SIT12- Introduction to Responsive Web Apps
Week 1 – Reflection

- Who the tutor are and the systems you need to use for this course
- Upload weekly material in GitHub to share on Week 7 with tutor
- Review the systems and languages you need for this subject, VueJS, Visual Studio, HTML, CSS, JavaScript
- Suggested online platforms provided.
- Top Tip: Where to go to validate errors in your web page

Shiva Pokhrel Unit chair details Campus: Melbourne Burwood 221 Burwood Highway BURWOOD VIC 3125 Email: shiva.pokhrel@deakin.edu.au Phone: +61 3 924 46281

Scheduled learning activities - cloud 1 x 1 hour scheduled online class/workshop per week

https://www.developerdrive.com/using-custom-attributes-in-html5/:~:text=Using%20Custom%20Attributes%20in%20HTML5%201%20Create%20Your,5%20Get%20the%20Dataset.%20...%206%20Conclusion.

Project APP - what to do in lock down ideas - rabbit hutch, cooking etc what tools I have solution
What area I have flat herb garden - groups areas

EACH WEEK REFLECTIONS AND CODE UNTIL WEEK 7 (UPLOAD FOLDER TO GIT)

System used : VUE

Week	Commencing	Topic	Assessment activity
1	12 July 2021	HTML, CSS and Responsive Web Apps, UX/UI design	
2	19 July	Networking Infrastructure and Protocols for Web	
3	26 July	CSS Functions and Capabilities	
4	2 August	Java Script Functions	Project 1 (First Version) Preliminary Project plan, analysis and design
5#	9 August	Vue Framework: Conditions and Loops	
6	23 August	Handling User Inputs	Project 1 (Improved Version)
7	30 August	Composing with Components	
8	6 September	Web Development Frameworks	Practical Portfolio
9	13 September	Advanced Programming 1	
10^	20 September	Advanced Programming 2	Project 2 Implementation and demonstration
11	27 September	Advanced Vue Features	Project 2 Implementation and demonstration

#Intra-trimester break: **Monday 16 August - Sunday 22 August 2021** (between weeks 5 and 6)
^ AFL Grand Final Eve (University closed) - **Friday 24 September** (date to be confirmed)

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 elements and all <p> elements

Web Process - Simple terms

HTML: Hypertext Markup Language (define the building blocks)
CSS: Cascading Style Sheets (make it visually appealing and consistent)
Javascript: Programing Lanuage - Functionality (make it interactive - hit like)

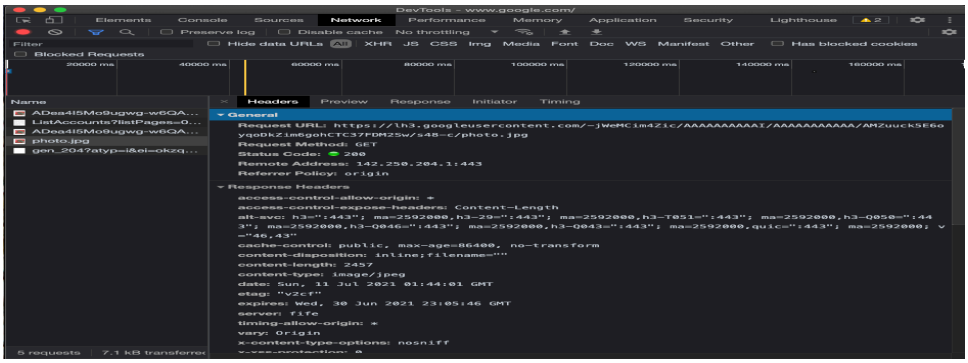
Frameworks: VUE, Angular and React (react is most popular)

Version Control System: Git

URL: Uniform Resource Locator
Browser (client) > Server (browser sends message to server) REQUEST & RESPONSE
HTTP: Hypertext Transfer Protocol (language) / HTTPS is with encryption (secure)
<!DOCTYPE html>
<html>
…
</html>

DOM: Document Object Model
Render: display

Chrome: View> Developer > DevTools



Visual Studio TOP TIPS:

Plugins: Live Server and Prettier

Save index.html
type ! and tab and your basic HTML skeleton will appear :) WOW

Non breaking space

< a href="images/example.jp" download>My Photo

Jump to a section
CSS

Jump to top of page
< a href="#">Back to Top

Notice:

- Comments can be embedded within a **<style>** section using the **/* */** start and end specifiers.
- Comments can be embedded within your **html document** using the **<!-- -->** start and end specifiers.

Changing the color of links

The LINK, VLINK and ALINK attributes can be inserted in the <BODY> tag to define the color of a link
- LINK defines the color of links that have not been visited
- VLINK defines the color of links that have already been visited
- ALINK defines the color of a link when a user clicks on it

To create a comment statement use the <!-- .... --> tags.

Don't forget target="_blank" to open links in new window

resize image
object-fit: cover;

Check errors in your web page

https://validator.w3.org/

ASCII - FIRST character set created (limited)
UTF-8 - represents almost all characters

What do you think are the advantages of externalizing your styles into a separate CSS file that can be linked into to HTML files ?

It takes less time to do updates to all your web pages if you use external styles and its easier to keep every page consistent, otherwise you might make a change but forget to do that same change on every page (change font of all your headers but one Is not)

Selectors re-watch when need to refresh learning:
Source: https://www.youtube.com/watch?v=l1mER1bV0N0

li:hover selecter highlights when user hovers over the line
input:required, hovered and focused good for forms and check box
2n select every second element etc

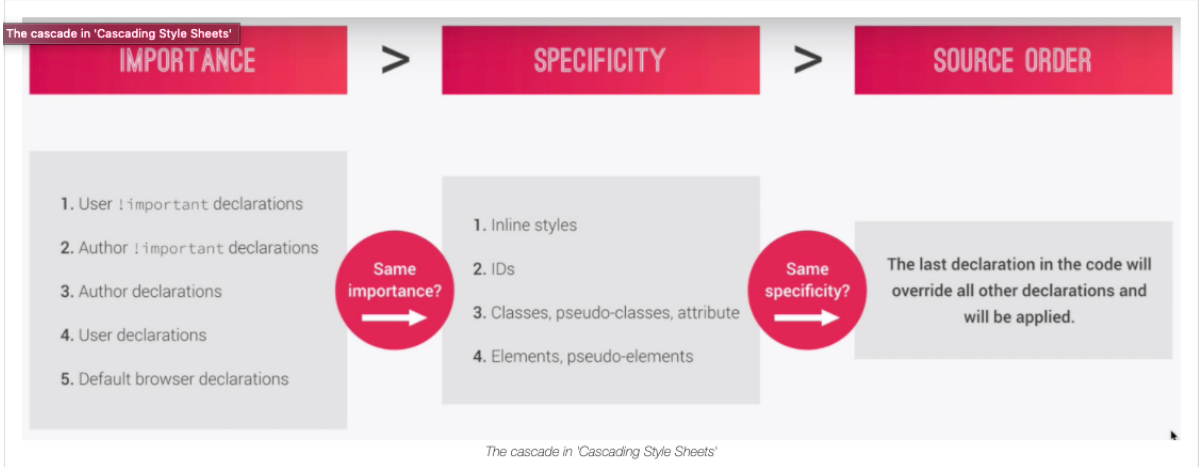
Select everything that IS NOT green
li:not(.green) {
background: red;
}

2.9 Activity

1. ::before styles applied to H3, class="seb_callout activitiies"

Importance

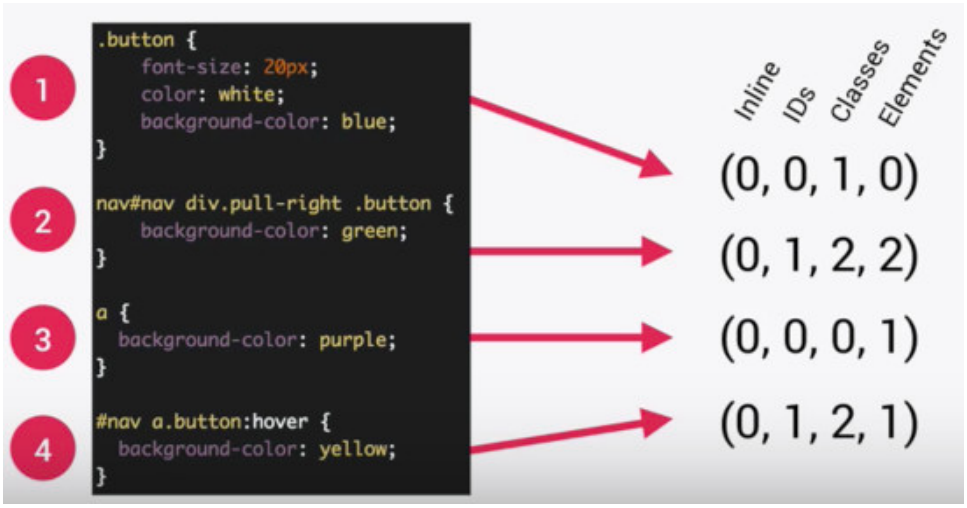
- A user agent (i.e. a browser) - has it's own built-in style sheet
- A user (browsers let users set their preferences within the browser, and this creates a 'user' style sheet)
- An author - in this case a web developer writing css code and applying it to the html page



There is a calculation you can perform to work out the specificity of different selectors. Each of the 4 different types of selectors shown in figure 1 are given a weighting by scoring them with a number as shown below in figure 2 below (1). Let's take for example, selector 2 in figure 2 below. Assuming there is no inline style defined, it is scored as having:

one ID selector:	nav**#nav** div.pull-right .button { background-color: green }
two class selectors:	nav#nav div**.pull-right .button** { background-color: green }
two element (type) selectors:	**nav**#nav **div**.pull-right .button { background-color: green }

These figures are placed in the columns as shown below in figure 2, which are ordered in terms of their precedence for specificity. In the end, we can compare different selectors with this final number. You can see in the example in figure 2, that 'selector 2' would be the most specific of the four selectors since it ends up with the highest number.



- Why shouldn't you pepper your code with !important declarations ? Too much importance will make nothing important as nothing will stand out it will be too busy.

- Why is it important to rely more on specificity of selectors than the order in which they appear? Specifiying makes it easier to make sense of what the style is for and where it will be used rather than relying on order which may not always select the correct response as its just based on order not specifics.

Recap Quiz - Cascade and Inheritance (cascade determines what overrides when there is conflict)

- ```
  section p {
    color: red;
  }
  .p1 {
    color: green;
  }
  ```

  ```
  <section id="section1" class="section1">
    <h2 class="title">About Yoda</h2>
    Paragraph 1
    Paragraph 2
  </section>
  ```

  Which color will be applied to the highlighted element?

  Red

  ✔ Green

  Both section p and .p1 target the highlighted element. The specificity of section p is 0.0.0.0.2 and .p1 is 0.0.0.1.0

- ```
  .section1 p {
    color: red;
  }
  .p1 {
    color: green;
  }
  ```

  ```
  <section id="section1" class="section1">
    <h2 class="title">About Yoda</h2>
    <p class="intro p1">Paragraph 1</p>
    <p class="intro p2">Paragraph 2</p>
  </section>
  ```

  Which color will be applied to the highlighted element?

  ✔ Red

  Both .section1 p and .p1 target the highlighted element. The specificity of .section1 p is 0.0.0.1.1 and .p1 is 0.0.0.1.0

  Green

```
.p1 {
  color: red;
}
section p {
  color: green;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="intro p1">Paragraph 1</p>
  <p class="intro p2">Paragraph 2</p>
</section>
```

Which color will be applied to the highlighted element?

✔ Red

Both .p1 and section p target the highlighted element. The specificity of .p1 is 0.0.0.1.0 and section p is 0.0.0.0.2

Green

```
#section1 {
  color: red;
}
p {
  color: green;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="intro p1">Paragraph 1</p>
  <p class="intro p2">Paragraph 2</p>
</section>
```

Red

✔ Green

The *#section1* selector targets the highlighted element(s) parent. *p* would inherit its parent's color only if there is no color applied directly to paragraphs.

Note:
* is the universal selector which targets *every* element type.

It has a specificity of 0 on the element type.

```
#section1 * {
  color: red;
}
p.intro.p1 {
  color: green;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="intro p1">Paragraph 1</p>
  <p class="intro p2">Paragraph 2</p>
</section>
```

✔ Red

Both *#section1 **** and *p.intro.p1* target the highlighted element. The universal selector (*) has no specificity value. The specificity of ***#section1 **** **is 0.0.1.0.0** and *p.intro.p1* is 0.0.0.2.1

Green

Note:
* is the universal selector which targets *every* element type.

It has a specificity of 0 on the element type.

```
section * {
  color: red;
}
p {
  color: green;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="intro p1">Paragraph 1</p>
  <p class="intro p2">Paragraph 2</p>
</section>
```

Red

✔ Green

Both *section* * and *p* target the highlighted element. The specificity of *section* * is 0.0.0.0.1. The specificity of ***p* is also 0.0.0.0.1 but it comes later in the stylesheet.**

```
section > p {
  color: red;
}
section p {
  color: green;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="intro p1">Paragraph 1</p>
  <p class="intro p2">Paragraph 2</p>
</section>
```

Red

✔ Green

Both *section > p* and *section p* target the highlighted element. The specificity of *section > p* is 0.0.0.0.2. The specificity of ***section p* is also 0.0.0.0.2 but it comes later in the stylesheet.**

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

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="green red">Paragraph 1</p>
</section>
```

Red

✔ Green

Both *.red* and *.green* target the highlighted element. The specificity of *.red* is 0.0.0.1.0. The specificity of ***.green* is also 0.0.0.1.0 but it comes later in the stylesheet.**

Note:

the :not(selectorName list) selector matches any element that does not have a list of selectorNames targeting it

```
section :not(.title) {
  color: green;
}
.red {
  color: red;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="red">Paragraph 1</p>
</section>
```

Which color will be applied to the highlighted elements?

Red

✔ Green

Both *section :not(.title)* and *.red* target the highlighted element. The specificity of ***section :not(.title)* is 0.0.0.1.1** and *.red* is 0.0.0.1.0

My code didn't show this, it went green?

Note:

the :not(selectorName list) selector matches any element that does not have a list of selectorNames targeting it

```
section :not(.title) {
  color: green;
}
p.red {
  color: red;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p class="red">Paragraph 1</p>
</section>
```

Which color will be applied to the highlighted elements?

✔ Red

Both *section :not(.title)* and *p.red* target the highlighted element. The specificity of *section :not(.title)* is 0.0.0.1.1 .The specificity of ***p.red* is also 0.0.0.1.1 but it comes later in the stylesheet.**

Green

```
p {
  color: green;
}
#section1 p {
  color: red;
}
```

```
<section id="section1" class="section1">
  <h2 class="title">About Yoda</h2>
  <p style="color: green;">Paragraph 1</p>
</section>
```

Red

✔ Green

Both *p* and *#section1 p* target the highlighted element. The specificity of *#section1 p* is 0.0.1.0.1 but the highlighted element has also an **inline style that wins (0.1.0.0.0)**

```
p {
  color: red !important;
}
#section1 p {
  color: green;
}
```

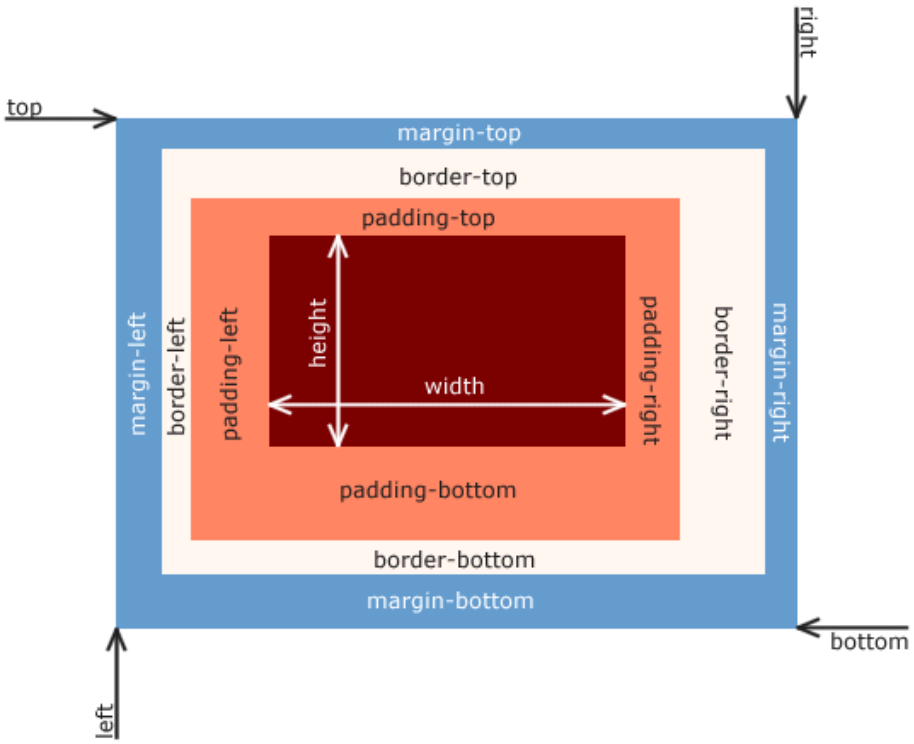
```
<section id="section1" class="section1">
   <h2 class="title">About Yoda</h2>
   <p style="color: green;">Paragraph 1</p>
</section>
```

✔ Red

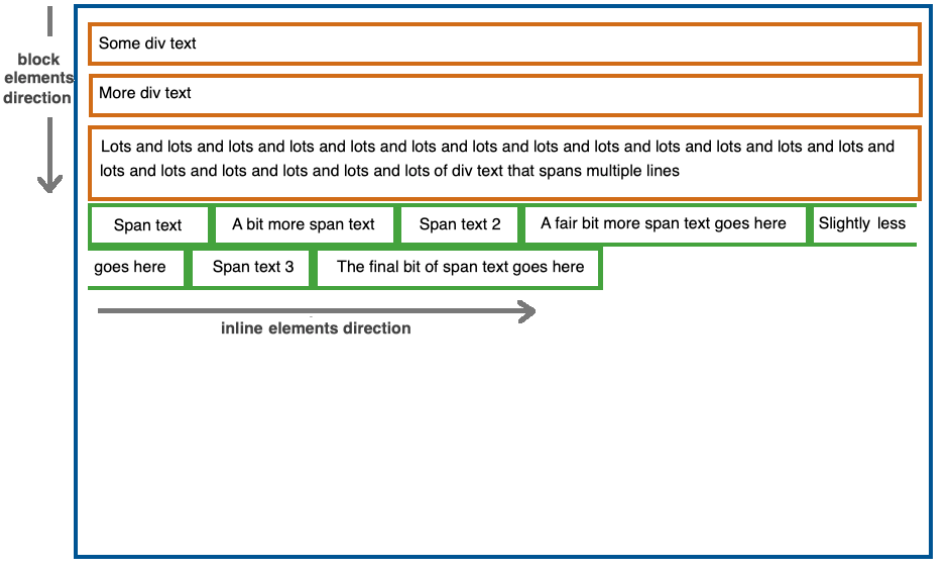
The highlighted element has an inline style (0.1.0.0.0) but the ***p*** **selector uses !important which wins even with the inline styles (1.0.0.0.1)**

Green

CSS Box Model

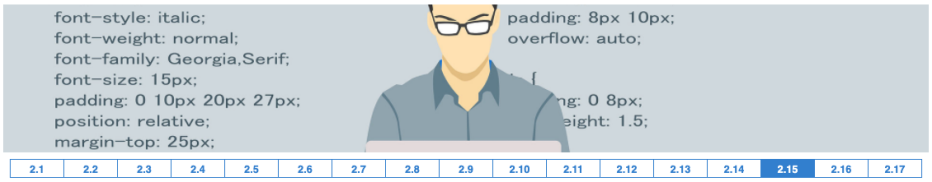


The W3C Box Model (Level 3)



2.15

2.15 Positioning elements with CSS: static, relative and absolute



Replace this with your page title (Hello 1)

[DONE]

Layout Reading: All options and why we use them
http://grid-layout.com/history.html

CSS {selectors: cheat-sheet}

By Web Dev Simplified https://courses.webdevsimplified.com

Basic

Name	CSS	Description	Results
Universal Selector	`*`	Select all elements	a b c d
Type Selector	`div`	Select elements of that type Select div elements	a div c div
Class Selector	`.c`	Select elements with that class Select elements with the c class	.a .b .c .d
Id Selector	`#i`	Select elements with that id Select elements with the i id *It is best practice to **not** use ids in CSS	#a #b #i #d

Combination

Name	CSS	Description	Results
Descendant Selector	`div a`	Select elements that are descendants of the first element Select anchors that are inside a div	div a b a c a d
Direct Child Selector	`div > a`	Select elements that are direct children of the first element Select anchors that are direct children of a div	div a b a c a d
General Sibling Selector	`div ~ a`	Select elements that are siblings of the first element and come after the first element Selects all anchors that are siblings of a div and come after the div	a div a a
Adjacent Sibling Selector	`div + a`	Select elements that are siblings of the first element and come **directly** after the first element Selects all anchors that are siblings of a div and come **directly** after the div	a div a a
Or Selector	`div, a`	Select elements that match any selector in the list Selects all anchors and all divs	a div a b
And Selector	`div.c`	Select elements that match all the selector combinations Selects all divs with the class c	.a div.c .c div

Attribute

Name	CSS	Description	Results
Has Attribute	[a]	Select elements that have that attribute Select elements with the a attribute	[a] [a="1"] [c] d
Exact Attribute	[a="1"]	Select elements that have that attribute with exactly that value Select elements with the a attribute with a value of 1	[a] [a="1"] [c] d
Begins With Attribute	[a^="1"]	Select elements that have that attribute which start with that value Select elements with the a attribute with a value that starts with 1	[a="12"] [a="21"]
Ends With Attribute	[a$="1"]	Select elements that have that attribute which end with that value Select elements with the a attribute with a value that ends with 1	[a="12"] [a="21"]
Substring Attribute	[a*="1"]	Select elements that have that attribute which contain that value anywhere Select elements with the a attribute with a value that contains a 1	[a="12"] [a="21"]

Pseudo Element

Name	CSS	Description	Results
Before Selector	div::before	Creates an empty element directly before the children of selected element	div before c after
After Selector	div::after	Creates an empty element directly after the children of selected element	div before c after

Pseudo Class State

Name	CSS	Description
Hover Selector	button:hover	Select elements that are hovered by the mouse Select buttons that are being hovered
Focus Selector	button:focus	Select elements that are focused. Select buttons that are being focused *Focus is set by either tabbing to an element or clicking an element such as a button or anchor tag
Required Selector	input:required	Select inputs that are required Select inputs with the required attribute
Checked Selector	input:checked	Select checkboxes/radio buttons that are checked Select inputs that are checked
Disabled Selector	input:disabled	Select inputs that are disabled Select inputs with the disabled attribute

Pseudo Class Position/Other

Name	CSS	Description	Results
First Child Selector	a:first-child	Select elements that are the first child inside a container Select anchors that are the first child	div div a b b a
Last Child Selector	a:last-child	Select elements that are the last child inside a container Select anchors that are the last child	div div a b b a
Nth Child Selector	a:nth-child(2n)	Select elements that are the nth child inside a container based on the formula Select anchors that are even numbered children	div a a b a
Nth Last Child Selector	a:nth-last-child(3)	Select elements that are the nth child inside a container based on the formula counting from the end Select anchors that are the third to last child	div a a b a
Only Child Selector	a:only-child	Select elements that are the only child inside a container Select anchors that are the only child	div div a b a
First Of Type Selector	a:first-of-type	Select elements that are the first of a type inside a container Select the first anchor in a container	div b a a b
Last Of Type Selector	a:last-of-type	Select elements that are the last of a type inside a container Select the last anchor in a container	div b a a b
Nth Of Type Selector	a:nth-of-type(2n)	Select elements that are the nth of a type inside a container based on the formula Select every second anchor	div b a a b
Nth Last Of Type Selector	a:nth-last-of-type(2)	Select elements that are the nth of a type inside a container based on the formula counting from the end Select the second to last anchor	div b a a b
Only Of Type Selector	a:only-of-type	Select elements that are the only of a type inside a container Select anchors that are the only anchor in a container	div b b a b
Not Selector	a:not(.c)	Select all elements that do not match the selector inside the not selector Select all anchor tags that do not have the c class	b a.c a a.d

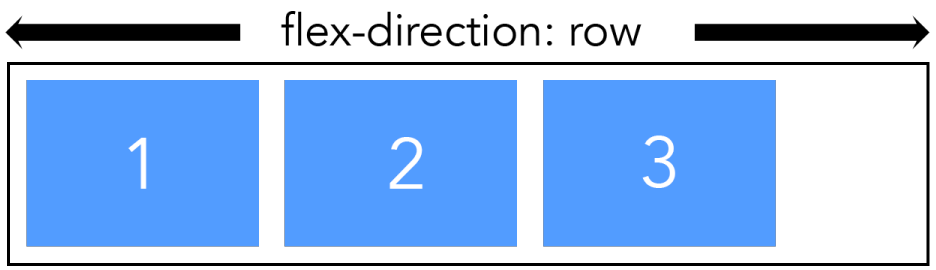
CSS Position Property

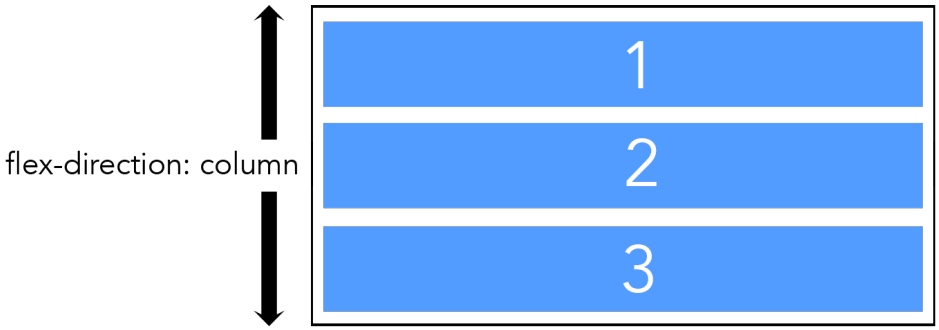
By default, s are block-level elements and they stack one above the other vertically on the page unless positioned in some way. The first CSS tool for handling alignment of these elements was the position property (i.e. position:fixed). The position property is still part of the spec and is still very useful today. There are five different position values:

- static

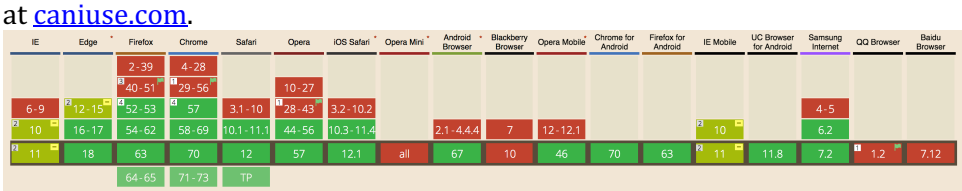
- relative
- fixed
- absolute
- sticky

position: static is the default way elements are positioned. position: relative positions elements relative to their normal position. position: fixed is positioned relative to the viewport (browser window), meaning it stays in place on the screen even when the page is scrolled. position: absolute positions elements relative to the nearest positioned ancestor. position: sticky positions elements based on the user's scroll position.
Sometimes elements need to overlap other elements. To determine the Z-axis (depth dimension) stacking order, z-index was created. For example, an element with a z-index: 2 will show up on top of an element with a z-index: 1.
Absolute and relative positioning are useful for aligning some elements, but are limited in their ability to effectively control layout for an entire page.





Browser support for CSS Grid made a huge step forward in 2017. Chrome, Firefox, Safari, and Opera all released versions with support in March, and Microsoft followed through with Edge in October. All major browsers now support it, as can be seen below and at caniuse.com.



CSS Grid is the first two-dimensional grid-based layout system, meaning it can handle columns and rows. It solves the layout challenges of the past and provides the tools needed to make a design vision a reality. No more floats or hacks or even frameworks. It is simply the most powerful tool made for web page layout and it's available now.

Float, Grid and Flex

Keep float for images in text not for text as it can get ugly when text is not the same size
Flex is good for forms and Nav bars
Whole page layout GRID

Flex Layouts Source: https://www.youtube.com/watch?v=hYJvxsgnGMA

Accessibility: the rules are not always correct for what users find easier to read/view

But the truth is that color-blind users can differentiate the contrasting colors quite clearly.
https://uxmovement.com/buttons/the-myths-of-color-contrast-accessibility/
<<CSS Selector Cheat Sheet - Dark.pdf>>