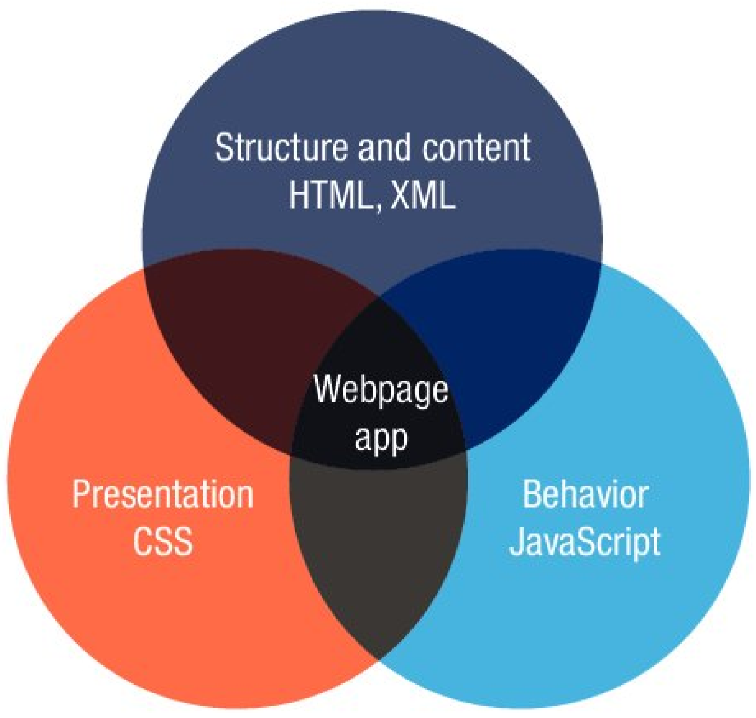
🖥️

Introduction to CSS

I. Cascading Style Sheet

What is Cascading Style Sheet ?

- CSS (Cascading Style Sheets) is a declarative language that controls how web pages look in the browser.
- Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents.
- Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.



The power of CSS ?

CSS is an incredibly powerful technology.

It can take the same HTML structure and present it in such drastically different ways that you would never even guess that the underlying structure of the content are exactly the same ones.

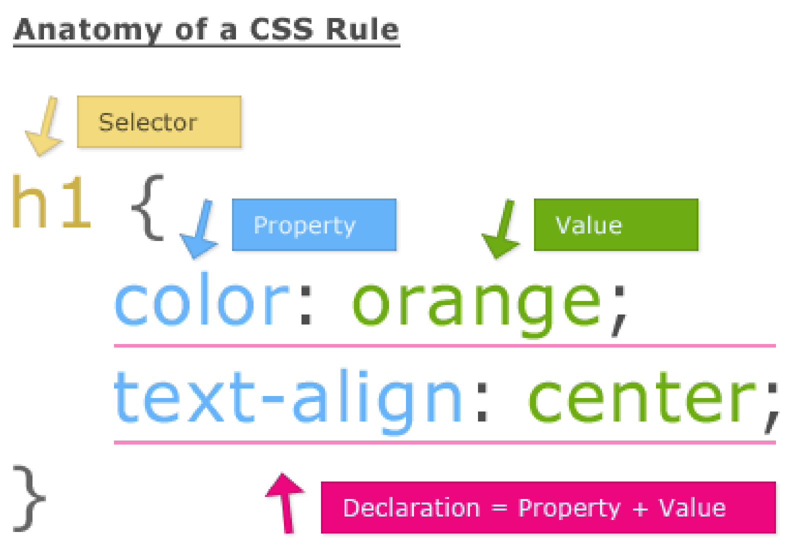
The following a website below that doesn't only illustrate this point, but should also amaze you as well.

- http://csszengarden.com/
- http://csszengarden.com/198/
- http://csszengarden.com/199/

The HTML stays exactly the same and the CSS file, or files, is what they provide.

The only difference is the CSS styling that got applied to it. This is the same website, again, with different CSS styling applied to it.

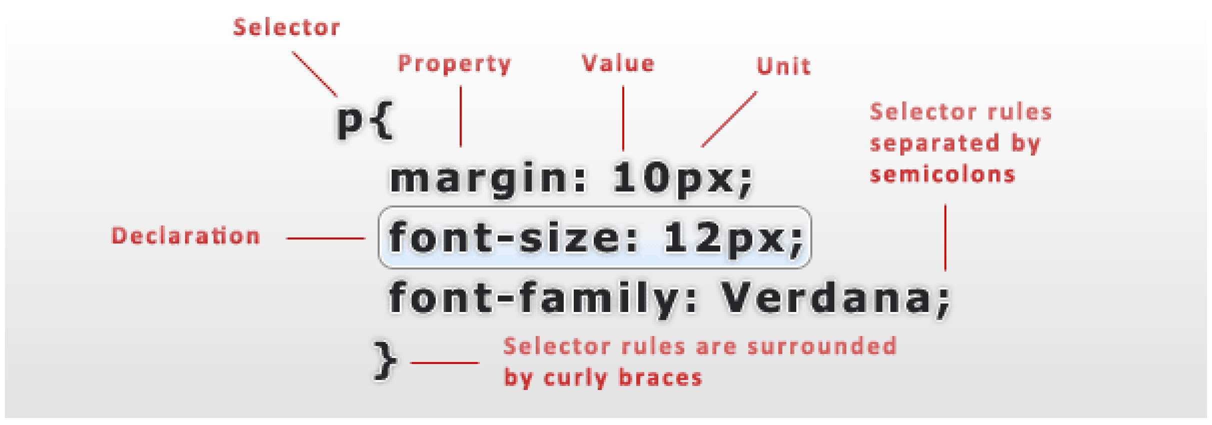
Anatomy of a CSS rule



The curly braces allow you to group several properties into the same rule (selector), the colon tells the parser where the property ends and the value starts, and the semicolon tells the parser where the value ends.

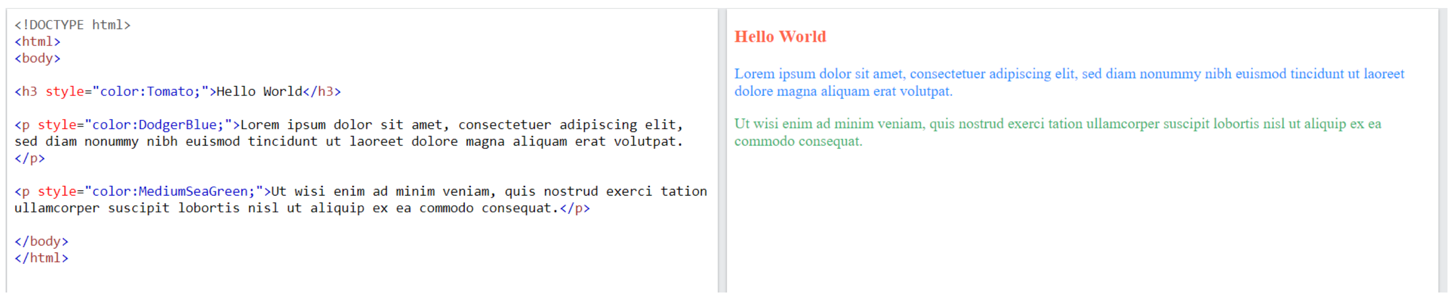
The number of declaration is zero of more that is allowed

Terminologies



- ▼ Each rule consists of a specific property, followed by a colon, followed by a value (or a list of values separated by spaces), followed by a semicolon.
- ▼ A ruleset consists of a selector (which we can think of as a tag name for now), an opening curly brace, a list of rules, and a final closing curly brace.
- ▼ A ruleset is simply a collection of style rules that are applied to some subset of elements in the DOM.
- ▼ A CSS file is a collection of rulesets.

A CSS file describes how specific elements of the HTML should be displayed by the browser.



```
<!DOCTYPE html>
<html>
<body>

<h3 style="color:Tomato;">Hello World</h3>

<p style="color:DodgerBlue;">Lorem ipsum dolor sit amet, consectetuer adipiscing elit,
sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.
</p>

<p style="color:MediumSeaGreen;">Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.</p>

</body>
</html>
```

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Validate CSS

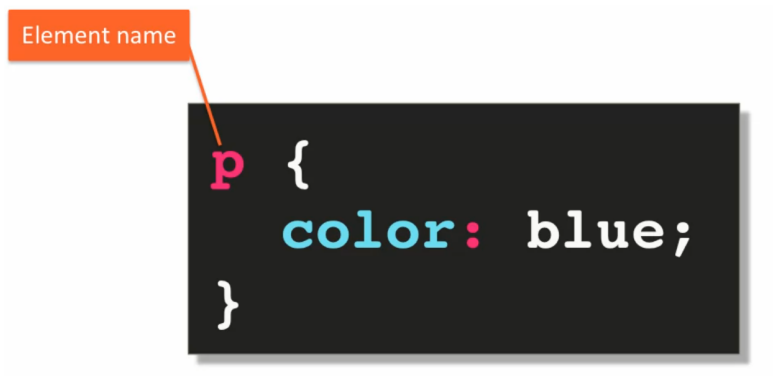
Go to the link to validate your CSS code: https://jigsaw.w3.org/css-validator/

II. Element Selector

An element type selector is a selector that targets an element by the tag name.

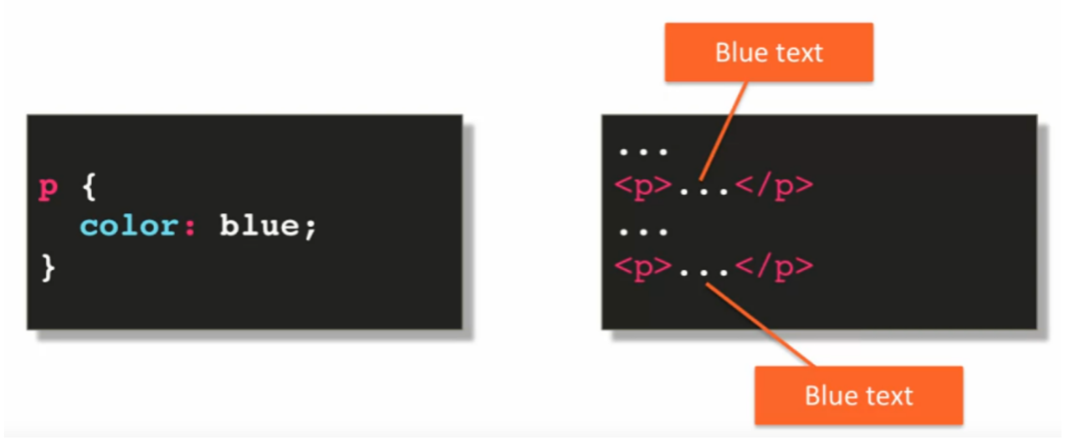
Element Selector Syntax

- also known as Type Selector



```
p {
  color: blue;
}
```

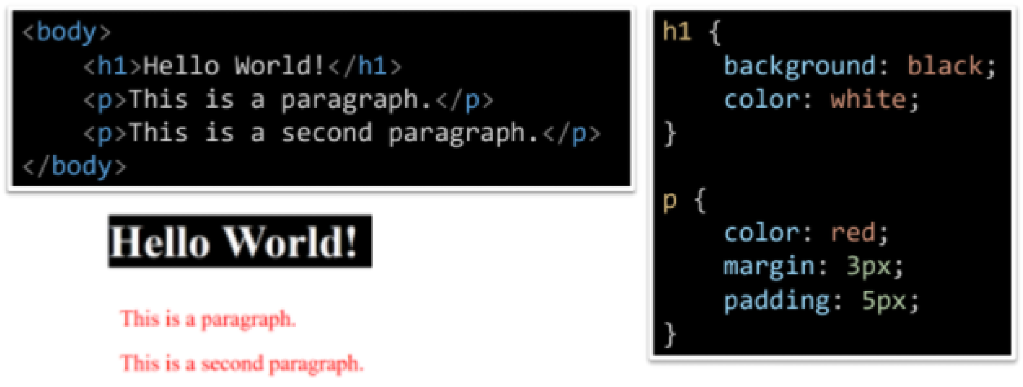
Then, it directly targets all HTML5 <p> elements: (Element Selector Effect)



```
p {
  color: blue;
}
```

```
...
<p>...</p>
...
<p>...</p>
```

Element Selector example:



```
<body>
    <h1>Hello World!</h1>
    <p>This is a paragraph.</p>
    <p>This is a second paragraph.</p>
</body>
```

```
h1 {
    background: black;
    color: white;
}

p {
    color: red;
    margin: 3px;
    padding: 5px;
}
```

Hello World!

This is a paragraph.

This is a second paragraph.

- The first ruleset styles the h1 element, and the second ruleset styles both p element

---

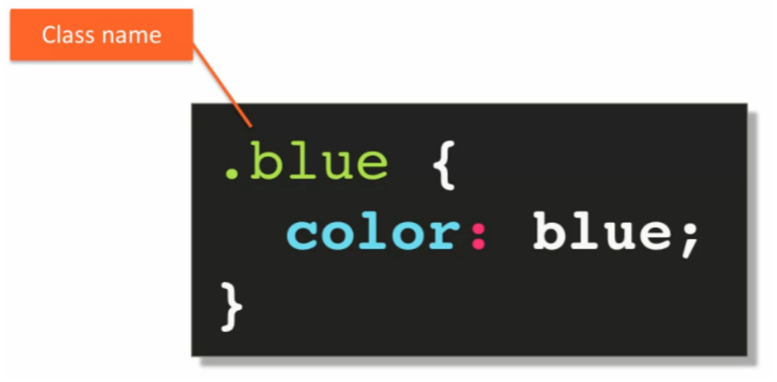
III. Class Selector

The `.class` selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

Class Selector Syntax

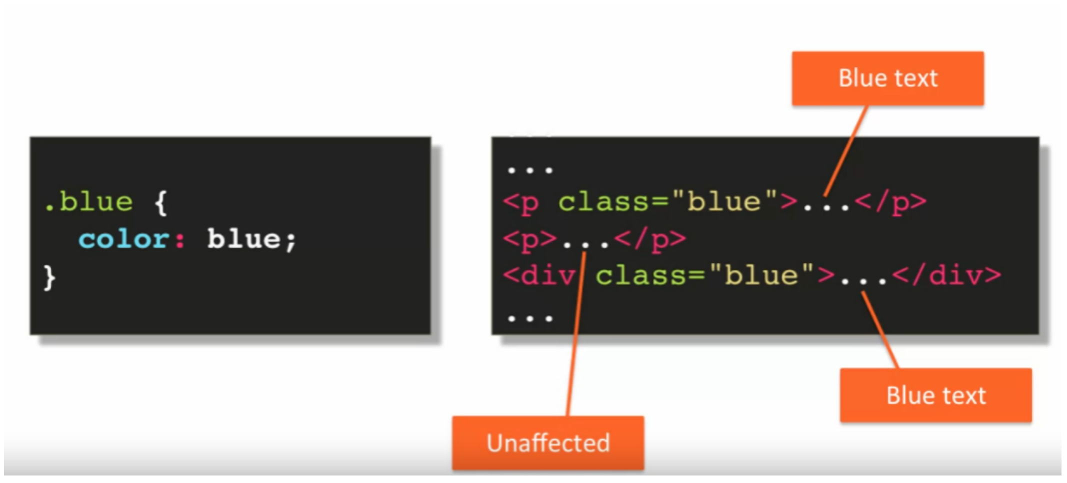
- Define using dot sign .



```
.blue {
  color: blue;
}
```

Then, it directly targets the class belong to (Class Selector Effect)

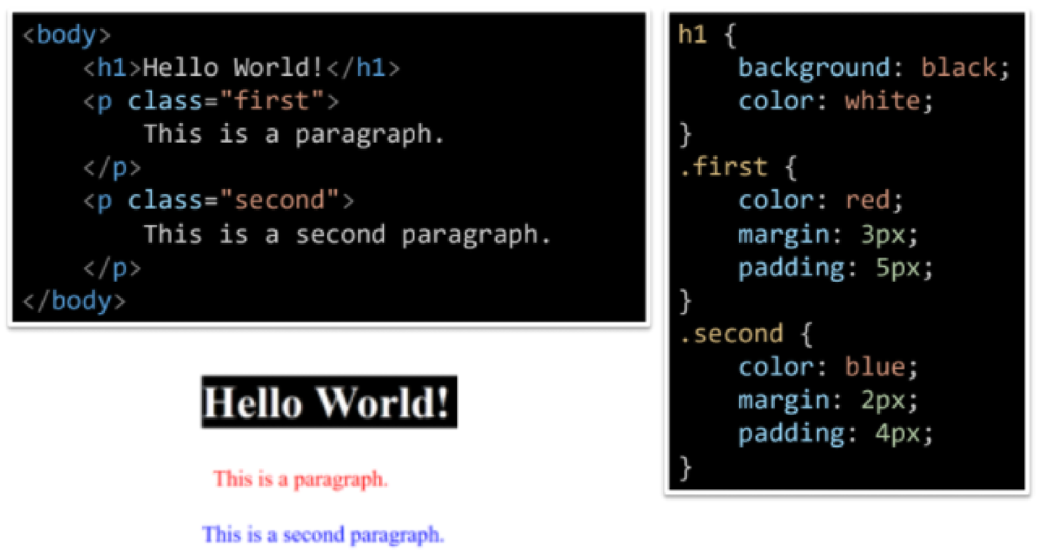
The class selector does require a little bit of a change to your HTML document in that each element that you want that class applied to has to have a attribute class equal to the class name.



```
.blue {
  color: blue;
}
```

```
...
<p class="blue">...</p>
<p>...</p>
<div class="blue">...</div>
...
```

Class Selector example:



```
<body>
    <h1>Hello World!</h1>
    <p class="first">
        This is a paragraph.
    </p>
    <p class="second">
        This is a second paragraph.
    </p>
</body>
```

```
h1 {
    background: black;
    color: white;
}
.first {
    color: red;
    margin: 3px;
    padding: 5px;
}
.second {
    color: blue;
    margin: 2px;
    padding: 4px;
}
```

- Notice that a different paragraph applied different class which have distinct affected

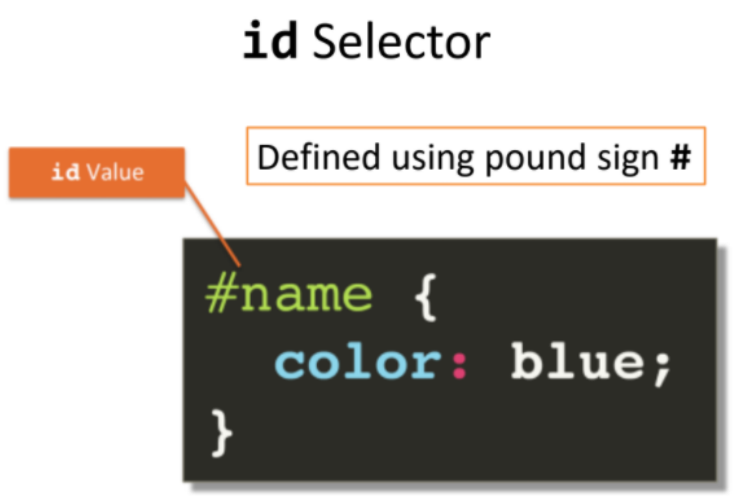
---

IV. Id Selector

The CSS ID selector matches an element based on the value of the element's id attribute. In order for the element to be selected, its id attribute must match exactly the value given in the selector.

Id Selector Syntax

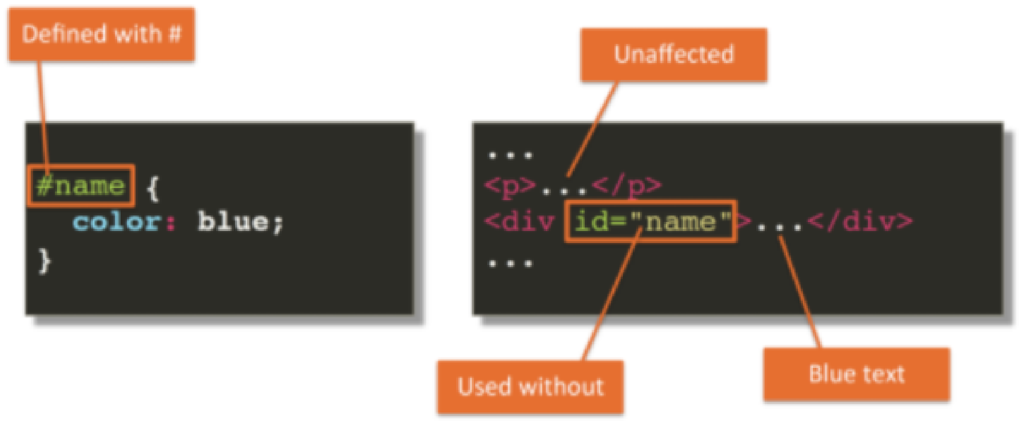
- Define using pound sign #



```
#name {
  color: blue;
}
```

> An id name cannot start with a number!

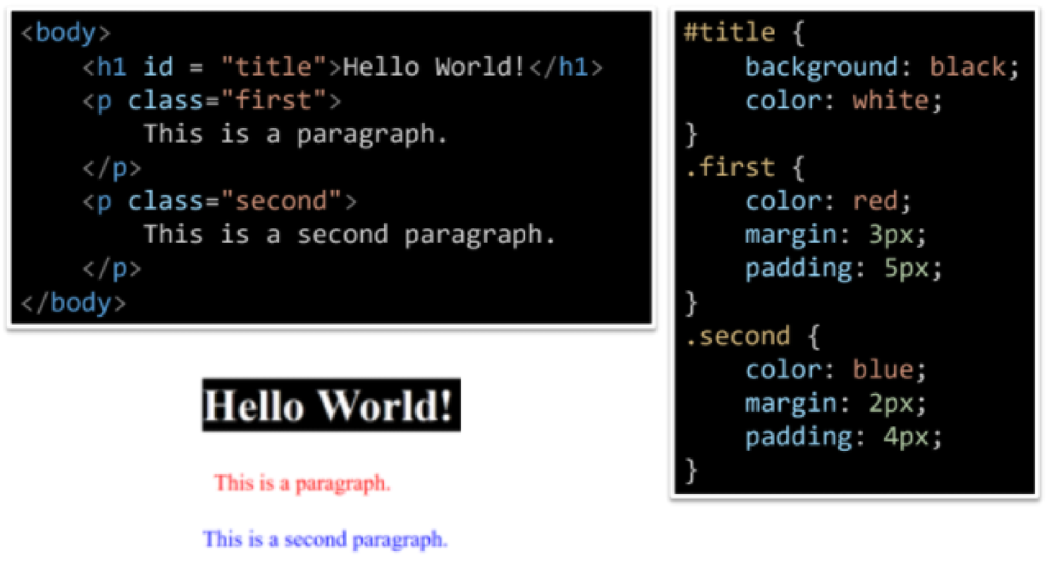
Id Selector Effect



```
#name {
  color: blue;
}
```

```
...
<p>...</p>
<div id="name">...</div>
...
```

Id Selector example:



```
<body>
    <h1 id = "title">Hello World!</h1>
    <p class="first">
        This is a paragraph.
    </p>
    <p class="second">
        This is a second paragraph.
    </p>
</body>
```

```
#title {
    background: black;
    color: white;
}
.first {
    color: red;
    margin: 3px;
    padding: 5px;
}
.second {
    color: blue;
    margin: 2px;
    padding: 4px;
}
```

Hello World!

This is a paragraph.

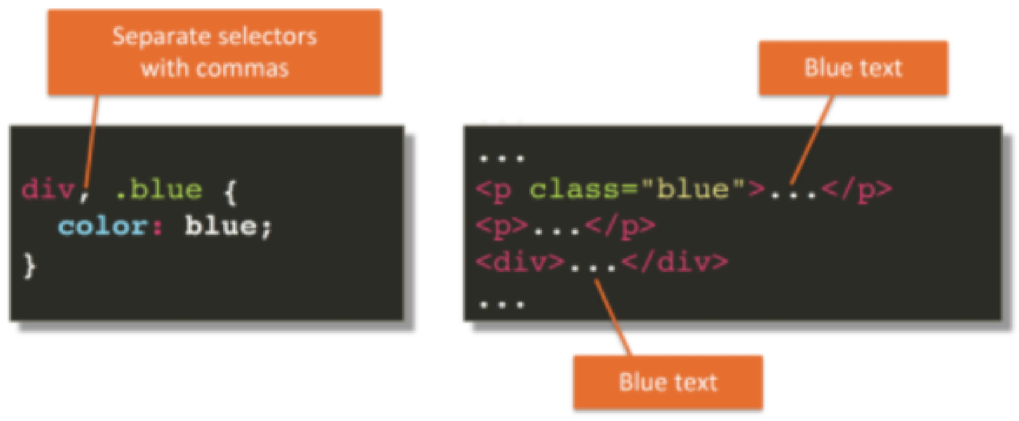
This is a second paragraph.

- Notice that a different paragraph applied different Id which have distinct affected

Grouping selectors

The grouping selector selects all the HTML elements with the same style definitions.

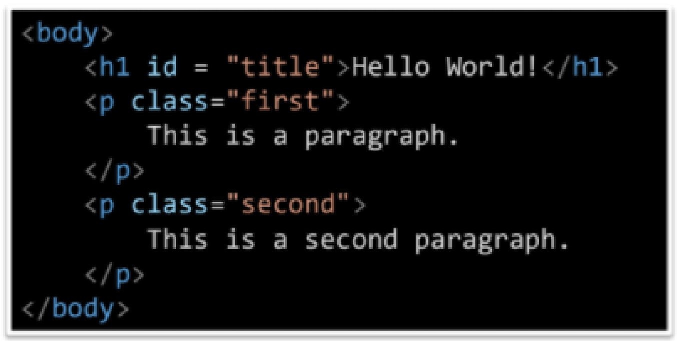
Look at the following CSS code (the div and the class .blue have the same style definitions):



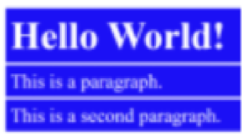
```
div, .blue {
  color: blue;
}
```

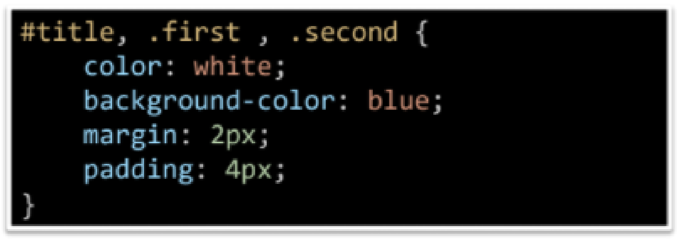
```
...
<p class="blue">...</p>
<p>...</p>
<div>...</div>
...
```

Grouping selector example



```
<body>
    <h1 id = "title">Hello World!</h1>
    <p class="first">
        This is a paragraph.
    </p>
    <p class="second">
        This is a second paragraph.
    </p>
</body>
```



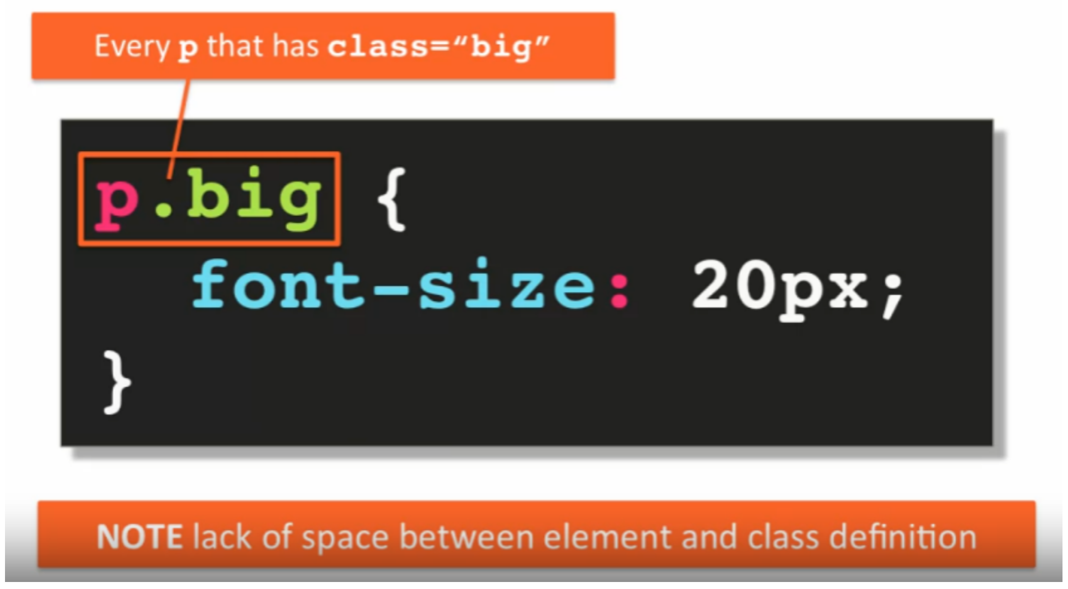


```
#title, .first , .second {
    color: white;
    background-color: blue;
    margin: 2px;
    padding: 4px;
}
```

It's affected those selectors that have definition

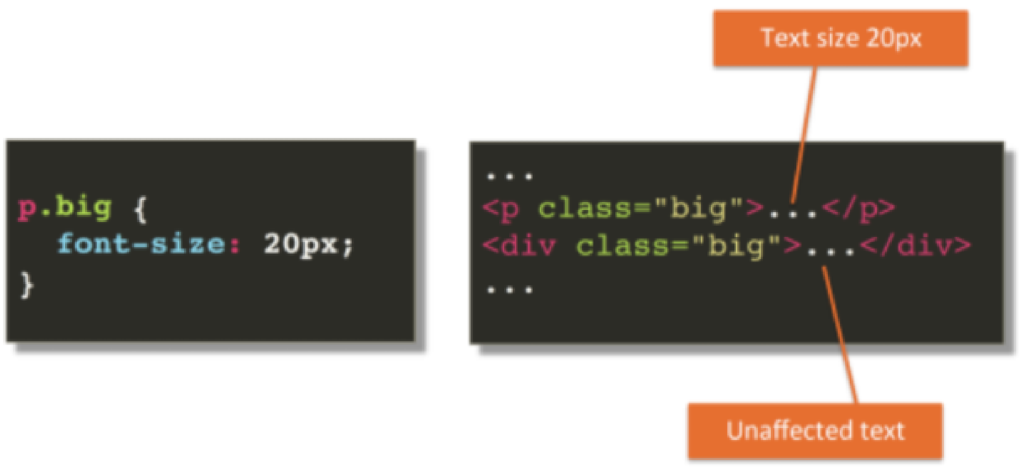
V. Element with Class Selector

The element.class selector is used to select the specified element with the the specified class.


Every p that has class="big"
NOTE lack of space between element and class definition

```
p.big {
    font-size: 20px;
}
```

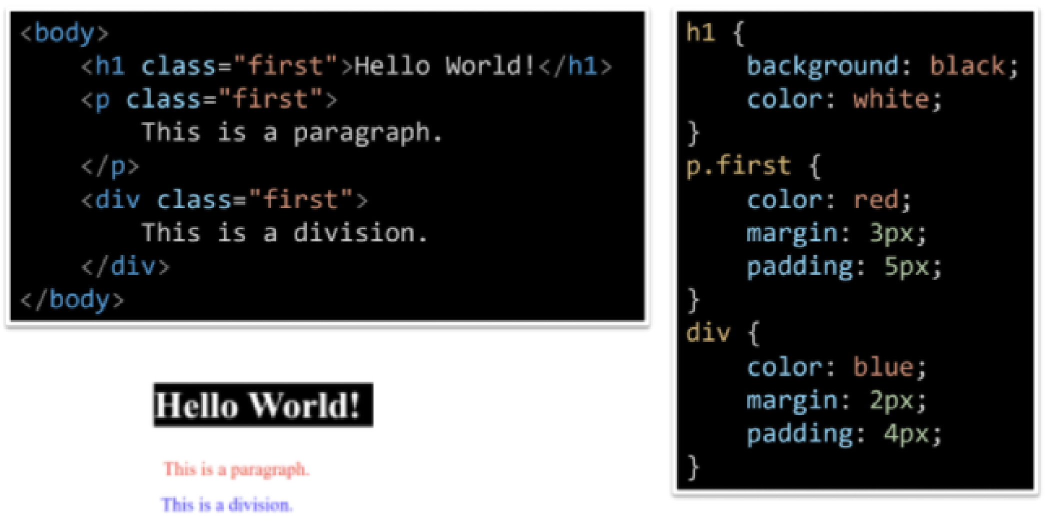
Similar to class selector, it's affected the every <p> element with the class "big":



```
p.big {
  font-size: 20px;
}
```

```
...
<p class="big">...</p>
<div class="big">...</div>
...
```

Element with class selector example:



```
<body>
    <h1 class="first">Hello World!</h1>
    <p class="first">
        This is a paragraph.
    </p>
    <div class="first">
        This is a division.
    </div>
</body>
```

```
h1 {
    background: black;
    color: white;
}
p.first {
    color: red;
    margin: 3px;
    padding: 5px;
}
div {
    color: blue;
    margin: 2px;
    padding: 4px;
}
```

Hello World!

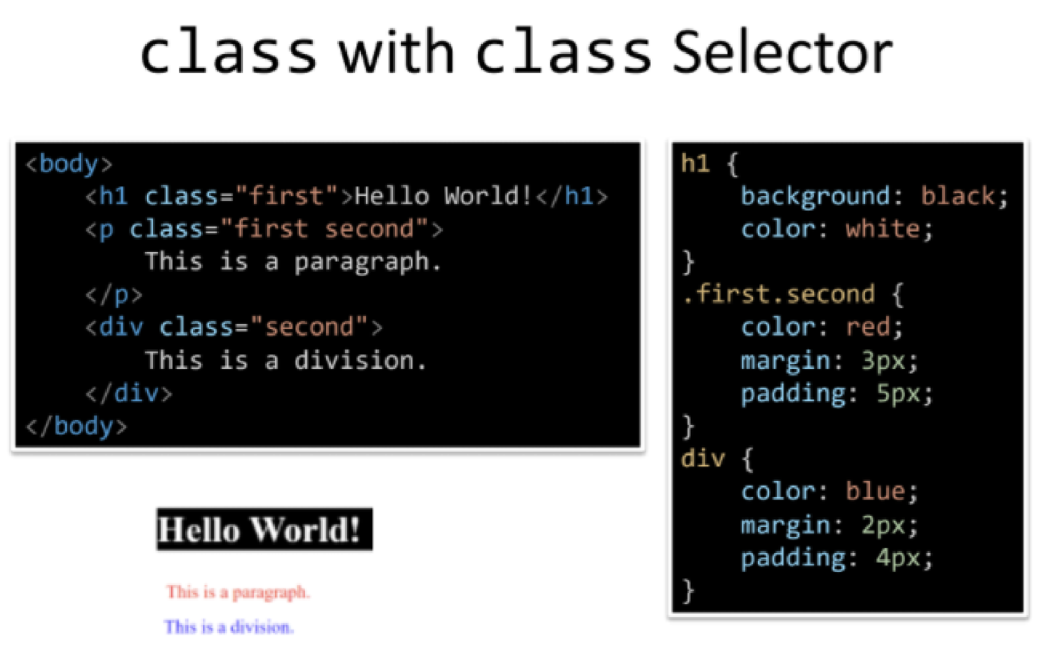
This is a paragraph.

This is a division.

VI. Class with Class Selector

Similar to class selector but class with class selector add class after class. That's it

The following below will help you understand the syntax



class with class Selector

```
<body>
    <h1 class="first">Hello World!</h1>
    <p class="first second">
        This is a paragraph.
    </p>
    <div class="second">
        This is a division.
    </div>
</body>
```

```
h1 {
    background: black;
    color: white;
}
.first.second {
    color: red;
    margin: 3px;
    padding: 5px;
}
div {
    color: blue;
    margin: 2px;
    padding: 4px;
}
```

Hello World!

This is a paragraph.

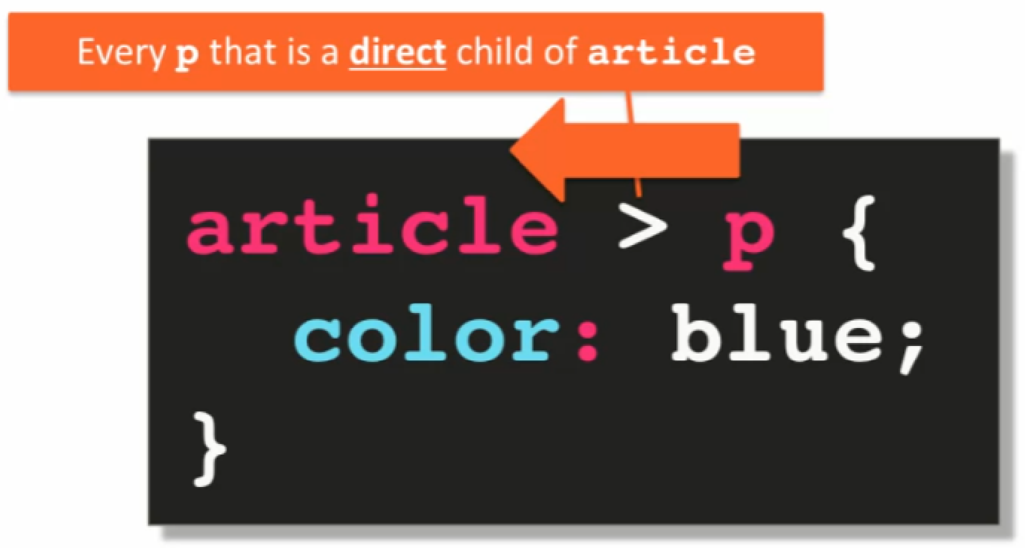
This is a division.

As you can see class .first.second affect the class with class selector and select class by "first second". There is space between 2 class in this example.

VII. Child Selector

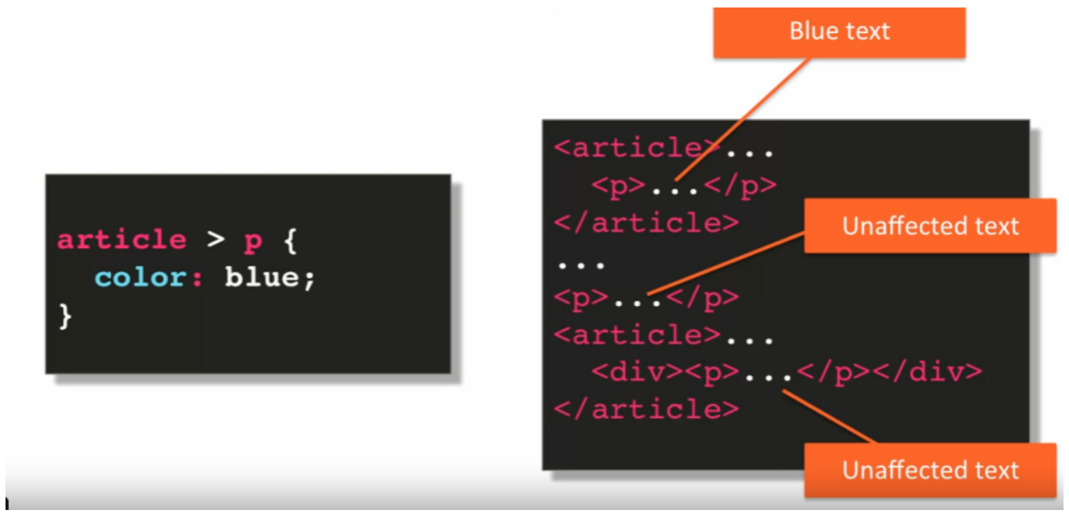
A child selector matches when an element is the child of some element. A child selector is made up of two or more selectors separated by ">".

The following rule sets the style of all P elements that are children of Article:



```
article > p {
   color: blue;
}
```

Child selector effect



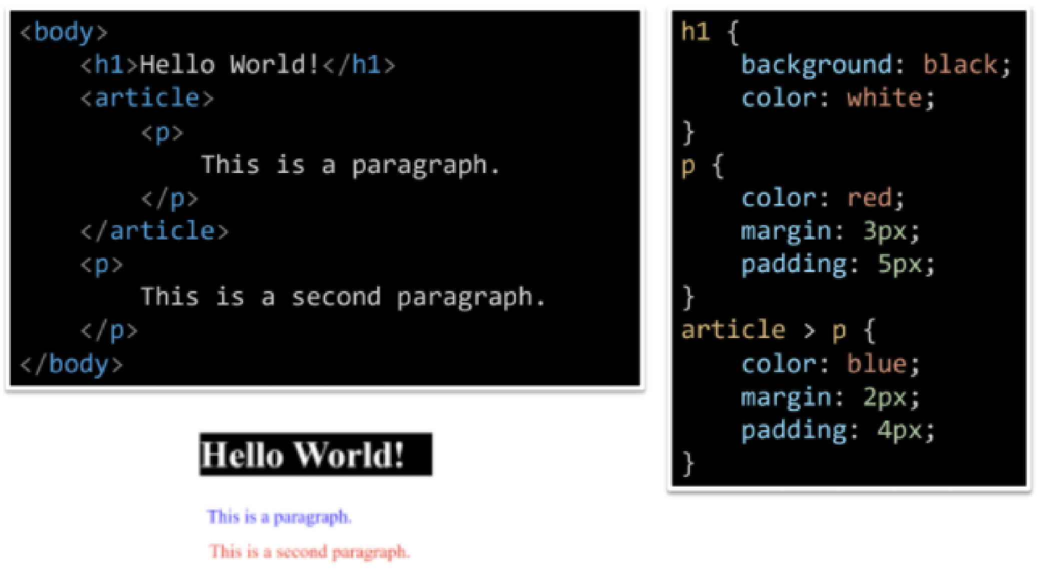
```
article > p {
  color: blue;
}
```

```
<article>...
  <p>...</p>
</article>
...
<p>...</p>
<article>...
  <div><p>...</p></div>
</article>
```

In this definition above, you can see that the P element that is a direct child of an article element will get the styling of color blue applied to it, as opposed to a regular P element that will obviously be unaffected.

Affected, but also a P element that is inside an article element but not a direct child of it, so there's a div element that's contained within the article element and only then is the P element contained within the div. So since the P element is not a direct child of the article in the second case, its font styling is unaffected.

Child selector example



```
<body>
    <h1>Hello World!</h1>
    <article>
        <p>
            This is a paragraph.
        </p>
    </article>
    <p>
        This is a second paragraph.
    </p>
</body>
```

```
h1 {
    background: black;
    color: white;
}
p {
    color: red;
    margin: 3px;
    padding: 5px;
}
article > p {
    color: blue;
    margin: 2px;
    padding: 4px;
}
```

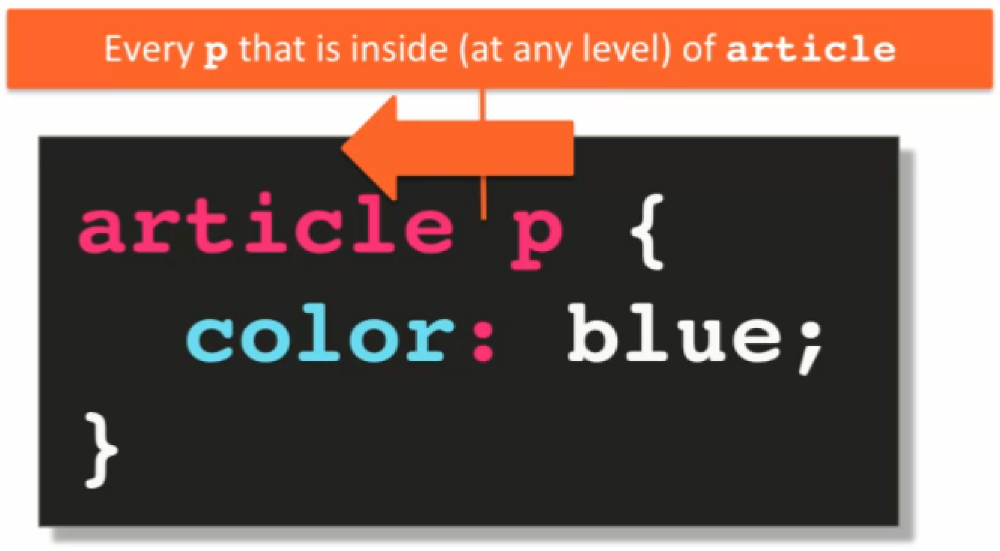
As you can see the p is the child of the article element is affected color: blue

VIII. Descendant Selector

Another type of selector id is a descendant selector.

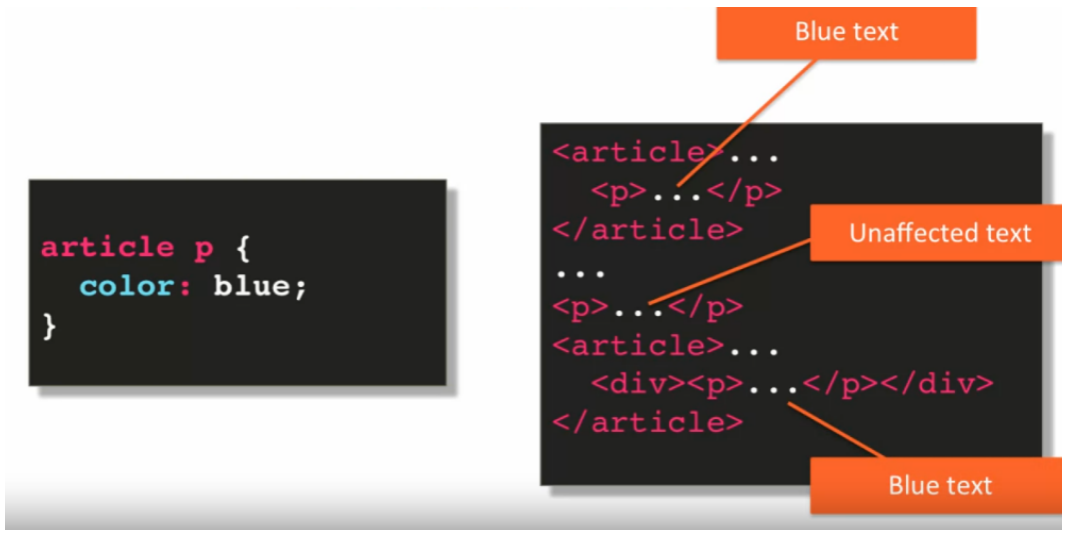
And the syntax of it is, selector space another selector.

So in this case below what we're trying to do is target every P element that is inside at any level. Of an article element, which means that even if it's not a direct child, even if it's deep down, as long as one of its parents in its ancestry is an article, this rule will apply



```
article p {
  color: blue;
}
```

Descendant selector effect

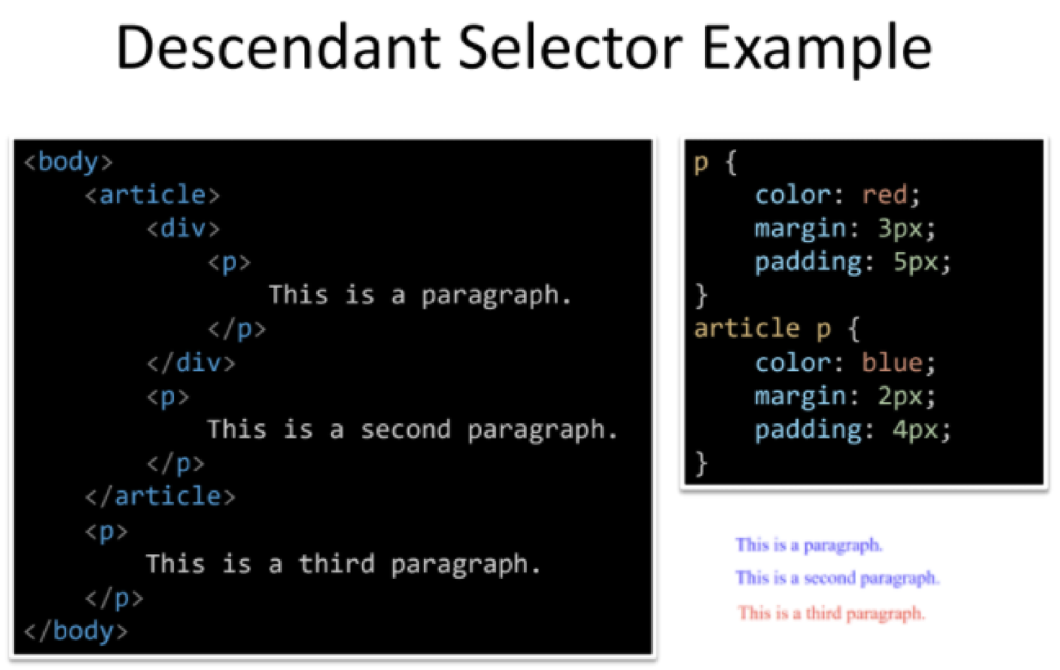


```
article p {
  color: blue;
}
```

```
<article>...
  <p>...</p>
</article>
...
<p>...</p>
<article>...
  <div><p>...</p></div>
</article>
```

In this case above, unlike the direct child that mention before is not affected but the descendant selector, it will still turn blue, that even though it's not a direct child or element. The P element is still technically inside the article even though it's not a direct child and will have its text turn blue.

Descendant selector example:



Descendant Selector Example

```
<body>
    <article>
        <div>
            <p>
                This is a paragraph.
            </p>
        </div>
        <p>
            This is a second paragraph.
        </p>
    </article>
    <p>
        This is a third paragraph.
    </p>
</body>
```

```
p {
    color: red;
    margin: 3px;
    padding: 5px;
}
article p {
    color: blue;
    margin: 2px;
    padding: 4px;
}
```

```
This is a paragraph.
This is a second paragraph.
This is a third paragraph.
```

In this example, any p element at any level inside the article element will be affected

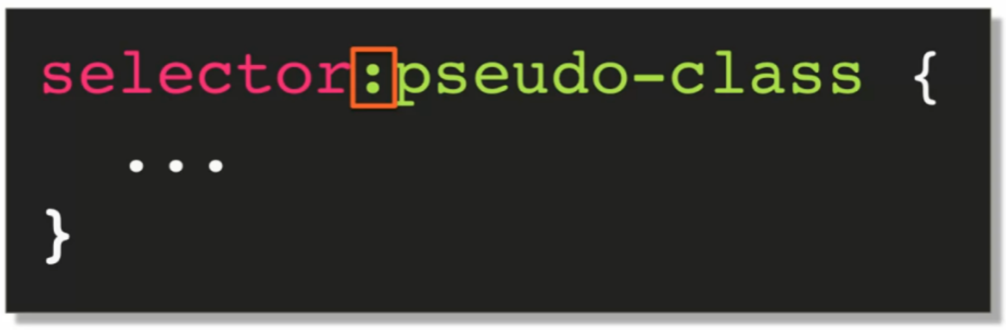
IX. Pseudo-Class Selector

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Pseudo-Class Selector syntax



```
selector:pseudo-class {
    ...
}
```

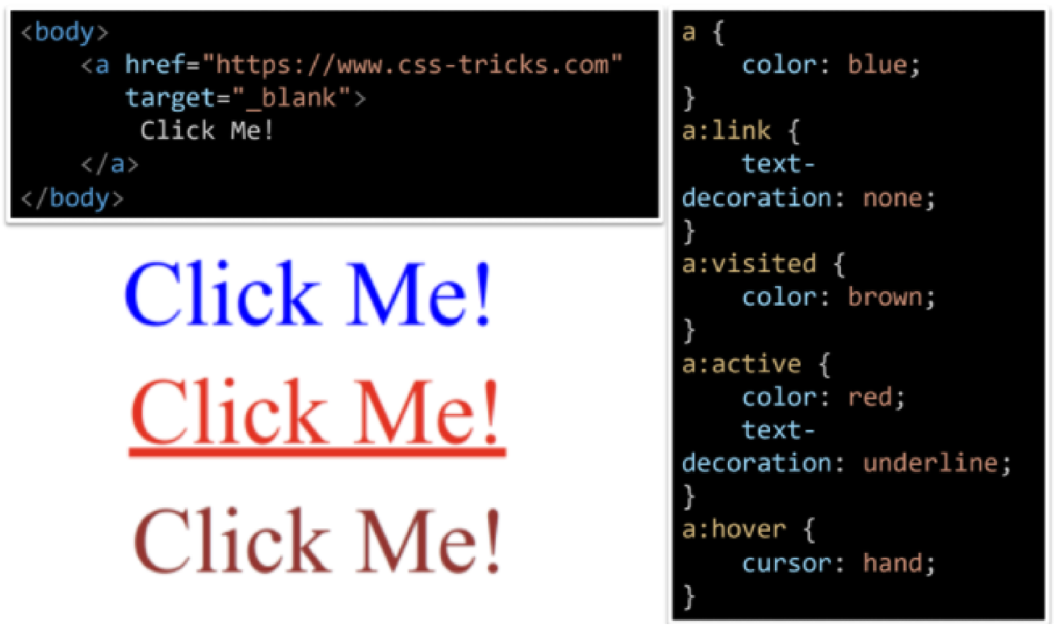
Many pseudo-class selectors exist

- :link represents an element (such as a link) that has not yet been visited.
- :visited

- :hover
- :active represents an element (such as a button) that is being activated by the user.

More pseudo-class selectors in this link: https://www.w3schools.com/css/css_pseudo_classes.asp

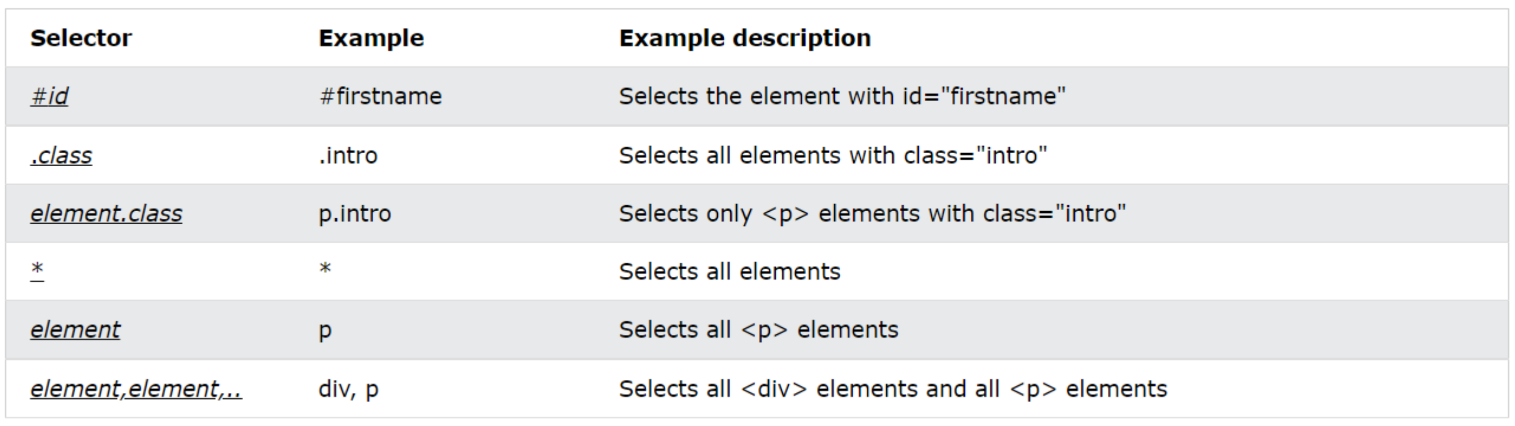
Pseudo-Class Selector example



```
<body>
    <a href="https://www.css-tricks.com"
       target="_blank">
         Click Me!
    </a>
</body>
```

```
a {
    color: blue;
}
a:link {
    text-
decoration: none;
}
a:visited {
    color: brown;
}
a:active {
    color: red;
    text-
decoration: underline;
}
a:hover {
    cursor: hand;
}
```

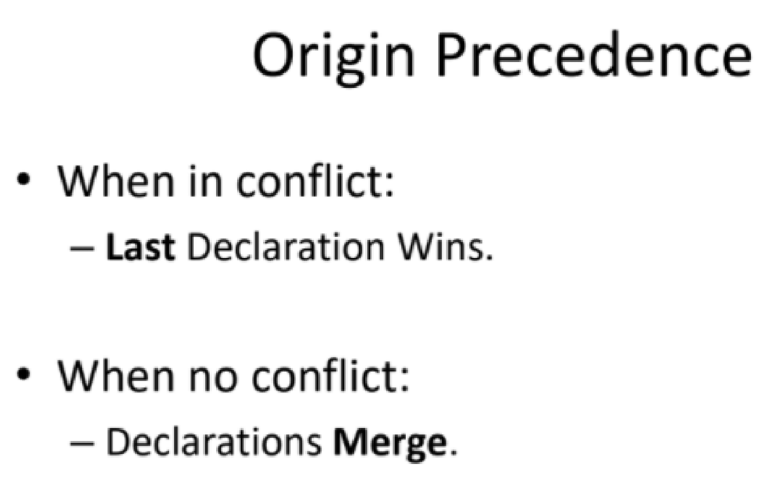
The following below show selectors in CSS



Selector	Example	Example description
#id	#firstname	Selects the element with id="firstname"
.class	.intro	Selects all elements with class="intro"
element.class	p.intro	Selects only <p> elements with class="intro"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,..	div, p	Selects all <div> elements and all <p> elements

X. Origin Precedence

When two declarations are in conflict, in other words they specify the same property for the same target, origin precedence rule kicks in, and it's a very simple rule. And the rule is, the last declaration wins.



Origin Precedence

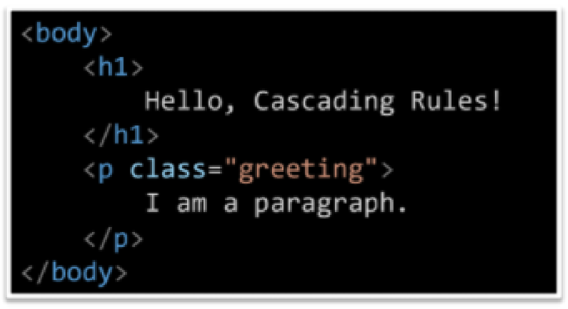
- When in conflict:
  - **Last** Declaration Wins.
- When no conflict:
  - Declarations **Merge**.

- when trying to figure out what the last declaration is, you have to remember that HTML is processed sequentially. That means top to bottom.
- for precedence, think of external CSS as declared at the spot where it's linked to.

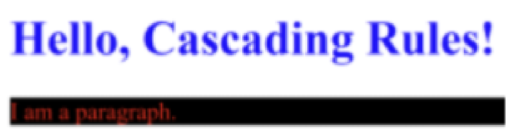
When different CSS declarations do not conflict, that is, they still target the same element, but the CSS properties with which they target that element are different, there's even a simpler rule.

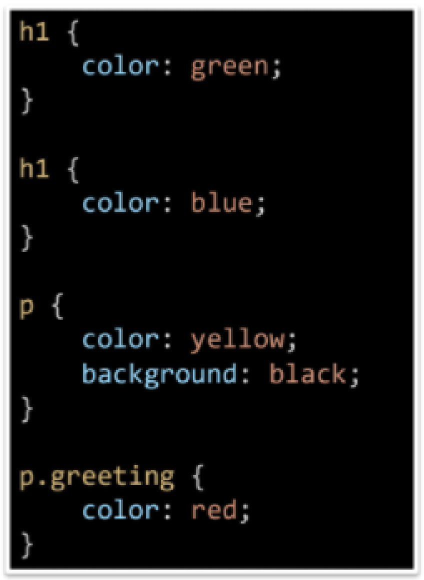
And that is that declarations merge. So a declaration for, for example, font size, and a declaration for color, since they're two different properties, when they're targeted to the same element, even if they're targeted from two different origins, they will merge into one

Example:



```html
<body>
    <h1>
        Hello, Cascading Rules!
    </h1>
    <p class="greeting">
        I am a paragraph.
    </p>
</body>
```



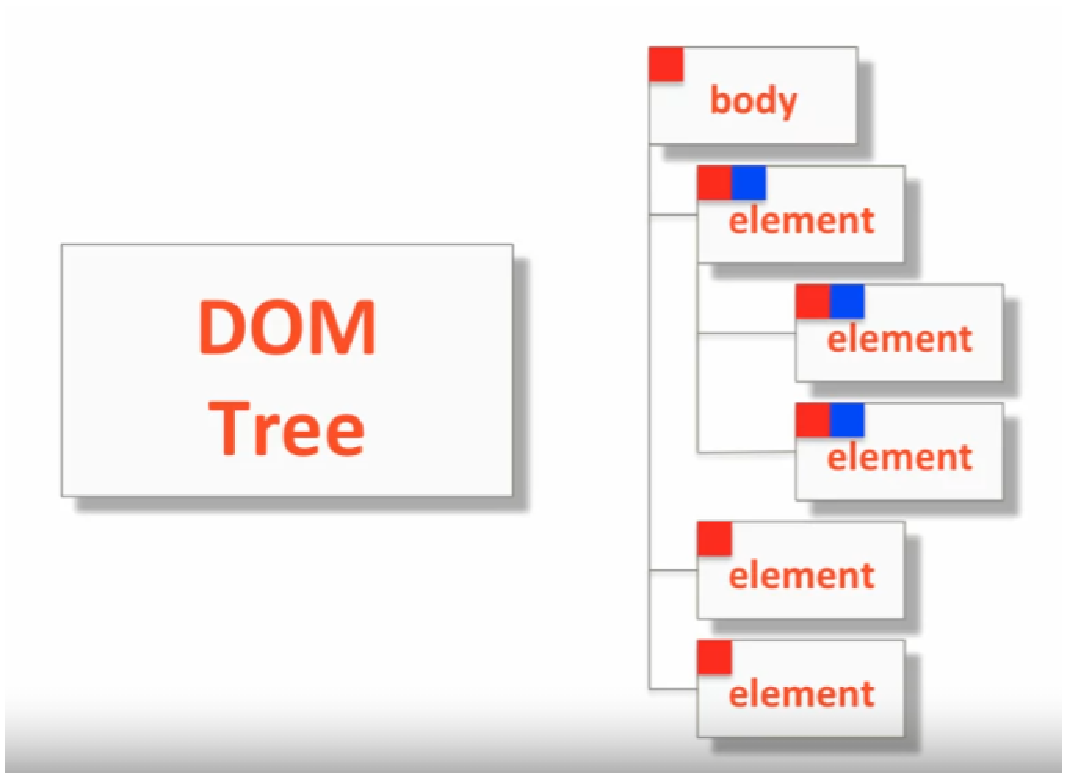


```css
h1 {
    color: green;
}

h1 {
    color: blue;
}

p {
    color: yellow;
    background: black;
}

p.greeting {
    color: red;
}
```

As you can see, when in conflict p and p.greeting then the last declaration win (that is p.greeting form the top to the bottom check)

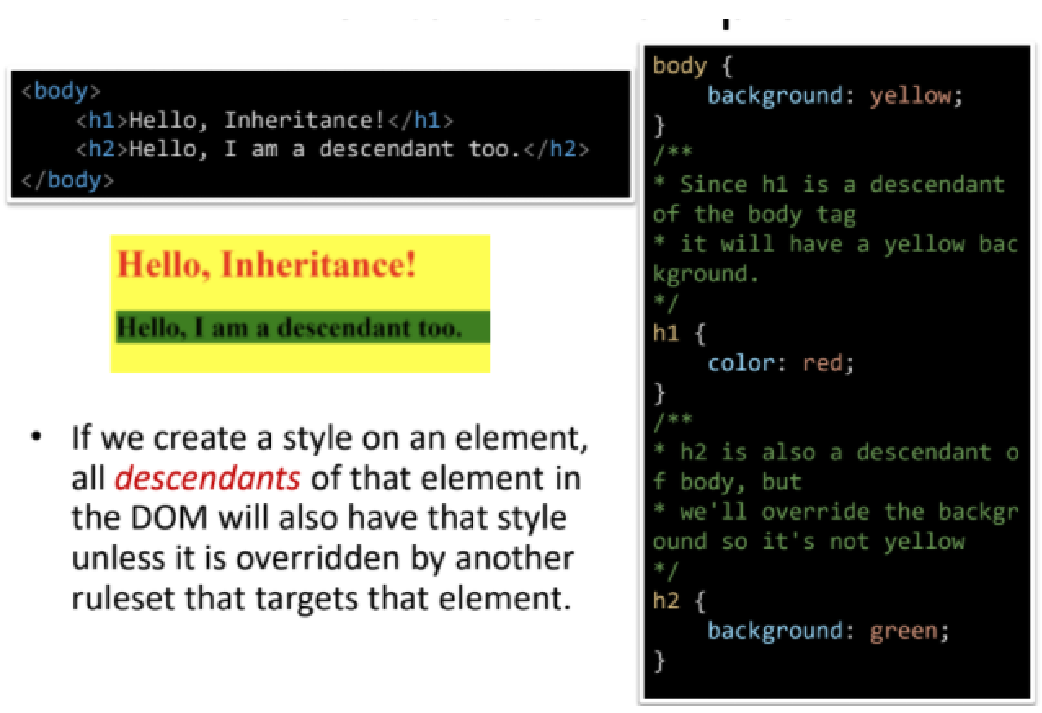
XI. Inheritance

The basic idea of this concept is that you have the document object model tree. And if you specify some CSS property on some element, all the children and grandchildren and so on and so on of that element will also inherit that property without you having to specify the property for each and every element.



For example, if I specify one property on the body tag, every element that is a child of a body tag, or even a child of a child of a body tag and so on, will inherit that property.

Inheritance example:



```
<body>
    <h1>Hello, Inheritance!</h1>
    <h2>Hello, I am a descendant too.</h2>
</body>
```

- If we create a style on an element, all *descendants* of that element in the DOM will also have that style unless it is overridden by another ruleset that targets that element.

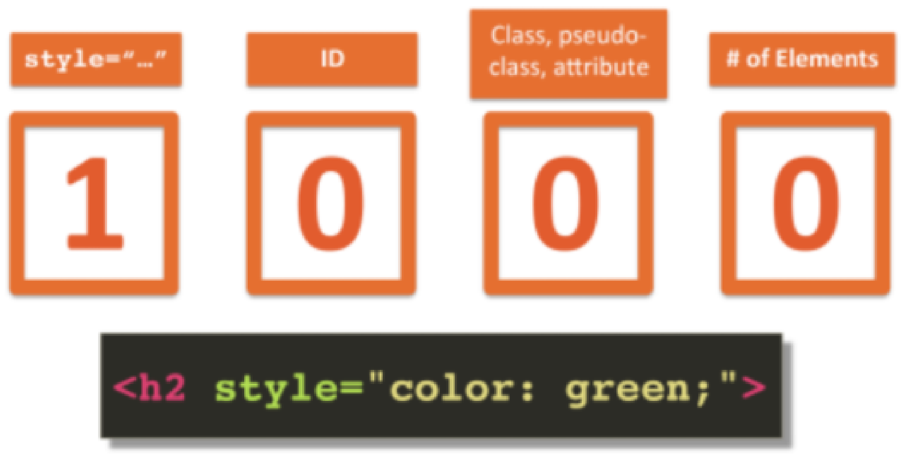
```
body {
    background: yellow;
}
/**
* Since h1 is a descendant
of the body tag
* it will have a yellow bac
kground.
*/
h1 {
    color: red;
}
/**
* h2 is also a descendant o
f body, but
* we'll override the backgr
ound so it's not yellow
*/
h2 {
    background: green;
}
```

XIII. Specificity

The specificity also has a pretty simple rule, and that is most specific selector combination wins.

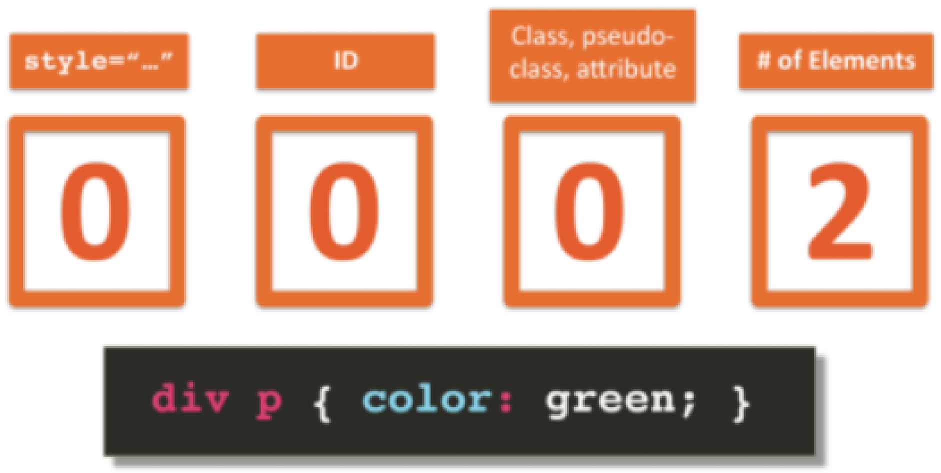
Here is some example to understand how to figure out the score

- The most specific selector combination wins



```
<h2 style="color: green;">
```

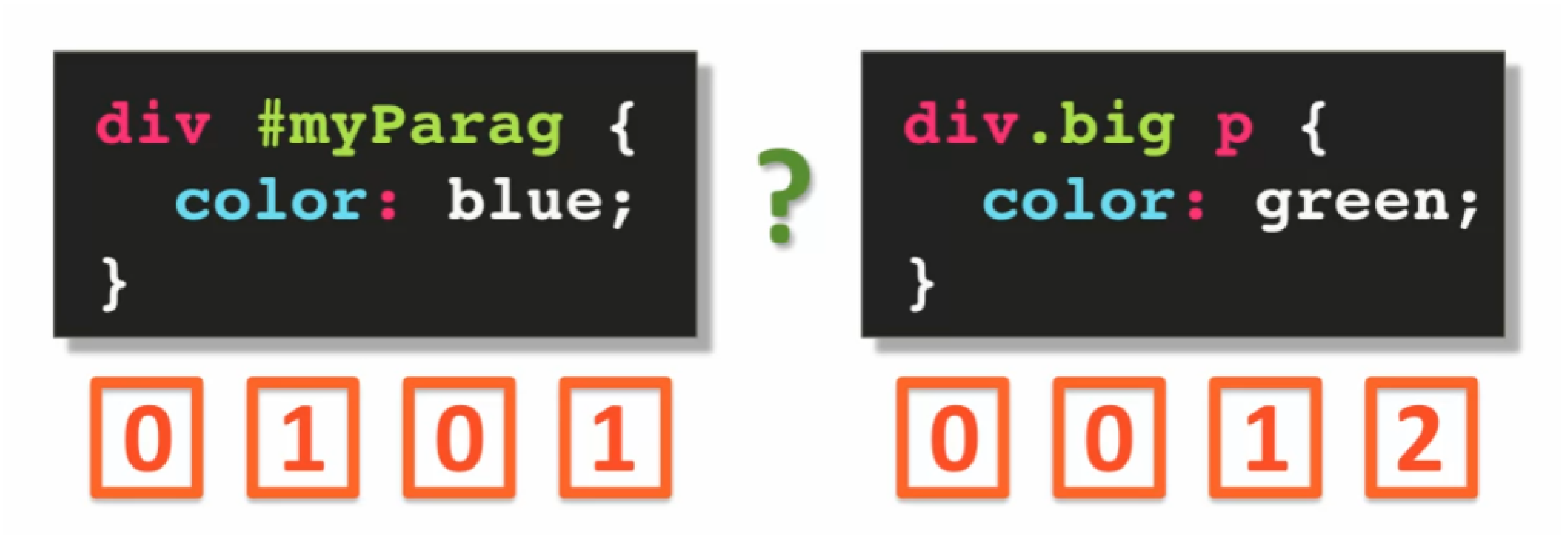
Which says h2 style equals color green we see that we're using the style attribute which means this box gets a one and the rest of them get a zero



```
div p { color: green; }
```

div p and color green, you see that it's not defined inside of a style attribute so that gets a 0. There's no ID selector so that gets a 0. There's no class definition selector either so that gets a 0. But there's two elements, here we're using a descendant combination of selectors, div and p. So, therefore, the number of elements that we have is two.

The following below show which one wins



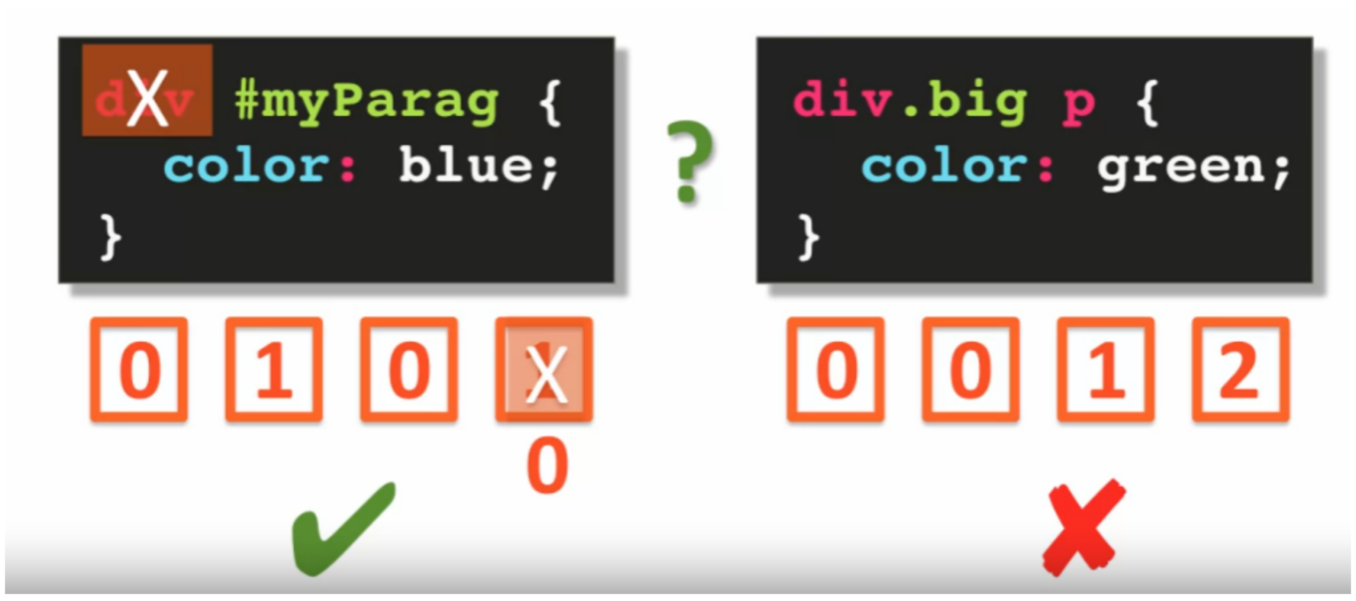
```css
div #myParag {
   color: blue;
}
```

```css
div.big p {
   color: green;
}
```

- The one on the left is not using the style attribute, so you know that gets a 0. It does use an idea attribute, so that gets a 1. There's no class So that's a 0, and the number of elements is a 1, so that gets a 1. So the final score of the left is 101
- On the right, likewise we're not using the style attribute so that gets a 0. We're not using an ID attribute, so that gets a 0 again. We're using one class, so that gets a 1. And we're using two elements so that gets a 2. So the score is, 101 versus 12

▼ Obviously the one on the left wins and the text color of our paragraph will be blue, not green

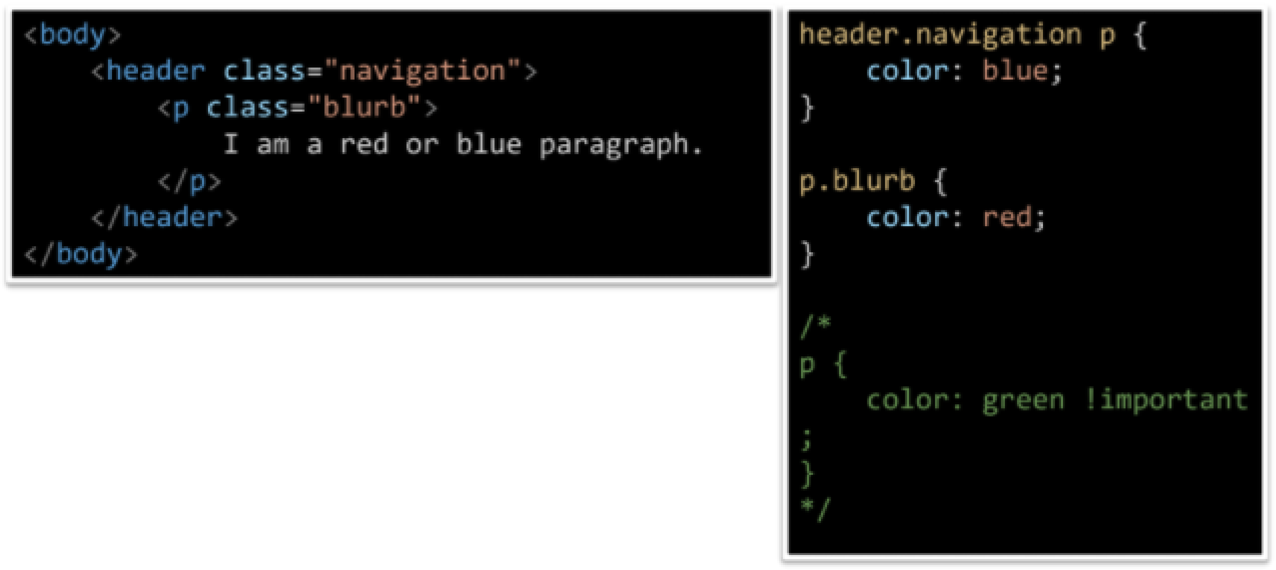
And if we not specifying the div there to begin with. We could have just expressed it with an id selector and still would have won this battle and therefore the color of the text to that paragraph would have still been blue.



```css
dXv #myParag {
   color: blue;
}
```

```css
div.big p {
   color: green;
}
```

Specificity example:



```html
<body>
    <header class="navigation">
        <p class="blurb">
            I am a red or blue paragraph.
        </p>
    </header>
</body>
```

```css
header.navigation p {
    color: blue;
}

p.blurb {
    color: red;
}

/*
p {
    color: green !important
;
}
*/
```

As you can see the selector header.navigation p not using style attribute, not using id, using 1 class and 2 element so the score is 12. The selector p.blurb not using style attribute and id as well, using 1 class and 1 element so the score is 11. Then the first selector wins the battle → the paragraph must blue color

But if you want to force even the score at any selector which is very high by using !important to force it render

XIV. CSS Font-Size

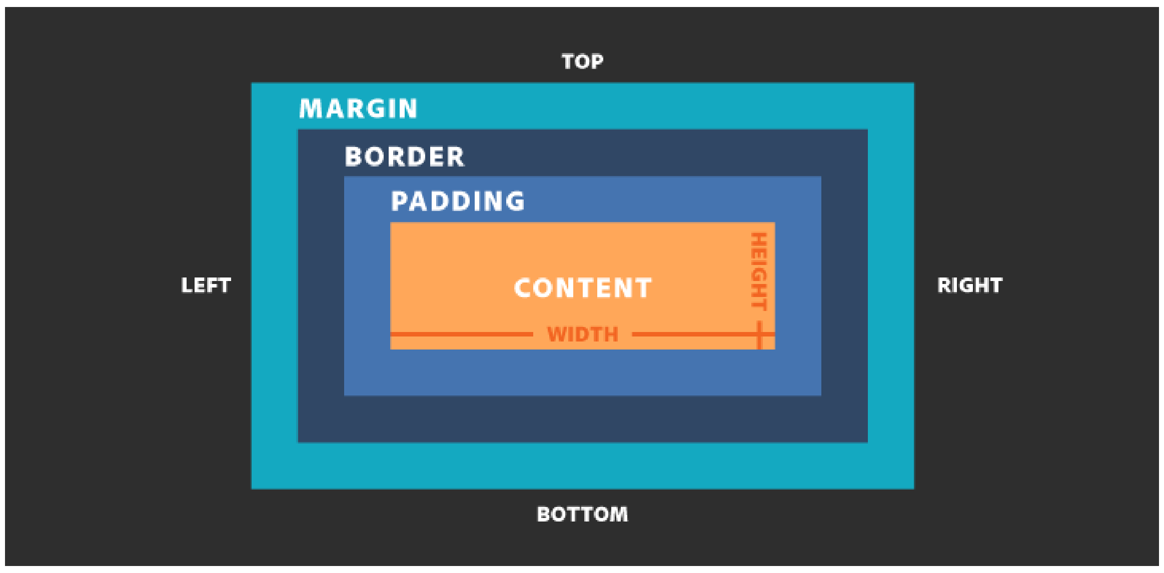
- Pixels (px): Pixels are fixed-size units that are used in screen media (i.e. to be read on the computer screen). One pixel is equal to one dot on the computer screen (the smallest division of your screen's resolution).
- Points (pt): Points are traditionally used in print media (anything that is to be printed on paper, etc.). One point is equal to 1/72 of an inch. Points are much like pixels, in that they are fixed-size units and cannot scale in size.
- "Ems" (em): The "em" is a scalable unit that is used in web document media. An em is equal to the current font-size, for instance, if the font-size of the document is 12pt, 1em is equal to 12pt.
- Percent (%): The percent unit is much like the "em" unit, save for a few fundamental differences. First and foremost, the current font-size is equal to 100% (i.e. 12pt = 100%).
- Generally, 1em = 12pt = 16px = 100%.

▼ To simplify use Pixels and Point for fixed-size unit, and use em, Percent for scalable unit. It's very helpful for responsive web page

XV. CSS box model

In CSS, the term "box model" is used when talking about design and layout.

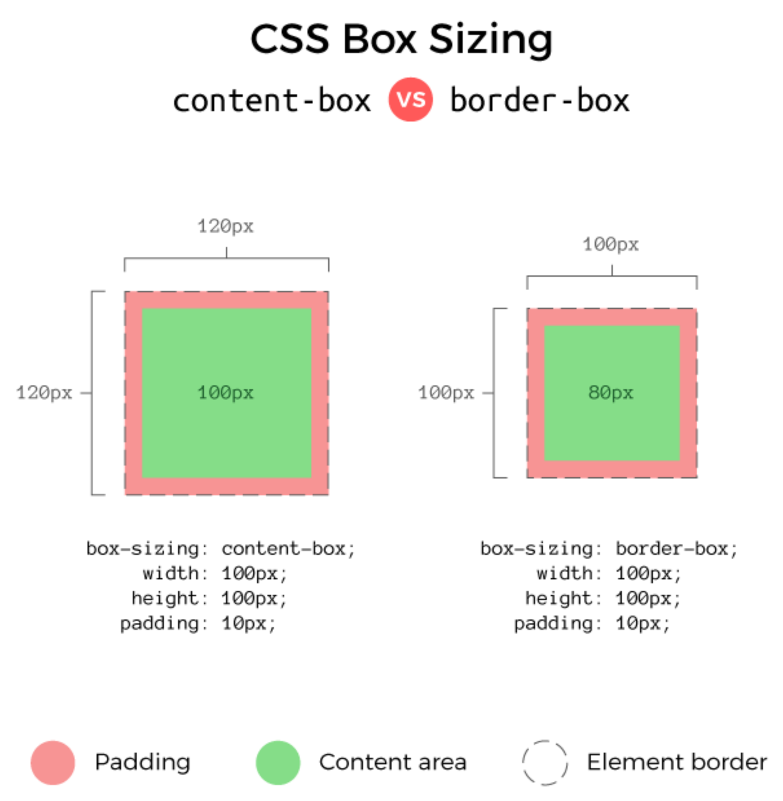
The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



▼ Margins do not define the width of the box, just define how far other elements should be pushed away from it.

Content box with Border box

- Content-box: The width & height of the element only include the content. In other words, the border, padding and margin aren't part of the width or height. This is the default value.
- Border-box: The padding and border are included in the width and height.

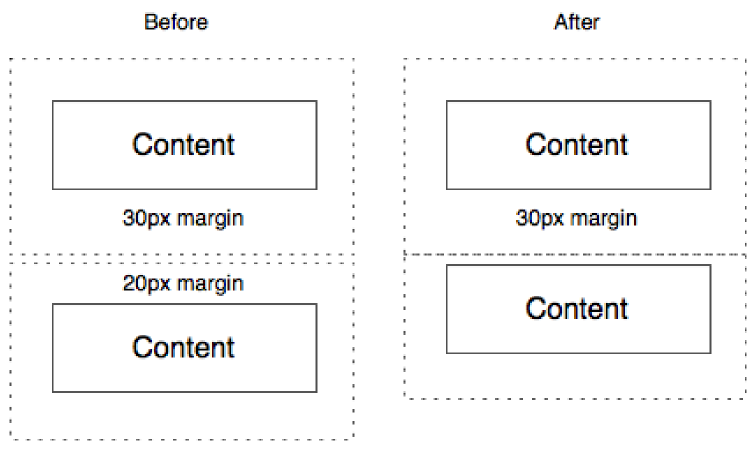


```
box-sizing: content-box;
     width: 100px;
    height: 100px;
   padding: 10px;
```

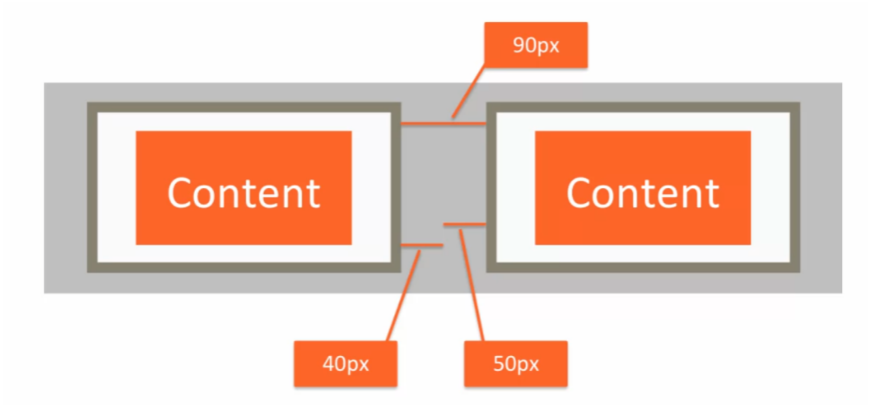
```
box-sizing: border-box;
     width: 100px;
    height: 100px;
   padding: 10px;
```

Collapsing Margins

- Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

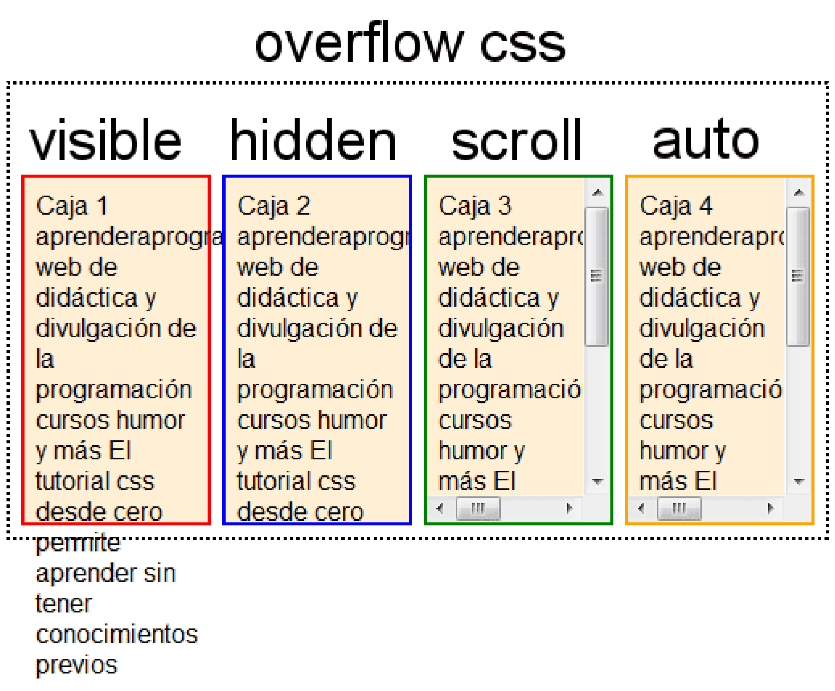


- This does not happen on left and right margins. It's combine together



The overflow property

- The `overflow` property specifies what should happen if content overflows an element's box.
- This property specifies whether to clip content or to add scrollbars when an element's content is too big to fit in a specified area.


overflow css
visible
hidden
scroll
auto
Caja 1 aprenderaprogra web de didáctica y divulgación de la programación cursos humor y más El tutorial css desde cero permite aprender sin tener conocimientos previos
Caja 2 aprenderaprogr web de didáctica y divulgación de la programación cursos humor y más El tutorial css desde cero
Caja 3 aprenderapro web de didáctica y divulgación de la programació cursos humor y más El
Caja 4 aprenderapro web de didáctica y divulgación de la programació cursos humor y más El

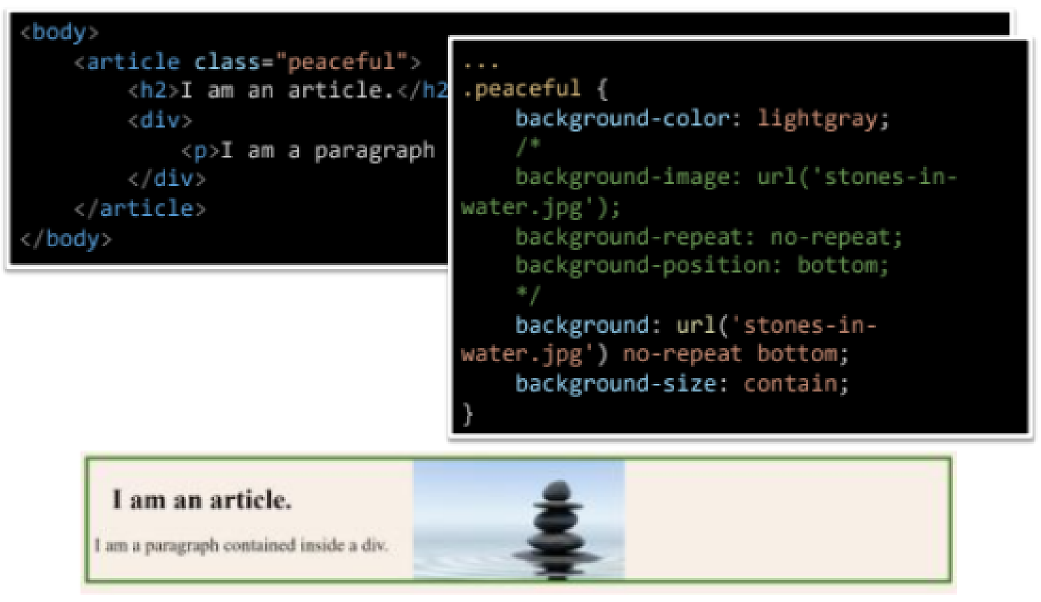
> The overflow property only works for block elements with a specified height.

The background property

The `background` property is a shorthand property for:

- background-color
- background-image
- background-position
- background-size
- background-repeat
- background-origin
- background-clip
- background-attachment

example below:



```
<body>
    <article class="peaceful">
        <h2>I am an article.</h2
        <div>
            <p>I am a paragraph
        </div>
    </article>
</body>
```

```
...
.peaceful {
    background-color: lightgray;
    /*
    background-image: url('stones-in-
water.jpg');
    background-repeat: no-repeat;
    background-position: bottom;
    */
    background: url('stones-in-
water.jpg') no-repeat bottom;
    background-size: contain;
}
```

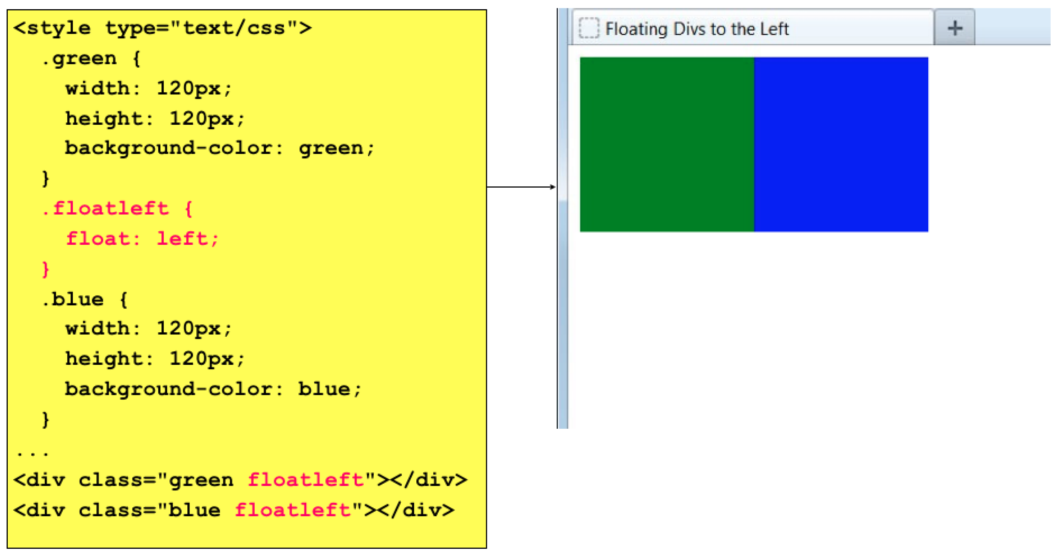
The Floating

The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container
- `none` - The element does not float (will be displayed just where it occurs in the text). This is default

- `inherit` - The element inherits the float value of its parent



```
<style type="text/css">
  .green {
    width: 120px;
    height: 120px;
    background-color: green;
  }
  .floatleft {
    float: left;
  }
  .blue {
    width: 120px;
    height: 120px;
    background-color: blue;
  }
...
<div class="green floatleft"></div>
<div class="blue floatleft"></div>
```

Position

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

- `static`: Static positioned elements are not affected by the top, bottom, left, and right properties.

  An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

- `relative`: An element with `position: relative;` is positioned relative to its normal position.

  Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

- `fixed`: An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

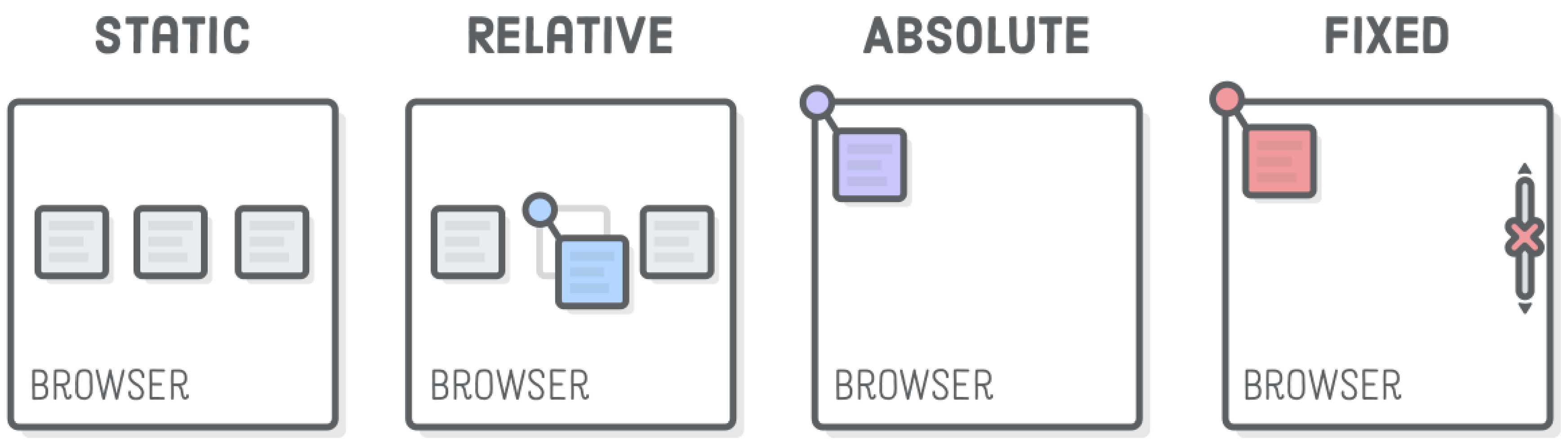
  A fixed element does not leave a gap in the page where it would normally have been located.

- `absolute`: An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

  However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

- `sticky`: An element with `position: sticky;` is positioned based on the user's scroll position.

  A sticky element toggles between `relative` and `fixed`, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).



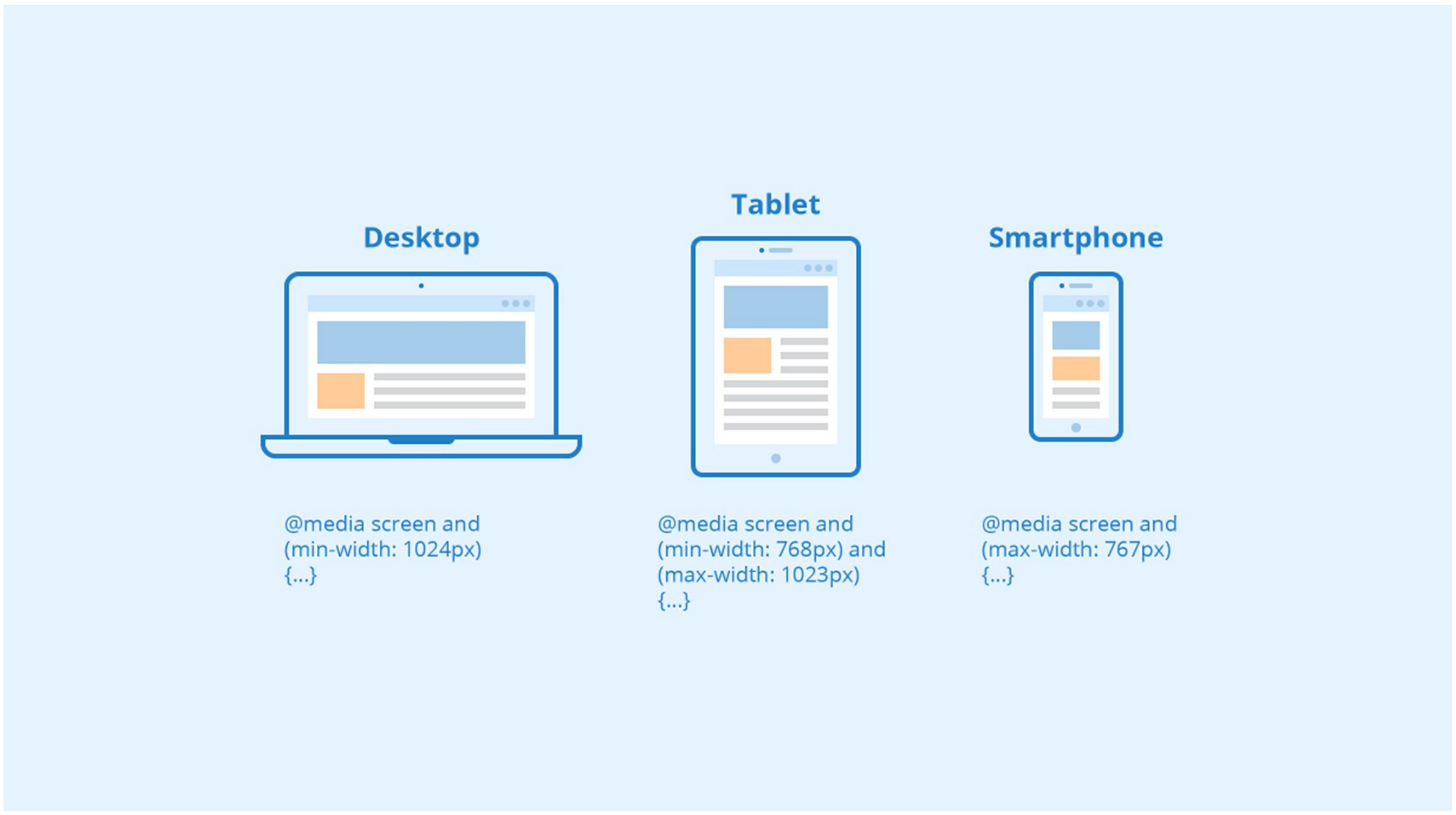
Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

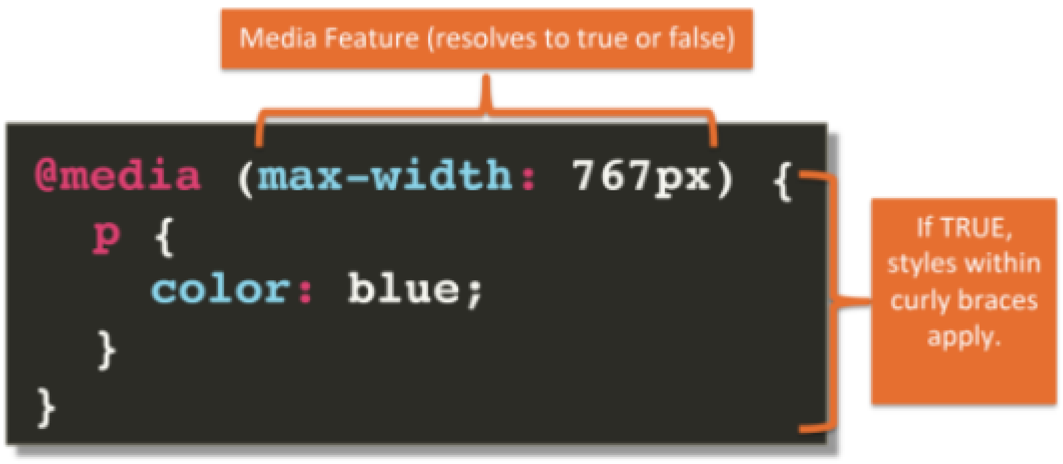
Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).


Desktop
Tablet
Smartphone
@media screen and (min-width: 1024px) {...}
@media screen and (min-width: 768px) and (max-width: 1023px) {...}
@media screen and (max-width: 767px) {...}

Media query syntax:

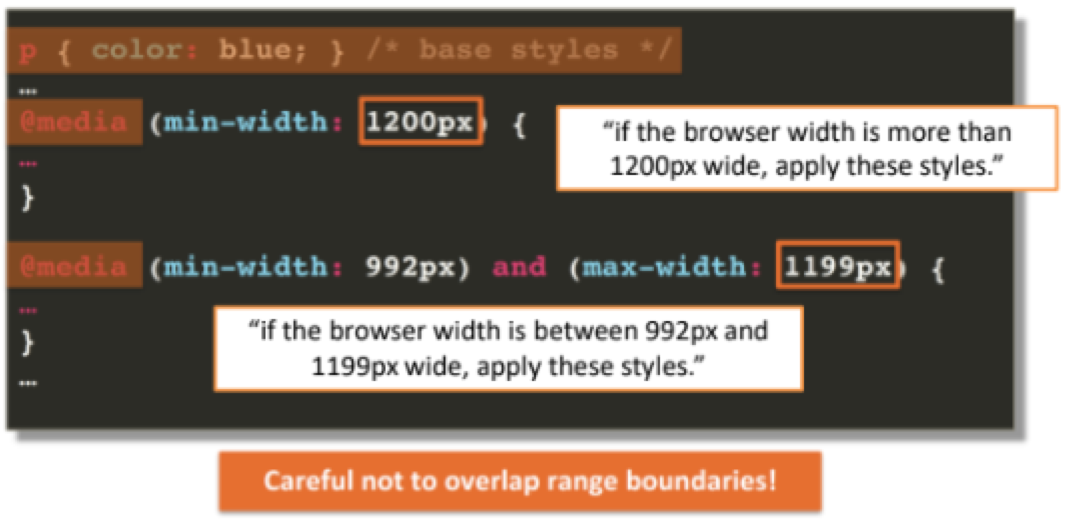


```
@media (max-width: 767px) {
   p {
     color: blue;
   }
}
```

Media Feature (resolves to true or false)
If TRUE, styles within curly braces apply.

Example:

Example

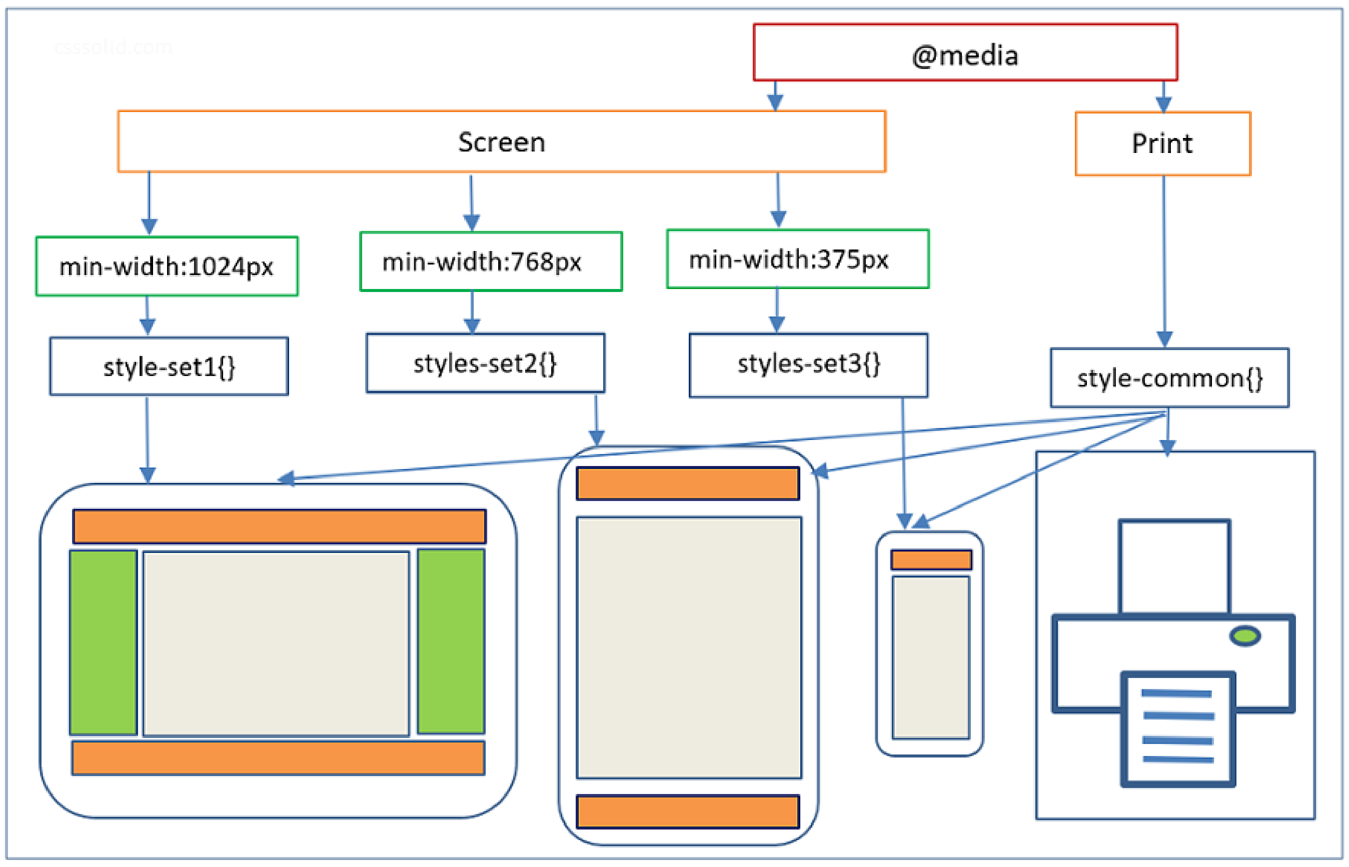


```
p { color: blue; } /* base styles */
...
@media (min-width: 1200px) {
...
}
@media (min-width: 992px) and (max-width: 1199px) {
...
}
...
```

"if the browser width is more than 1200px wide, apply these styles."

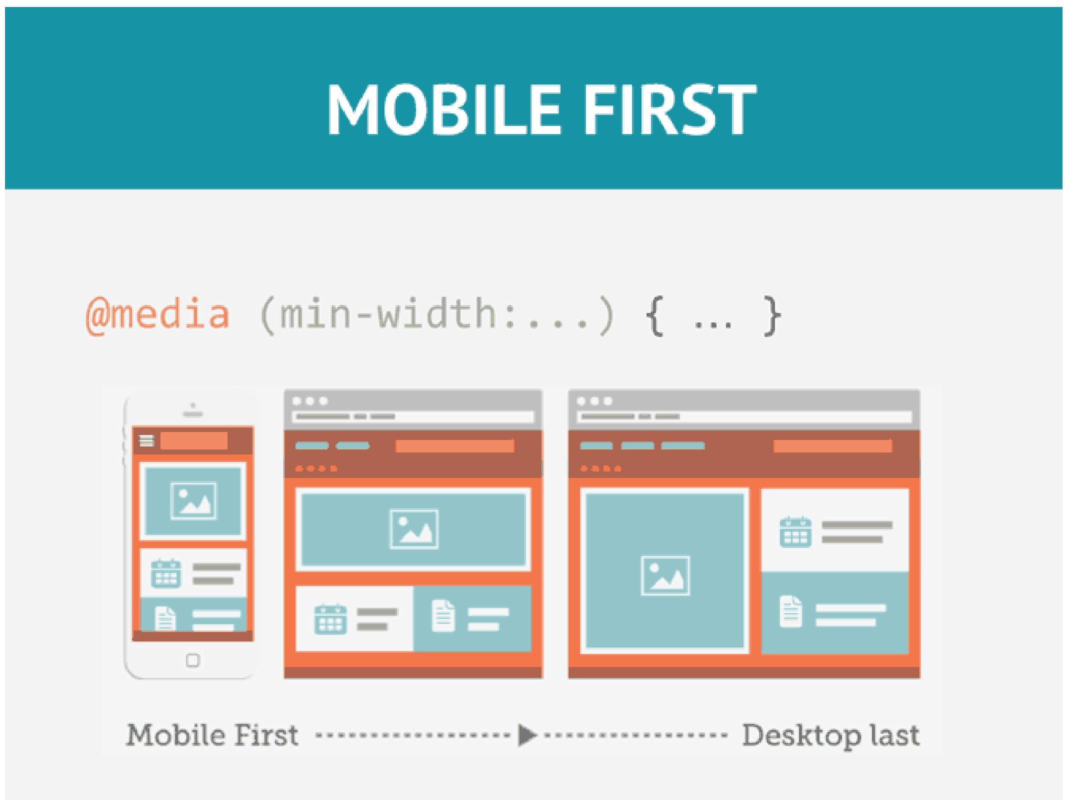
"if the browser width is between 992px and 1199px wide, apply these styles."

Careful not to overlap range boundaries!



Responsive Web Design

Responsive Web Design is the methodology that recommends the design and implementation of a website that responds to user behavior and environment based on the screen size, orientation and operating system of their device.

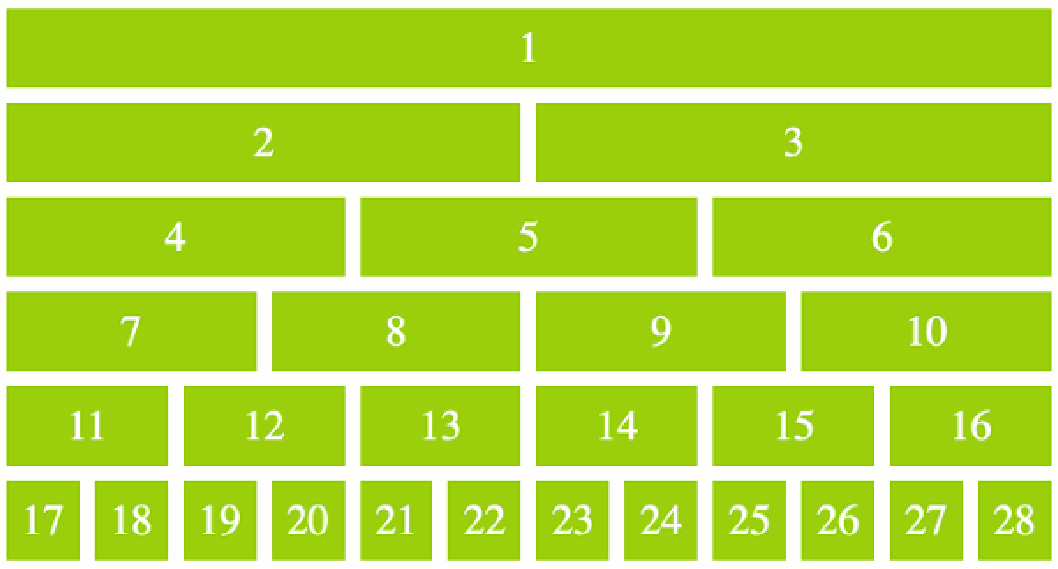


Technologies for Responsive Web Design

- From a technology standpoint, the framework consists of a combination of flexible grids, flexible layouts, images and intelligent use of CSS media queries.
- Site's layout adapts to the size of the device.
- Content verbosity or its visual delivery may change.

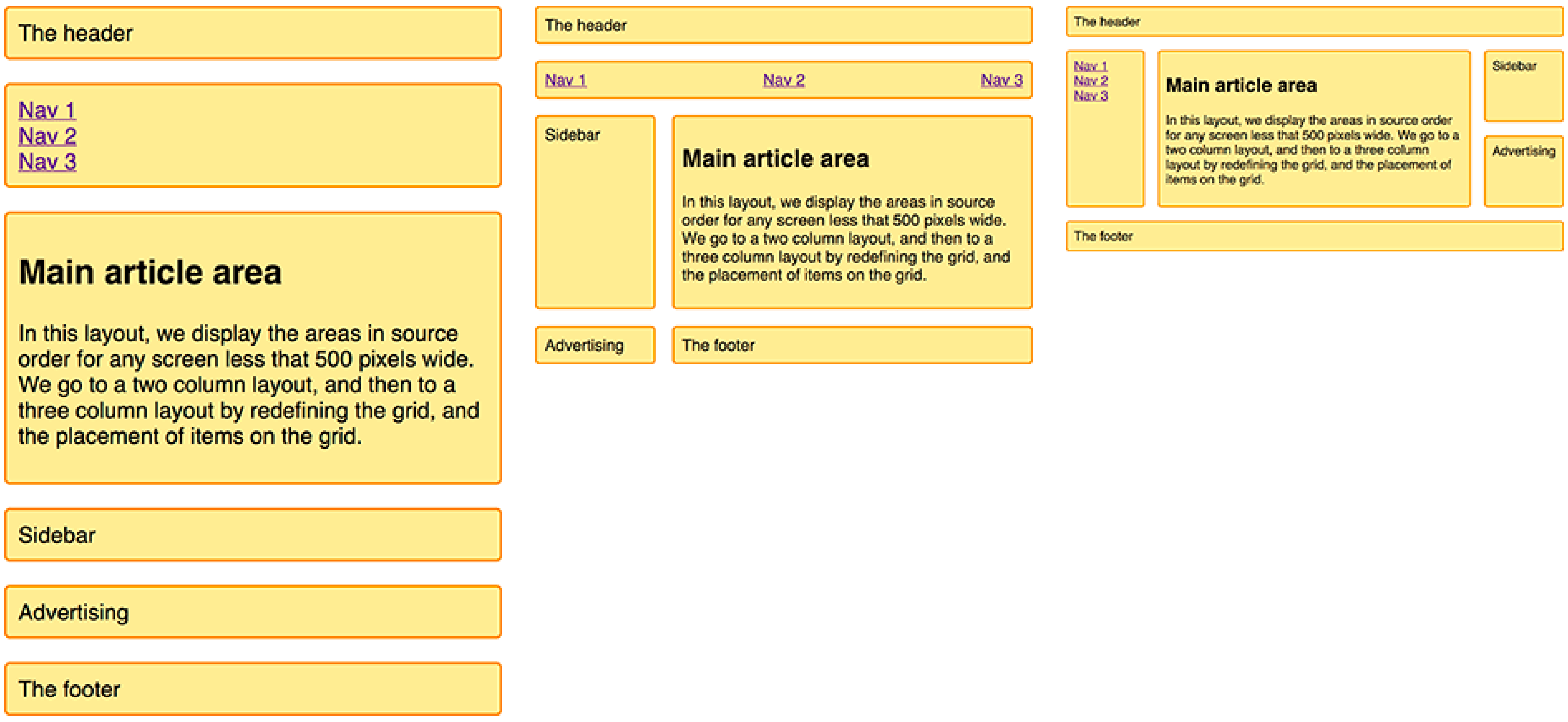
12 - Column Grid Responsive Design

Many web pages are based on a grid-view, which means that the page is divided into columns:



Using a grid-view is very helpful when designing web pages. It makes it easier to place elements on the page.

A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.


The header

Nav 1
Nav 2
Nav 3

Main article area

In this layout, we display the areas in source order for any screen less that 500 pixels wide. We go to a two column layout, and then to a three column layout by redefining the grid, and the placement of items on the grid.

Sidebar

Advertising

The footer

The header

Nav 1 Nav 2 Nav 3

Sidebar

Main article area

In this layout, we display the areas in source order for any screen less that 500 pixels wide. We go to a two column layout, and then to a three column layout by redefining the grid, and the placement of items on the grid.

Advertising

The footer

The header

Nav 1
Nav 2
Nav 3

Main article area

In this layout, we display the areas in source order for any screen less that 500 pixels wide. We go to a two column layout, and then to a three column layout by redefining the grid, and the placement of items on the grid.

Sidebar

Advertising

The footer

Viewport meta tag

HTML5 introduced a method to let web designers take control over the viewport, through the `<meta>` tag.

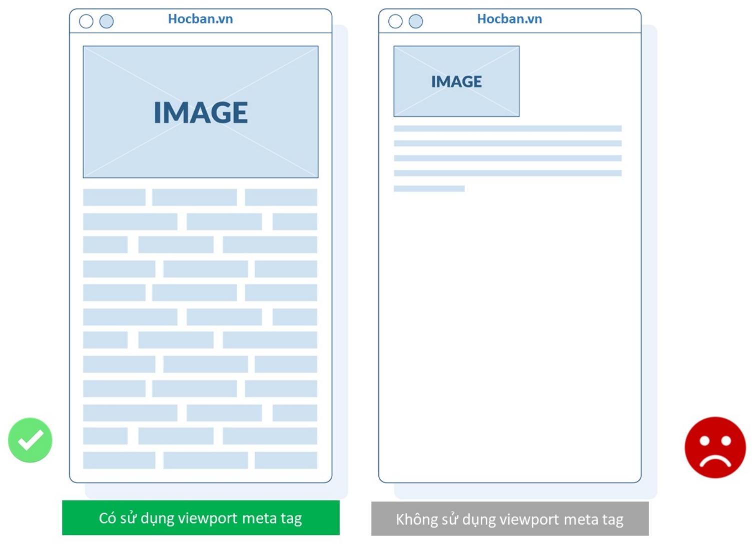
You should include the following `<meta>` viewport element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This gives the browser instructions on how to control the page's dimensions and scaling.

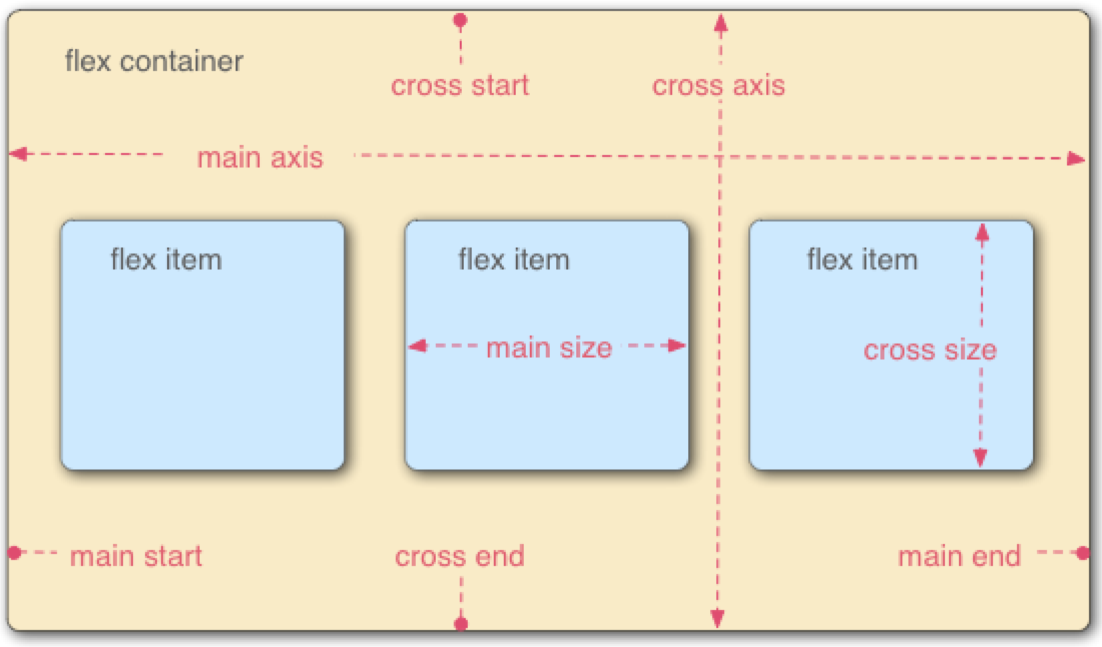
The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The `initial-scale=1.0` part sets the initial zoom level when the page is first loaded by the browser.



Flex

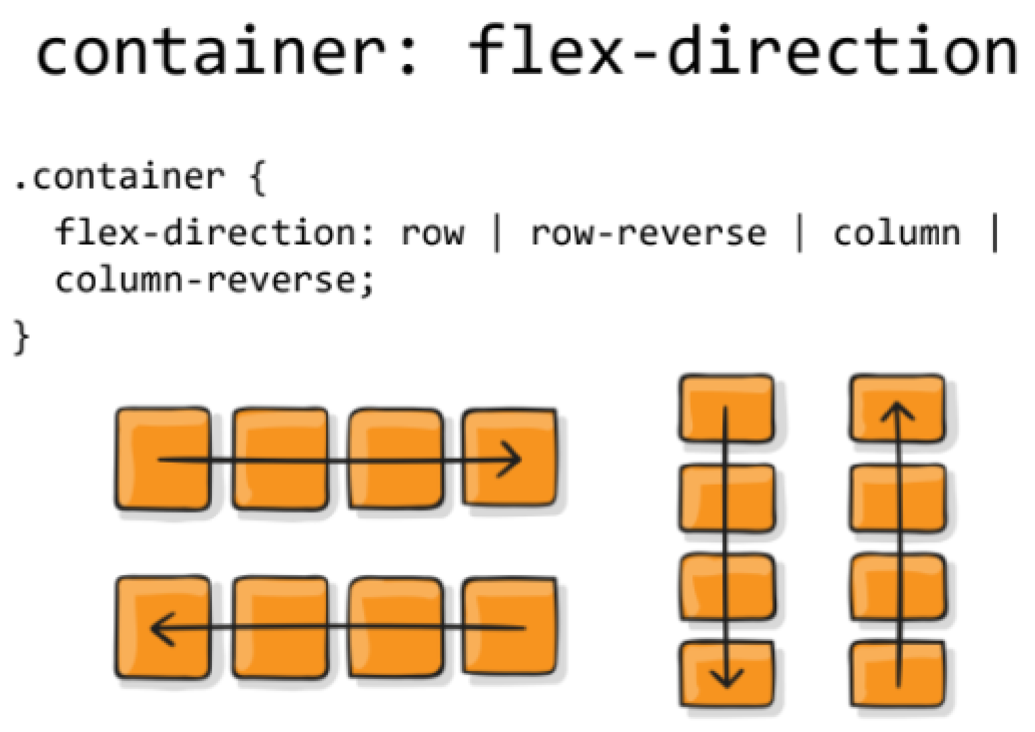
In flexbox, there are mainly two main components: the parent container (flex container) and the inner elements (flex items)



Some of the properties of the Container:

- display
- flex-direction
- flex-wrap
- flex-flow
- justified-content
- align-items
- align-content

Using flex-direction:



container: flex-direction

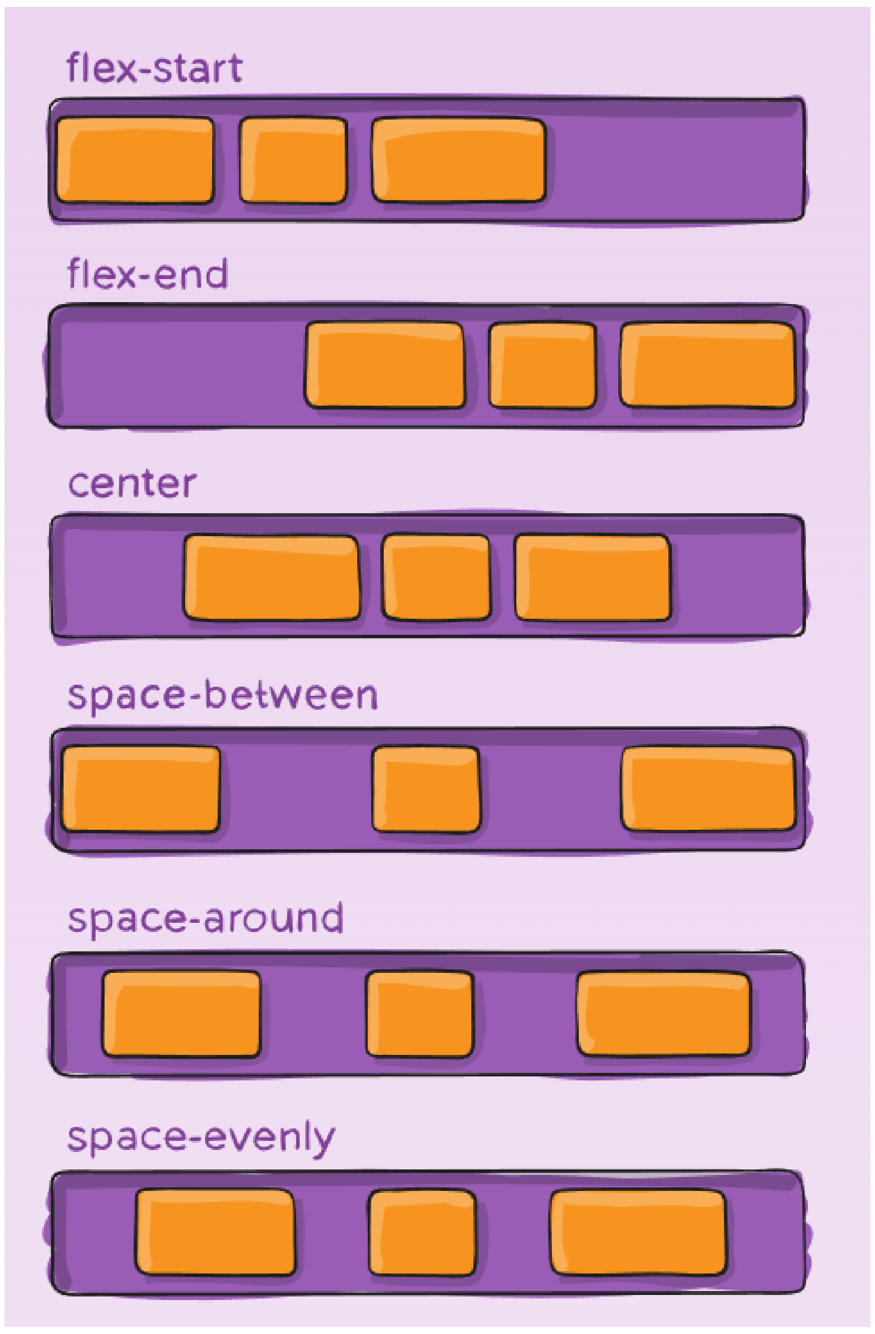
```
.container {
  flex-direction: row | row-reverse | column |
  column-reverse;
}
```

if flex-direction is row then all the item in container lay on a row, otherwise if using row-reverse then all item in container lay on a row but reverse position. If flex-direction is column so all the items on container lay on a column

justified-content is used to align the position of items relative to the main axis. There are six possible values for the justification-content attribute:

- flex-start: will set the item to start from main start (and this is also the default value)
- flex-end: will place item starting from main end
- center: will place all items in the center of the main axis
- space-between: will divide the extra space evenly and add it between items
- space-around: will split the spacing at the beginning and the end. The distance at the beginning and the end will be half of the distance between the two items together

- space-evenly: will evenly divide the distance between items and items, item and main start, item and main end are equal

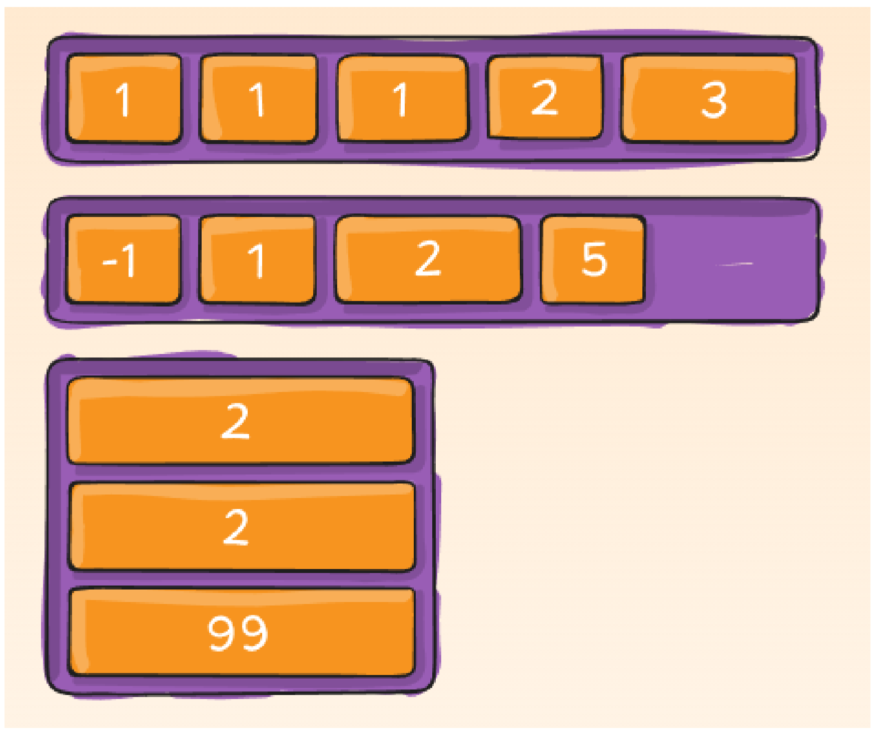


Order

By default, flex items are laid out in the source order. However, the `order` property controls the order in which they appear in the flex container.

```
.item {
  order: 5;/* default is 0 */}
```

Items with the same `order` revert to source order.



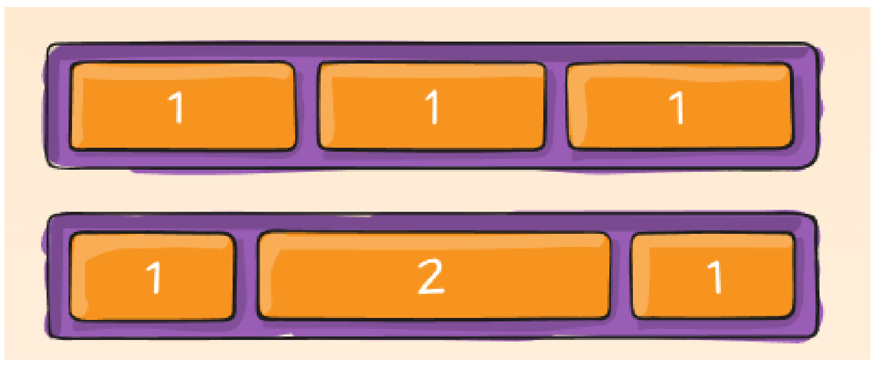
flex-grow

This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

If all items have `flex-grow` set to 1, the remaining space in the container will be distributed equally to all children. If one of the children has a value of 2, the remaining space would take up twice as much space as the others (or it will try to, at least).

```
.item {
  flex-grow: 4;/* default 0 */}
```

> Negative numbers are invalid.



Go to this link to learn more flex-box properties: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Recommend book:

1. ***Jon Duckett (2014). HTML & CSS - Design and Build Websites. John Wiley & Sons.***
2. ***Semmy Purewal (2014). Learning Web App Development. O'Reilly Media***