HTML / CSS Worksheet 2

This week's worksheet we will look at:

Making a `<!DOCTYPE>` for an html5 document

Rundown of html and how to code it in

block level elements:

`<h>` `<p>` `<ol>` `<ul>`

Inline elements

`<em>` `<strong>` `<small>`

Other

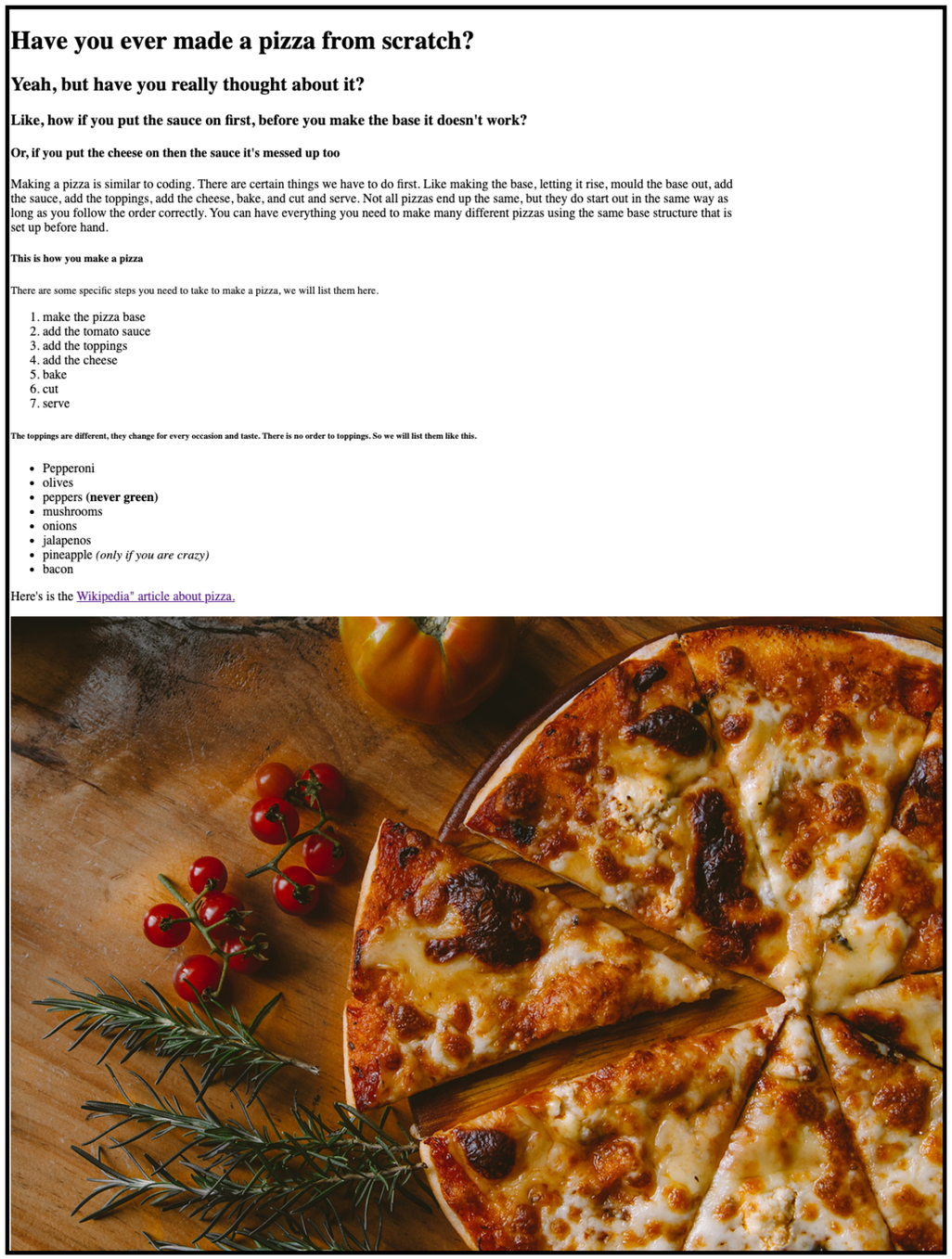
add a **link** `<a>` / add an **image** `<img>` `src` `alt`

GitHub

Visual Studio Code (VSC)

HTML

****Note** in order to succeed, read the ENTIRE STEP FIRST, THEN code everything by hand or by direction. It's the only way to be sure.**



Have you ever made a pizza from scratch?

Yeah, but have you really thought about it?

Like, how if you put the sauce on first, before you make the base it doesn't work?

Or, if you put the cheese on then the sauce it's messed up too

Making a pizza is similar to coding. There are certain things we have to do first. Like making the base, letting it rise, mould the base out, add the sauce, add the toppings, add the cheese, bake, and cut and serve. Not all pizzas end up the same, but they do start out in the same way as long as you follow the order correctly. You can have everything you need to make many different pizzas using the same base structure that is set up before hand.

This is how you make a pizza

There are some specific steps you need to take to make a pizza, we will list them here.

1. make the pizza base
2. add the tomato sauce
3. add the toppings
4. add the cheese
5. bake
6. cut
7. serve

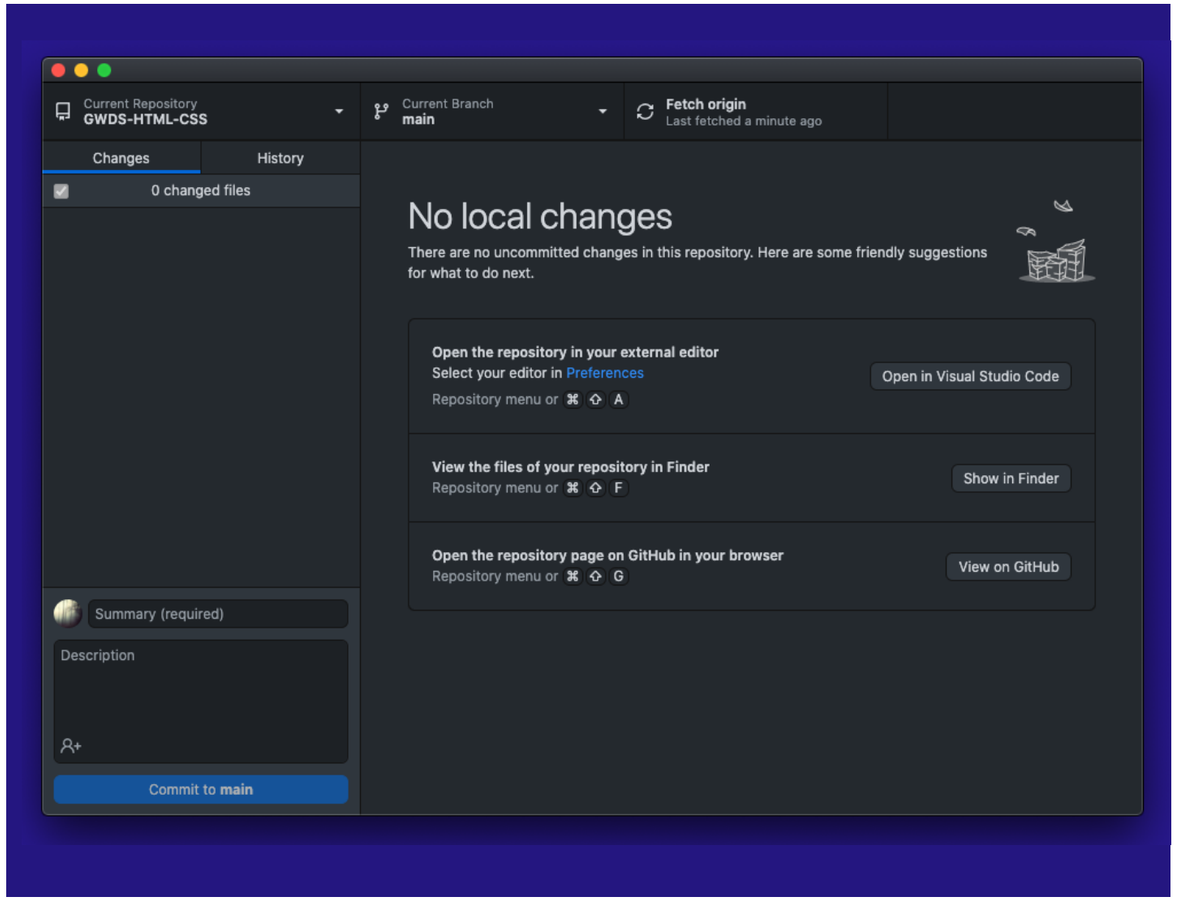
The toppings are different, they change for every occasion and taste. There is no order to toppings. So we will list them like this.

- Pepperoni
- olives
- peppers **(never green)**
- mushrooms
- onions
- jalapenos
- pineapple *(only if you are crazy)*
- bacon

Here's is the Wikipedia" article about pizza.

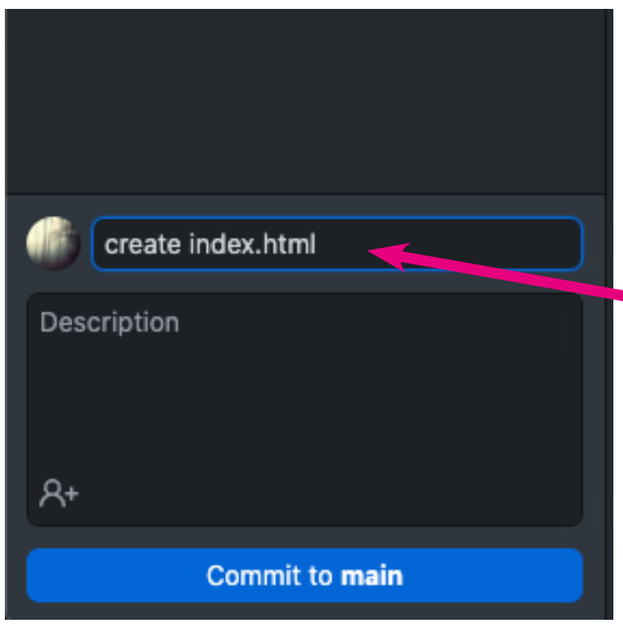
1

And we're back. Let's jump right in. Last week you created your first repository in GitHub. Now we are going to do it again. Make a new repository in GitHub. If you can't remember, go visit the week 1 tutorial pdf.
Name it **HTML/CSS**
¡¡Don't put it inside your first Repo!!
This will also be a practice so you can organise your files for this class.



2

Let's power on. Now that's set up, open VSC and create an index.html file within your new repository. (Again, consult the week 1 tutorial if you don't remember how to do this.) **Name it index.html *save***, and make sure it is in your repo file so GitHub picks it up. Before you commit you will need to make a summary. Then ***commit to main*** and ***push origin***.



it's required, folks.

3

Awesome. Now it's time to start with your `<!DOCTYPE>` This is just a special string that browsers look for when they try to display your web page, and it always needs to look exactly like it does here to the right. Boy howdy that's a lot of typing. BUT

In VSC if you type

`!`

then use your **tab** button on your keyboard VSC will auto fill it for you! Try it now.

TA DA!

Commit to main with a comment (like create doctype), and ***push to origin.*** You have now just made your main html body structure.

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

4

Now let's look at filling some of this:

Like in the chat before your website will be wrapped in an `<html></html>` tag. That's all set up in the right place for us.

The `<head></head>` tag contains all of the metadata, like the page title, any CSS stylesheets, and other things that are required to render the page but you don't really want the user of your website to see. Next week when we get into CSS this will come into play.

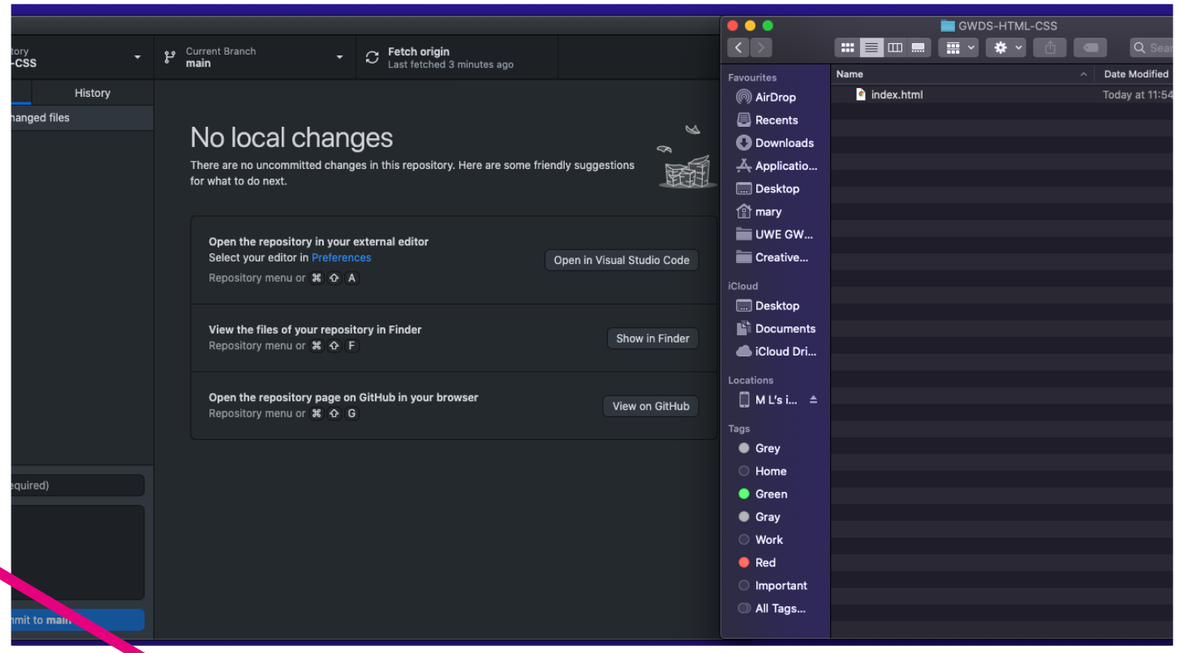
Now we are going to put our first code in. Go to `<title></title>` and type what you see on the right. **Save.** Go to **GitHub Desktop**, add your ***comment***, like **"add title"** ***commit to main***, ***push origin***

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

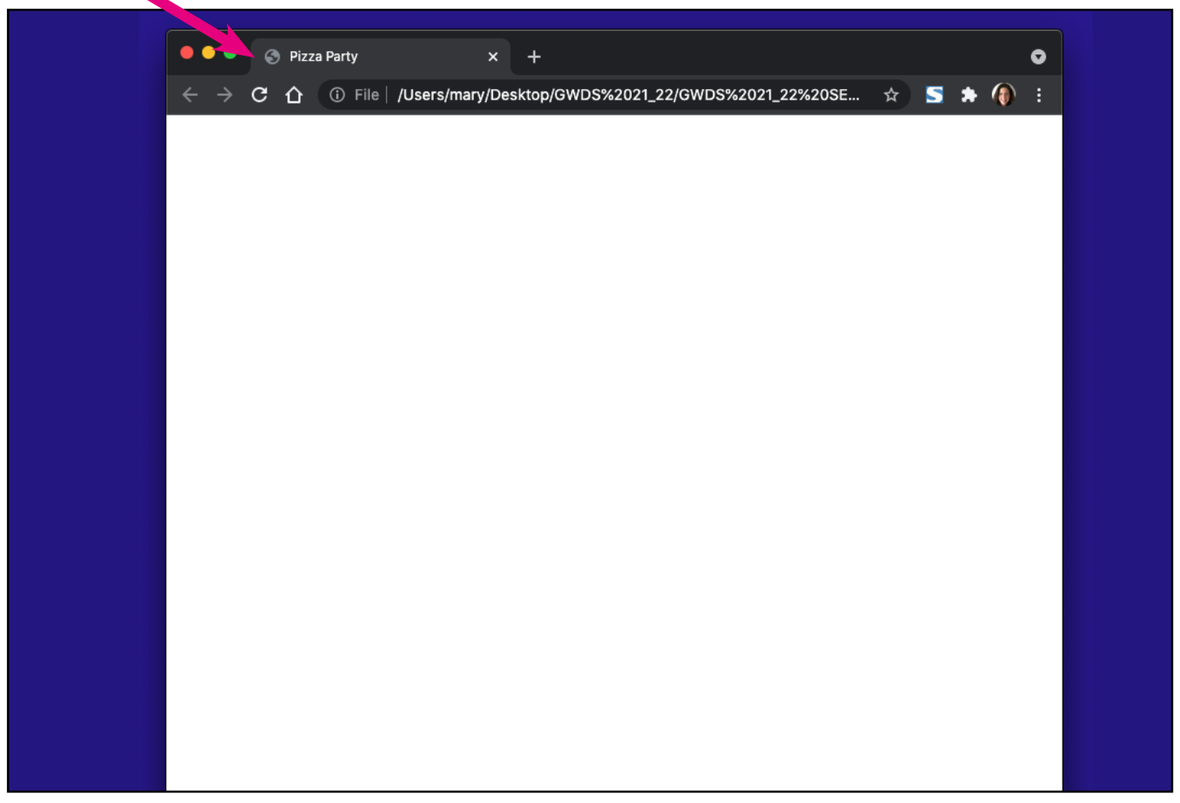
```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible"
    content="IE=edge">
    <meta name="viewport"
    content="width=device-width, initial-scale=1.
    0">
    <title>Pizza Party</title>
</head>
<body>

</body>
</html>
```

But wait! One more thing to do before we go on: **find your file locally.** You can navigate to your file or use “show in finder” using GitHub Desktop. Double click on your file. This is your local file being presented in a webpage. It gives you a preview of what you are working on.



DON’T CLOSE IT! KEEP IT UP.

Leave the web browser page up, and either put it next to your VSC file or somewhere on your desktop you can go and look at quickly as you make changes.



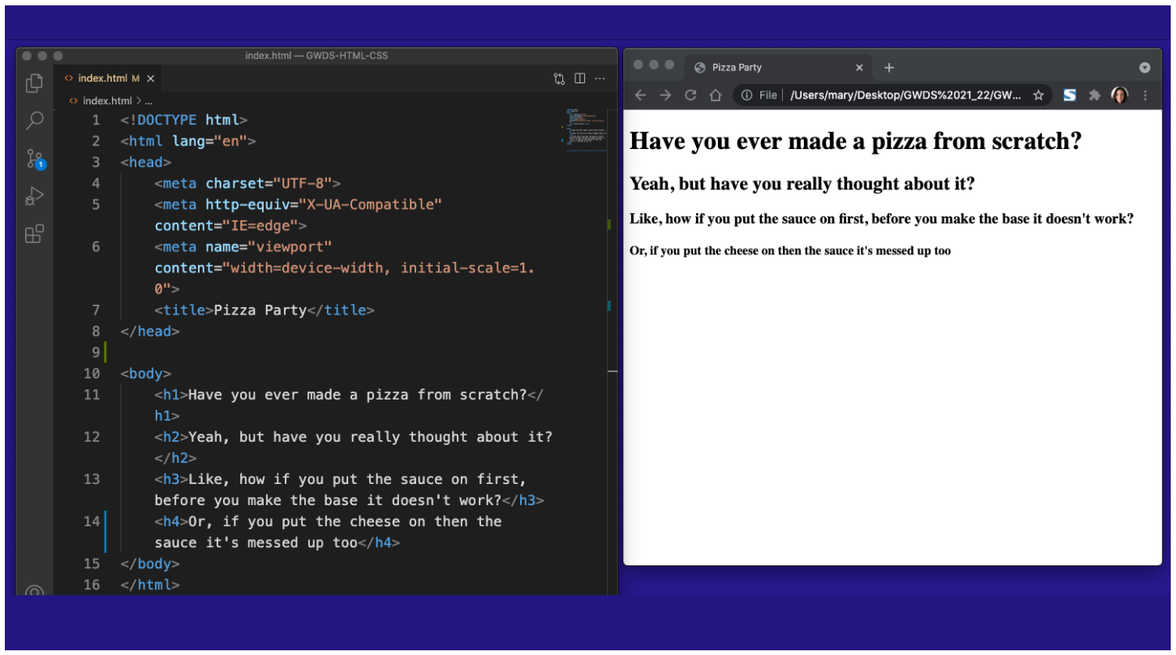
5

Next let's move to the <body></body> tag.
That's where the bulk of our code will end up.
We will be adding some bock level elements.

The first thing we will look at is <h></h>tags.
The **h stands for headings**.
There are six levels of headings, ranging from 1-6. You can learn more about headings here at W3 schools.

Let's add 4 different headings: type exactly what you see here in this window to the right.
Make sure you have an opening <h> tag and a closing </h> tag, and that everything is inside the <body></body> tag.
Save your file.
Commit with a comment (like 'add headings')
Push origin.
Now look at your web browser and hit the refresh button.
(look like this? Good. If not, don't worry, just retrace your steps)

```
</head>

<body>
    <h1>Have you ever made a pizza from scratch?</h1>
    <h2>Yeah, but have you really thought about it?</h2>
    <h3>Like, how if you put the sauce on first, before you make the base it doesn't work?</h3>
    <h4>Or, if you put the cheese on then the sauce it's messed up too</h4>
</body>
</html>
```



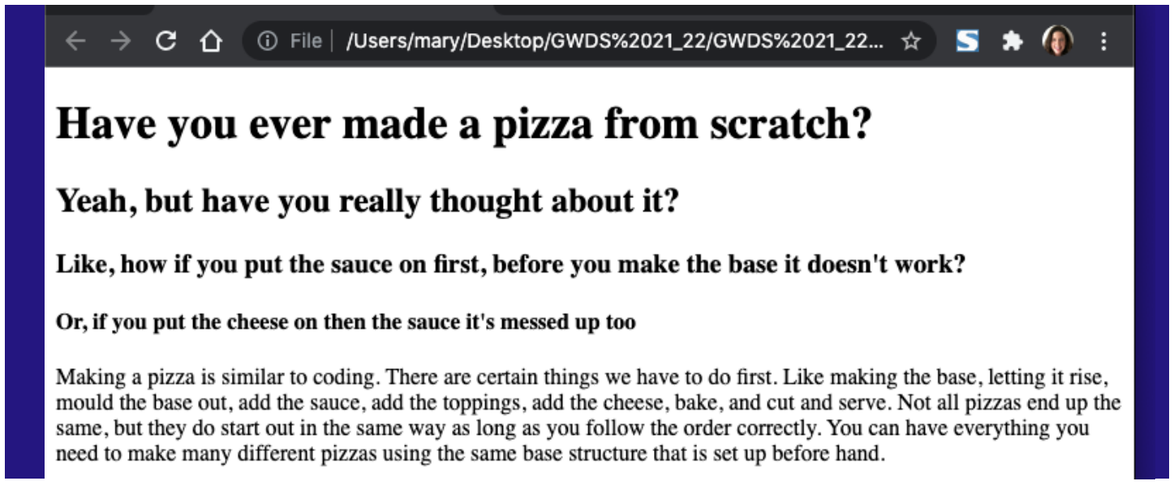
6

Let's move to the tag. **P stands for paragraph.** When you are designing your website, any body copy, or bulk of information or, reflection paragraphs (like in your assignment) will go inside a (or variation of) You can get more info about paragraphs here.

Type exactly what you see in the image to the right. Make sure it is inside the <body></body> and below the headings.
Save your file.
Commit with a comment (like 'add copy')
Push origin.
Now look at your web browser and hit the refresh button.
(look like this? Good. If not, don't worry, just retrace your steps)

What's that? It's a **comment**. It's like a note you can leave for your future self, or for someone who might be also working on your code. No one will see it but you. We'll use those more later.

```
    <h4>Or, if you put the cheese on then the sauce it's
    messed up too</h4>

<!-- Add some copy here -->
    <p>Making a pizza is similar to coding. There are
    certain things we have to do first. Like making the
    base, letting it rise, mould the base out, add the
    sauce, add the toppings, add the cheese, bake, and cut
    and serve. Not all pizzas end up the same, but they do
    start out in the same way as long as you follow the
    order correctly. You can have everything you need to
    make many different pizzas using the same base
    structure that is set up before hand.</p>



</body>
</html>
```




```
15
16  <!-- Add some copy here -->
17      <p>Making a pizza is similar to
```

7

On to lists. Lists do what it says on the tin. They make lists. There are **ordered lists** and **unordered lists**. We'll do both. You can read about lists at W3 schools here. As before, code in exactly what you see in the image on the right and then:

Save your file.

Commit with a comment (like 'add ordered list')

Push origin.

Now look at your web browser and hit the refresh button. You know the drill by now.

```
structure that is set up before hand.</p>


<!--  Make lists here   -->
<h5>This is how you make a pizza</h5>
<p>There are some specific steps you need to take to make a
pizza, we will list them here.</p>

<ol>
    <li>make the pizza base</li>
    <li>add the tomato sauce</li>
    <li>add the toppings</li>
    <li>add the cheese</li>
    <li>bake</li>
    <li>cut</li>
    <li>serve</li>
</ol>



</body>
```

What's going on here?

There is a `<ol></ol>` tag which is an ordered list. It lists things with numbers or letters (which you can find more info on here). Inside the `<ol>` tag is a `<li></li>` tag which is a list item. **It is the individual item that is listed within your ordered list.** They present indented in basic html.

```
<ol>
    <li>make the pizza base</li>
    <li>add the tomato sauce</li>
    <li>add the toppings</li>
    <li>add the cheese</li>
    <li>bake</li>
    <li>cut</li>
    <li>serve</li>
</ol>
```

The `<ol></ol>` always wraps around the `<li></li>`

8

We don't always want numbers, sometimes, like toppings, we just want a list of options. More about unordered lists here. Code in exactly what you see in the image on the right and:

Save your file.

Commit with a comment ('add unordered list')

Push origin.

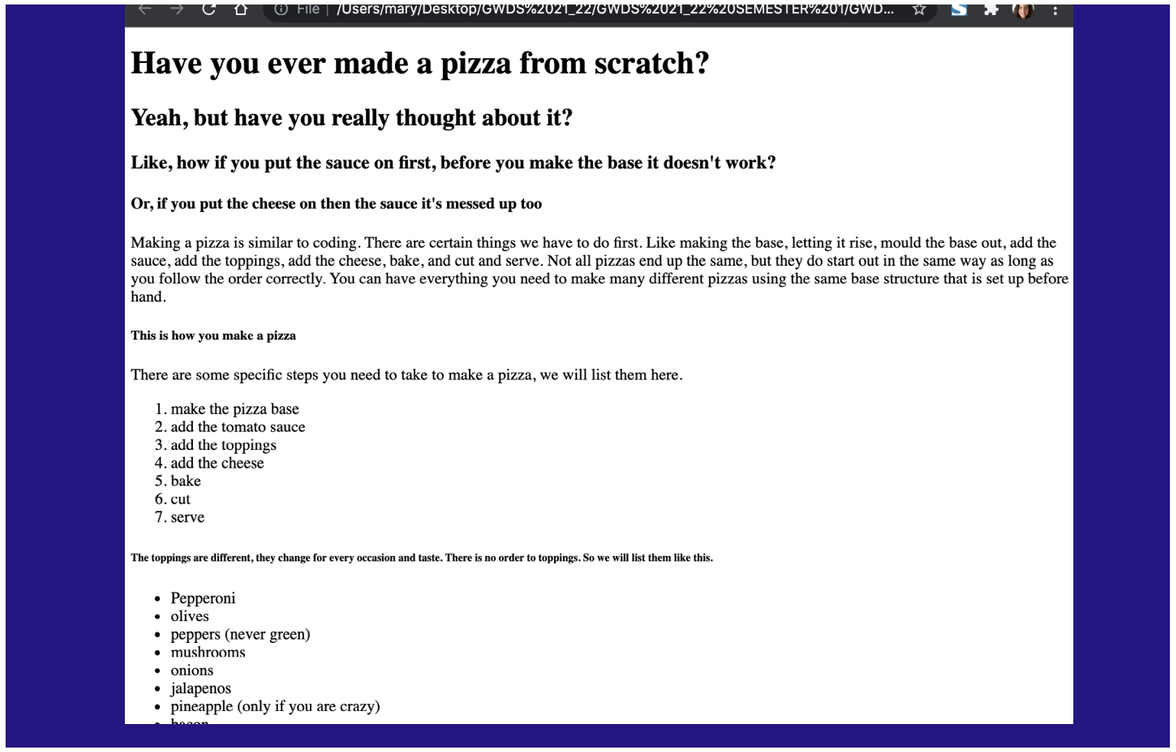
Now look at your web browser and hit the refresh button.

The `<ul></ul>` tag does a similar thing to the `<ol></ol>` but no numbers. The nesting is the same as the ordered list.

If you haven't checked recently, here's where we should be. If you are there, great! Go get a cuppa. If not, no problem, just retrace your steps.

Also for good measure, double check your GitHub Desktop to make sure you've done a most recent ***commit*** and ***push***.

```
    </ol>

    <h6>The toppings are different, they change for every
    occasion and taste. There is no order to toppings. So we
    will list them like this.</h6>

    <ul>
        <li>Pepperoni</li>
        <li>olives</li>
        <li>peppers (never green)</li>
        <li>mushrooms</li>
        <li>onions</li>
        <li>jalapenos</li>
        <li>pineapple (only if you are crazy)</li>
        <li>bacon</li>
    </ul>

</body>
</html>
```



9

Now it's time to add some inline elements. **Inline elements always live inside block elements**. Add the inline elements shown to the right in the correct places, then **Save**, ***Commit*** with comment, and ***push to origin***. **Refresh your web page** and have a look. What's going on here?

The `<em></em>` tag replaced the `<i>` tag. Though it looks like italics, it is something that has emphasis placed upon it.

The `<strong></strong>` tag is for content that is for strong importance. Like how green peppers are disgusting no matter how you look at it.

The `<small></small>` tag is usually for small print, copyright and legal text, but I just added it in here for funsies. It's usually rendered one font size smaller.

They all must have closing tags.

```
        <li>jalapenos</li>
        <li>pineapple <em>(only if you are crazy)</em></li>
        <li>bacon</li>
    </ul>
```

```
    <li>Pepperoni</li>
    <li>olives</li>
    <li>peppers <strong>(never green)</strong></li>
    <li>mushrooms</li>
```

```
<!--  Make lists here   -->
<h5>This is how you make a pizza</h5>
<p><small>There are some specific steps you need to take to make a pizza, we will list them here.</small></p>
```

WRONG

```
<h5>This is how you make a pizza</h5>
<small><p>There are some specific steps you need to take to make a pizza, we will list them here.</p></small>
```

WHY IS THIS WRONG?

10

Only a few more things to do before we are done for this week.

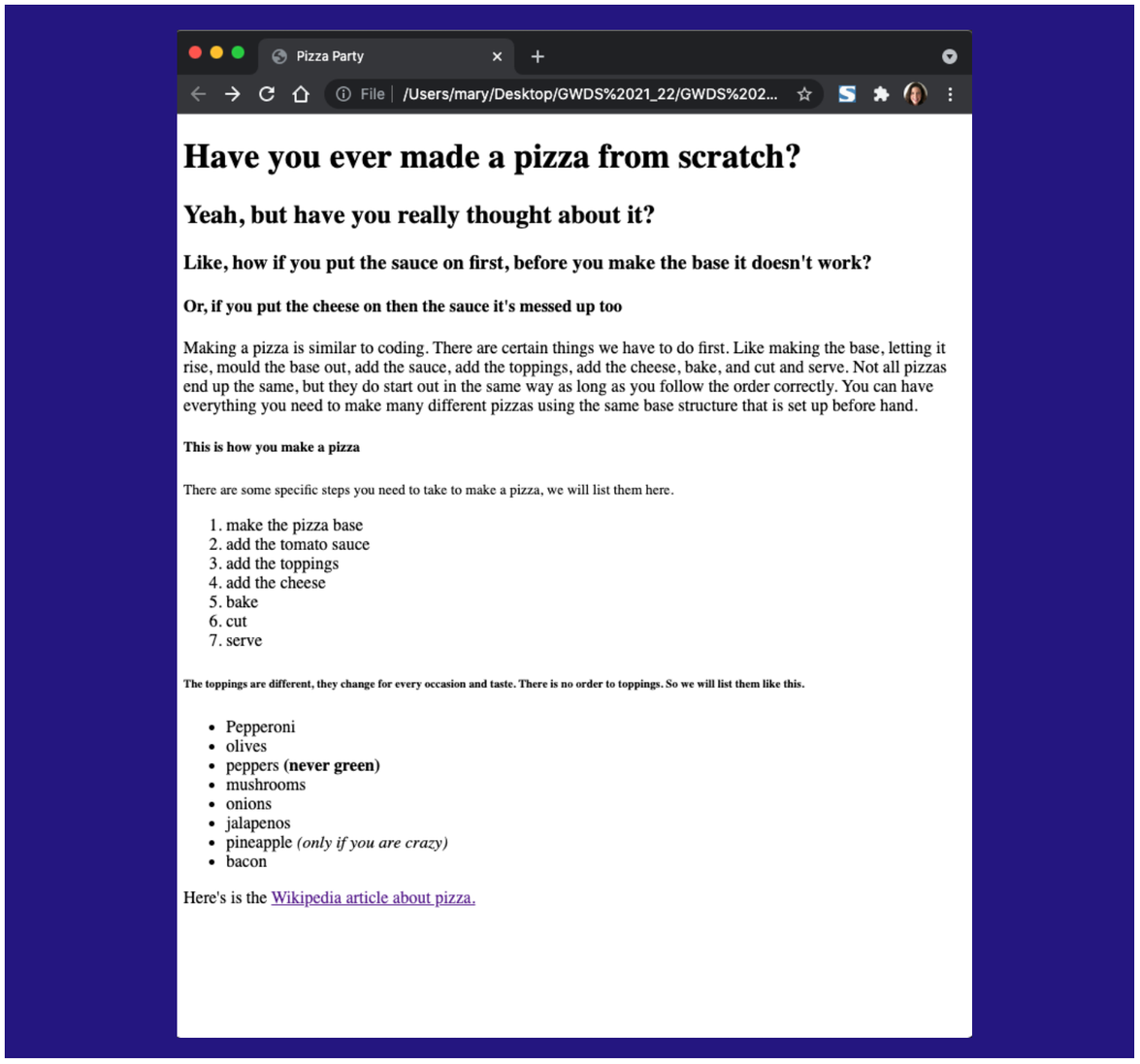
Now we are going to add a link to a website outside of our own website. Add exactly what you see in the image on the right and then **save**, ***commit*** with comment, and ***push***. Then refresh your local web page. What's going on?

The `<a>` tag defines a **hyperlink**. The syntax is as follows `<a href="url">link text</a>` the `<a></a>` has other uses, but this is the one we are looking at right now. This is an **absolute url**. Which means it's a full web address. A **relative url** means it's a link to a page within the same website (for next week).

Now add the code in the second image. **Save.** What happens? You can read more about hyperlinks here. We will get more in depth about all the uses the `<a>` tag has in future worksheets.

```
        <li>bacon</li>
    </ul>

    <p>Here's is the <a href="https://en.wikipedia.org/wiki/Pizza">Wikipedia article about
    pizza.</a></p>

</body>
```

```
46
47      <p>Here's is the <a href="https://en.wikipedia.org/wiki/Pizza" target="_blank">Wikipedia
        article about pizza.</a></p>
48
49  </body>
```



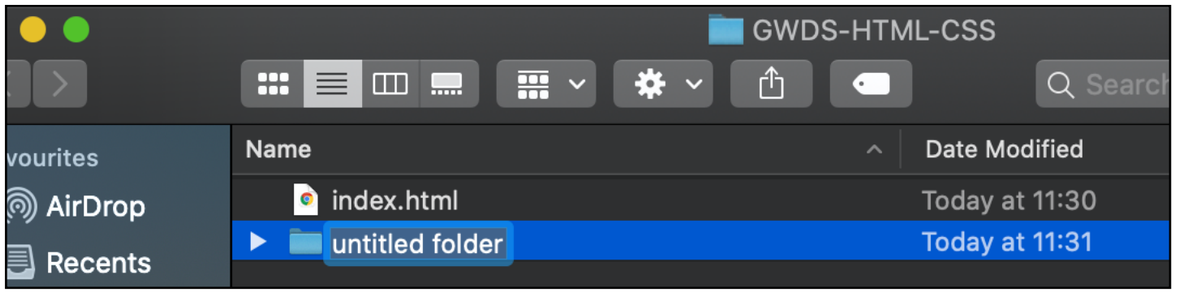
11

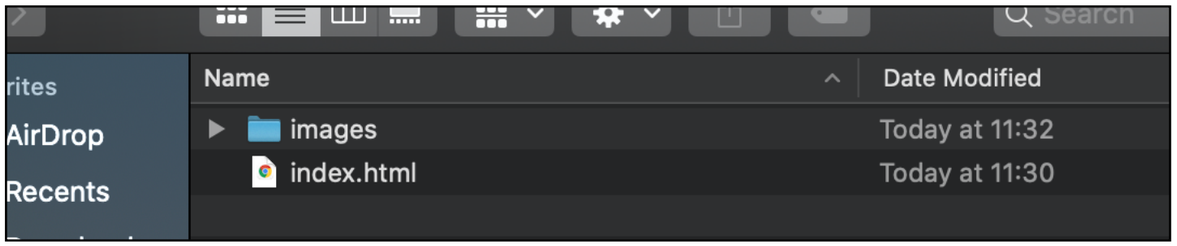
One last thing we are going to do. We will be adding an image to our website. There are a few steps to this process.

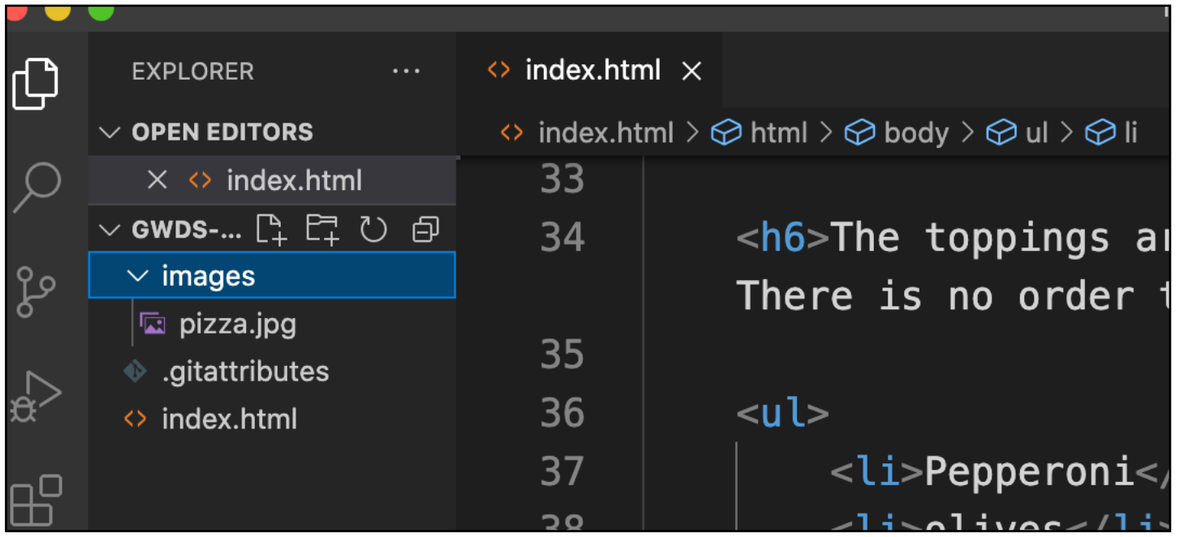
1) Find your folder on your local drive. Where your index.html lives. You can find that by clicking “show in finder” on GitHub desktop. When you get there, create a new folder using file>new folder. Name it **images**.

2) Go here and download this great image (it will happen automatically and end up in your downloads folder) I’ve found for you on unsplash. Add this image to your newly created images folder. ****GitHub knows your have added a folder and image.**** **Save**, ***commit*** with comment, and ***push***. Now when you go back to your VSC the left column will recognise you have an images folder and an image.

   Click on **pizza.jpg** in VSC. What happens?

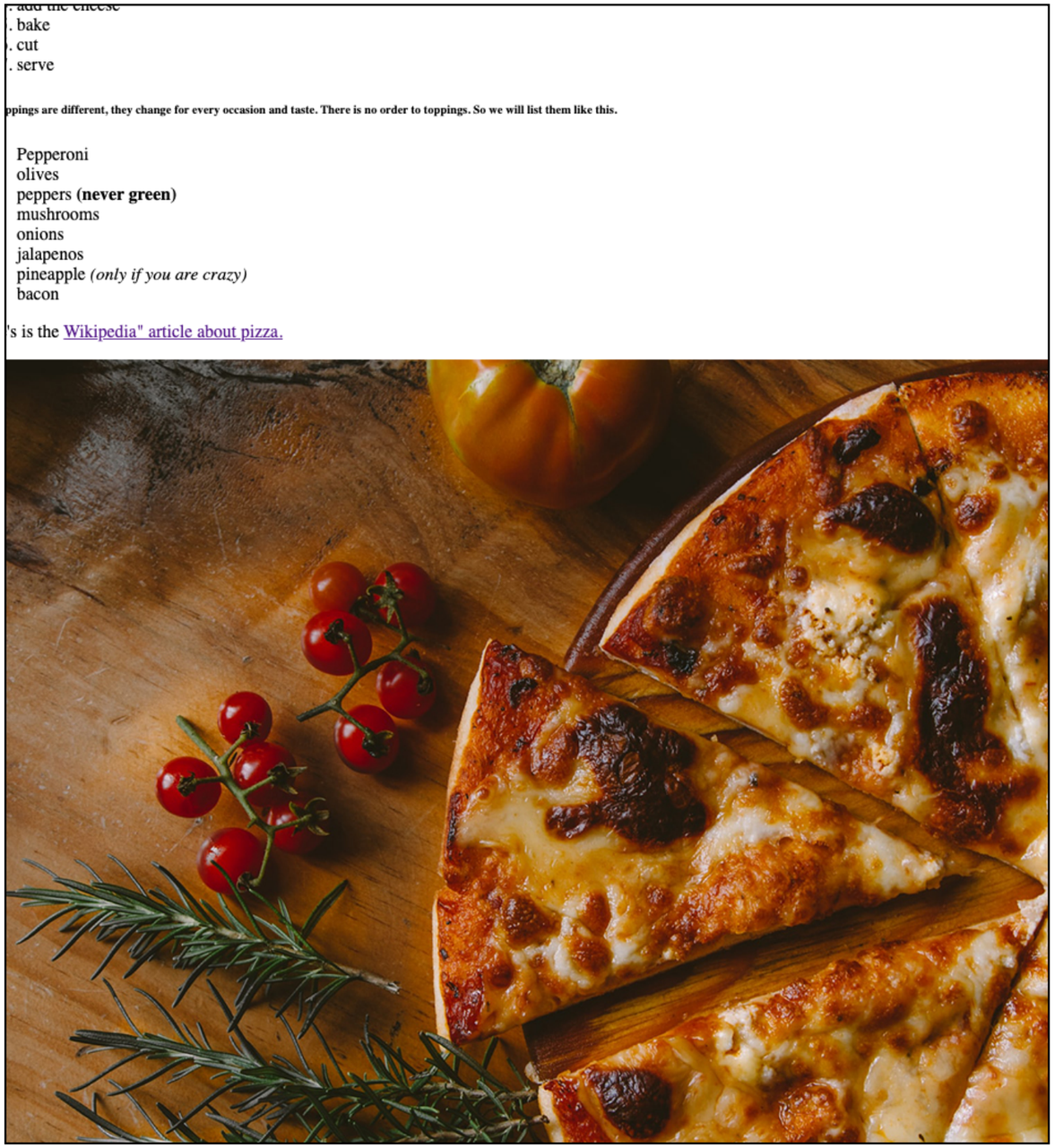






3) After the url link we just added but above the `</body>` tag code in exactly what you see in the image on the right.
Save, ***commit*** with comment, and ***push***.
Now refresh your local webpage. What's going on here.

```
    article about pizza.</a></p>

    <img src="images/pizza.jpg" alt="a picture of a pizza">


</body>
</html>
```



`<img>` is a tag that is used to link an image to a website. It's like a "holding space" for the image to appear in.
`src` specifies the path to the image
`alt` specifies an alternate text for the image

The `alt` is super important for accessibility, screen readers and people who are using devices where the image is not being pulled in. Get used to adding alt text to your images now. They are industry standard and best practices.

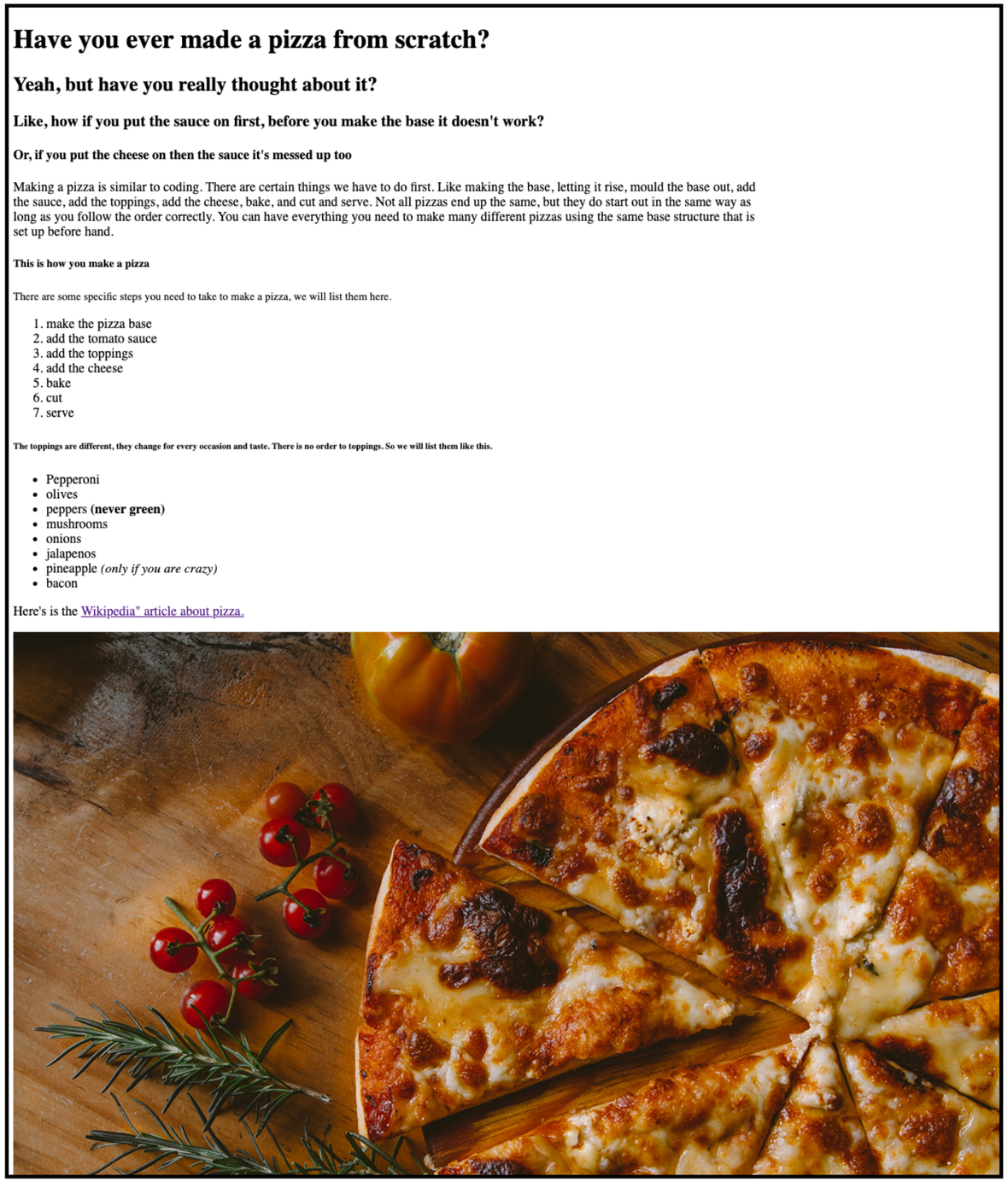
In conclusion.....

This is how images are placed into a website.
This image is HUGE for a reason for next week).

We will be doing this a lot more in the coming weeks so you can get more and more used to how this all works. But for now, pat yourself on the back.

You have now run through the basics of html. And you made a web page. Pretty grotty looking, eh? That's where next week and **CSS** comes in to play.

Don't forget to **save**, ***commit*** and ***push!***
See you next week!



Have you ever made a pizza from scratch?

Yeah, but have you really thought about it?

Like, how if you put the sauce on first, before you make the base it doesn't work?

Or, if you put the cheese on then the sauce it's messed up too

Making a pizza is similar to coding. There are certain things we have to do first. Like making the base, letting it rise, mould the base out, add the sauce, add the toppings, add the cheese, bake, and cut and serve. Not all pizzas end up the same, but they do start out in the same way as long as you follow the order correctly. You can have everything you need to make many different pizzas using the same base structure that is set up before hand.

This is how you make a pizza

There are some specific steps you need to take to make a pizza, we will list them here.

1. make the pizza base
2. add the tomato sauce
3. add the toppings
4. add the cheese
5. bake
6. cut
7. serve

The toppings are different, they change for every occasion and taste. There is no order to toppings. So we will list them like this.

- Pepperoni
- olives
- peppers (**never green**)
- mushrooms
- onions
- jalapenos
- pineapple *(only if you are crazy)*
- bacon

Here's is the Wikipedia" article about pizza.