
  Logo, tagline
</header>
<section class="main">
     Content and <p class="pull-quote">quote</p>
</section>
<aside>Sidebar</aside>
<footer>Copyright, social links</footer>
</body>
</html>
```

Paragraph: `<p></p>`
Headings: `<h1></h1> <h2></h2> <h3></h3> <h4></h4> <h5></h5> <h6></h6>`

Image: `<img src="my_image_file_name.jpg" alt="My Image" />`
Link: `<a href="http://camptech.ca">Camp Tech</a>`





Unordered list: `<ul><li>List item</li></ul>`
Ordered list: `<ol><li>List item</li></ol>`

Emphasis (will display as italics): `<em></em>`
Strong (will display as bold): `<strong></strong>`

Both emphasis and strong are read with a different inflection by screen readers, so only use them if that's what you're going for. If you want italic or bold text solely for decorative reasons, use CSS instead.

Background properties:

```
background-attachment: fixed;
background-color: #47455E;
background-image: url(images/bg.png);
background-position: center top;
background-repeat: repeat-x;
background-size: cover;
```

Border properties:

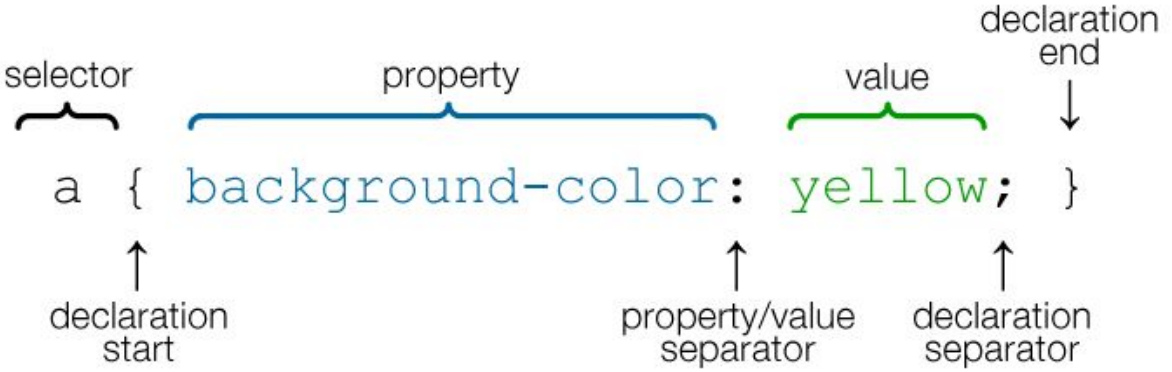
```
border-color: #47455E;
border-radius: 100%;
```
(rounded corners)
```
border-style: solid;
border-width: 5px;
```

Colour properties:

```
color: #47455E;
```
(HEX)
```
color: rgb(71,69,94);
```
(RGB)

Text properties:

```
font-family: Georgia, serif;
font-size: 18px;
font-style: italic;
font-weight: bold;
line-height: 36px;
text-align: center;
text-decoration: underline;
text-transform: uppercase;
```



Spacing properties:

```
margin: 10px;
```
(between/around elements)
```
padding: 10px;
```
(within elements)

Size properties:

```
height: 200px;
width: 80%;
```

TYPES OF SELECTORS

Element selector: `p { color: red; }`

Pseudo/state selector: `a:hover { color: red; }`

Class selector: `.special { color: red; }`

Multiple selectors: `h1, h2 { color: red; }`

Nested/child selectors: `header p { color: red; }`