At this stage, you should be familiar

with different layouts and grid structures,

and you should be ready to start using flexbox.

Let's explore a few practical examples

of how you can use it.

In the next few minutes, you will explore what

the three most common uses of flexbox are.

As mentioned earlier, flexboxes are more suitable to use

for simple layout or designing simple elements in a page.

In line with that idea,

let's explore a few commonly used design elements

that you see on a page.

You will discover simple ways in which you can utilize

flexboxes for binding elements

together or creating an easy layout.

Let's explore the three most common uses

of flexbox is in CSS.

The first application of flexbox that you will

cover that is commonly used is a search bar.

You can use flex in

search bars because it ties up all the elements,

such as the small Search icon,

the search input area,

and the Submit button.

Next, you can create the navigation bar using flex,

which consists of several different layouts,

and it makes your navigation bar

highly responsive on different devices.

Finally, another type of flexbox that

is commonly used is an image gallery.

Flexes helpful here is it can realign

itself as you change the size of the window.

In the next few minutes,

you will discover how to create

these flexboxes to ensure a simple layout.

In the first example of a flexbox file,

I will create a search bar,

I add a reference to the CSS file

in the head section of the HTML page.

After that, I add content inside

the body section of our HTML code.

I first create a container class with a div tag.

Inside this container, I have three different elements,

which are the search icon,

the search box where I typed my search,

and finally the Submit button.

In the CSS code,

I first write rules for the container,

then for the search icon,

the search box, and finally for the button.

Most of the properties defined here are

settings for the alignment of the selectors.

There are just a few important things to take note of.

The display property I use is

inline flex instead of flex,

which makes the flex container

behave like an inline element.

Another property to take note of is the overflow.

The overflow property clips the overflowing content,

which in this case will be

the text I enter in the search query.

Then I add the properties for

the icon and the search box.

Finally, I add the properties for the Search button.

Let's remove this CSS code

momentarily to check how the output displays with aided.

Although all the elements are there

the page is standard and playing,

I add the code back to

the CSS file and check the output one more time.

Now the search box displays

the properties that I defined in my CSS file.

Notice that when I change the size of the page,

the search area doesn't change.

Another place where flexbox is used quite

frequently is in navigation menus.

In this case, I created

an unordered list that consists of four items.

Notice that in my CSS code I use

something called a star or universal selector.

The universal selector applies the rules

to every element in my CSS code.

I use this to declare

any formatting that is browser specific.

Now I define the rules for the container.

It's important to notice that I use the flex-flow,

which is a shorthand property specifying the direction

of the flex container and its behavior for wrapping.

Another important property is justify content,

which aligns the flexible containers items

along the main axis.

In this case, I have set it to stretch.

Since the individual elements are part of the container,

I applied the rules both to the container,

onto the elements which are the list items,

anchor, tag, and so on.

It's also important to notice

that the page is responsive.

As the size of the browser window gets smaller,

the items stacked on top of each other,

and if I expand the window,

it changes to a standard horizontal navigation menu.

One final flexbox example that I will

demonstrate is a responsive image gallery.

All the elements for the gallery is

in the body of the HTML file.

There is a div element with a class container that

consists of six images that are

saved inside the project folder.

Now, let's open the CSS file.

First, I remove all browser-specific settings that may

be there by using

the star selector just like I did earlier.

I set the value of both the margin

and the padding to zero.

I then reset the basic alignment options

for the images by setting the border to zero.

Next, let's focus on the container.

First, I set the value of display to flex,

then I add the flex-wrap

property which determines whether

the flex items should be forced in

one line or should wrap over multiple lines.

Next, I justify the content property,

which aligns the flexible container items

on the horizontal main axis,

I set it to space dash between.

Finally, I add some padding.

Let's inspect the output.

The output window is narrow at the moment,

so the six images are stacked on top of each other.

But when I expand the window, they spread out.

One can notice the effect better in a browser,

so I copy the link and paste it into my browser.

When I change the size of the browser window,

it is clear that the alignment of

these images is responsive.

These are just some of the ways in which

you use flexboxes to provide

responsiveness to websites and bind

different elements together on web pages.

You should now be more familiar with

some simple ways in which you can use flexboxes to

bind elements together to create

an easy responsive layout on your webpage. Good job.

Correct

Correct! The most common uses of flexbox in CSS are creating a responsive search bar, navigation bar and image gallery.

eate complex page layout



Image gallery

Correct

Correct! The most common uses of flexbox in CSS are creating a responsive search bar, navigation bar and image gallery.



Navigation bar