CSS: Flexbox

Picking the right display context and managing the layout of a web page can be extremely inconvenient for a web developer. However, it’s part of the job and we have to face it head on.

Luckily, the flexbox is the right tool for overcoming this obstacle and is the subject of this skill. Here’s an overview of the content to come:

- CSS Flexbox
- Display flex
- CSS position
- CSS grid

Flexbox

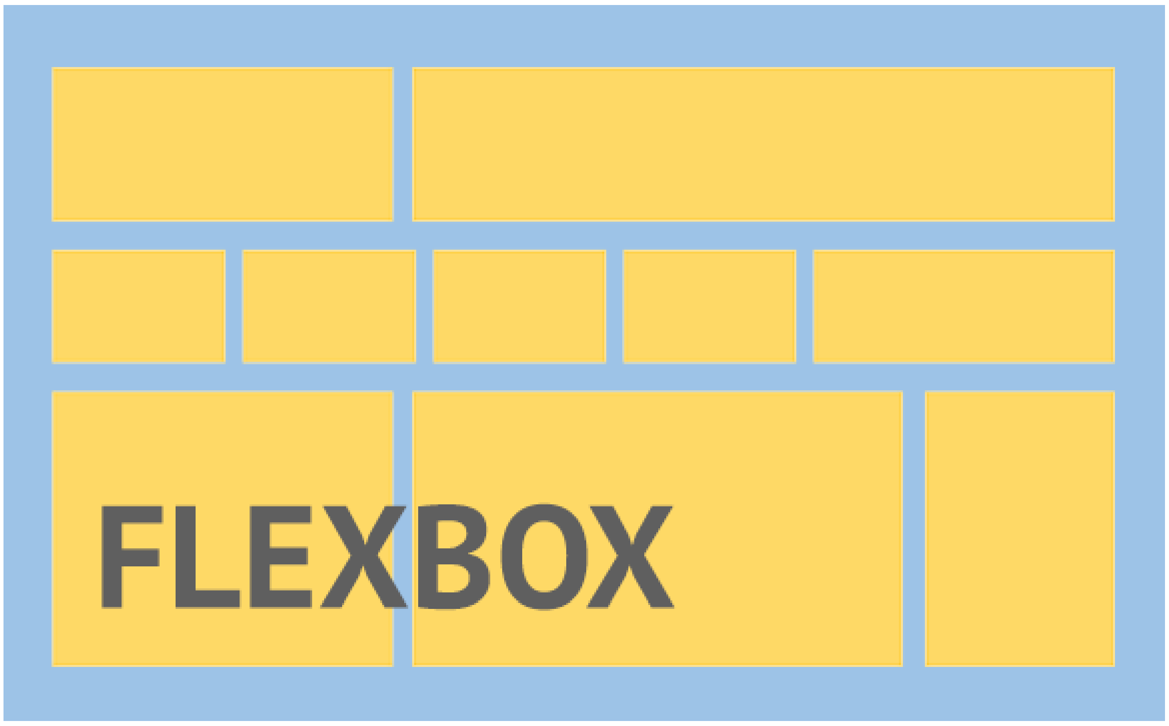
Time to meet Flex!

As we have already seen, we can display HTML elements using layout modes: block for sections in a web page, inline for text, etc.

Flexbox will make it easier to design flexible, dynamic, and responsive layout structures without using float or positioning.

Learning Flexbox can be difficult for most people as it’s not as enjoyable as other skills we will learn. But, it challenges you to rethink your approach when it comes to layouts in CSS.

Don’t worry, we will walk you through everything that you need to know.
So, let’s jump right in!



Since Flexbox is a whole module and not a Single Property , it involves a lot of things, such as its whole set of properties. Some of them are meant to be set on the Container(parent element, known as “flex container”) whereas the others are meant to be set on the Children (said “ Flex Items ”)..

CSS display: flex

What Is Flexbox?

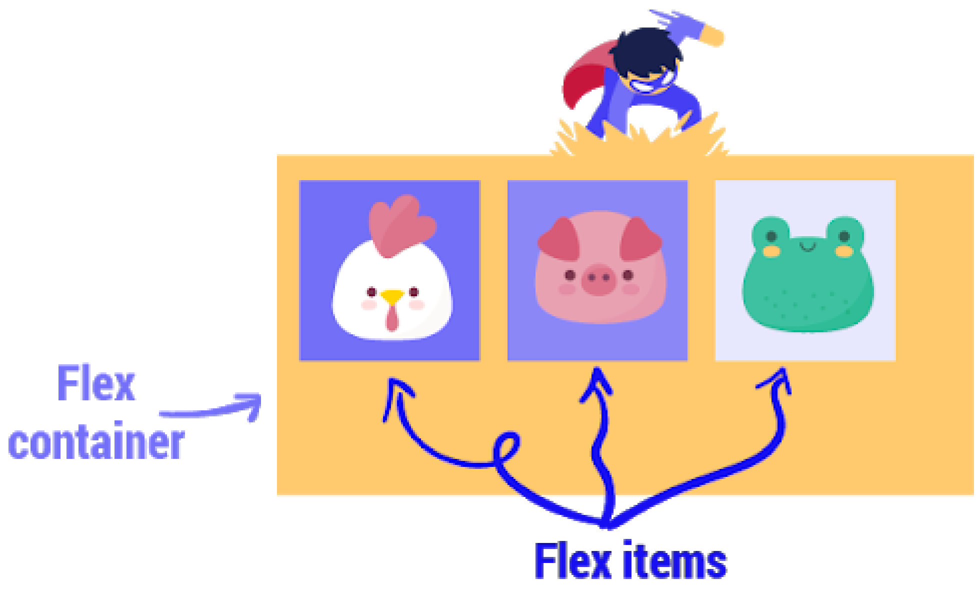
Flexbox has an overly technical definition. For the sake of simplifying this course, we'll skip the technicalities.

You want to imagine Flexbox as the layout ninja for CSS. It's incredibly practical, dynamic, and has a lot of advantages.

When you need to deal with the layout in your styles, the CSS Flexbox model is likely to be your best bet.

How does it work?

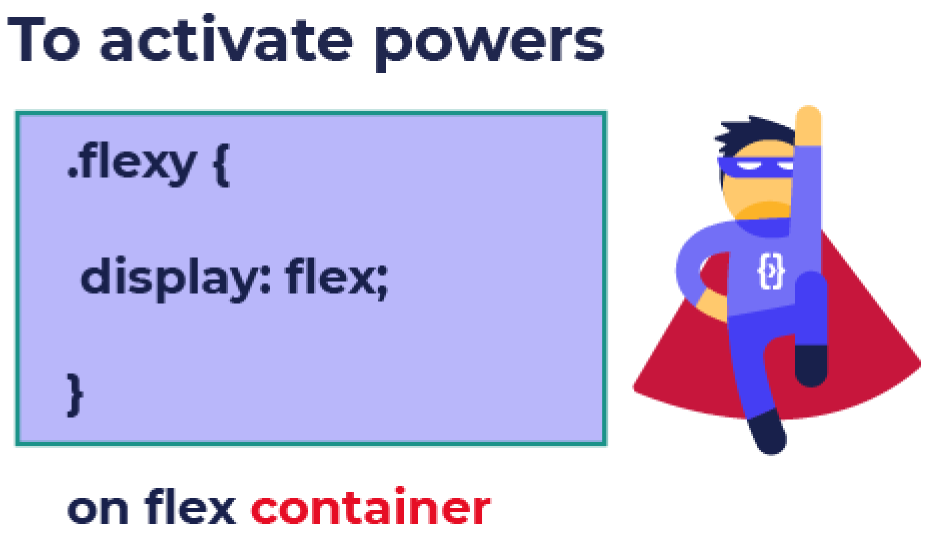
Flexbox starts by making a parent element into a flex container.



Use display Flex

How can we access the Flexbox's super powers?

Pretty simple, let's assume the parent element in question is a div with a class name of `flexy`, we simply apply the following:

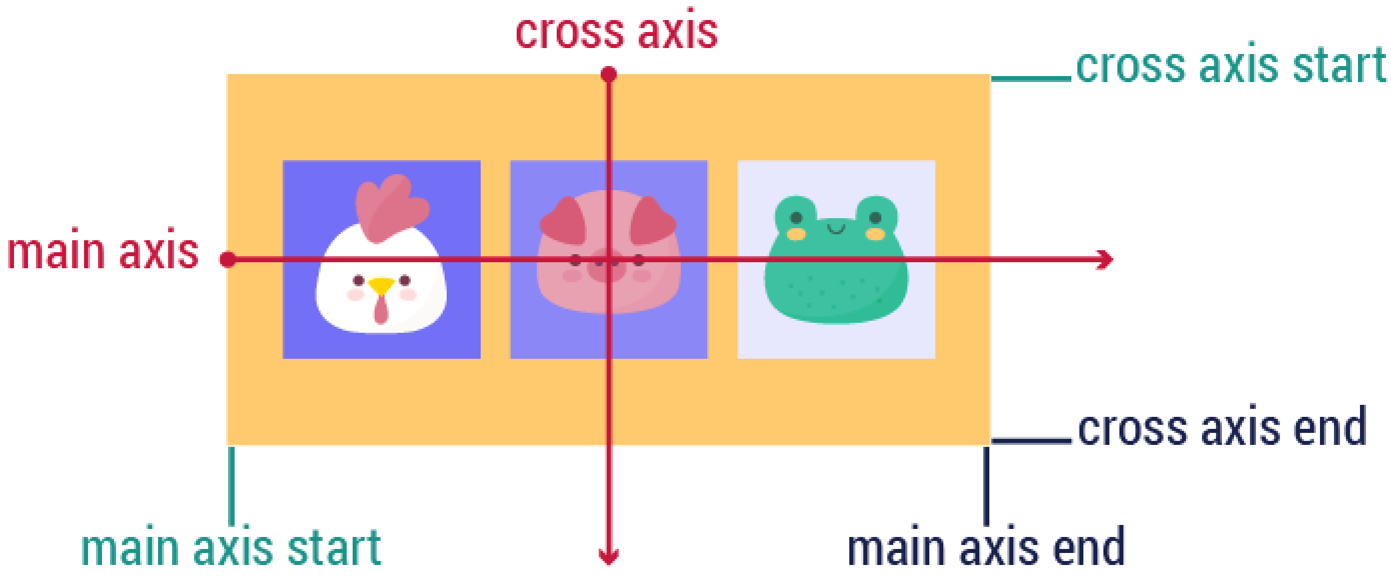


```
.flexy {
  display: flex;
}
```

This one line of code enables a flex context for all its direct children. So if all items need to shrink, each one will shrink equally and if there's no space they will be allowed to shrink horizontally until they all fit.

Why Do We Use Flexbox and What Are Its Super Powers?

After we have made the parent a flex container, we can proceed to using it. One of the many things we can do with Flexbox is to center a child element within a parent. This centering will be along both sides, vertical and horizontal, all according to the main axis.



https://www.youtube.com/watch?list=PL-w_yrNy8uTam5zYatOodamOW9-J9lHb_&v=CBp5qgWQ_7s&feature=emb_imp_woyt

In Flexbox, the parent element is called:

Flex container

Flex parent

Flex Block

Display: flex...

Enables a flex context for all its direct children

Enables a flex context for the parent element

The flex container has two axes: main axis and cross axis.

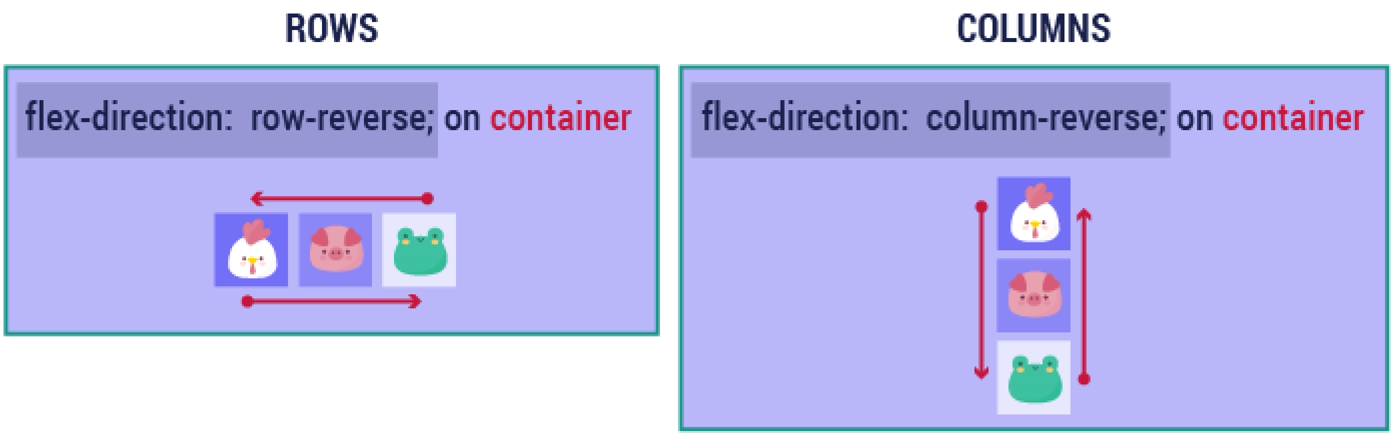
True

False

Css flexbox: columns and rows:

Columns and Rows

By default, flex items are laid out in their source order. Think of flex items as primarily laid out either in horizontal rows or vertical columns.
Therefore, if we want to have rows or columns, we apply the following rules:


ROWS
flex-direction: row-reverse; on container
COLUMNS
flex-direction: column-reverse; on container

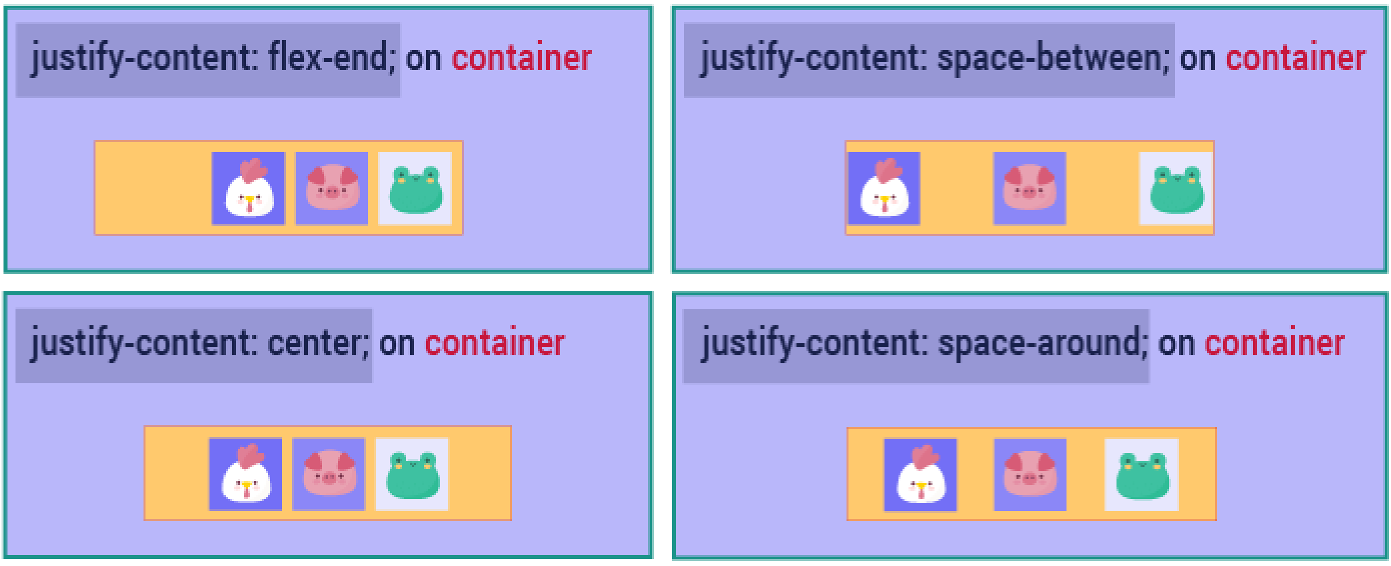
Think of flex items as primarily laid out either In Horizontal rows or vertical Columns . So if you want to have the elements organized as Rows you apply Flex-Direction:row , and if you want to have the elements organized as columns you apply Flex-Direction:column.

Css flexbox :Justify content

Justify-content

Justify-content defines how elements are aligned along the main axis.

It helps distribute extra free space when either all the flex items on a line are inflexible or when they're flexible but have reached their maximum size.


justify-content: flex-end; on container
justify-content: space-between; on container
justify-content: center; on container
justify-content: space-around; on container

Justify-content...

Defines how the elements are aligned along the main axis.

Helps distribute extra free space when either all the flex items on a line are inflexible, or they are flexible and have reached their maximum size.

Which one of these is not a justify-content value?

Flex-start

Flex-end

Flex-middle

Center

The `space-evenly` attribute makes the browser distribute the space between the elements.

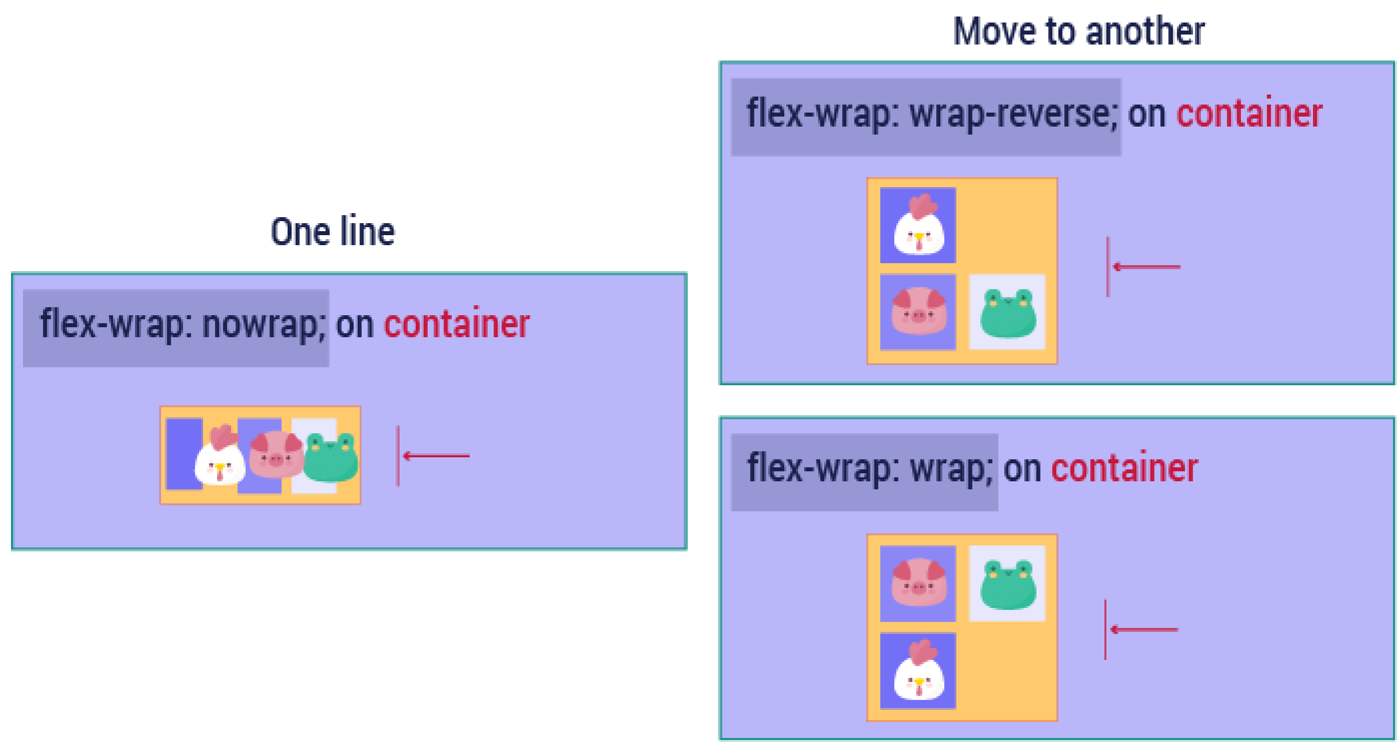
True

False

Css Flexbox: Flex-wrap:

Flex-wrap:

One of flex’s properties is that all items will automatically try to fit into one line.
You can change that and allow the items to wrap as needed. So we have two choices: either we decide for them to be in one line or move to another line.


Move to another
flex-wrap: wrap-reverse; on container
One line
flex-wrap: nowrap; on container
flex-wrap: wrap; on container

The `flex-wrap` attribute makes the container wrap the items in one line.

True

False

The `wrap-reverse` attribute makes the item revert from the last one to the first.

True

False

Items will automatically try to fit into one line. You can change that and allow the items to wrap as needed by applying:

flex:wrap

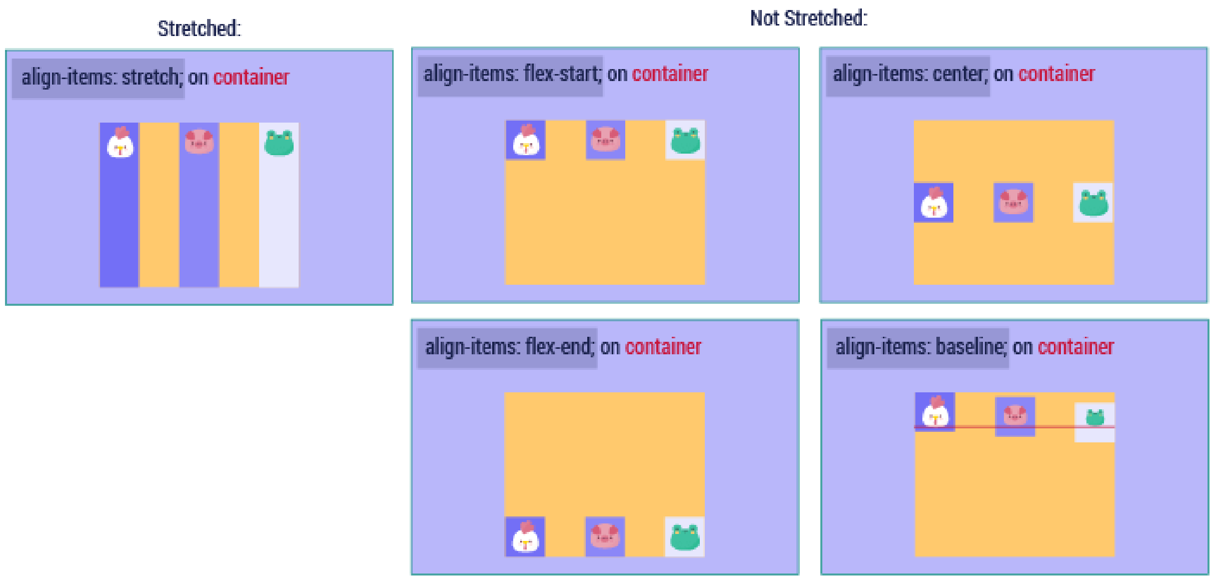
flex-wrap:wrap

flex: re-wrap

CSS Flexbox: align-items

Align-items:

Align-items determines the default behavior for how flex items are laid out along the cross axis on the current line. Think of it as the justify-content version for the cross axis.



Stretched:
Not Stretched:
align-items: stretch; on container
align-items: flex-start; on container
align-items: center; on container
align-items: flex-end; on container
align-items: baseline; on container

Align-items are the equivalent of :

justify-content

align-text

Which one of these is not an `align-item` value?

Flex-start

Stretch

Flex-begin

The `align-item` attribute does work on the main axis.

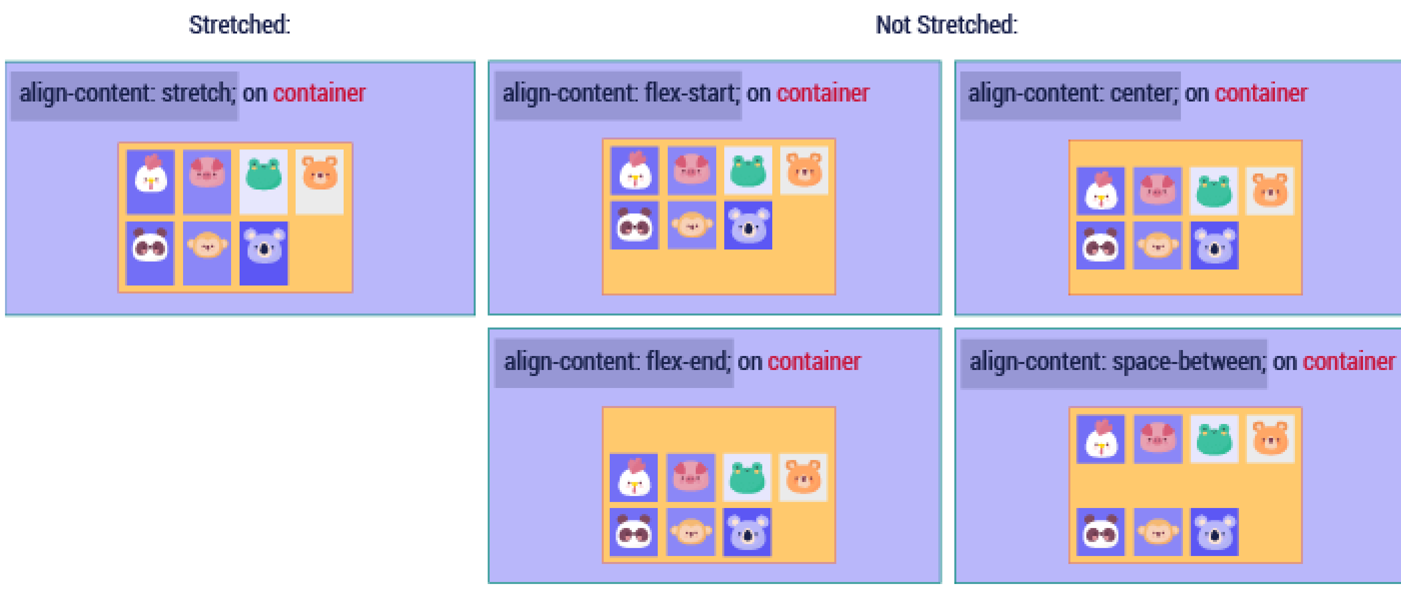
True

False

Align-content:

What if now we have multiple lines of content and we want to align them?
Piece of cake! We naturally use the align-content property.

Keep in mind, this property has no effect when there is only one line of flex items.


Stretched:
Not Stretched:
align-content: stretch; on container
align-content: flex-start; on container
align-content: center; on container
align-content: flex-end; on container
align-content: space-between; on container

Align-content has no effect when there is only one line of flex items.

True

False

The stretch value in align-content will distribute the item all over the container.

True

False

The `align-item` works on the cross axis.

True

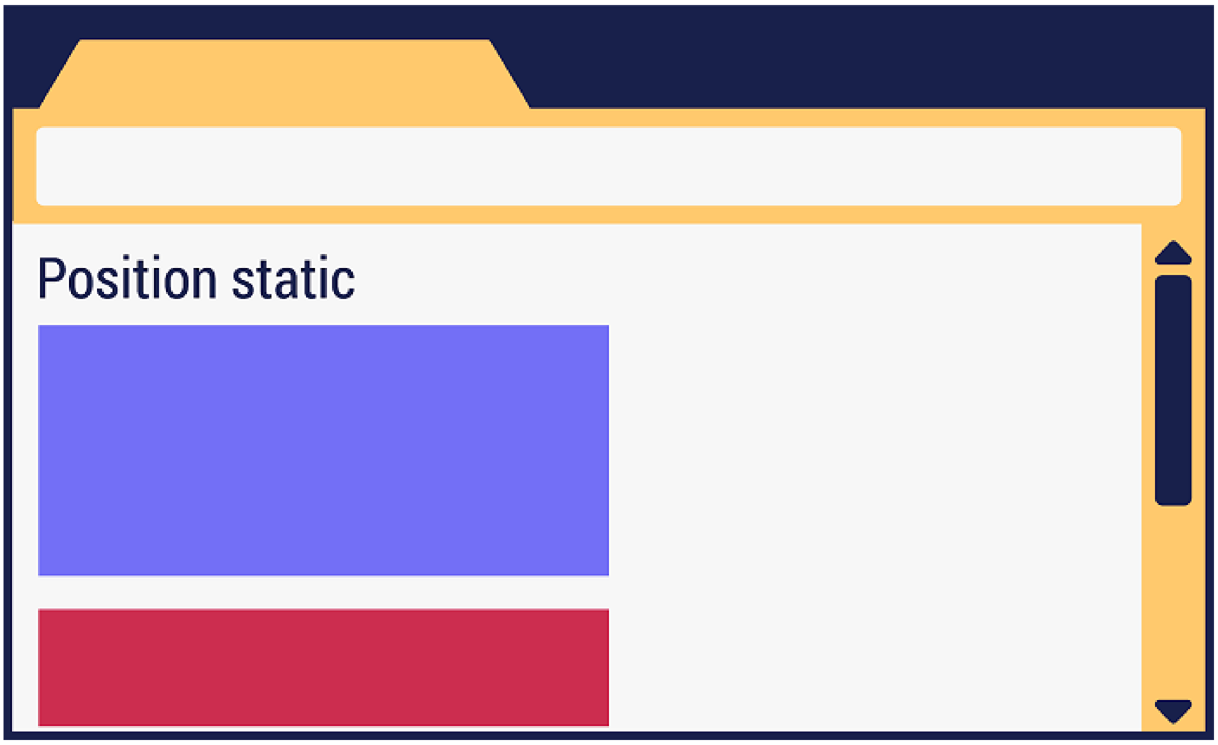
False

Static Position

The **static** position is the default positioning for all the elements on a web page.

The elements are fixed objects within the scene until you decide for them to be a part of a positioning context.

Static means elements are static according to the normal flow of the page. If one moves, another will move as well.



Css positioning

Position Absolute

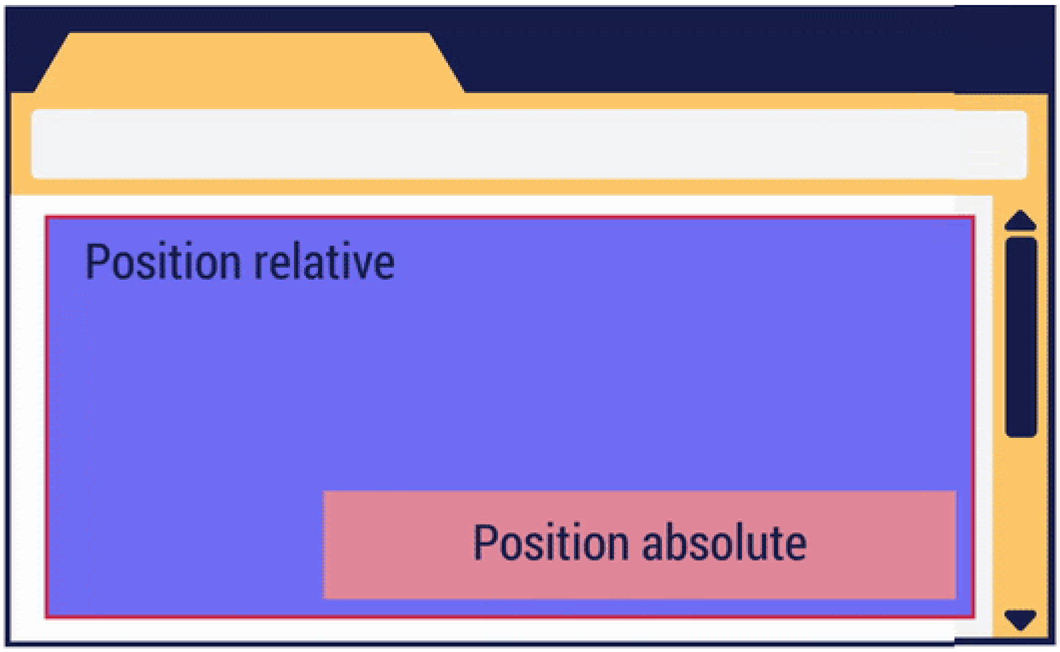
By default, all HTML elements are in a static position. The element is not positioned in any special way and is always positioned according to the normal flow of the page.
In this case the parent is given position: static, and the child is given position: absolute and according to that, it will be absolute to the parent element that has a position of relative, absolute, or fixed.
If there is no position specified, it will be absolute to the page (meaning that it won't be affected by anything around it).

Position Relative

As we have already said, elements are statically positioned by default.

In this case, the parent is given a value of position:relative, and the child is given a value of position:absolute.

When you style an element with position: relative, it creates a positioning context for every child element within the element.
Using this positioning you can go ahead and accurately position the child element with respect to any sides of the reference object.
Relatively positioned elements behave exactly like static ones, but they serve as a local frame of reference for absolutely positioned child elements.

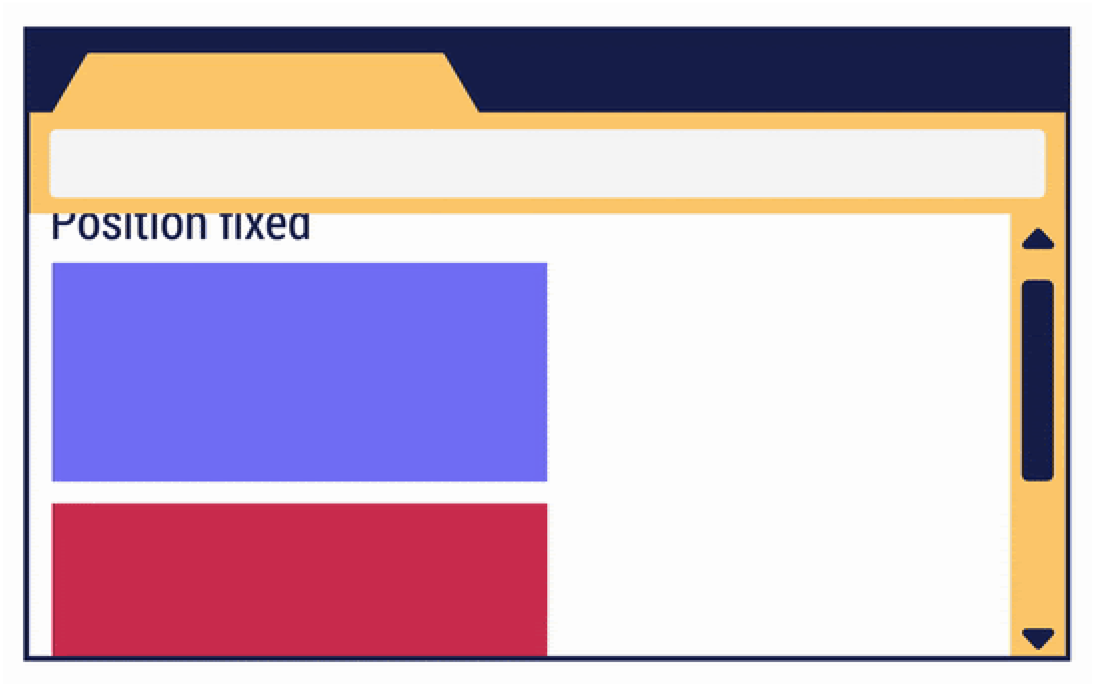


Position Relative VS Position Absolute

https://www.youtube.com/watch?list=PL-w_yrNy8uTam5zYatOodamOW9-J9lHb_&time_continue=5&v=Of8mr6QFkuo&feature=emb_logo

Position Fixed

If we position our element as **fixed**, it means the position will always be locked to the size of the browser's window. If you scroll up ,down ,left or right, the element will stay in its position.



Position Sticky

An element with position: sticky is positioned based on the user's scroll position.
In other words, as soon as the scroll reaches the sticky element, it will stick to the screen.

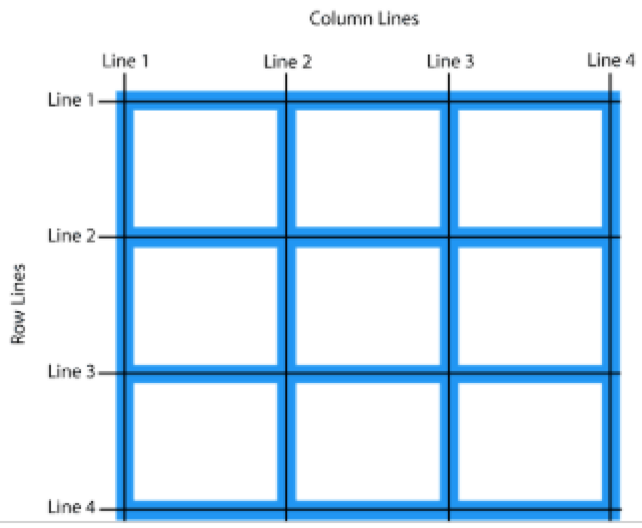
The Static position is The Default positioning for all the elements on a web page. Static means elements are always positioned according to the normal Flowof the page. If one Moves, another will move as well.

CSS Grid System

Every website or application layout you make (or have seen) is essentially boxes placed within certain defined boundaries.

In very simple terms, a grid is just horizontal and vertical lines that define the placement of other design elements.

In the context of the CSS Grid layout, a grid container is a parent that contains all items within it. The grid container defines the initial placement of the grid lines, both vertical and horizontal.



So How Can We Define a Grid?

Just like Flexbox, everything begins with a one-line **display:grid** or **display:inline-grid** for an inline version.

For example, to make a certain div a grid container,

- **CSS**

```
div {

  display: grid;

}
```

https://www.youtube.com/watch?list=PL-w_yrNy8uTam5zYatOodamOW9-J9lHb_&time_continue=35&v=AI-d6FluSDk&feature=emb_logo
In the context of the CSS Grid layout, a grid container is a Parent that contains All The Items
within it. The grid Container defines the initial placement of The Grid Lines , both vertically and horizontally.

CSS Grid System: Rows and columns

Columns and Rows

First we have to define columns and rows, because a grid without them is entirely pointless.

To create columns and rows within a grid container you use these two new CSS properties: `grid-template-columns` and `grid-template-rows`.

So how do you use them? Pretty simple.

`grid-template-columns` determines the placement of the columns while `grid-template-rows` determines the placement of the rows within the grid container.

All we need to do is type in the length of these properties,

- CSS

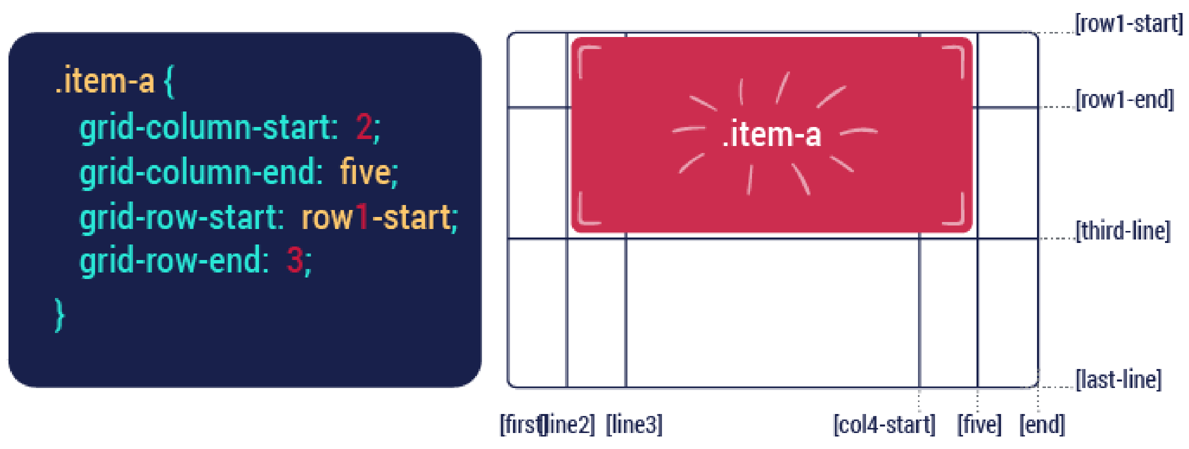
```
      .grid-container {

        display: grid;

        grid-template-columns: 100px 200px 300px;

        grid-template-rows: 100px 200px 300px;

      }
```

Grid-Template-Columns and grid-template-rows determine the columns and Rows
 of the Gridwith a space-separated list of values. The values represent the track Size
 , and the space between them represents the Grid Line.

CSS Grid Systems : Display

Properties of the Grid Items

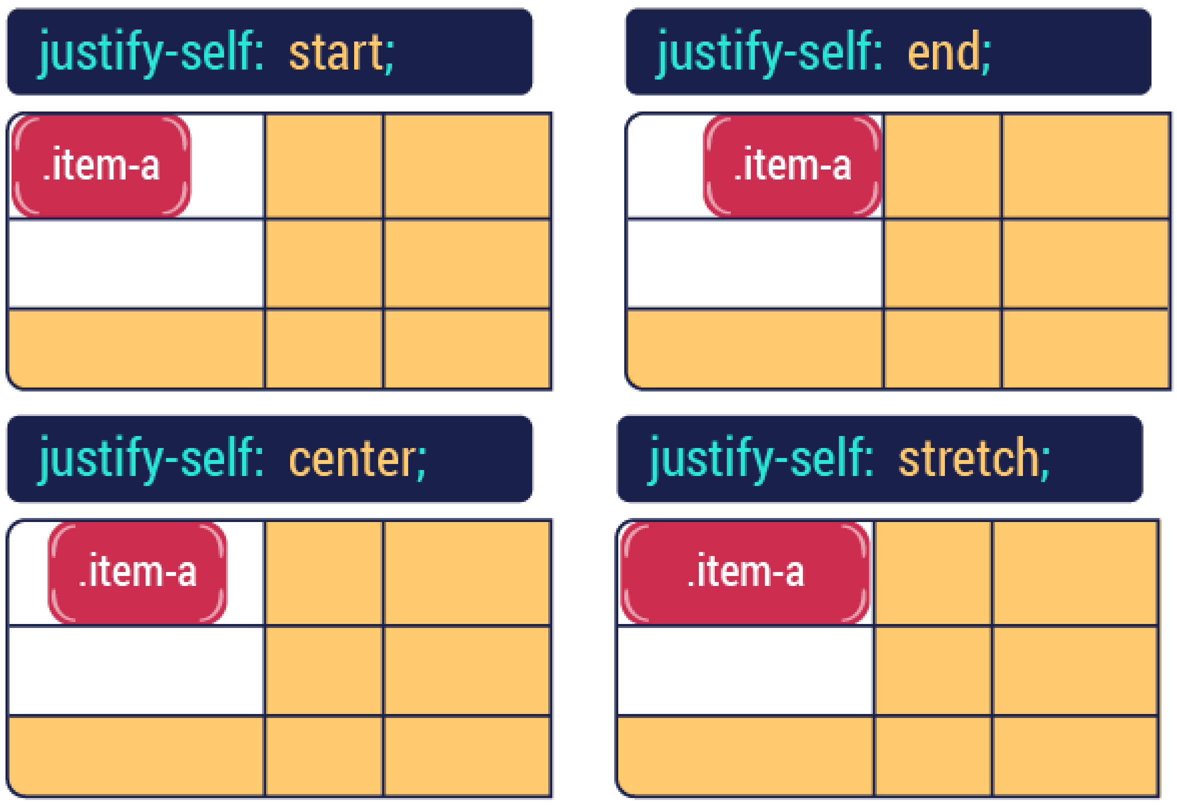
We have:
grid-column-start/grid-row-start the line where the item begins
grid-column-end/grid-row-end the line where the item ends



```
.item-a {
  grid-column-start: 2;
  grid-column-end: five;
  grid-row-start: row1-start;
  grid-row-end: 3;
}
```

Justify-self Properties

Justify-self aligns a grid item inside a cell along the inline. It can have the value`center`,`end`,`start` or `stretch`.



Grid-Column-Start, grid-column-end, Grid-Row-Start, and grid-row-end determine a grid Item's location within the Grid by referring to a specific grid's Lines. grid-column-start/grid-row-start is the line where the Item Begins
, and grid-column-end/grid-row-end is the line where the item ends.

What Is Responsive Design?

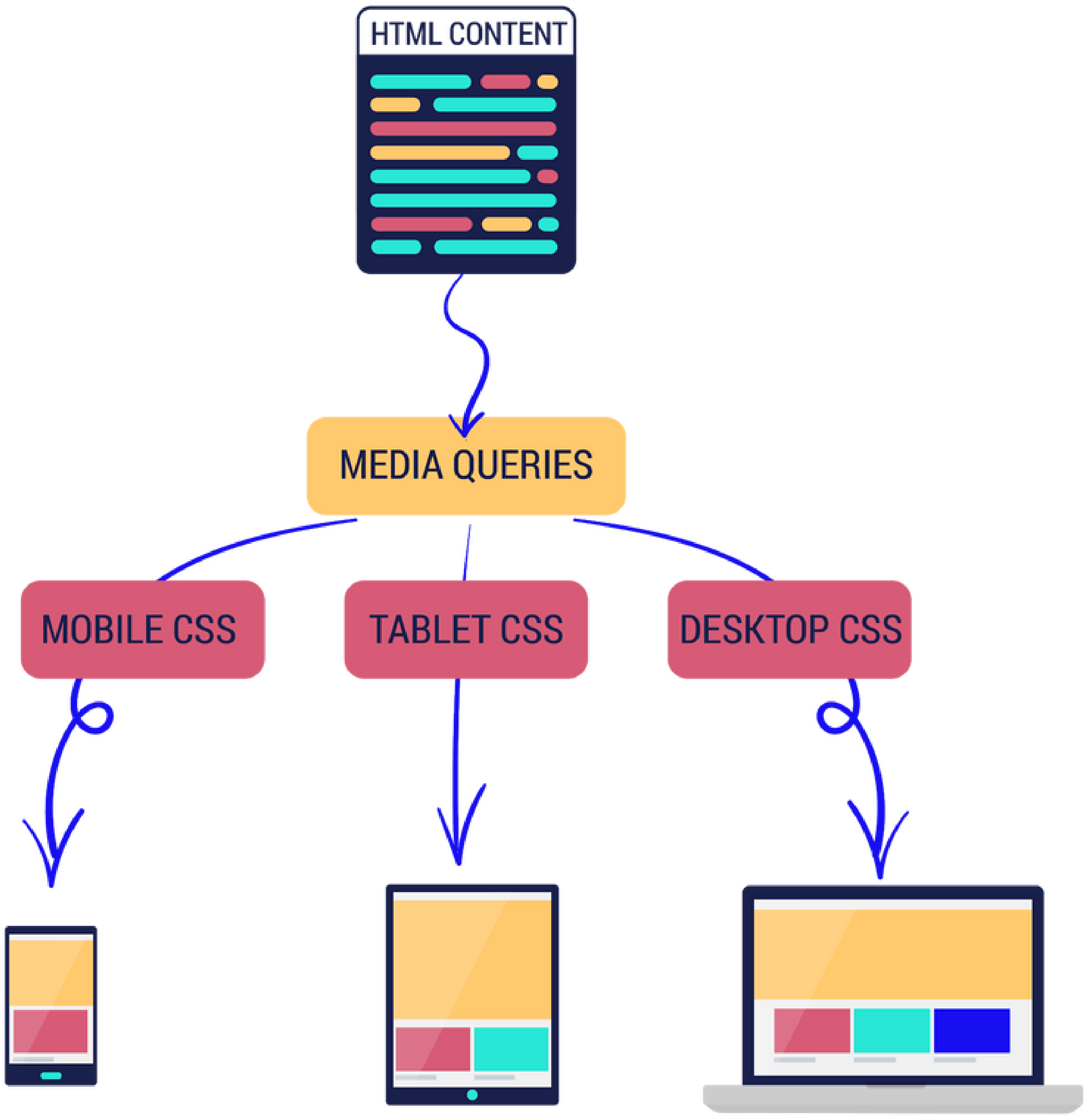
Responsive web design is an *approach* whereby a designer creates a web page that is highly interactive with the user. Meaning it "*responds to*" or resizes itself depending on the type of device it is being displayed on and depending on the user's behavior and preferences.

The devices could be a large desktop computer monitor, a laptop or device with small screen such as a smartphone or tablet.

Responsive web design has become an essential tool for anyone with a digital presence. With the growth of smartphones, tablets, and other mobile computing devices, more people are using smaller-screens to view web pages.

How Can We Make It Responsive?

Responsive design is accomplished through CSS `media queries`. Think of media queries as a way to conditionally apply CSS rules. They tell the browser that it should ignore or apply certain rules depending on the user's device.



What is responsive design in web development?

Responsive design makes your website look great on every device you are using

Responsive design makes your website respond fast to the user's needs.

Responsive design is a language.

True

False

The keyword for responsive design is

Media query

Responsive

responsive_design

What Is Media Query?

Media queries are at the heart of responsive design. They let you target specific screen sizes and specify CSS rules that will be executed on that screen.

The most popular way that media queries are used is something called the @media rule.

- CSS

```css
@media screen and (max-width: 300px) {

    /* write your CSS in this code block */

}
```

What Does It Really Do?

So to translate the code below, it means “For a screen device with a maximum width of 300px … implement the following actions”
Any styles within the code block will only apply to devices that match the expression, screen, and max-width (300px) i.e. screen devices with a maximum width of 300px.

- CSS

```css
@media screen and (max-width: 300px) {

    /* write your CSS in this code block */

}
```

@media rule is a a part of the media queries

True

False

@media screen defines only the maximum width

True

False

@media screen and (max-width: 300px) means :

For a screen device with a maximum width of 300px

For a screen device with a minimum width of 300px

Css :Media Query Breakpoints

What is a breakpoint?
CSS breakpoints are points where the website content responds according to the device width, allowing you to show the best possible layout to the user.

These are the most used breakpoints:

```
/* Extra small devices */

        @media (max-width: 576px) { /* ... */ }

/* Small devices (landscape phones, 576px and up)  */

        @media (min-width: 576px) and (max-width: 768px) { /* ... */ }

/* Medium devices (tablets, 768px and up)  */

        @media (min-width: 768px) and (max-width: 992px)  { /* ... */ }

/* Large devices (desktops, 992px and up)  */

        @media (min-width: 992px) and (max-width: 1200px)  { /* ... */ }

/* Extra large devices (large desktops, 1200px and up)  */

        @media (min-width: 1200px) { /* ... */ }
```

Assessment Solution

@media (max-width: 576px) { ... }

@media (min-width: 576px) and (max-width: 768px) { ... }

@media (min-width: 768px) and (max-width: 992px) { ... }

@media (min-width: 992px) and (max-width: 1200px) { ... }

@media (min-width: 1200px) { ... }

CSS:Useful Resources

Font-Awesome

Now, we will be sharing with you some helpful and useful resources to make your web page look “awesome”.
Time to meet Font-Awesome! It is the most popular way to add font icons to your website.
Font Awesome icons are created using scalable vectors, so you can use high quality icons that work well on any screen size.

How Do We Use It?

First, we add the following line inside the < head > section of your HTML page:

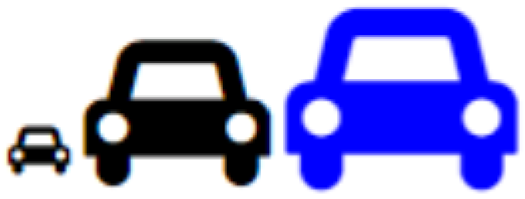
```
<link rel="stylesheet"  href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
```

Then, to display cool icons we simply use the material available on https://fontawesome.com/. You will find everything you will need on that website. Here's an example:

- HTML

```
<i class="fa fa-car"></i>

<i class="fa fa-car" style="font-size:48px"></i>

<i class="fa fa-car" style="font-size:60px; color:blue"></i>
```

- Output



Font-face

Let’s get to know another useful tool, it’s called the font-face rule.

With `font-face` , web developers or designers have the freedom to use new and interesting fonts, and throw the classic and dull fonts right out the window.

It’s very easy to use the font-face rule. First you must define a name for the font (example: Chathura Bold). Then, point it to the font file (Src: url(../Path to your font file).

And then you simply use it as a normal font in your CSS file.

```
@font-face {

    font-family: "chathurabold";

    src: url("../font/chathura_bold_macroman/chachathura-bold-webfont.eot");

    font-weight: normal;

    font-style: normal;

}
```

What Is Bootstrap?

Remember when we spoke about responsive design? Well, it’s time to meet Bootstrap. It will be your responsive design guide.

How will Bootstrap make your life easier?

Bootstrap code is already written and is available for everyone to use. So, we’ll be treating it like a library and borrow only what we need from it.
Everything we need is already on the Internet. We just have to be selective, thorough, and pick out only what we need. This is called “snippeting”.

Snippeting is basically copying code from another source and pasting it to your code. It is ethical and unproblematic in Bootstrap, no need to worry.

How Is It Used?

We will now look at a few of the many examples of how to use Bootstrap. You can find more information on Bootstrap’s official website. Feel free to try new things!
Like every library, to have access to its elements you must add the following line inside the <head> section of your HTML page:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
```

Font Awesome is a library of fonts:

True.

False

Bootstrap is a CSS rule that can be applied only on forms

True.

False.

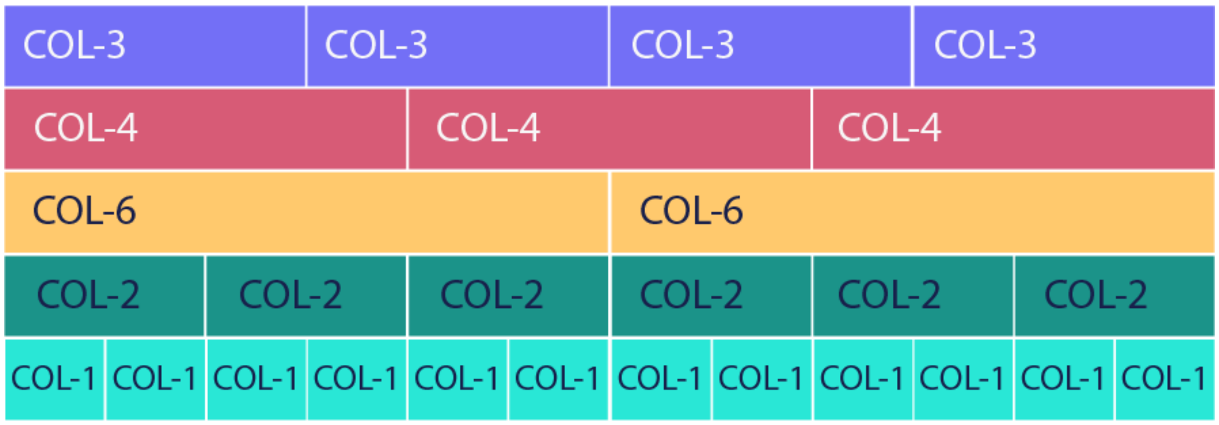
Bootstrap Grid System

One of Bootstrap's most important is that it makes your website responsive using its grid system.

The main idea is to divide the screen into twelve equal columns, and this gives us a new unit that works independently from the size of the screen.

There are four types of resolutions (or classes) in Bootstrap:

- xs (for phones - screens less than 768px wide)
- sm (for tablets - screens equal to or greater than 768px wide)
- md (for small laptops - screens equal to or greater than 992px wide)
- lg (for laptops and desktops - screens equal to or greater than 1200px wide)



How do you use it?

First Step

To apply Bootstrap's grid system to our web page we need to use Bootstrap's container and rows.
First thing we need to do is create a div with `class="container"`.It is as essential as the body tag when we're working in Bootstrap.

So, our Bootstrap must be inside the div with the class container. It makes a margin on the right and on the left so that our content is centered.

If we want our page to take up the whole screen, we write `container-fluid`.

```
<body>

  <div class="container">

    <!-- Some BootStrap Classes here -->

  </div>

</body>
```

How do you use it?

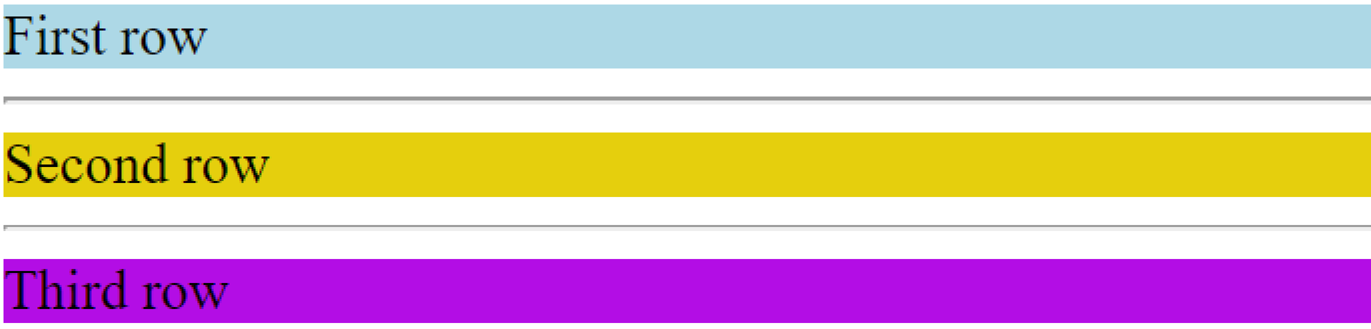
The Second Step

After creating the container, we can add rows.
Inserting a row, like the following picture demonstrates, means a line across the screen. That row will be divided into twelve equal parts where we can put our HTML elements and that will replace the margin and padding.

- HTML

```
<div class="container-fluid">

  <div class="row" style="background-color:lightblue">

    First row

  </div>

  <hr />

  <div class="row" style="background-color:#e5cf0d">

    Second row

  </div>

  <hr />

  <div class="row" style="background-color:#b30de5">

    Third row

  </div>

</div>
```

- Output



```
First row

Second row

Third row
```

How Does It Work?

So, how does the grid system work?

Each row that we add can be divided into 12 equal parts called "columns". From this moment onwards, we can say things like, "This element should take 3 columns on a big screen and 12 columns on a small screen."

Since the row is divided into 12 columns, if an element gets 12 columns of width it will be as wide as the whole screen. If it takes 6, then it’s as wide as half the screen. Here’s an example, col-8 will take 8 columns, and col-4 will take 4 columns.

- HTML

```
<div class="row">

  <div class="col-8">col-8</div>

  <div class="col-4">col-4</div>
```

```
    </div>
```

- Output

How do you use it?

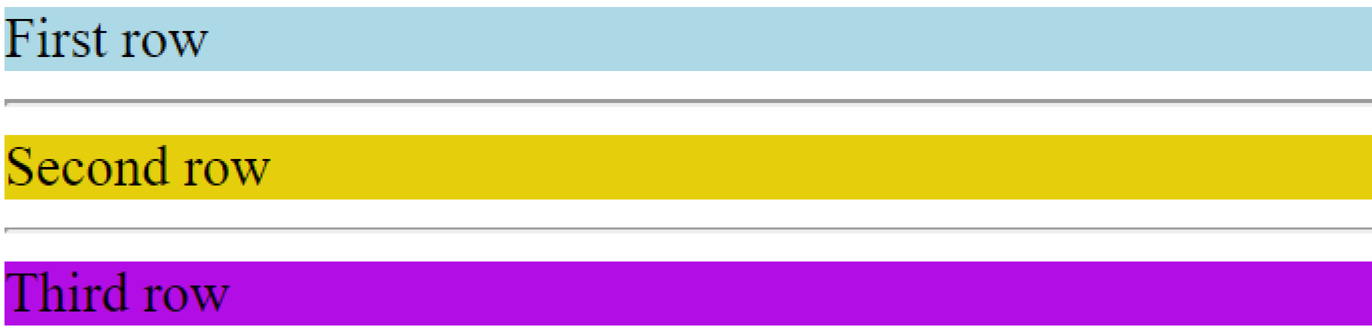
The Second Step

After creating the container, we can add rows.
Inserting a row, like the following picture demonstrates, means a line across the screen. That row will be divided into twelve equal parts where we can put our HTML elements and that will replace the margin and padding.

- HTML

```
    <div class="container-fluid">

      <div class="row" style="background-color:lightblue">

        First row

      </div>

      <hr />

      <div class="row" style="background-color:#e5cf0d">

        Second row

      </div>

      <hr />

      <div class="row" style="background-color:#b30de5">

        Third row

      </div>

    </div>
```

- Output


First row

Second row

Third row

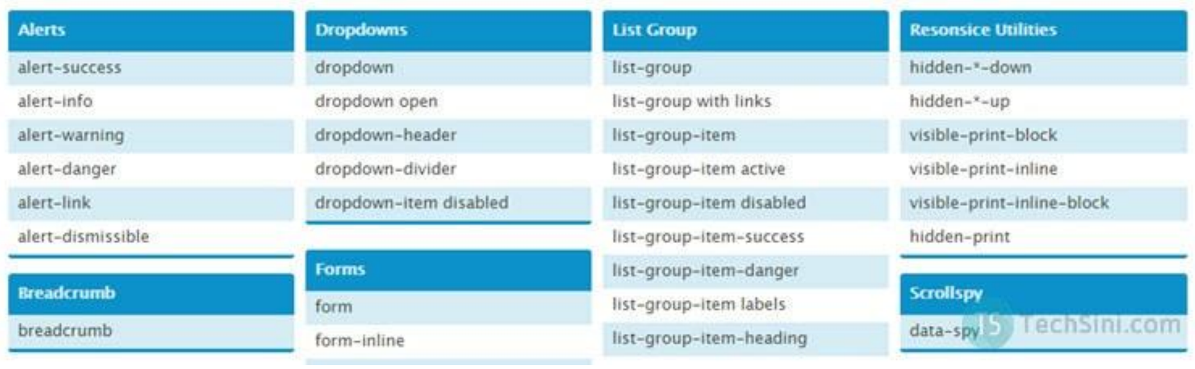
Bootstrap ’s grid system uses a series Of Containers , rows, and Columnsto layout and align content. It’s built with Flexbox
 and is fully Responsive

Bootstrap Classes

As we have already mentioned, Bootstrap is a massive collection of reusable and versatile pieces of code which are written in CSS, HTML, and JavaScript. With that said, developers have a range of components and tools at their disposal, which can be used on the website. These include, but are not limited to:

- Drop-down menus
- Navigation bars
- Progress bars
- Thumbnail images

Let’s take a look at some of these free tools.



Alerts	Dropdowns	List Group	Resonsice Utilities
alert-success	dropdown	list-group	hidden-*-down
alert-info	dropdown open	list-group with links	hidden-*-up
alert-warning	dropdown-header	list-group-item	visible-print-block
alert-danger	dropdown-divider	list-group-item active	visible-print-inline
alert-link	dropdown-item disabled	list-group-item disabled	visible-print-inline-block
alert-dismissible		list-group-item-success	hidden-print
	Forms	list-group-item-danger	
Breadcrumb	form	list-group-item labels	**Scrollspy**
breadcrumb	form-inline	list-group-item-heading	data-spy

TechSini.com

uttons

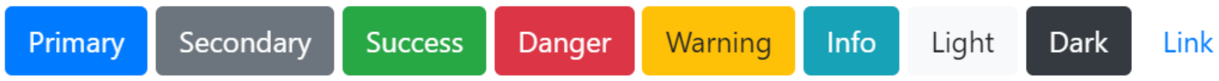
So in HTML, buttons are basic, dull, and uninteresting. They also don’t have a meaning behind them. Why not improve them and make them more awesome and more advanced with Bootstrap?

- HTML

```
<button type="button" class="btn btn-primary">Primary</button>

<button type="button" class="btn btn-secondary">Secondary</button>

<button type="button" class="btn btn-success">Success</button>

<button type="button" class="btn btn-danger">Danger</button>

<button type="button" class="btn btn-warning">Warning</button>

<button type="button" class="btn btn-info">Info</button>
```

```
<button type="button" class="btn btn-light">Light</button>

<button type="button" class="btn btn-dark">Dark</button>

<button type="button" class="btn btn-link">Link</button>
```

- Output



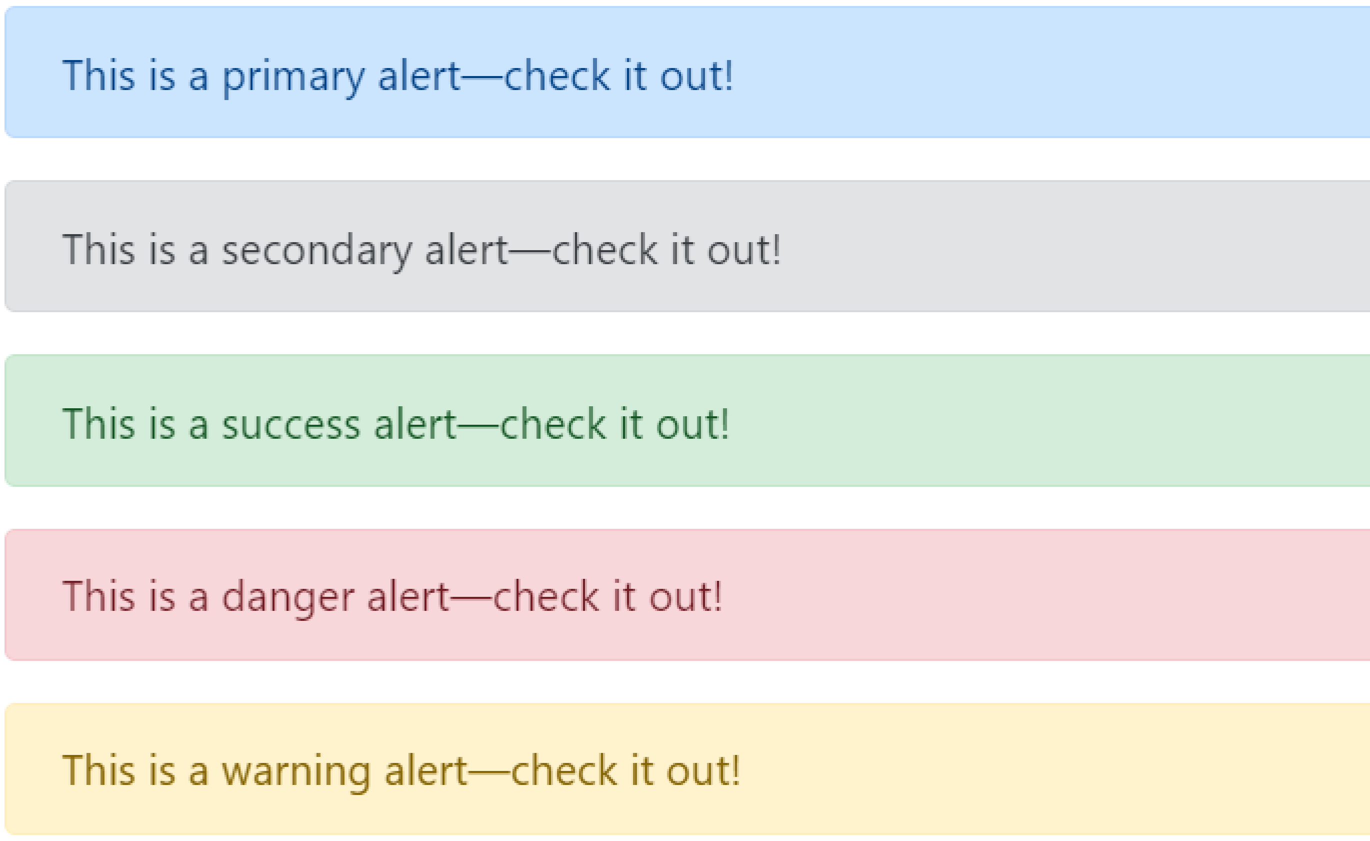
Alert

Alerts can be used in different situations. Let’s introduce some of them. You can imagine each scene yourself.

- HTML

```
<div class="alert alert-primary" role="alert">


  This is a primary alert—check it out!

</div>

<div class="alert alert-secondary" role="alert">

  This is a secondary alert—check it out!

</div>

<div class="alert alert-success" role="alert">

  This is a success alert—check it out!

</div>

<div class="alert alert-danger" role="alert">

  This is a danger alert—check it out!

</div>

<div class="alert alert-warning" role="alert">

  This is a warning alert—check it out!

</div>
```

- Output


This is a primary alert—check it out!

This is a secondary alert—check it out!

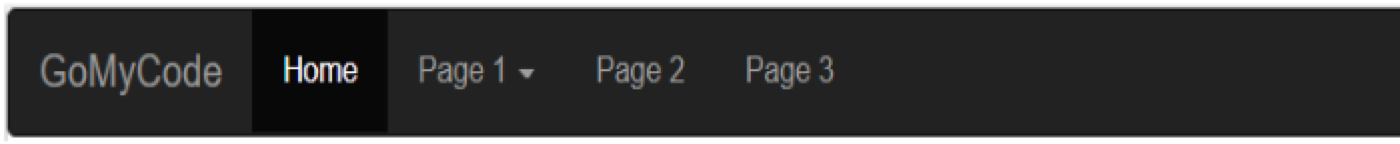
This is a success alert—check it out!

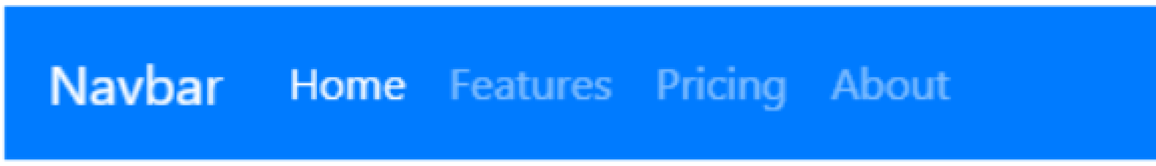
This is a danger alert—check it out!

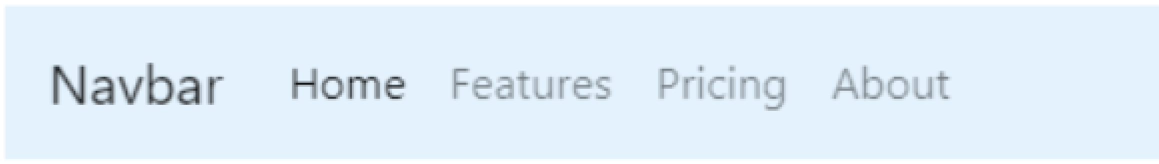
This is a warning alert—check it out!

Navbar

Bootstrap also offers a large range of navigation bars to make things easier.
A navigation bar is a navigation header placed at the top of the page.
Here's an example: (You will find the needed code in the Bootstrap's official website)







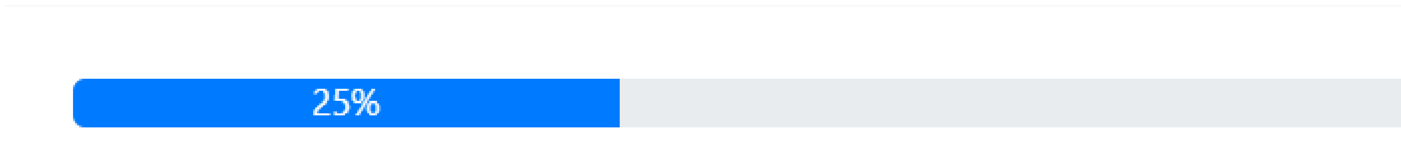
Progress Bar

Another useful feature is a progress bar. They can be used for creating your resume page.
For example:

- HTML

```
<div class="progress">

  <div class="progress-bar" role="progressbar" style="width: 25%;" aria-valuenow="25"
aria-valuemin="0" aria-valuemax="100">25%</div>

</div>
```

- Output



o use bootstrap classes, we have to give the proper class_name to the attribute class

True.

False.

What does the following code display? <button class='danger'>Click Me!</button>

A red button

A blue button

A grey button

The progress bar contains an attribute named aria-valuenow that controls the percentage of its filled status. (i.e. how the much the status bar has filled)

True

False

Conclusion

What you should know by now :

[CSS layout Resume - YouTube](https://www.youtube.com/watch?time_continue=862&v=HG53uZi2QS0&feature=emb_logo)
https://www.youtube.com/watch?time_continue=862&v=HG53uZi2QS0&feature=emb_logo

CHCKPOINT

Objective

In this checkpoint, we are going to reproduce the docplanner website using HTML and CSS.

Here is a link to the site.

Docplanner Group

https://www.docplanner.com

You'll find the following instructions helpful.

Instructions

1. *Try to reproduce and replicate the webpage.*
2. *This video can help you better understand how to divide your website using divs: link*

   (50) How do I separate different sections of my webpage using DIVs? - YouTube

   https://www.youtube.com/watch?v=iRPPq9vYig4

3. *Here you can find the best practices in HTML: link*

   30 HTML5 Best Practices for efficient Coding | Themelocation

   https://www.themelocation.com/best-html5-practices/