FRONT-END FOUNDATIONS

In this course

- Making a web page with HTML and CSS
- Writing HTML text tags and CSS selectors
- Laying out a web page with multiple sections
- Working with images on a web page
- Customizing fonts and building HTML forms

LEVEL 1

HTML

A Quick Story of the Web

Tim Berners-Lee

In 1980, Tim worked for CERN, where he proposed a better way for researchers there to share and read documents.

CERN

You know, the same people who run the Large Hadron particle accelerator.

ENQUIRE

So in 1984, he built something called ENQUIRE for CERN, which was made up of Cards (Documents) and Hyperlinks (which connected the Documents).

But Tim realized he could do better.

The World Wide Web

With Robert Cailliau, Tim Berners-Lee wrote a proposal in 1990 for the creation of the **World Wide Web**.

When it started, the WWW was a way for scientists around the world to create and share their own webpages.

The World Wide Web's First Logo

The language of a webpage is HTML

HTML allows you to take a plain text document created in any simple text editor...

```
Learn By Doing

No setup. No hassle. Just Learning.

Code School teaches web technologies in the
comfort of your browser with video lessons,
coding challenges, and screencasts.
```

.. and organize it into lists, link to other webpages, include images, and more.

But what does HTML mean?

HTML stands for **H**yper**T**ext **M**arkup **L**anguage

Hypertext

Overcoming the constraints of written text. *Interactive*.

Markup Language

A way to literally "Mark Up" a document to specify attributes, like different font sizes, lists, links to other webpages, and images.

HTML is written in text files

Just like most programming languages, we type a bunch of HTML into a file (aka. document) so we can send it around.

index.html

Name of the file

File extension
(tells the browser the file contains HTML)

To display an HTML file we need a browser

Web Browsers are basically "HTML Readers."

They understand how to read HTML and display it for you.

Hopefully they all display

the same thing (that's the idea).

A guide to the slides

In this course, we're going to show you HTML, CSS and what those look like in the browser.

Look for these symbols to help you figure out what you're looking at:

HTML — HTML code

CSS — CSS code

What it looks like displayed in the browser

Building Recipe World

First things first: get some content!!

Before you start making a web page, you've got to figure out what content it is going to show.

We'll start with a single recipe with these sections.

Page Title:	`Recipe World`
Recipe Name:	`Magic Cake`
Section Name:	`Ingredients`
Section Name:	`Directions`

Writing HTML

Most of the time, you'll put your content in between **HTML tags**, which have corresponding opening and closing versions.

HTML

```
<h1> Recipe World </h1>
```

opening tag

the closing version has a slash before the tag name

closing tag

Recipe World

Use heading tags to define your content hierarchy

Higher heading numbers mean the content that appears between the headings is less important than lower numbers.

HTML

```
<h1>Recipe World</h1>
<h2>Magic Cake</h2>
<h3>Ingredients</h3>
<h3>Directions</h3>
```

More important

Less important

Recipe World

Magic Cake

Ingredients

Directions

Generally speaking:

The **page title/company name** goes in the **<h1>**

The **page main subject** goes in the **<h2>**

<h3> through **<h6>** are used to organize other divisions of page content

Use paragraph tags for non-heading text

<p> is called the paragraph tag.

HTML

```
<h1>Recipe World</h1>
<h2>Magic Cake</h2>
<p>Magic Cake is one of the tastiest...</p>


<h3>Ingredients</h3>
...
```

This paragraph describes Magic cake

Add paragraph content (where necessary) in between heading tags.

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

Use an unordered list to display a list of things

<ul> stands for unordered list.

Each list item needs to also be put inside of an **<li>** tag.

HTML

```
<h3>Ingredients</h3>
<ul>
  <li>2 eggs</li>
  <li>1 pound of sugar</li>
  <li>3 sticks of butter</li>
</ul>
```

unordered list

list items

This list of ingredients doesn't need to be in a specific order

Ingredients

- 2 eggs
- 1 pound of sugar
- 3 sticks of butter

Nesting tags

HTML tags don't have to always just contain text, they can contain other HTML tags.

parent

children

HTML

```html
<p>Magic Cake is one of...
<h3>Ingredients</h3>
<ul>
  <li>2 eggs</li>
  <li>1 pound of sugar</li>
  <li>3 sticks of butter</li>
</ul>
<h3>Directions</h3>
...
```

when you indent, it's easy to see that 3 <li> tags are nested inside of a <ul> tag

A tag that contains other tags is called the **parent**.

The tags contained in a parent tag are called **children**.

Not indenting child tags makes HTML hard to read

HTML

```html
<h1>Recipe World</h1>
<h2>Magic Cake</h2>
<p>Magic Cake is one of...
<h3>Ingredients</h3>
<ul><li>2 eggs</li><li>1 pound of
sugar</li><li>3 sticks of butter</li>
</ul>
<h3>Directions</h3>
...
```

This is valid HTML, but it's really hard to read

Use an ordered list to show list items in a certain order

If the content in your list does refer to steps to be followed, use an ordered list.

HTML

```
<h3>Ingredients</h3>
<ul>...
<h3>Directions</h3>
<ol>
  <li>Mix eggs, sugar, and butter in
      a large bowl.</li>
  <li>Spread into a non-stick dish</li>
  <li>Bake at 350
  <li>Let sit at r
      minutes</li>
  <li>Eat and enjo
</ol>
```

ordered list

list items

these steps need to be followed in order or the cake will be ruined

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

Wrapping everything in the body

Any content that appears on a web page should be in between a <body> tag.

the parent of all visible content

HTML

```
<body>
  <h1>Recipe World</h1>
  <h2>Magic Cake</h2>
  <p>Magic Cake is one of...</p>
  <h3>Ingredients</h3>
  <ul>...</ul>
  <h3>Directions</h3>
  <ol>...</ol>
</body>
```

<body> doesn't display anything, but helps keep the page content organized.

Adding a head tag

Non-visible stuff goes in the <head> tag.

HTML

```
<head>
</head>
<body>
  <h1>Recipe World</h1>
  ...
</body>
```

we'll discuss what goes in the <head> later when we need it

You'll eventually use the head tag to load other useful scripts, like CSS and JavaScript.

Make everything a child of a single parent tag

All of your HTML goes inside of the **<html>** tag.

HTML

```html
<html>
  <head>
  </head>
  <body>
    <h1>Recipe World</h1>
    ...
  </body>
</html>
```

Notice that we're indenting again so it's easy to see the parent/child relationship

Setting the DOCTYPE to html

The DOCTYPE sets the HTML version.

HTML

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <h1>Recipe World</h1>
    ...
  </body>
</html>
```

The browser can make better decisions about how to display your page when it knows which version of HTML you're using.

The recipe page so far

Recipe World

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

Ingredients

- 2 eggs
- 1 pound of sugar
- 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

FRONT-END FOUNDATIONS

Adding links to our recipe site

Recipe World

- Home
- Recipes
- Suggest

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

We also need to stress that whatever happens to you as a result of consuming Magic Cake is your responsibility. You've been warned, and you can view our legal page for more information.

Ingredients

- 2 eggs
- 1 pound sugar
- 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

What happens when you type a URL in the Browser

1 Enter a URL into the browser's address bar.

Browser

No page is loaded yet

The URL contains a **protocol**, **server** and a **file** that is being requested.

What happens when you type a URL in the Browser

2

A request is sent to the server, and the server looks for the file that's being requested.

What happens when you type a URL in the Browser

3 The server returns the file to the browser and the browser displays it.

What happens when you click a link in the Browser

Clicking a link is like typing a URL in the address bar, only the URL is already pre-entered in HTML as part of link.

1. Click the link on the page, and a request for the file in that link is sent to the server.

What happens when you click a link in the Browser

2 The server returns the file to the browser and the browser replaces the current page with the new file.

How to make a link

Anything that appears between the opening and closing **a** tags is the part that will be a clickable link on the web page.

HTML

```
<p>...can view our <a>legal page</a> for more...</p>
```

link text

a is short for **anchor.**

How to make a link

The **href** attribute sets what page should open when the link is clicked.

the value of the attribute
should always be in quotes

HTML

```
<a href="http://example.com/legal.html">legal page</a>
```

HTML attribute

HTML **attributes** let you add additional features or information to a tag beyond just the tag name and text content.

How to make a link

The **target** attribute sets where the page you set in the href should open when the link is clicked.

HTML

```
<a href="http://..." target="_blank" >legal page</a>
```

HTML attribute

_blank means *open this page in a new tab/window.*

_self means *replace the currently open page in this tab/window.*

if you don't add the target attribute, the default is _self

Demo of a link opening in a new tab

Absolute vs. Relative paths

Absolute paths ask for a file from a specific server

HTML

```
<a href="http://example.com/legal.html">legal page</a>
```

absolute always includes the protocol and server

Relative paths ask for a file with no server specified.

HTML

```
<a href="legal.html">legal page</a>
```

relative never includes the protocol and server

Since no server name is included, the browser assumes you mean the same server that your page is on.

Links using relative paths

If no protocol/server is included, the browser assumes the protocol and server are the same as the page you're requesting from.

```
<a href="legal.html">legal page</a>
```

HTML

Figuring out the site navigation

Recipe World server files

All of our HTML files are currently in our main site folder on the server.

These are the three main pages that should go in our site navigation.

Adding site navigation with relative links

Relative links should almost always be used when linking to pages that are located on the same site.

add this block of HTML to all pages

HTML

```
<body>
  <h1>Recipe World</h1>
  <ul>
    <li><a href="home.html">Home</a></li>
    <li><a href="recipes.html">Recipes</a></li>
    <li><a href="suggest.html">Suggest</a></li>
  </ul>
  <h2>Magic Cake</h2>
  ...
</body>
```

The Magic Cake recipe page with site navigation

Recipe World

- Home
- Recipes
- Suggest

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

We also need to stress that whatever happens to you as a result of consuming Magic Cake is your responsibility. You've been warned, and you can view our legal page for more information.

Ingredients

- 2 eggs
- 1 pound sugar
- 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

Creating a link to another web site

First, find and copy the URL to the other site

Creating a link to another web site

Absolute links need to always be used when linking to pages that are located on another site/server.

HTML

```
<h3>Ingredients</h3>
<ul>
  <li>2 eggs</li>
  <li>1 pound of
      <a href="http://www.amazon.com/Domino-..."">sugar</a>
  </li>
  <li>3 sticks of butter</li>
</ul>
```

absolute path

link text

FRONT-END FOUNDATIONS

LEVEL 2

CSS

The early days of style on the web

When the Web was brand new, there was no separation between HTML and the **presentation** of that HTML.

HTML

```
<H3 COLOR=RED><CENTER>Ingredients</CENTER></H3>
```

caps-lock got stuck a lot back then too

HTML shouldn't determine how something **looks**,
it should just determine how content is structured.

Separating content from presentation

Thankfully, the authors of the web realized that, and created a way to make rules for how tags should look and put them in a separate place from the HTML.

HTML

```
<h3>Ingredients</h3>
```

CSS

*Make the **h3** tag green*

*Make the **h3** a bigger font*

*Make the **h3** have a border at the bottom*

Cascading Style Sheets

CSS is code that changes the **appearance** of HTML.

HTML without CSS

HTML with CSS

Writing your first selector

Tags are selected by creating something called a **selector**.

HTML

```
<p>Magic Cake is one of the tastiest...</p>
```

the simplest selector is the **type selector**, and is just the tag name without <> brackets.

CSS

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

this is one of many properties that can be applied to selectors

each property can usually be set to one of a few different values

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

The syntax of a selector

Selectors have a very specific syntax that needs to be followed or the CSS won't work.

an open-curly-bracket goes after the selector name

CSS

```
selector {
  property: value;
}
```

A semi-colon goes after the value of a property

a closed-curly-bracket goes after the property list

A colon goes after the property name and before the value

Using multiple properties in one selector

A single selector can change multiple properties.

HTML

```
<p>Magic Cake is one of the tastiest...</p>
```

CSS

```
p {
  text-decoration: underline;
  color: red;
}
```

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

multiple properties applied to a single selector

Selecting multiple tags with one selector

Selectors will select all matching tags on the page and apply properties.

HTML

```html
<h3>Ingredients</h3>
<ul>
  <li>2 eggs</li>
  <li>1 pound of sugar</li>
  <li>3 sticks of butter</li>
</ul>
```

CSS

```css
li {
  font-size: 24px;
}
```

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

Ingredients

- 2 eggs
- 1 pound sugar
- 3 sticks of butter

bigger font size now for all <li> tags

Directions

1. Mix eggs, sugar, and butter in a large bowl
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

Selecting only tags that are children of another tag

A **descendent selector** can be used to select tags only if they are children of another tag.

HTML

```html
<h3>Ingredients</h3>
<ul>
  <li>2 eggs</li>
  <li>1 pound of sugar</li>
  <li>3 sticks of butter</li>
</ul>
```

select only li tags that are children of ul tags

CSS

```css
ul li {
  font-size: 24px;
}
```

Ingredients

- 2 eggs
- 1 pound sugar
- 3 sticks of butter

only the <ul> list items are bigger now

Directions

1. Mix eggs, sugar, and butter in a large bowl
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

TIP: read the selector list from right-to-left. The left-most selector is the parent, with children following to the right.

Selecting tags based on actions and conditions

A **pseudo-selector** is a modifier that can be added to a selector to select a tag only when a certain condition has occurred.

CSS

```
a {
  text-decoration: none;
}
```

removes the underline from all links on a page

```
a:hover {
  text-decoration: underline;
  color: darkred;
}
```

adds an underline and changes the color of all links on a page ONLY when the mouse is over the link

Using pseudo-selectors to narrow selection criteria

The **:first-child** pseudo-selector can be applied to narrow the amount of child tags selected

HTML

```html
<h3>Directions</h3>
<ol>
  <li>Mix eggs, sugar...</li>
  <li>Spread into a...</li>
  <li>Bake at 350...</li>
</ol>
```

select only the first li tag

CSS

```css
ol li:first-child {
  color: red;
}
```

Directions

1. Mix eggs, sugar, and butter in a large bowl
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

Where do we put all this stuff?

One place you can put CSS is in a **style** tag that's a child of the **head** tag.

HTML

```
<html>
  <head>
    <style type="text/css">
      a {
        color: red;
      }
    </style>
  </head>
  <body>
    ...
  </body>
</html>
```

The type attribute lets the browser know that CSS is coming

Your CSS selectors can be written in between the **style** opening and closing tags.

FRONT-END FOUNDATIONS

Hexadecimal colors

A popular way to choose colors in CSS is to use hexadecimal numbers.

CSS

```
p {
  color: red;
  color: #FF0000;

  color: black;
  color: #000000;

  color: yellow;
  color: #FFFF00;
}
```

red

black

yellow

Here's three keyword colors and their hexadecimal equivalents.

Hexadecimal colors

Each hex color is actually three different parts.

Hexadecimal colors - Minimum and Maximum

Color values for red, green, and blue run from **0** - **255** decimal, or **00** - **FF** hexadecimal.

#FFFF00

This color is maximum red, maximum green, and no blue.

That's equivalent to the color **Yellow**

Breaking down decimal numbers

Decimal numbers run from 0-9.

76 (decimal)

multiply the left by 10

(7 x 10)

multiply the right by 1

(6 x 1)

Add them together, and you get **76** decimal

Breaking down hexadecimal numbers

Since we're so used to reading decimal numbers, sometimes it makes sense to convert hexadecimal to decimal.

76 (hexadecimal)

multiply the left by 16

`(7 x 16)`

multiply the right by 1

`(6 x 1)`

Add them together, and you get **118** decimal

Breaking down hexadecimal numbers

Since we're so used to reading decimal numbers, sometimes it makes sense to convert hexadecimal to decimal.

FF (hexadecimal)

multiply the left by 16

(F x 16)

multiply the right by 1

(F x 1)

???? How do we multiply by a letter?!$#?$!$?

Understanding letter-based hexadecimal numbers

Hexadecimal numbers run from 0-15, but use letters for numbers above 9.

A = 10	C = 12	E = 14
B = 11	D = 13	**F = 15**

FF

multiply the left by 16

(15 x 16)

multiply the right by 1

(15 x 1)

Add them together, and you get 240 + 15, or **255** decimal

Calculating the color from hexadecimal numbers

#7403AB

Red: (7 x 16) + (4 x 1) = 116

Green: (0 x 16) + (3 x 1) = 3

Blue: (10 x 16) + (11 x 1) = 171

This color is some red, almost no green, and more blue than red.

That's a color that looks **Purple**

Calculating the color from hexadecimal numbers

#FF00B3

Red: (15 x 16) + (15 x 1) = 255

Green: (0 x 16) + (0 x 1) = 0

Blue: (11 x 16) + (3 x 1) = 179

This color is lots of red, no green, and a little less blue than red.

That's a color that looks **Hot Pink**

Colors on the web

By using hexadecimal numbers to set colors, we get 256 possible combinations for each of the three color channels.

or…

```
256 x 256 x 256 = 16,777,216
```

FRONT-END FOUNDATIONS

Our page after changing some styles with CSS

Questions

How does our page know how much vertical and horizontal space to put between tags?

How can we control that spacing?

Why this much space?

Answer

The box model

The Box

Every tag shown in the body is contained in an invisible rectangle that we'll call **the box**.

HTML

```
<p>Magic Cake is one of...</p>
<p>We also need to stress...</p>
```

These two paragraph tags are shown as two boxes stacked on top of each other.

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

We also need to stress that whatever happens to you as a result of consuming Magic Cake is your responsibility. You've been warned, and you can view our legal page for more information.

content boxes

The boxes on our page

Here are the boxes for the current version of the magic-cake.html page.

HTML

```
<body>
  <h1>Recipe World</h1>
  <ul>...</ul>
  <h2>Magic Cake</h2>
  <p>Magic Cake is one of...
  <p>We also need to stress...
  <h3>Ingredients</h3>
  <ul>...</ul>
  <h3>Directions</h3>
  <ol>...</ol>
</body>
```

Block-level tags

The content of **block-level** tags take up the entire width of the **container**.

the container here is <body>

the box takes up the entire width of the container

new boxes are pushed to the line below

The tags that contain this content are all block-level

h1	**h2**	**h3**
p	**ul**	**li**

Inline-level tags

If a tag is not block-level, it's **inline-level**.

We also need to stress that whatever happens to you as a result of consuming Magic Cake is your responsibility. You've been warned, and you can view our legal page for more information.

inline-level tags don't try to take up more width than they need

HTML

```
<p>...can view our <a href="legal.html">legal page</a>
for more information</p>
```

Some common inline-level tags

a	**img**
input	**label**

Turning block-level into inline-level

Sometimes, you'll want block-level tags to be inline-level.

Example: list items displaying horizontally instead of vertically

block-level list items

the default display is block-level

inline-level list items

CSS

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

Block-level vertical spacing

What we know

- Each tag's content fits in an invisible box
- Each block-level tag's box takes up an entire line (horizontal space)

Back to our problem

How does the browser know how much space goes between the boxes?

The Box Model

The **box model** is a way to describe the borders and spacing in between the **boxes** of each tag.

- There are 4 parts of the box model:

1. Content area

   The **content area** contains your actual content (text, images, etc.)

Magic Cake

The content area will only take up as much vertical space as it needs to display the content inside.

The Box Model

- There are 4 parts of the box model:

2. Padding

   Padding is added to the **top**, **right**, **bottom**, or **left** of the content area.

The Box Model

- There are 4 parts of the box model:

3. Border

   Borders are added around the **top**, **right**, **bottom**, or **left** of the padding

The Box Model

- There are 4 parts of the box model:

4. Margin

   Margins are added to the **top**, **right**, **bottom**, or **left** of the <u>border</u>.

How to calculate the size of the box

The full size of a box after these four properties have been set can be calculated like this:

+ content area width
+ padding-left + padding-right
+ border-left + border-right
+ margin-left + margin-right

= the full box width

use top, bottom, and height to calculate full box height

Applying the box model properties

Apply padding to one side at a time...

CSS

```
h2 {
  padding-top: 6px;
  padding-right: 3px;
  padding-bottom: 0;
  padding-left: 0;
}
```

If we want to put some space above and to the right of some text, one option is to adjust the padding.

...or all at once in a clockwise order

CSS

```
h2 {
  padding: 6px    3px    0    0;
}
```

top right bottom left

Applying the box model properties

You can apply borders all at once and not specify a side.

CSS

```
h2 {
  border-width: 6px;
  border-style: solid;
  border-color: black;
}
```

Creating borders has a shortcut version, too.

CSS

```
h2 {
  border: 6px    solid    black;
}
```

width style color

Applying the box model properties

You can also just pick one side and just add a border there.

CSS

```
h2 {
  border-bottom: 6px    solid    black;
}
```

width style color

Applying the box model properties

Add margins the same way you add padding...

CSS

```
h2 {
  margin-top: 6px;
  margin-right: 0;
  margin-bottom: 6px;
  margin-left: 0;
}
```

...or use the same shortcut syntax as padding.

CSS

```
h2 {
  margin:   6px    0    6px    0;
}
            top  right bottom left
```

When should you use padding?

Padding is used to control the size of a box *without* adjusting the size of the content inside the box.

CSS

```
h2 {
  padding: 0 0 0 0;
}
```

CSS

```
h2 {
  padding: 10px 0 10px 0;
}
```

When should you use margin?

Margin is used to control the space between boxes.

CSS

```
h2 {
  margin: 20px 0 20px 0;
}
```

CSS

```
h2 {
  margin: 4px 0 4px 0;
}
```

Summary of our new box model knowledge

What we now know:

- Each tag's content fits in an invisible box
- Each block-level tag's box takes up an entire line
- Padding can be used to adjust spacing within a container
- Margin can be used to adjust spacing between containers

Why is there spacing here?

We didn't set any margins or padding?!@!#?#

Default browser styles

Browsers actually have a default stylesheet for when no custom styles are set.

Default box styles left on

Default box styles turned off

Resetting default browser styles

Add this to the very top of your CSS file to reset default styles.

include each tag that's in your HTML for that page

CSS

```
html, body, h1, h2,
h3, p, ol, ul, li, a {
  padding: 0;
  border: 0;
  margin: 0;
}
```

Now, all of the default box properties are reset, and we need to set the box model properties ourselves.

Default box styles turned off

A Box Model process - Start with the body

The order that you apply the box model properties is personal preference, but here's one approach:

Start from the highest parent element
This is often **<body>**

```
body {
  padding: 20px 20px 20px 20px;
}
```

CSS

This padding has the effect of pushing all of the children away from the edges of the **<body>**.

this padding is applied to every side of the body

A Box Model process - Next, look at headings

Next, focus on the heading tags

<h1>

CSS

```
h1 {
  margin: 10px 0 15px 0;
}
```

Use margin to put vertical space between tags.

Before

no margins

After

margins

A Box Model process - Next, look at headings

Next, focus on the heading tags

<h2>

CSS

```
h2 {
  margin: 10px 0 20px 0;
}
```

Before

no margins

Home Recipes Suggest

Magic Cake

Magic Cake is one of the tastiest cakes the world ha
to tell about it, and these are their stories.
We also need to stress that whatever happens to you

After

margins

Home Recipes Suggest

Magic Cake

Magic Cake is one of the tastiest cakes the world ha
to tell about it, and these are their stories.

A Box Model process - Next, look at headings

Next, focus on the heading tags

<h3>

CSS

```
h3 {
  margin: 15px 0 15px 0;
}
```

no margins

Before

Magic Cake is one of the tastiest cakes
about it, and these are their stories.
We also need to stress that whatever hap
responsibility. You've been warned, and

Ingredients

2 eggs 1 pound of sugar 3 sticks of butte

margins

After

Magic Cake is one of the tastiest cakes
to tell about it, and these are their stories.
We also need to stress that whatever happ
your responsibility. You've been warned,
information.

Ingredients

2 eggs 1 pound of sugar 3 sticks of butter

A Box Model process - borders

After your margins are in place, think about where borders might be necessary.

CSS

```
h3 {
  margin: 15px 0 15px 0;
  border-bottom: 1px solid #CCCCCC;
}
```

this bottom border visually separates the sections

A Box Model process - padding

Use padding to put more space between the content and border.

CSS

```
h3 {
  margin: 15px 0 15px 0;
  border-bottom: 1px solid #CCCCCC;
  padding-bottom: 3px;
}
```

Before

After

now we've got a little
more space between
the content and border

A Box Model process - padding

Padding is also used to adjust list and list item indentation.

Before

Ingredients

2 eggs 1 pound of sugar 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

After

Ingredients

2 eggs 1 pound of sugar 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

A Box Model process - padding

Padding is also used to adjust list and list item indentation.

After

Ingredients

2 eggs 1 pound of sugar 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

CSS

```
ul {
  padding: 0 0 0 50px;
}

ol {
  padding: 0 0 0 50px;
}
```

left padding

Result after adjusting box model properties

Here's where our page stands right now:

Recipe World

Home Recipes Suggest

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and live
We also need to stress that whatever happens to you as a result of consuming Magic Cake is
view our legal page for more information.

Ingredients

2 eggs 1 pound of sugar 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

FRONT-END FOUNDATIONS

Fixing page style problems

PROBLEM

The **<ul>** padding is being applied to the navigation and ingredients.

We need a way to add padding to **one** <ul> but <u>not</u> the other.

Using classes to differentiate between tags

Add a **class** attribute to an HTML tag so that you can access it directly in CSS...

HTML

```
<ul class="nav">
```

... then access this class in CSS by putting a period in front of the class name.

CSS

```
.nav {
  padding-left: 0;
}
```

padding-left is gone from the nav <ul>

padding-left remains on the ingredients <ul>

Using classes in descendent selectors

Classes can be used interchangeably with tags, so the way descendant selectors work doesn't change.

HTML

```html
<ul class="nav">
  <li><a href="home.html">Home</a></li>
  <li><a href="recipes.html">Recipes</a></li>
  <li><a href="suggest.html">Suggest</a></li>
</ul>
```

CSS

```css
.nav a {
  color: #0000af;
  text-decoration: none;
}
```

This selector says *find all **a** tags that are children of **any tag** with a **class of "nav"***, and set the color to blue and turn off the underline.

Using classes to differentiate between tags

Problem: We don't want the ingredients list to display inline

our current rule for list items

CSS

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

should be the default block-level, not inline

Using classes to differentiate between tags

Classes can be used interchangeably with tags, so the way descendant selectors work doesn't change.

CSS

```
ul li {
  display: inline;
}
.nav li {
  display: inline;
}
```

still displayed inline

displaying as a block-level tag again (the default)

Writing class and type selectors in the right order

When you're just using type selectors, the order doesn't really matter.

When you start combining them with class selectors - the order matters.

CSS

```
.nav {
  padding-left: 0;
}

ul {
  padding: 0 0 0 50px;
}
```

padding-left!!!

Padding for **.nav** before **ul** rule

```
padding-left: 0;
```

Padding for **.nav** after **ul** rule

```
padding-left: 50px;
```

This **ul** padding overrides the **.nav** padding because it comes <u>after</u> the **.nav** in the stylesheet.

Writing class and type selectors in the right order

Generally speaking, **first declare the broadest rules with type selectors**, and **then get more specific with class selectors**.

flip them so the ul comes first, and then the .nav

CSS

```
ul {                broader
  padding: 0 0 0 50px;
}

.nav {              more specific
  padding-left: 0;
}
```

Padding for **.nav**

before **.nav** rule

```
padding-left: 50px;
```

Padding after **.nav** rule

```
padding-left: 0;
```

This way, the broader rules **cascade** down and become defaults, and more specific rules can change the defaults.

Creating a page that lists all recipes

Our recipes.html page will have a list of links to individual recipe pages.

recipes.html

HTML

```html
<html>
  <head>
  </head>
  <body>
    <h1>Recipe World</h1>
    <ul class="nav">
      <li><a href="home.html">Home</a></li>
      <li><a href="recipes.html">Recipes</a></li>
      <li><a href="suggest.html">Suggest</a></li>
    </ul>
    <h2>Recipes</h2>
  </body>
</html>
```

right now, this page just has the main site navigation and an h2 describing the page content

Creating the list of recipes

The list of recipes can be an unordered list where each list item contains a few other block-level tags and a link to the individual recipe page.

recipes.html

HTML

```
<h2>Recipes</h2>
<ul>
  <li>
    <h3><a href="magic-cake.html">Magic Cake</a></h3>
    <p>This is a dangerously delicious cake.</p>
  </li>
  <li>...</li>
  <li>...</li>
  <li>...</li>
</ul>
```

copy this general structure for each recipe

The recipes page has a problem

Problem:

None of our styles are in recipes.html because we put our CSS in the **<head>** of magic-cake.html.

Solution:

We need to put our CSS in a separate file and link that file to each of our HTML pages.

all of our CSS rules are missing

Move the CSS into a single file

Here we've created a single file called main.css so we have one place to put all of our CSS.

magic-cake.html

HTML

```
<html>
  <head>
    <style type="text/css">
      a {
        color: red;
      }
      all other styles
    </style>
  </head>
  ...
</html>
```

main.css

CSS

Move all of the selectors and rules into the CSS file, but **not** the HTML <style> tag.

Visualizing our site files

Recipe World server files

main.css will hold all of our CSS and be linked in the **<head>** of every page.

main.css

CSS

```
a {
  color: red;
}
all other styles
```

Where do you write CSS?

While CSS can be written inside of a **<style>** tag in an HTML file, most of the time it's written in a separate file and connected to the HTML file with a **link** tag.

recipes.html

HTML

```
<html>
  <head>
    <link>
  </head>
  <body>
    . . .
  </body>
</html>
```

The link tag should be a child of the head tag

The link tag is an **empty** tag, meaning it is written with only an opening tag and has no closing tag.

```
<link></link>
```

Where do you write CSS?

Since the tag is empty, the way you use it is by setting different **attributes**.

recipes.html

HTML

```
<html>
  <head>
    <link type="text/css">
  </head>
  <body>
    ...
  </body>
</html>
```

this is called a MIME type

The **type** attribute set to **text/css** lets the browser know we're going to be loading a CSS file.

Where do you write CSS?

Since the tag is empty, the way you use it is by setting different **attributes**.

recipes.html

HTML

```html
<html>
  <head>
    <link type="text/css" rel="stylesheet">
  </head>
  <body>
    ...
  </body>
</html>
```

The **rel** attribute is short for **relationship**.

Setting rel to **stylesheet** tells the HTML that it should use the linked file to determine page styles.

Where do you write CSS?

Since the tag is empty, the way you use it is by setting different **attributes**.

recipes.html

HTML

```
<html>
  <head>
    <link type="text/css" rel="stylesheet" href="main.css">
  </head>
  <body>
    ...
  </body>
</html>
```

this is a relative path, so we know here that main.css should be in the same folder as our HTML files

The **href** attribute works just like it does with the **<a>** tag.

Before-and-after linking the CSS file

Now recipes.html is using the general styles we've already created

Before linking main.css

Recipe World

- Home
- Recipes
- Suggest

Recipes

- **Magic Cake**

  This is a dangerously delicious cake.

- **Tacosaurus Mex**

  It's always taco night somewhere!

- **Florida Omelette**

  Omelette you in on a little secret - this is one tasty egg.

- **Mystery-Wich**

After linking main.css

Recipe World

Home Recipes Suggest

Recipes

- **Magic Cake**

  This is a dangerously delicious cake.

- **Tacosaurus Mex**

  It's always taco night somewhere!

- **Florida Omelette**

  Omelette you in on a little secret - this is one tasty egg.

- **Mystery-Wich**

Dealing with similar styles on multiple pages

Often using the exact same styles on all pages doesn't look right

recipes.html

HTML

```html
<h2>Recipes</h2>
<ul>
  <li>
    <h3><a href="magic-cake...
    <p>This is a dangerously...
  </li>
  <li>...</li>
  <li>...</li>
  <li>...</li>
</ul>
```

The **ul**, **ul li**, **h3**, and **p** styles are cascading down from our broader selectors.

Recipe World

Home Recipes Suggest

Recipes

- **Magic Cake**

  This is a dangerously delicious cake.

- **Tacosaurus Mex**

  It's always taco night somewhere!

- **Florida Omelette**

  Omelette you in on a little secret - this is one tasty egg.

- **Mystery-Wich**

  It's a sandwich. That's all we can really say right now.

A proposal for changing the recipes page styles

no list-style-type for the <ul>

border around the <li>

left padding between the li border and content area

no border under the h3

our green heading color used for the <a> tag

Dealing with similar styles on multiple pages

First, we'll give the recipes.html **<ul>** a unique class.

recipes.html

HTML

```html
<h2>Recipes</h2>
<ul class="recipes">
  <li>
    <h3><a href="magic-cake.html">Magic Cake</a></h3>
    <p>This is a dangerously delicious cake.</p>
  </li>
  <li>...</li>
  <li>...</li>
  <li>...</li>
</ul>
```

Dealing with similar styles on multiple pages

Then, write some more specific CSS for **.recipes** and any children.

existing ul styles

CSS

```
ul {
  padding: 0 0 0 50px;
}
```

remember, .recipes is a specific <ul>

CSS

```
.recipes {
  list-style-type: none;
  padding: 0;
}
```

resets all sides to 0

Recipes

Magic Cake

This is a dangerously delicious cake.

Tacosaurus Mex

It's always taco night somewhere!

Florida Omelette

Omelette you in on a little secret - this is one tasty egg.

Mystery-Wich

It's a sandwich. That's all we can really say right now.

Dealing with similar styles on multiple pages

Then, write some more specific CSS for **.recipes** and any children.

existing ul li styles

CSS

```
ul li {
}
```

the .recipes <li>'s will inherit the margins and add a new border and padding-left

CSS

```
.recipes li {
  border: 1px solid #b56663;
  padding-left: 15px;
}
```

Dealing with similar styles on multiple pages

Then, write some more specific CSS for **.recipes** and any children.

existing h3 styles

CSS

```
h3 {
  margin: 15px 0 15px 0;
  border-bottom: 1px solid #cccccc;
  padding-bottom: 3px;
}
```

the .recipes h3 will inherit all of the h3 properties, but remove the bottom border

CSS

```
.recipes h3 {
  border-bottom: 0;
}
```

Recipes

Magic Cake

This is a dangerously delicious cake.

Tacosaurus Mex

It's always taco night somewhere!

Florida Omelette

Omelette you in on a little secret - this is one tasty egg.

Mystery-Wich

It's a sandwich. That's all we can really say right now.

Dealing with similar styles on multiple pages

Then, write some more specific CSS for **.recipes** and any children.

We don’t currently have a general CSS rule for **a**, so we can just add our specific version.

CSS

```
.recipes a {
  color: #7facaa;
  text-decoration: none;
}
```

FRONT-END FOUNDATIONS

Organizing the page layout

In our current layout, we've got a clear separation between the header and main content.

The div tag

div is short for **division**, and it's a block-level tag that's a generic way to group related content into sections on a page.

HTML

```
<body>
  <div>        header
    <h1>Recipe World</h1>
    <ul class="nav">...
  </div>
  <div>        main content
    <h2>Magic Cake</h2>
    ...
  </div>
</body>
```

Use classes to describe what divs contain

A common way to differentiate between tags is to add a class.

HTML

```
<body>
  <div class="header">
    <h1>Recipe World</h1>
    <ul class="nav">...
  </div>
  <div class="main-content">
    <h2>Magic Cake</h2>
    ...
  </div>
</body>
```

let's build this style

Classes will let us style each differently

Use classes to describe what divs contain

A common way to differentiate between tags is to add a class.

let the divs handle padding now and not the body

HTML

```html
<body>
  <div class="header">
    <h1>Recipe World</h1>
    <ul class="nav">...
  </div>
  <div class="main-content"
    <h2>Magic Cake</h2>
    ...
  </div>
</body>
```

CSS

```css
body {
  padding: 0 0 0 0;
}
.header {
  padding: 10px 10px 10px 10px;
}
.main-content {
  padding: 20px 20px 20px 20px;
  border: 1px solid #dddddd;
  margin: 30px 0 0 0;
}
```

Giving the header a separate style

Let's make our header styles look different so we can easily see that the **header** is different from the **main-content**.

CSS

```
.header {
  padding: 10px 10px 10px 10px;
  background-color: #b56663;
}
.header a {
  color: #ffffff;
}
h1 {
  color: #ffffff;
}
```

Adjusting the size of divs

The width and height of any tag's content area can be changed in CSS.

So far, we've just wanted our tags to take up 100% of the available width, but let's make our **main-content** div a fixed size.

CSS

```
.main-content {
  width: 500px;
  padding: 20px;
  border: 1px solid #dddddd;
  margin: 30px 0 0 0;
}
```

Centering content

There's two main ways to center things, but the CSS you write depends on what you're centering.

If: you want to center an <u>entire</u> block-level tag

And: it is fixed-width

Then: set the left and right margins to auto

CSS

```
.main-content {
  width: 500px;
  padding: 20px;
  border: 1px solid #dddddd;
  margin: 30px auto 0 auto;
}
```

Centering with margin auto

CSS

```
margin: 30px auto 0 auto;
```

right left

auto here means *take as much space as you can.*

Centering content

There's two main ways to center things, but the CSS you write depends on what you're centering.

If: you want to center children <u>inside</u> a block-level tag

Then: set the **text-align** of the children to **center**

CSS

```
h1 {
  color: #ffffff;
  margin: 0 0 15px 0;
  text-align: center;
}
```

centered!

Centering content

There's two main ways to center things, but the CSS you write depends on what you're centering.

CSS

```
.nav {
  padding-left: 0;
  margin: 5px 0 20px 0;
  text-align: center;
}
h2 {
  margin: 10px 0 20px 0;
  text-align: center;
}
```

centered!

A note about fixed widths

If you've been reading about or making websites, you might be wondering why we're using fixed widths instead of relative widths.

Fixed widths are great for learning how page layout works, and after this course, you can continue on to our **Journey Into Mobile** course where you'll learn how to convert your fixed widths to relative ones!

FRONT-END FOUNDATIONS

LEVEL 4
IMAGES

The three types of webpage images

Most images on webpages fall into one of these three categories

- Content Images
- Layout Images
- User Interface Images

Content images

Content images are any images that are just as necessary to the page as all of the text.

Layout images

These are in the background, and are not necessary to understand the content of the page.

actual important content

User Interface images

These assist with the interface of the webpage, but are not required to understand the content of the page.

These downward arrows direct how to work with the interface.

Creating content images

Content images are created in HTML with the **<img>** tag.

```
<img>
```

<img> is an **empty tag**.

```
<img></img>
```

remember, that means it just has an opening tag (no closing)

Creating content images

Content images are created in HTML with the **<img>** tag.

magic-cake.html

HTML

```
<img src="cake.png">
```

relative path

Instead of open/close tags, add an **src attribute** to set which image is used.

Recipe World server files

Putting images in their own folder

It makes sense to organize your images so they aren't in the main site folder.

magic-cake.html

HTML

```
<img src="images/cake.png">
```

Recipe World server files

now all of your images can go in the images folder

How images are loaded

Adding alt attributes to content images

The alt attribute should contain text that describes the image or the purpose of the image.

magic-cake.html

HTML

```
<img src="images/cake.png" alt="Magic Cake Photo">
```

Why?? **Accessibility**

Not everyone who views your page will be able to view the images.

Screen readers rely on alt tags to describe images to visitors that might not be able to view images.

Laying out content images

<img> is an inline-level tag, but they are often put inside block-level tags.

This is similar to **<a>** tags, which almost always appear as children of block-level tags.

magic-cake.html

HTML

```
<h3>Your Photos</h3>
<ul class="photos">
  <li><img src="images/cake1.png" alt="Cake Photo 1"></li>
  <li><img src="images/cake2.png" alt="Cake Photo 2"></li>
  <li><img src="images/cake3.png" alt="Cake Photo 3"></li>
</ul>
```

<li> is a block-level tag

Laying out content images

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

Your Photos

-
-

Problem

We want these photos to display side-by-side, like this

Directions

1. Mix eggs, sugar, and butter in a large bowl.
2. Spread into a non-stick dish
3. Bake at 350 degrees for 1 hour
4. Let sit at room temperature for 20 minutes
5. Eat and enjoy!

Your Photos

Laying out content images

magic-cake.html

HTML

```html
<h3>Your Photos</h3>
<ul class="photos">
  <li><img...
</ul>
```

CSS

```css
.photos {
  list-style-type: none;
  padding: 0;
}
.photos li {
  display: inline;
  padding-left: 11px;
}
```

remove bullets and padding

make the images show horizontally with a little space in between

Using an image as a site/logo mark

We can replace the **<h1>** with an **<img>** tag.

magic-cake.html

HTML

```
<div class="header">
  <h1>Recipe World</h1>
  <ul class="nav">...
```

HTML

```
<div class="header">
  <img src="images/logo.png" alt="Recipe World Logo">
  <ul class="nav">...
```

Centering an image inside a block-level tag

<img> is an inline-level tag so it can't be centered with text-align.

Instead, set just that image to **display: block** and center it with the **margin: auto** approach.

magic-cake.html

HTML

```html
<div class="header">
  <img src="images/logo.png" alt="Recipe World Logo">
  <ul class="nav">...
```

CSS

```css
.header img {
  display: block;
  margin: 0 auto 0 auto;
}
```

FRONT-END FOUNDATIONS

Creating layout images

Layout images are created in CSS with the **background** property.

Understanding the background property

There’s several different properties that relate to background images.

background-color works just like the color property.

CSS

```
body {
  background-color: #5f5f5f;
}
```

TIP

You should always set a background color in case the background image fails to load.

Understanding the background property

background-image can point to a relative or absolute path.

CSS

```
body {
  background-color: #5f5f5f;
  background-image: url(images/gobbler.png);
}
```

the path has to be inside this url() code

notice that quotes don't need to be around the path

Check out subtlepatterns.com for this and other patterns!

Understanding the background property

background-position can be used to position the image in a container.

CSS

```
body {
  background-color: #5f5f5f;
  background-image: url(images/gobbler.png);
  background-position: top left;
}
```

the first word can be **top**, **center**, or **bottom**

the second word can be **left**, **center**, or **right**

Understanding the background property

background-position can be used to position the image in a container.

CSS

```
body {
  background-color: #5f5f5f;
  background-image: url(images/gobbler.png);
  background-position: center right;
}
```

this shows the image vertically centered and pushed to the right

Understanding the background property

background-repeat can be used to repeat tiled images.

CSS

```
body {
  background-color: #5f5f5f;
  background-image: url(images/gobbler.png);
  background-position: top left;
  background-repeat: repeat;
}
```

Understanding the background property

background-repeat can be used to repeat tiled images.

CSS

```
body {
  background-color: #5f5f5f;
  background-image: url(images/gobbler.png);
  background-position: top left;
  background-repeat: repeat-y;
}
```

Other values

repeat-x	tile the image horizontally
repeat-y	tile the image vertically
no-repeat	don’t tile or repeat anything

Understanding the background property

All four of those can be shortened into a single **background** property.

CSS

```
body {
  background-color: #5f5f5f;
  background-image: url(images/gobbler.png);
  background-position: top left;
  background-repeat: repeat;
}
```

CSS

```
body {
  background: #5f5f5f url(images/gobbler.png) top left repeat;
}
```

color image position repeat

Creating different tiled backgrounds

We can set different backgrounds for different containers.

Here, we're using one repeated background for the body, another for the main-content, and another for the divs on the recipes page.

CSS

```
body {
  background: #000000 url(images/stardust.png) top left repeat;
}
.main-content {
  background: #ffffff url(images/crossword.png) top left repeat;
}
.recipes li {
  background: #ffffff url(images/li-bg.png) top left repeat;
}
```

Result of using three different layered backgrounds

body background

.main-content background

.recipes li background

Writing on top of background images

Some sites like to write text over images.

This list on gamespot.com has text for the game name and release date appearing over a background image from each game.

Writing on top of background images

Some sites like to write text over images.

One way to do this is to create an empty **div** container that completely contains a background image, and put text tags inside of it.

home.html

HTML

```
<div class="main-content">
  <h2>Home</h2>
  <div class="featured-image">
  </div>
</div>
```

First, add the background image to the container

home.html

HTML

```html
<div class="main-content">
  <h2>Home</h2>
  <div class="featured-image">
  </div>
</div>
```

CSS

```css
.featured-image {
  width: 630px;
  height: 246px;
  background: #ffffff url(images/featured-cake.png)
              top left no-repeat;
}
```

these should be equal to the width/height of the image

Then, add a tag that shows text as a child of that container

home.html

HTML

```html
<div class="featured-image">
  <h3>Featured: Magic Cake</h3>
</div>
```

CSS

```css
.featured-image h3 {
  margin: 0;
  background-color: #333333;
  color: #ffffff;
  padding: 5px 0 5px 15px;
  text-transform: uppercase;
}
```

FRONT-END FOUNDATIONS

Displaying an image to the side of block-level tags

We know that tags are either block or inline.

This is an image to left of multiple block-level tags.

How can we make this?

Floating images

You might try setting two divs to inline.

left div set to inline

right div set to inline

Floating images

HTML

magic-cake.html

```html
<ul class="recipes">
  <li>
    <div>
      <img src=...>
    </div>
    <div>
      <h3><a href=...>Magic Cake</a></h3>
      <p>...</p>
    </div>
  </li>
</ul>
```

CSS

```css
.recipes div {
  display: inline;
}
```

Setting these two divs side-by-side with **display:inline** seems like it would work.

Floating images

Turns out, we can't just flip things to inline and then put block-level tags inside of them.

Time to try something else.

left div set to inline

right div set to inline

Floating images

Put the img, h3, and p all in one li and **float** the img left.

magic-cake.html

HTML

```html
<ul class="recipes">
  <li>
     <img src=...>
     <h3><a href=...>Magic Cake</a></h3>
     <p>...</p>
  </li>
</ul>
```

CSS

```css
.recipes img {
  float: left;
}
```

This makes the image take up some of the left space that the **h3** and **p** boxes would normally eat up as block-level tags.

Floating images

Success! We've also added some right padding so the image doesn't touch right up against the text.

CSS

```
.recipes img {
  float: left;
  padding-right: 10px;
}
```

Float is a tricky property that is covered in much greater detail in the CSS Cross Country course!

Magic Cake

This is a dangerously delicious cake.

FRONT-END FOUNDATIONS

LEVEL 5
FONTS AND FORMS

Fonts on the Web

Fonts and font styles can be controlled with CSS.

Our current page's fonts

Since we haven't set a font, the browser is just picking a default.

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

We also need to stress that whatever happens to you as a result of consuming Magic Cake is your responsibility. You've been warned, and you can view our legal page for more information.

Ingredients

- 2 eggs
- 1 pound sugar
- 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl

Changing the font with CSS

The **font-family** property is used to set the ideal font and a few fallback options.

CSS

```
body {
  font-family: Helvetica, Arial, sans-serif;
}
```

The ideal font

The browser will try to load the ideal font first.

Changing the font with CSS

The fallback fonts are used if the ideal font isn't available.

CSS

use quotes if fonts are multi-word

```
body {
  font-family: Helvetica, Arial, "Times New Roman", sans-serif;
}
```

Fallback fonts

Not all fonts are installed in all browsers!!!

Our page after changing the font

Here’s magic-cake.html after changing the main font to Helvetica.

Magic Cake

Magic Cake is one of the tastiest cakes the world has ever known. Few have tried it and lived to tell about it, and these are their stories.

We also need to stress that whatever happens to you as a result of consuming Magic Cake is your responsibility. You've been warned, and you can view our legal page for more information.

Ingredients

- 2 eggs
- 1 pound sugar
- 3 sticks of butter

Directions

1. Mix eggs, sugar, and butter in a large bowl

Browser defaults - the font edition

Just like with the box model properties, the browser has default font styles - and you should reset these too!!

These weights and sizes are the browser defaults.

Resetting default browser font styles

Now the font weights and sizes won't follow an arbitrary default.

CSS

```
html, body, h1, h2,
h3, p, ol, ul, li, a {
  padding: 0;
  border: 0;
  margin: 0;
  font-size: 100%;
  font: inherit;
}
```

Changing font sizes

The size can be set with the **font-size** property.

CSS

```
h2 {
  color: #7facaa;
  margin: 0 0 20px 0;
  text-align: center;
  font-size: 26px;
}
```

Pixels aren't the only size options

percentages

ems

Changing font thickness

The thickness can be set with the **font-weight** property.

CSS

```
h2 {
  color: #7facaa;
  margin: 0 0 20px 0;
  text-align: center;
  font-size: 26px;
  font-weight: bold;
}
```

Weight values

normal	lighter
bold	bolder
100, 200... 900	

Changing other font properties

Here's a few other font properties:

CSS

```
.footer p {
  color: #aaaaaa;
  text-align: center;
  font-weight: bold;
  font-size: 12px;
  font-style: italic;
  text-transform: uppercase;
}
```

BEFORE

AFTER

HTML

```
<div class="footer">
  <p>&copy; 2014 - Recipe World</p>
</div>
```

snazzy way to show a copyright symbol

Adjusting the line height

line-height is like margin for each line in a box.

You can adjust the vertical spacing between different text tags by changing the margin.

margin

Margin can’t adjust the spacing between lines in a single box.

Adjusting the line height

line-height is like margin for each line in a box.

CSS

```
.main-content p {
  line-height: 16px;
}
```

Magic Cake is one of the tastiest cakes
lived to tell about it, and these are their

We also need to stress that whatever h
is your responsibility. You've been war
information.

CSS

```
.main-content p {
  line-height: 26px;
}
```

Magic Cake is one of the tastiest cakes

lived to tell about it, and these are their

We also need to stress that whatever h

is your responsibility. You've been war

information.

FRONT-END FOUNDATIONS

Web Forms

Forms are a way for a web page to get input from a user.

Forms usually contain things like labels, inputs of several different types, text areas, and a submit button.

Recipe World's form

Disclaimer

Making forms actually process user input requires server-side code.

We can still design how the form will look in the browser now.

The HTML for a simple form

This HTML will produce the form shown below *(but some extra attributes would needed to send the form data to the server).*

suggest.html

HTML

```html
<form>
  <label>Recipe Name</label>
  <input type="text">
  <input type="submit" value="Click to Submit">
</form>
```

Common form input types

The **type** attribute sets the kind of input field that will display.

```
<input type="text">
<input type="checkbox">
<input type="radio">
<input type="file">
<input type="password">
<input type="submit">
```

The for and id attributes

The value of the **for** attribute in the label should be the same as the value of the **id** attribute in an input field to associate the label and input.

Each for/id pair has to be unique on the page.

HTML

this associates the label to the input

```
<form>
  <label for="recipe-name">Recipe Name</label>
  <input type="text" id="recipe-name">
  <input type="submit" value="Click to Submit">
</form>
```

Recipe Name

Click to Submit

Using a textarea instead of an input

input tags should be used for short or single-line user input.

The **textarea** tag is used for multi-line user input.

HTML

textareas also don't need to define a type

```
<form>
  <label for="ingredients">Ingredients</label>
  <textarea id="ingredients"></textarea>
  <input type="submit" value="Click to Submit">
</form>
```

Unlike inputs, **textarea**s need an opening and closing tag.

Styling forms

labels and **input**s are inline-level tags, but it usually makes sense to display one on top of the other like block-level instead of side-by-side.

Before

CSS

```
label, input {
  display: block;
}
```

After

Styling forms

Once you've got both tags displaying as block-level, you can style them just like you would any other block-level tag.

CSS

```
label, input {
  display: block;
}
label {
  margin-bottom: 10px;
}
input {
  width: 500px;
  margin-bottom: 25px;
}
```

Styling the submit button separately

Since the submit button is technically an input tag, our input selector properties are affecting the way it is displayed.

Attribute selectors are a way to style a tag based on one of its attributes.

CSS

```
input {
  width: 500px;
  margin-bottom: 25px;
}

input[type=submit] {
  width: 120px;
  font-size: 30px;
}
```

attrib. name attrib. value

Before

After

Styling inputs

The container around an input is actually just a border, so you can style it with the **border** property.

CSS

```
input[type=text] {
  border: 2px solid #7facaa;
}
```

Styling inputs

To adjust the height of the input, style the text inside.

CSS

```
input[type=text] {
  border: 2px solid #7facaa;
  font-size: 24px;
}
```

Recipe Name

Magic Cake

Your Name

Jon Friskics

Styling inputs

And, since the green line is a border, use padding to put some space between the border and the text.

CSS

```
input[type=text] {
  border: 2px solid #7facaa;
  font-size: 24px;
  padding: 7px;
}
```

Recipe Name

Magic Cake

Your Name

Jon Friskics

Styling textareas

You can set a width and height for **textarea**s, but otherwise the styles behave similar to regular inputs.

CSS

```
textarea {
  width: 500px;
  height: 400px;
  padding: 7px;
  border: 2px solid #7facaa;
  margin-bottom: 25px;
  font-size: 24px;
}
```

Creating a separate style for a checkbox input

It looks awkward if the newsletter **label** and **input** are on separate lines since the checkbox is so small, so we can use attribute selectors again to make just this input and label display inline.

HTML

```html
<form>
  <label for="newsletter">Get Newsletter?</label>
  <input type="checkbox" id="newsletter">
  ...
</form>
```

CSS

```css
input[type=checkbox], label[for=newsletter] {
  display: inline;
}
```

Before

After

The final form

Suggest a recipe

Recipe Name

Ingredients

Your Name

Your Email

Get Newsletter?

Submit the recipe