



Keywords/Questions:

border-radius:100%;
 Alt+mouse cursor

 Ctrl+Alt+mouse cursor

 Carousel
 data-bs-interval="5000"

 data-bs-pause="hover"
 data-bs-ride="carousel"
 data-bs-ride="false"
 class="carousel slide"
 class="carousel"
 <a> can replace <button>
 <div class="carousel-inner">
 <div class="carousel-item">
 class="carousel-item active"
 class="carousel-control-prev"
 data-bs-slide="prev"
 class="carousel-control-prev-icon"

 aria-hidden="true"

 class="visually-hidden"

Notes:

border-radius:100%; :-used to make our image a circle.

-it curves the border and makes it a circle.

Alt+ mouse cursor click and drag:- if you have consecutive lines you can drag the cursor
on all the lines and write the same thing on every line.

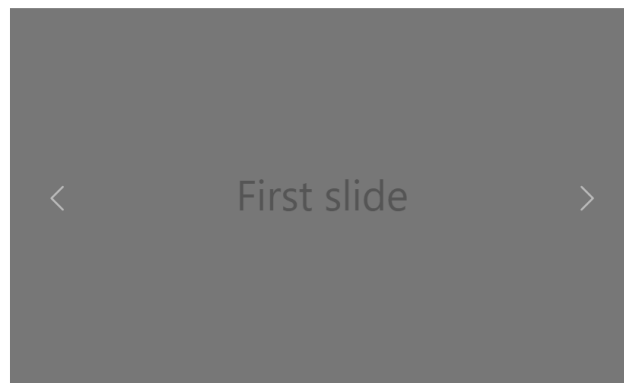
-good for naming classes.

Ctrl+Alt+mouse cursor:- you can do the above for both nonconsecutive and consecutive
lines.

Carousel:-is used to create a slideshow

-we can get codes for it from [getbootstrap.com>docs>components>carousel](https://getbootstrap.com/docs/5.0/components/carousel/)

-there are different ones but this is the one Angela liked:



-we can go to getbootstrap.com and copy and edit the codes in whatever way we want.

Summary:

border-radius:100%:-creates a circle

Alt+Mouse cursor:-allows writing the same thing on consecutive lines

Ctrl+Alt+Mouse: allows writing same thing on both consecutive or non-consecutive lines: for all d/f lines

Carousel:-used to create a slide show.

data-bs-interval="1000":- works in milliseconds controls time for each slideshow. default is "5000"=5 seconds

data-bs-pause='hover' :-pauses slide show when we hover over

More Summary

`data-bs-ride="false"`: - stops autoplay

`data-bs-ride="carousel"`: - starts autoplay

`class="carousel slide"`: - allows sliding when moving from one slide to the next slide

`class="carousel"`: - doesn't allow sliding during transition between different slides.

`<a>` can be used instead of `<button>`. To do this, we need to replace `type="button"` by `role="button"`.

`<div class="carousel-inner">`: contains all the `carousel-item` divs.

`<div class="carousel-item">`: - contains each slideshow elements.

`<div class="carousel-item active">`: - the slide show item that is first seen when we load the page.

`<button class="carousel-control-prev"....>` : - controls left button

`<button class="carousel-control-next"....>` : - controls right button

`data-bs-slide="prev"`

- used to indicate that the button moves the slide show to the left.

- used with the `<button>` element.

`data-bs-slide="next"`

- used to indicate that the button moves the slide show to the right.

- used with the `<button>` element.

`class="carousel-control-prev-icon"`: - creates the icon that lets us move to the left.

`class="carousel-control-next-icon"`: - creates the icon that lets us move to the right.

`aria-hidden="true"`: - indicates the element won't be read by a screen reader for the blind, but visible to the eye

`class="visually-hidden"`

- used to indicate an element that won't be seen to our eyes but will be read by a screen reader for the visually impaired

- Applying padding to a single `carousel-item` alone might be better than applying it to the whole section including the buttons, if we wanna get more space between the buttons and text

```
<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="next">
  <span class="carousel-control-next-icon" aria-hidden="true"></span>
  <span class="visually-hidden">Next</span>
</button>
```

More Notes

-Carousel Settings:

-Go to [getbootstrap.com>docs>components>Carousel>Options](https://getbootstrap.com/docs/4.0/components/carousel/#options)

-data-bs-interval="5000"

-means it will stay 5seconds before it slides to the next slide show page.

-we can just add it to the first line of our carousel code like this:


```
<div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel" data-bs-target="#2000">
```

-data-bs-pause='hover'

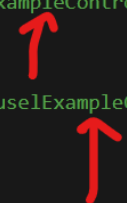
-means it will pause when we hover over the slide.

-it can be added like the above.

-If we change **class="carousel slide"** to **class="carousel"** in the first line of the carousel code, it will stop sliding when moving to the next image. It will just change to the next image without the sliding effect.



```
<div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner">
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <a class="carousel-control-prev" role="button" data-bs-target="#carouselExampleControls" data-bs-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Previous</span>
  </a>
  <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="visually-hidden">Next</span>
  </button>
</div>
```



-the button href and the parent's div id should be the same to allow the buttons to control the slide show



Carousel Setting Continued:

–data-bs-ride:false;

–it means it doesn't auto-play unless you touch the buttons.

–added to our code in the same way as data-bs-target.

–<div class="carousel-inner">

–parent container for the carousel items.

–<div class="carousel-item active">

–the item with the keyword "active" will be the first to appear when we load the page.

–at-least one carousel item should have the keyword active.

–<button class="carousel-control-prev">

–controls the left button

```
<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="prev">
  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
  <span class="visually-hidden">Previous</span>
</button>
```

–<button class="carousel-control-next">

–controls the right button

```
<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="next">
  <span class="carousel-control-next-icon" aria-hidden="true"></span>
  <span class="visually-hidden">Next</span>
</button>
```

–You can use an <a> tag instead of <button>. To do this, you should:

1)change <button> to <a>

2)change type="button" to role="button"

3)change data-bs-target to href (this is an optional step, it doesn't have any effect, they are both interchangeable)

```
<a class="carousel-control-prev" role="button" href="#carouselExampleControls" data-bs-slide="prev">
  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
  <span class="visually-hidden">Previous</span>
</a>
```

```
<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="prev">
  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
  <span class="visually-hidden">Previous</span>
</button>
```

–both codes have the same function they serve as a slide show button, the first one is with an anchor tag and the second one is with a button.

Carousel Settings Continued:



-class="carousel-control-prev-icon"

-creates the icon that lets us move left

-written as ``

-its right side equivalent is class="carousel-control-next-icon"

-aria-hidden="true"

-aria-hidden basically means that when a visually impaired person comes across your web site, they usually have something called a screen reader which goes through the web site and reads what is on there so they can hear what we can see. Now for things like buttons you don't really want the screen reader to read out left button, right button, or, you know, previous button, next button. So you have this thing called aria-hidden, which basically means, make the screen reader skip over this button span so that the visually impaired person doesn't have to listen to what all the buttons are. And this goes towards, you know, making your web site more accessible and friendly to people with disabilities.

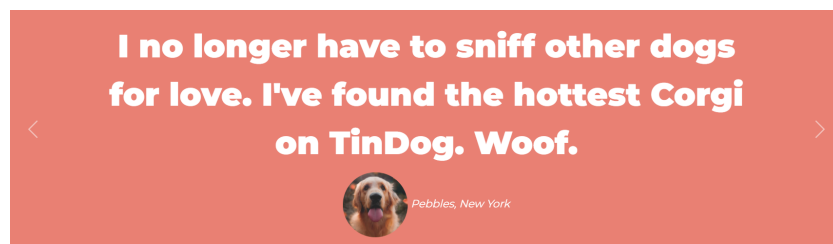
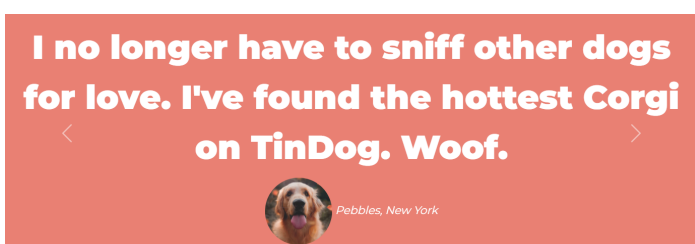
-class="visually-hidden"

-used to include what we want the screen reader of a visually impaired person to read.

-written as `Previous`

-the screen reader would say "Previous" when it sees the button. but the element won't be visible to our eyes.

-Applying padding to a single carousel-item alone might be better than applying it to the whole section including the buttons, if we wanna get more space between the buttons and text



```
#testimonials {
  padding: 7% 15%;
  text-align: center;
  background-color: #ef8172;
  color: #fff;
}
```

```
.carousel-item {
}
```

```
<section id="testimonials">
  <div id="carouselExampleControls" class="carousel slide" data-bs-ride="carousel" data-bs-target="#carouselExampleControls">
    <div class="carousel-inner">
      <div class="carousel-item">
        <h2>I no longer have to sniff other dogs for love. I've found the hottest Corgi on TinDog. Woof.</h2>
        
        <em>Pebbles, New York</em>
      </div>
      <div class="carousel-item active">
        <h2 class="testimonial-text">My dog used to be so lonely, but with TinDog's help, they've found the love of their life. I think.</h2>
        
        <em>Beverly, Illinois</em>
      </div>
    </div>
    <a class="carousel-control-prev" role="button" data-bs-target="#carouselExampleControls" data-bs-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Previous</span>
    </a>
    <button class="carousel-control-next" type="button" data-bs-target="#carouselExampleControls" data-bs-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="visually-hidden">Next</span>
    </button>
  </div>
</section>
```

```
#testimonials {
  text-align: center;
  background-color: #ef8172;
  color: #fff;
}
```

```
.carousel-item {
  padding: 7% 15%;
}
```



Keywords/Questions:

Pricing

bootsnipp.com

Card

class="card"

class="card-header"

class="card-body"

class="card-footer"

class="card-group"

class="invisible"

visibility:hidden;

class="w-72"

class="col-lg-4 col-md-6"

Notes:

Pricing:-

-getbootstrap.com>Examples>Pricing

-you can inspect the code and copy, edit it and style it up in a way that fits your website

-inspect and copy the pricing table for our project.

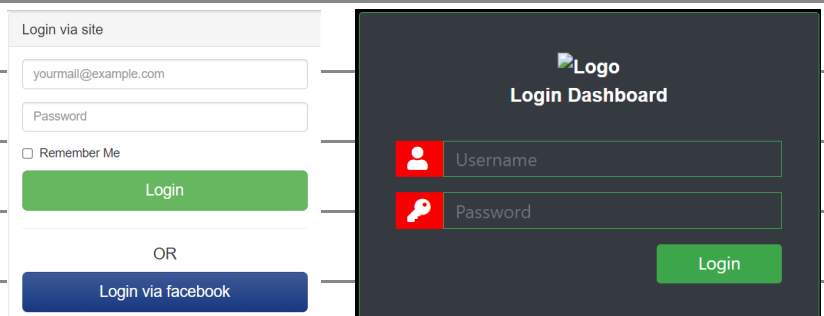
Free	Pro	Enterprise
\$0/mo	\$15/mo	\$29/mo
10 users included 2 GB of storage Email support Help center access	20 users included 10 GB of storage Priority email support Help center access	30 users included 15 GB of storage Phone and email support Help center access
Sign up for free	Get started	Contact us

bootsnipp.com

-website to find different snippets of code for different parts of your website.

-you can check the html and css code needed to create the website or website part

-For example we can find code for different type of login pages by searching "login"



Summary:

Pricing:-a price card you can either make it by yourself or get it from getbootstrap.com>Examples>Pricing

bootsnipp.com :- a website for finding codes for different website parts i.e. a login page.

Card:- used to create a pricing table. -has a header, body and footer -getbootstrap.com>docs>components>card

<div class="card"> :- parent div for a card. -<div class="card-header">:-header div for a card

<div class="card-body">:- body div for a card -<div class="card-footer">:-footer div for a card

<div class="card-group">:- makes a group of cards in the same row. Using the grid system is better than this

class="hidden" & visibility:hidden; :- the element will occupy space that is visible, but we can't see its content.

class="w-75" :-occupies 75% width of the parent container

More Notes

Card: `-getbootstrap.com>docs>components>card`

-we can use it to create our own pricing table

-it has a header, body and sometimes a footer just like our html code

`<div class="card">`:- parent div for a card

`<div class="card-header">`:-header div for a card

`<div class="card-body">`:-body div for a card

`<div class="card-footer">`:-footer div for a card

`<div class="card-deck">`:- not supported on bootstrap anymore.

-used to make a bunch of cards on the same row. They will be side by side & have same height.

-`<div class="card-group">` :-is an alternative that works

Chihuahua	Labrador	Mastiff
Free 5 Matches Per Day 10 Messages Per Day Unlimited App Usage Sign Up	\$49 / mo Unlimited Matches Unlimited Messages Unlimited App Usage Sign Up	\$99 / mo Priority Listing Unlimited Matches Unlimited Messages Unlimited App Usage Sign Up

```
<div class="card-group">
  <div class="card">
    <div class="card-header">
      <h3>Chihuahua</h3>
    </div>
    <div class="card-body">
      <h2>Free</h2>
      <p>5 Matches Per Day</p>
      <p>10 Messages Per Day</p>
      <p>Unlimited App Usage</p>
      <button type="button">Sign Up</button>
    </div>
  </div>

  <div class="card">
    <div class="card-header">
      <h3>Labrador</h3>
    </div>
    <div class="card-body">
      <h2>$49 / mo</h2>
      <p>Unlimited Matches</p>
      <p>Unlimited Messages</p>
      <p>Unlimited App Usage</p>
      <button type="button">Sign Up</button>
    </div>
  </div>

  <div class="card">
    <div class="card-header">
      <h3>Mastiff</h3>
    </div>
    <div class="card-body">
      <h2>$99 / mo</h2>
      <p>Priority Listing</p>
      <p>Unlimited Matches</p>
      <p>Unlimited Messages</p>
      <p>Unlimited App Usage</p>
      <button type="button">Sign Up</button>
    </div>
  </div>
</div>
```

-A disadvantage of using `<div class="card-group">` is that it is not as adaptable as the grid system. They are all evenly stretched apart and have the same height.

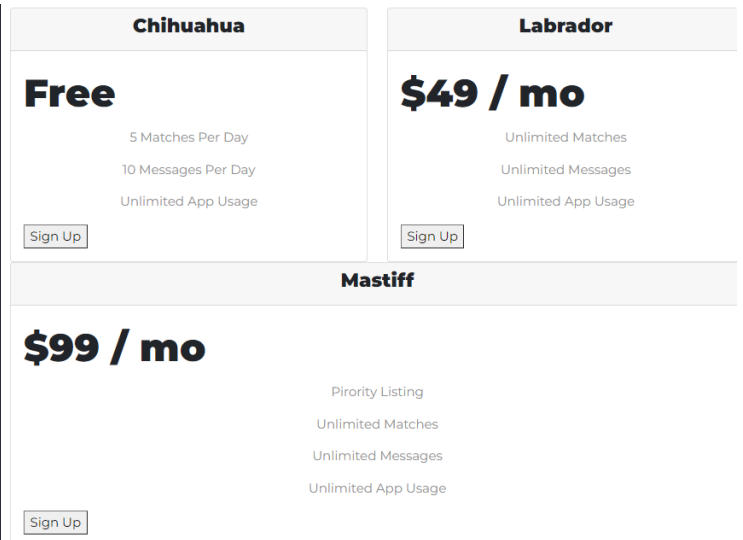
-It is better to use rows and cols (aka the grid system) to better size them.

<div class="col-lg-4 col-md-6">

–means that the horizontal size would be 33.3% of the screen for laptop and desktops, but 50% of the screen for tablets. Don't write just **md-6**, you need to write **col-md-6**.

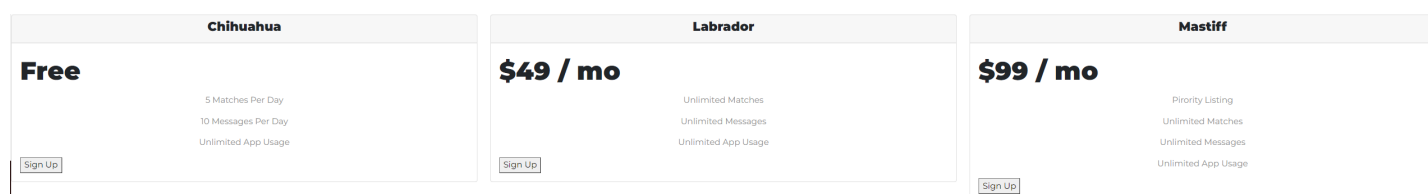
–any screen size below medium (small or mobile screen) is going to cover 100% of the screen.

```
<div class="row">
  <div class="col-lg-4 col-md-6">
    <div class="card">
      <div class="card-header">
        <h3>Chihuahua</h3>
      </div>
      <div class="card-body">
        <h2>Free</h2>
        <p>5 Matches Per Day</p>
        <p>10 Messages Per Day</p>
        <p>Unlimited App Usage</p>
        <button type="button">Sign Up</button>
      </div>
    </div>
  </div>
  <div class="col-lg-4 col-md-6">
    <div class="card">
      <div class="card-header">
        <h3>Labrador</h3>
      </div>
      <div class="card-body">
        <h2>$49 / mo</h2>
        <p>Unlimited Matches</p>
        <p>Unlimited Messages</p>
        <p>Unlimited App Usage</p>
        <button type="button">Sign Up</button>
      </div>
    </div>
  </div>
  <div class="col-lg-4 col-md-12">
    <div class="card">
      <div class="card-header">
        <h3>Mastiff</h3>
      </div>
      <div class="card-body">
        <h2>$99 / mo</h2>
        <p>Priority Listing</p>
        <p>Unlimited Matches</p>
        <p>Unlimited Messages</p>
        <p>Unlimited App Usage</p>
        <button type="button">Sign Up</button>
      </div>
    </div>
  </div>
</div>
```



–For a medium sized screen it will be like the one right above.

–As we can see here using the grid system is better for grouping cards than using the "card-group" keyword.



`class="w-100"`

–means the element will occupy a 100% width of the parent container.

–if we write `class="w-50"`, it will occupy 50% width of the parent container

`class="invisible"`

–the element occupies space but won't be seen to our eyes

–similar effect to the css property `visibility:hidden`.



Keywords/Questions:

Notes:

z-index:1;

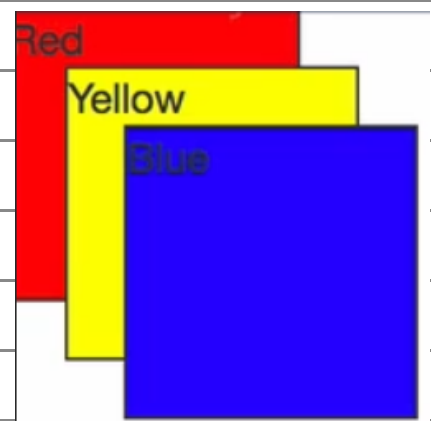
Things that come first in html will be on the back or will be stacked at the back.

```
.red {
  background-color:red;
}

.yellow {
  background-color: yellow;
  left: 20px;
  top: 20px;
}

.blue {
  background-color: blue;
  left: 40px;
  top: 40px;
}
```

```
<div class="red">
  Red
</div>
<div class="yellow">
  Yellow
</div>
<div class="blue">
  Blue
</div>
```



z-index:0; is default

```
div {
  height: 100px;
  width: 100px;
  border: 1px solid;
  position:absolute;
}
```

z-index:-element with the highest z-index will be placed in front of every other element.

first div will be on botthom

-default z-index for every element is 0.

-doesn't work for position:static;, but works for absolute, relative and fixed.

background-color:#fff;

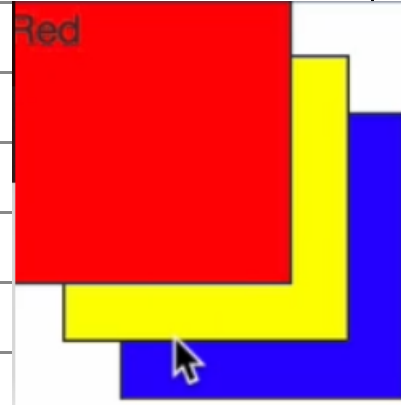
```
<div class="red">
  Red
</div>
<div class="yellow">
  Yellow
</div>
<div class="blue">
  Blue
</div>
```

```
div {
  height: 100px;
  width: 100px;
  border: 1px solid;
  position:absolute;
}
```

```
.red {
  background-color:red;
  z-index: 1;
}

.yellow {
  background-color: yellow;
  left: 20px;
  top: 20px;
}

.blue {
  background-color: blue;
  left: 40px;
  top: 40px;
  z-index: -1;
}
```



Summary:

-Things that come first in html will be on the bottom

-Things that come last in html will be on the top.

-Default z-index is z-index:0;

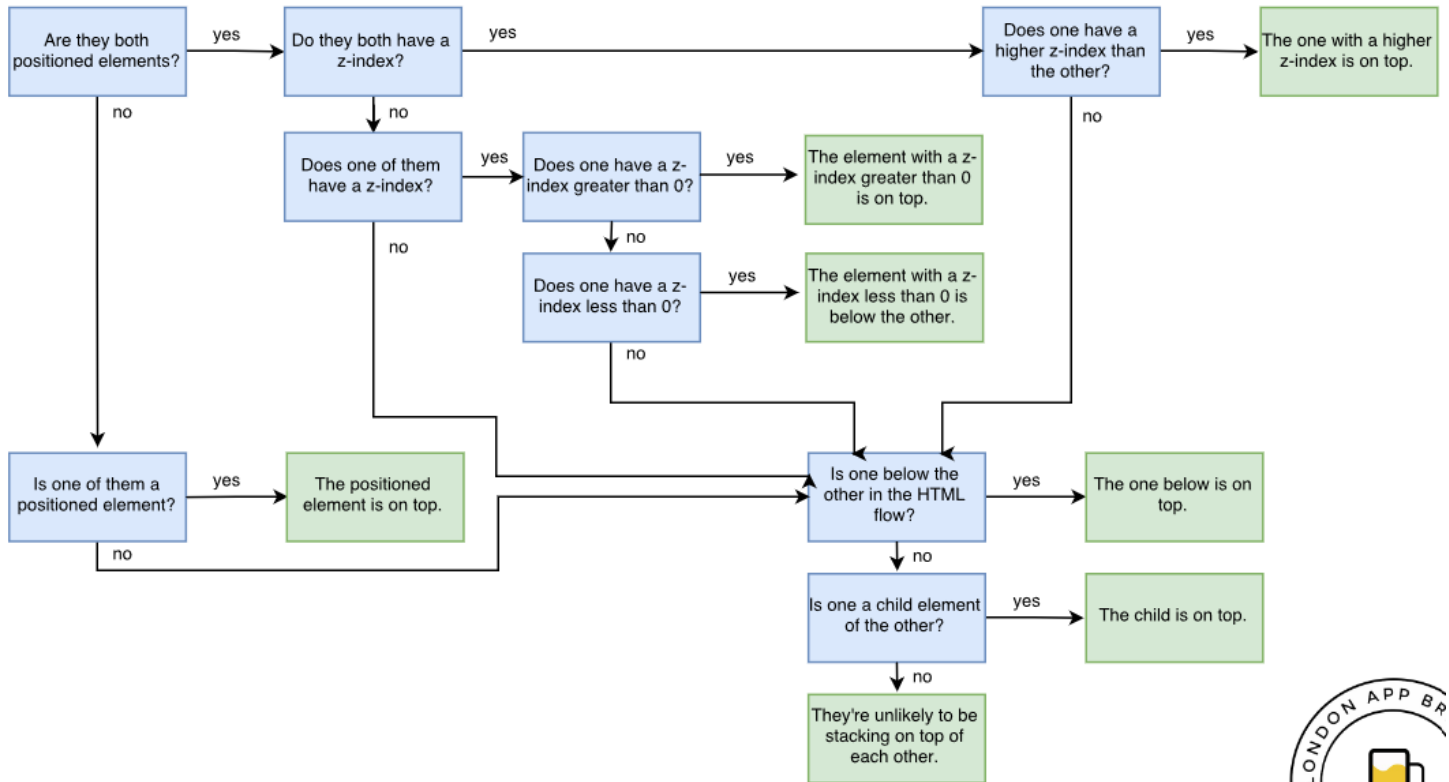
-Element with highest z-index will be stacked on the top

-z-index doesn't work for position:static;

-background-color:#fff; means white

- If all the divs have the same z-index. we revert back to the original stacking order that is based off the order of our code in the HTML file.
- Positioned element
 - every positioning except position: static;

Which one is on top?



background-color:#fff;

- means white that is non-transparent.



Keywords/Questions:	Notes:
Mobile-friendly test	Google rankings are affected by whether if your web site is mobile friendly or not.
Making separate mobile site	Mobile-Friendly test:- tells you whether your website is mobile friendly or not.
	-just type mobile friendly test on google or use this link:
	- https://search.google.com/test/mobile-friendly
Responsive Design	Ways to make your website mobile-friendly:-
CSS breakpoints (Media query breakpoints)	-Making a separate mobile site
	-it will redirect users if they're trying to browse from mobile
@media print{ }	-many big websites do this for example when you try to access Facebook
@media speech{ }	from a mobile phone browser it will take us to m.facebook.com instead
@media (max-width:900px){ }	of facebook.com. You can access the mobile site from desktop by going
@media(max-width:90px) and (min-width:50px) { }	to m.facebook.com
	-this is a lot of work since you are designing 2 separate web sites.
	-Making your website responsive
	-this is what Angela recommends as the best solution.
Mobile-friendly first design	Making websites responsive with CSS native functionality:
Laptop-friendly first design	-Media query breakpoints(aka CSS breakpoints)
	-when using CSS media queries you need to turn off or delete code that
	runs bootstrap because it might affect the media queries.
Summary:	Mobile-Friendly test:-tests whether your website is mobile friendly. -important for google search website ranking
	- https://search.google.com/test/mobile-friendly
	Making websites mobile-friendly:-making a separate mobile site -making your website responsive
	Media query breakpoints(CSS breakpoints):-make your website more responsive -turnoff bootstrap when using this
	@media print{ } :- property displayed while printing.
	@media speech{ } :- applied when using a screen reader.
	@media (min-width:900px) and (max-width:1000px){ }:-when both properties are satisfied for our browser screen

More Notes

@media print

- the property will be displayed only when we print the file
- don't forget to turn off or delete bootstrap

```
@media print{  
  h1{  
    color:red;  
  }  
}
```

```
<h1>hale</h1>
```



@media speech

- activated if the website is being read to a visually impaired person by a screen reader

@media <type> <feature>

:- Usually either a <type> or a <feature> is applied at once. We don't use them both in the same line. (Not sure this is my opinion)

- @media: – this keyword says that everything that comes afterwards is a media query.
- <type>: – is the type of media or medium we are selecting on

- the code should only be activated if the web site is being displayed on a screen, or if it's being printed, or if it's on a screen reader, all of those kind of thing

- <feature>: – a condition for the property to be displayed. For example: –

```
@media screen (min-width: 900px) {  
  
  //Change Something  
  
}
```

– the property within this { } is applied when we have a large screen, a screen > 900px (a laptop screen).

```
@media (min-width: 900px) {  
  h1{  
    color:red;  
  }  
}
```

– we shouldn't include screen when we write our code

@media(max-width:900px):

–displayed when the screen size is less than 900px(less than a laptop screen)

We can use more than one media queries in the same line

```
@media (min-width: 900px) and (max-width: 1000px) {  
h1 {  
  font-size: 60px;  
}  
}
```

–applied when we have browser screen size that is between 900 and 1000px.

–We can check the screen size of our browser on chrome by touching inspect(or going to developer tools) and trying to increase the screen size of my browser. On the right hand top edge you will be able to see the size of your browser.

–When when you're designing a responsive web site there's really two ways or two directions where you can go about doing this. You can either go **mobile first**, so you start designing a web site at sort of this size, and then you start looking at how you can make it look good on laptop, or the other way, which is what we've done basically, is designed our web site **for desktop and then made it responsive and look good on the smaller sizes**. Now there's a lot of debate about which direction is better, but what Angela recommends is that you actually try doing it both ways, so starting by designing and creating your HTML and CSS for a mobile size web site, and then trying to make it responsive on tablet and laptop, and also going the other direction for maybe another web site project, and you'll find out which you prefer.



Keywords/Questions:

Padding

Notes:

Padding:- is inside the element. For example, the gap between the border of the element and the text inside the element. It will both add additional gap and also push the text from the sides.


Margin

Margin:- is adding space outside the border of an element. It can be added in any side.

Use the box model through inspect element to edit your codes margin and padding before editing it on atom

Box Model

```
<a class="nav-link" href="#pricing">Pricing</a>
```



-this will take us to the div or section with id="pricing"

-when we use an <a> tag with href="#someid" it will take us to the div or section with that id.

When searching through websites if we know the id of the part of the webpage we want to access, we can write it next to the url with a # followed by the id. For example, in the website <https://github.com/vicc/chameleon> if we want to go to the part with id="-product-features", we can write the url as: <https://github.com/vicc/chameleon#-product-features>

URL+#+id

Summary:

Padding:- gap between the border and inside content. Increasing it will p

Margin:- added space outside the border of the element

Box model: use it first before going to atom

:- can be used to go to a particular section of the website.

-very good to use as a navigation link

URL+#+id :- we can use it to access parts of a website



Keywords/Questions:

Notes:

Code Refactoring Principles

5aVWVUSfad`YBd`UbWZ

#Z DVSTI[1fk

ZeWVkfZ`YadS`lIW`S`aYUS`iSk1

ZeWVkfZ`YUa_ _WfWea fZSf kag US`g`WHS`Vi ZSf WUZ bScf aX

Readability

fZWaWWeSTagf1

-When you come back in a year to try and understand your code, can you quickly understand what's going on?

Modularity

SZ ? aVg`Sdfk

-Is how easy it is to narrow down your code when you encounter errors

-If one particular part of your web site breaks down, is your code modular

Efficiency

enough that you would be able to narrow down on the exact section of code or code file that's responsible for the problems that's occurring

-Is how divided your code is to different parts.

Length

%Z 7XUWk

-How fast does your code run?

-It is less important compared to keeping your code modular & readable.

Code golf

&Z >WYZ- don't repeat yourself in your code so that you keep your code well

structured. If it will become less readable, don't reduce length.

Summary:

-Code golf:-game writing shortest code, codegolf.stackexchange.com

4 Code Refactoring Principles:-

1. Readability:- using comments -can you read and understand your code easily after a year

2. Modularity:- your code is divided to different smaller parts. -easy to track errors and mistakes.

3. Efficiency:- speed of your code -C is faster -not as important as the above 2

4. Length:- avoid repeating. Don't make it short, if it makes your code incomprehensible.

Code golf:- writing a code as short as possible competition.



Keywords/Questions:

Notes:

Refactoring code rules

Rules for Refactoring code

1) Put all your HTML elements together

2) For your HTML elements(aka body, h1, h3,... tags) place only broad styling:

-like color, font-family, ...

3) If there is a repetition of the same property for different elements for example

let it be font family we can write it as

```
h1, h2, h3 {
  font-family: "Montserrat";
  font-weight: 900;
}
```

4) When using specific styling use classes to style each common occurrence of a

particular type of element. So this is what we mean about modularity, so it allows

you to drill down on a specific section easily and figure out what the problems are or

change the style of one particular section without affecting the rest of the web site.

5) When writing your code, when you see opportunities for refactoring you would do

it right there and then rather than waiting for a larger refactoring session

6) Combining selectors to specify settings

```
#title .container-fluid{
  padding: 3% 15% 7%;
}
.container-fluid{
  padding: 7% 15%;
}
```

-This targets the container-fluid div under the title section.

Advantage of using this is that the padding from the 2

won't add up. The more specific one will override it. If we

just use an id, it would result in addition of padding

Summary:

Refactoring Code rules:-Put HTML elements in style sheet together & use them for broad styling like color and font-family

-We can use multiple selectors in the same line i.e. **h1, h2 {font-family:"Montserrat"; }**

-Use specific classes for common occurrences

-Refactor code as you write it

-Combining selectors helps avoid addition of properties like padding

```
#title .container-fluid{
  padding: 3% 15% 7%;
}
.container-fluid{
  padding: 7% 15%;
}
```



Keywords/Questions:

Notes:

Multiple Selectors

Multiple Selectors

```
selector1, selector2 {
  h1, h2, h3, h4, h5, h6 {
    font-family: "Montserrat-Bold";
  }
}
```

-the space in between selectors is not mandatory.

Hierarchical Selectors

-used for applying the same property for 2 or more selectors.

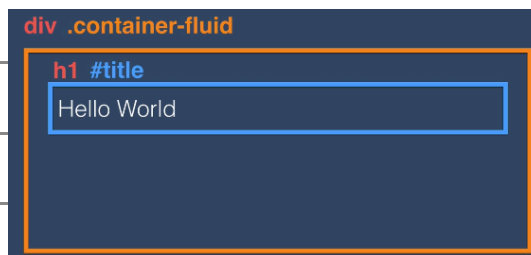
Hierarchical Selectors

-spacing between selectors is very important. -read from right to left.

-you should have one space and the order is that the **first selector is from the parent**

and the **second selector is from the child.**

Combined Selectors



```
div h1 {color: red;}
```

-this above code means that all

h1 tags inside a div tag will

have a red color.

Selector Priority:

```
.container-fluid h1 {color: red;} .container-fluid #title {color: red;}
```

1st ID

-this code means that all h1 tags inside a - You would never write this code, you

2nd Class

parent that has the class "container-fluid" would simply just target the id title.

3rd HTML tags

should have this style.

Summary: Multiple Selectors:- applying the same property for 2 or more different selectors. Spacing doesn't matter.

```
selector1, selector2 {
}
```

Hierarchical Selectors:- 1 space in between selectors

1st selector is from parent & 2nd selector from child.

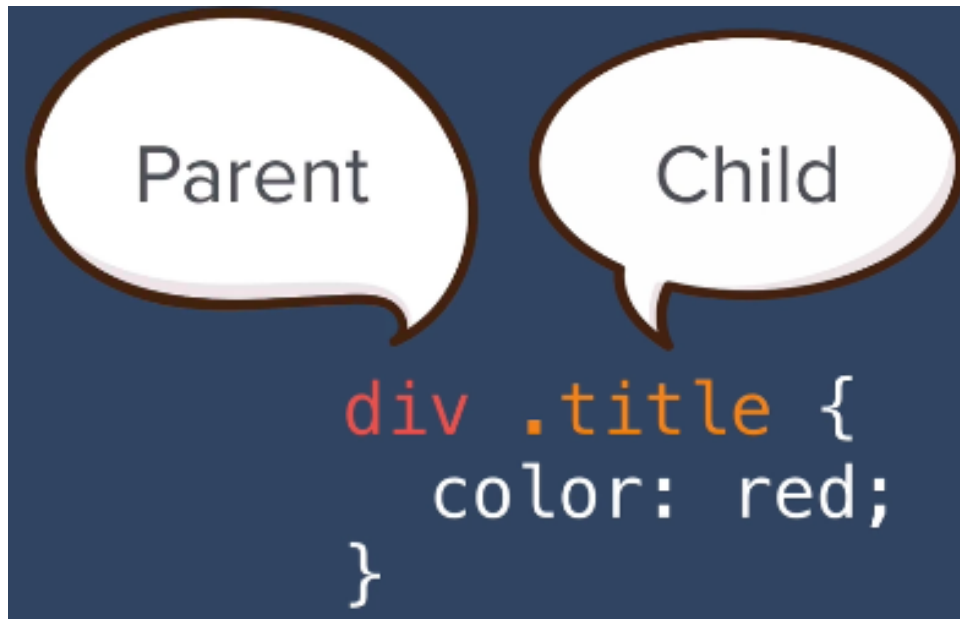


Combined Selectors:- no spacing between selectors & all the selectors must come from 1 element.

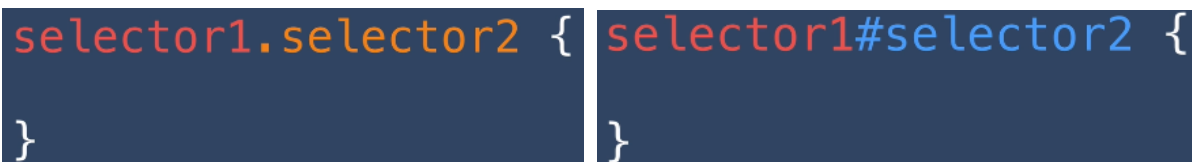
```
selector1.selector2 {
} selector1#selector2 {
}
```

Priority of Selectors in order: 1st id 2nd class 3rd html tags

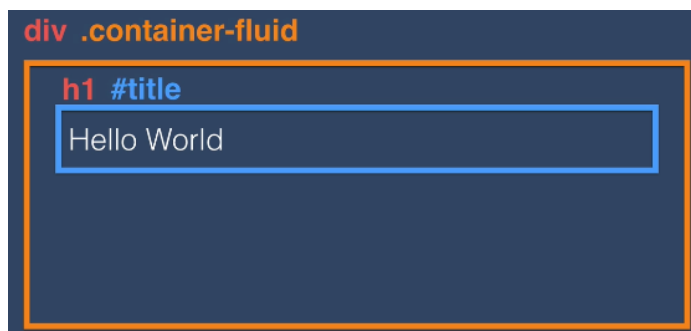
Hierarchical Selectors Continued :-



Combined Selectors



-have to all occur in the same element.



-This targets an h1 tag with the id title.



-This targets the div that has a class container-fluid.

-In this case we're saying that the div that has a class of container-fluid should have the text color of red being applied to all of its content.



-But this doesn't work if you're saying that the div with an id of title should have the text color red. This is not a valid code because currently on screen we don't have a single div that has the id of title.

element .class VS. element.class

```
<div class="container">
  <h1 class="title">Hello World!</h1>
</div>
<div class="container-fluid">
  <h1 class="title">Good Bye World!</h1>
</div>
```

```
.container .title{
  color:red;
}
```

Hello World!

Good Bye World!

```
<div class="container">
  <h1 class="title container">Hello World!</h1>
</div>
<div class="container-fluid">
  <h1 class="title">Good Bye World!</h1>
</div>
```

```
h1.container.title{
  color:red;
}
```

Hello World!

Good Bye World!

—an element inside the HTML that has not only a class of container but also a class of title.

```
<div class="container">
  <h1 id="heading" class="title">Hello World!</h1>
</div>
<div class="container-fluid">
  <h1 class="title">Good Bye World!</h1>
</div>
```

```
h1.title{
  color: red;
}
```


Hello World!

Good Bye World!

Prioritization of selectors

- ids come first
- then classes
- then at last html tags

```
<section class="colored-section" id="testimonials"> == $0
```

```
#testimonials {  
    background-color:  #ef8172;  
}
```

```
.colored-section {  
 background-color:  #ff4c68;  
 color:  #fff;  
}
```

-For example here the background-color is overwritten by the id. We can check this by inspecting the code and going into styles.

Topic/Title: Advanced CSS-Selector Priority
Completing the Website
Download the Completed Website
Tip from Angela-Building a Programming Habit



Keywords/Questions:

Notes:

```
h1{  
  color:blue;  
  color:red;  
}
```

If you have two different CSS values for a single property in a single selector, the last property will be applied.

```
<h1 id="heading" class="title">Hello World</h1>
```

```
h1{  
  color:red;  
  color:green;  
}
```

Hello World

ID>Class>HTML tag

-The color green property will be applied

Inline>Internal>External

Order of priorities:

-ID>Class>HTML tag

-Inline CSS>Internal CSS>External CSS

-Inline CSS>ID>Class>HTML tag

avoid ids unless it is for navigation or carousel

```
<h1 id="heading" class="title" style="color:orange">Hello World</h1>
```

avoid inline css

```
h1{  
  color:red;  
}  
.title{  
  color:yellow;  
}  
#heading{  
  color:blue;  
}
```

Hello World

one class for a single item

Summary:

If there are 2 values for the same property under the same tag the last one will be applied.

Order of properties -Inline CSS>Internal CSS>External CSS -Inline CSS>ID>Class>HTML tag

Preventing conflicting rules:- avoid using IDs unless you need it for bootstrap carousel or navigation

-apply only a single custom class to each of your elements..

-avoid inline CSS

Just because sth repeats itself isn't a good enough reason to get rid of it.

More Notes

How can we prevent creating conflicting rules?

- Use IDs sparingly, don't try and use it when you can use a class. for example, in our case we've really only got ids for our sections, and, in part, that's because **it helps us with our navigation.**
- just because you only have one of something isn't good enough to give it an id instead of a class.
- if you're working with Bootstrap carousels or Bootstrap elements, then they do need an id to target from the href. **They need that navigational ability of the id**, and that is a case where you might consider using it, but always first consider using class instead of going straight to an id, even if it only appears once.
- apply only a single custom class to each of your elements.

```
<h1 class="title hale">Hello World</h1>
```

- This is bad practice. There are two classes for a single element. Unless it is when you are applying bootstrap to your elements don't do that.

- avoid inline styles. Using inline CSS is laziness and bad practice

Just because something repeats itself isn't itself a good enough reason for you to try and get rid of it. Don't group things that became the same by chance together