

# Cascading Style Sheets

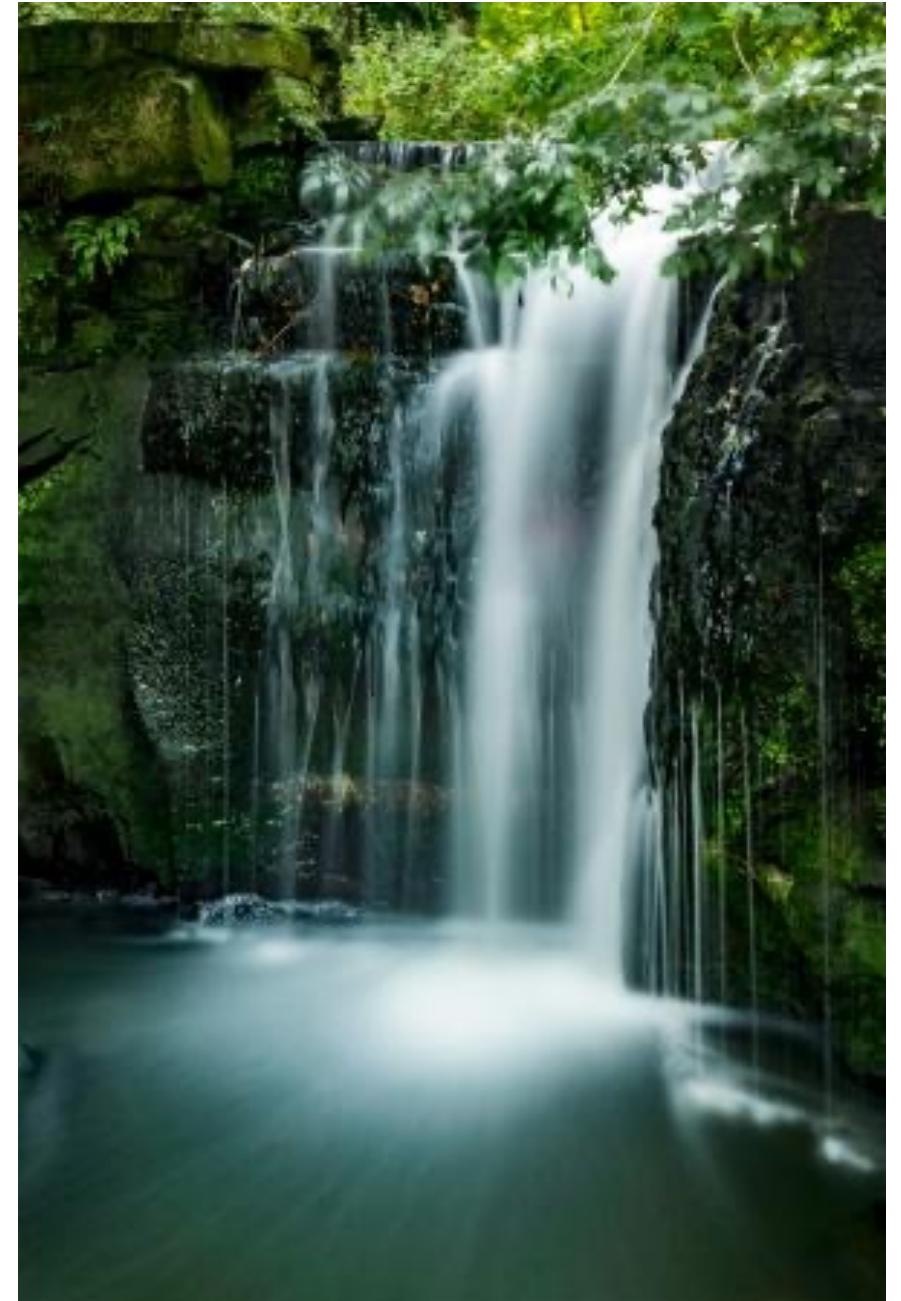
## Styling the Web

Fulvio Corno

Luigi De Russis

Enrico Masala

*Some slides adapted from Laura Farinetti*



# Goal

- Styling web content - CSS
- Advanced layout in web pages
- Responsive layouts

# Outline

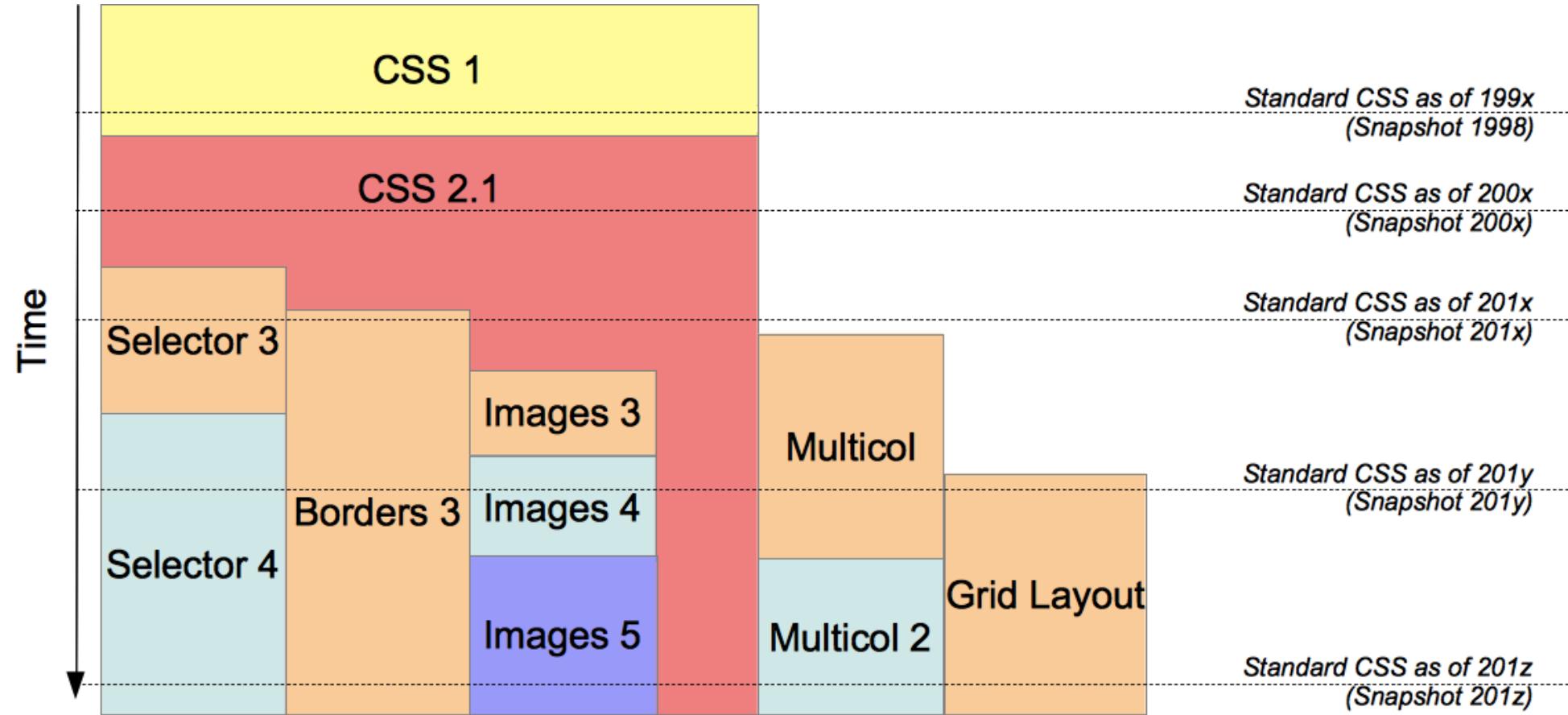
- CSS syntax
- CSS selectors
- CSS cascading
- CSS box model
- Page layout with grid
- CSS Flexbox
- Responsive layout



# Cascading Style Sheets

- CSS 1: W3C recommendation (17 Dec 1996)
- CSS 2.1: W3C Recommendation  
(7 June 2011)
- CSS 3: modular approach, different stages (REC, PR, CR, WD) for different parts
  - See <https://www.w3.org/Style/CSS/>
- Resources:
  - <https://developer.mozilla.org/en-US/docs/Web/CSS>
  - <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>

# Overview of CSS development approach



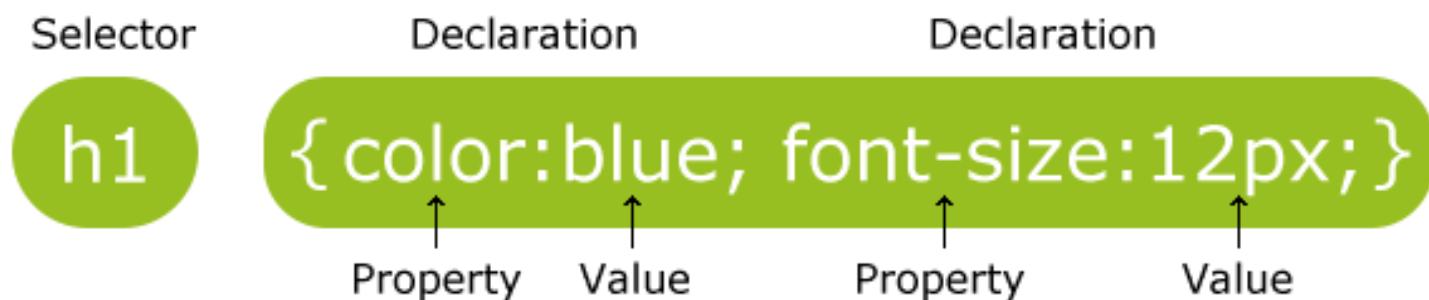
<https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3>

Cascading Style Sheets

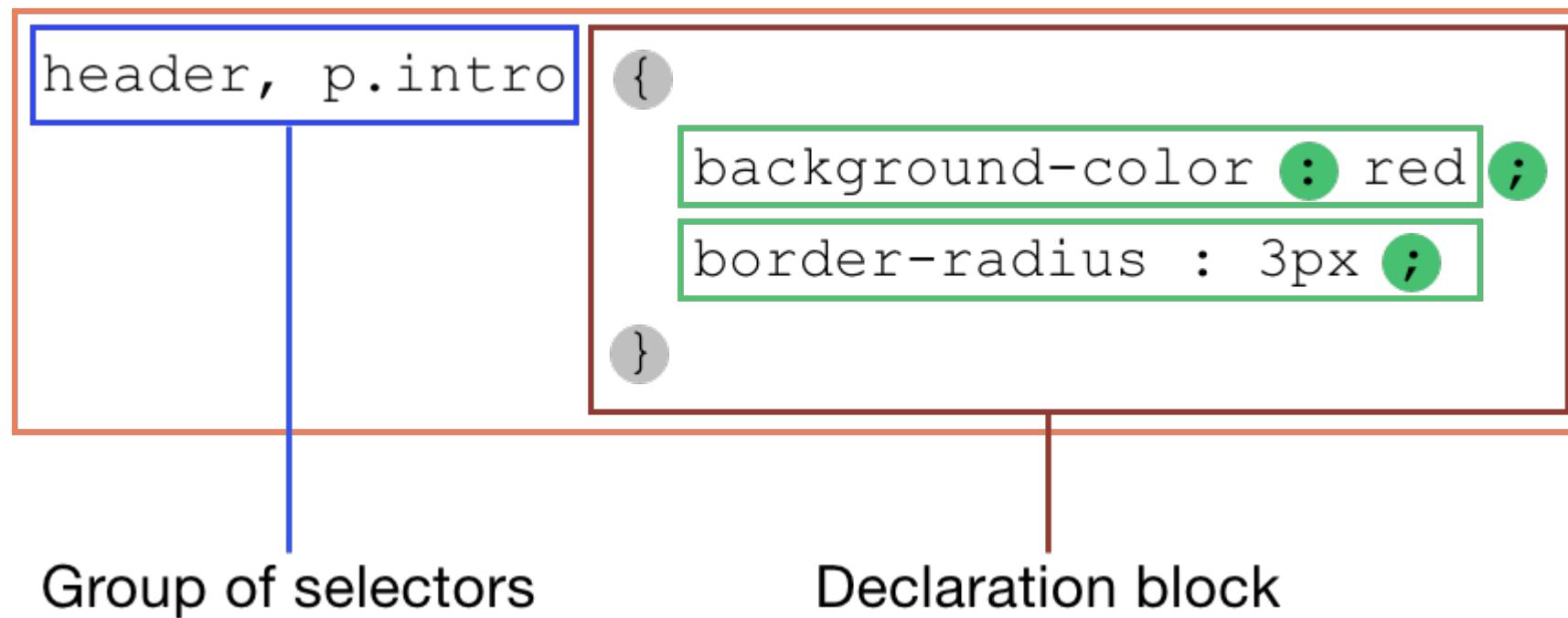
# **CSS SYNTAX**

# CSS Syntax

- CSS is based on **rules**
- A rule is a statement about one [stylistic] aspect of one or more HTML element
  - **Selector + Declaration(s)**
- A style sheet is a set of one or more rules that apply to an HTML document

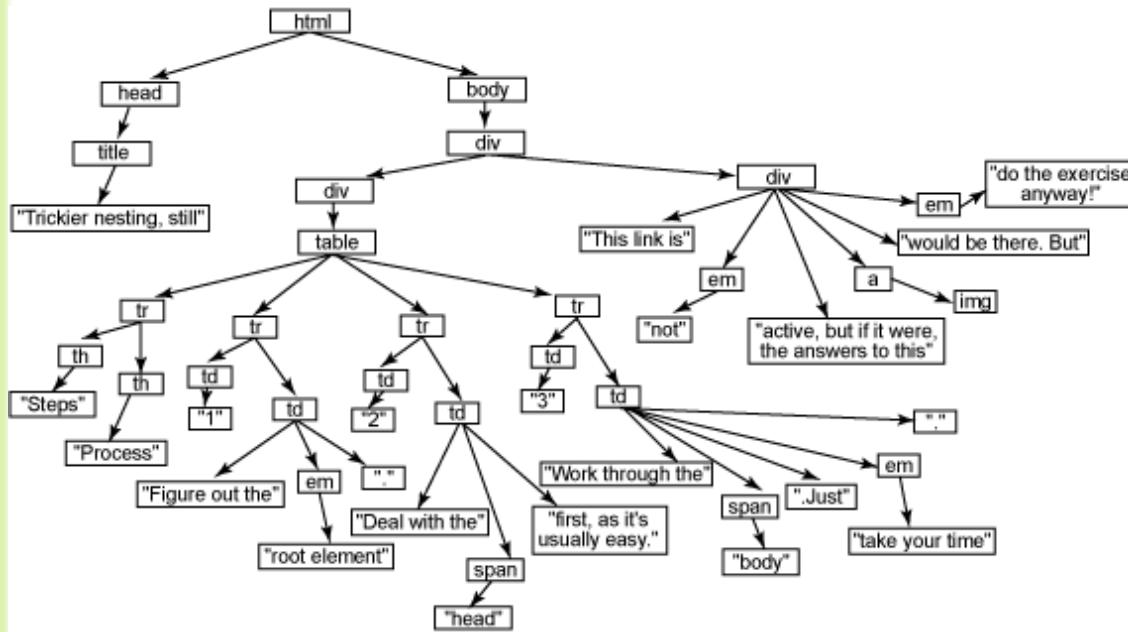


# General syntax



# HTML nested structure

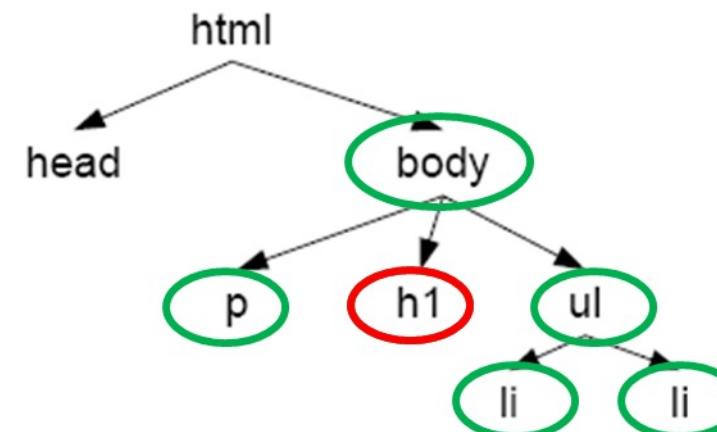
```
<html lang="en">
<head>
  <title>Trickier nesting, still</title>
</head>
<body>
  <div>
    <div>
      <table>
        <tr><th>Steps</th><th>Processes</th></tr>
        <tr><td>1</td><td>Figure out the <em>root element</em>. </td></tr>
        <tr><td>2</td><td>Deal with the <span>head</span> first as it's
usually easy.</td></tr>
        <tr><td>3</td><td>Work through the <span>body</span>. Just <em>take
your time</em>. </td></tr>
      </table>
    </div>
  </div>
  This link is <em>not</em> active, but if it were, the answer to this
<a></a> would be there. But <em>do the exercise
anyway!</em>
</div>
</div>
</body>
</html>
```



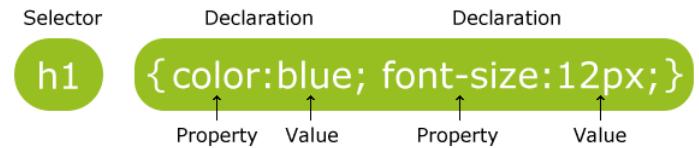
# Tree structure and inheritance

- HTML documents are trees
- **Styles are inherited along trees**
- Override: when two rules are in conflict the most specific wins
- Example
  - `body {color: green}`
  - `h1 {color: red}`

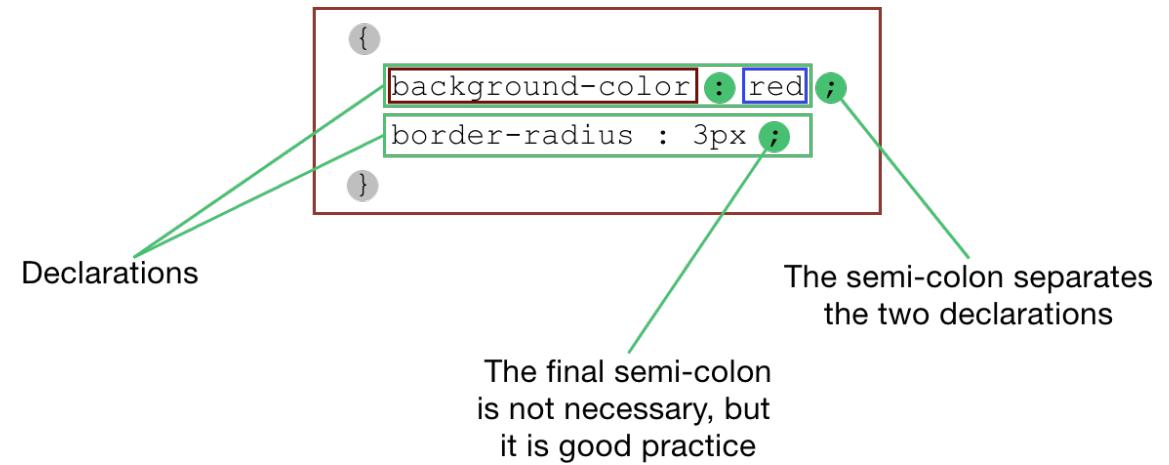
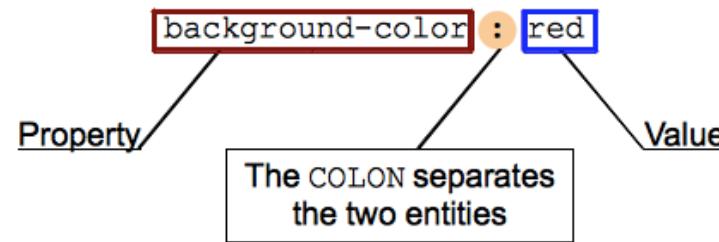
That's why it is called CASCADING!



# Declaration Syntax



A CSS declaration :



# CSS properties (200+)

- Allowed Values (and Default Value):
  - Numbers, measurements, percentage
  - Enumerated options (strings)
  - Colors (name, RGB[A], HSL[A])
- Applies to:
  - Which elements may have this property
- Inherited:
  - Does also apply to children elements?

```
align-content align-items align-self all animation animation-delay animation-
direction animation-duration animation-fill-mode animation-iteration-count
animation-name animation-play-state animation-timing-function backface-visibility
background background-attachment background-blend-mode background-clip
background-color background-image background-origin background-position
background-repeat background-size border border-bottom border-bottom-color
border-bottom-left-radius border-bottom-right-radius border-bottom-style border-
border-top border-collapse border-color border-image border-image-outset
border-image-repeat border-image-slice border-image-source border-image-width
border-left border-left-color border-left-style border-left-width border-radius
border-right border-right-color border-right-style border-right-width border-
spacing border-style border-top border-top-color border-top-left-radius border-
top-right-radius border-top-style border-top-width border-width bottom box-
decoration-break box-shadow box-sizing break-after break-before break-inside
caption-side caret-color @charset clear clip clip-path color column-count column-
fill column-gap column-rule column-rule-color column-rule-style column-rule-width
column-span column-width columns content counter-increment counter-reset cursor
direction display empty-cells filter flex flex-basis flex-direction flex-flow
flex-grow flex-shrink flex-wrap float font @font-face font-family font-feature-
settings font-kerning font-size font-size-adjust font-stretch font-style font-
variant font-variant-caps font-weight grid grid-area grid-auto-columns grid-auto-
flow grid-auto-rows grid-column grid-column-end grid-column-gap grid-column-start
grid-gap grid-row grid-row-end grid-row-gap grid-row-start grid-template grid-
template-areas grid-template-columns grid-template-rows hanging-punctuation
height hyphens @import isolation justify-content @keyframes left letter-spacing
line-height list-style list-style-image list-style-position list-style-type
margin margin-bottom margin-left margin-right margin-top max-height max-width
@media min-height min-width mix-blend-mode object-fit object-position opacity
order outline outline-color outline-offset outline-style outline-width overflow
overflow-x overflow-y padding padding-bottom padding-left padding-right padding-
top page-break-after page-break-before page-break-inside perspective perspective-
origin pointer-events position quotes resize right scroll-behavior tab-size
table-layout text-align text-align-last text-decoration text-decoration-color
text-decoration-line text-decoration-style text-indent text-justify text-overflow
text-shadow text-transform top transform transform-origin transform-style
transition transition-delay transition-duration transition-property transition-
timing-function unicode-bidi user-select vertical-align visibility white-space
width word-break word-spacing word-wrap writing-mode z-index
```



- <http://www.w3schools.com/cssref/>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
- <https://www.tutorialrepublic.com/css-reference/css3-properties.php>

# CSS properties by category

- Animation Properties
- Background Properties
- Border Properties
- Color Properties
- Dimension Properties
- Generated Content Properties
- Flexible Box Layout
- Font Properties
- List Properties
- Margin Properties
- Multi-column Layout Properties
- Outline Properties
- Padding Properties
- Print Properties
- Table Properties
- Text Properties
- Transform Properties
- Transitions Properties
- Visual formatting Properties

[https://www.tutorialrepublic.com/  
css-reference/css3-properties.php](https://www.tutorialrepublic.com/css-reference/css3-properties.php)

# CSS Units: Most Used

- CSS has several different **units** for expressing a length
  - format: a number followed by a unit (e.g., 10px)
- Two types of length units
  - absolute (fixed)
  - relative
- The most common fixed unit is pixel (**px**)
  - they are relative to the viewing device
  - for low-dpi devices, 1px is one device pixel (dot) of the display
  - for printers and high-resolution screens, 1px implies multiple device pixels
  - not well suited for responsive design (later on)

```
{  
width: 100%,  
font-size: 2em,  
margin: 10px 20px 10px 20px,  
padding: 10px, ...  
}
```

<https://developer.mozilla.org/en-US/docs/Web/CSS/margin>

# CSS Units: Most Used

Viewport: The browser's viewport is the area of the window in which web content can be seen. This is often not the same size as the rendered page, in which case the browser provides scrollbars for the user to scroll around and access all the content.

- The most common relative units, instead:

Unit	Description
em	Relative to the font-size of the element. 2em means 2 times the size of the font of the current element
rem	Relative to font-size of the root element of the HTML page (Root EM)
vw	Relative to 1% of the width of the viewport, i.e., the browser window size
vh	Relative to 1% of the height of the viewport
%	Relative to the <u>parent</u> / container element

- Suggestions:
  - prefer relative units to absolute ones, when possible
  - rem is (nowadays) preferred to em

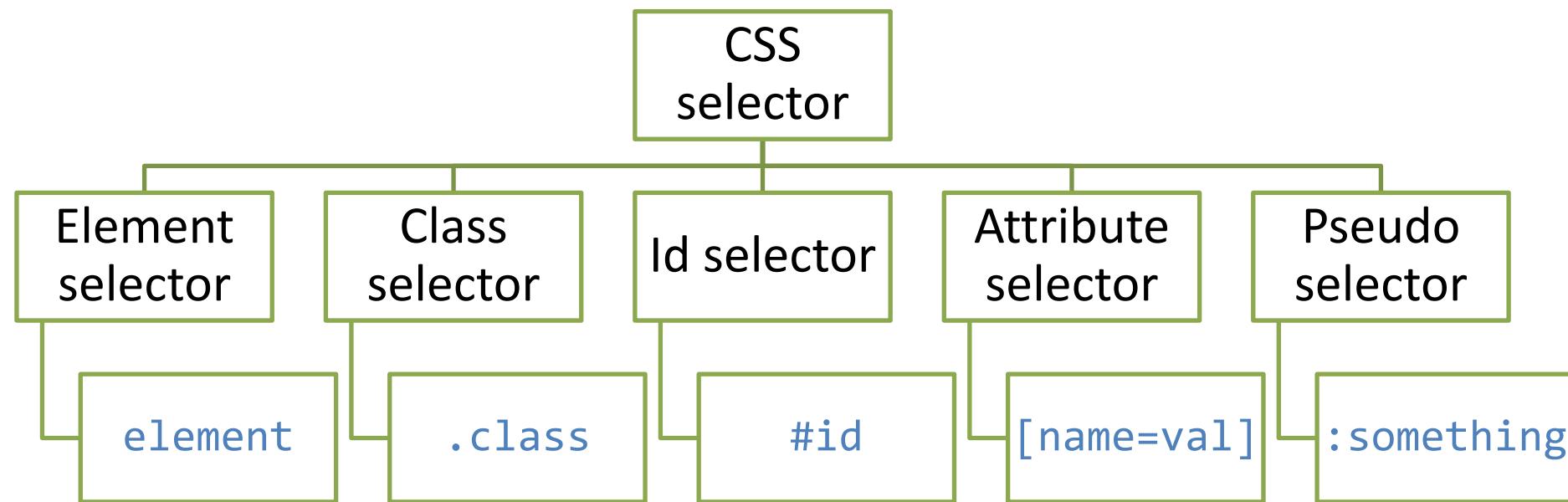
Cascading Style Sheets

# CSS SELECTORS

[See Developer Console > Styles](#)

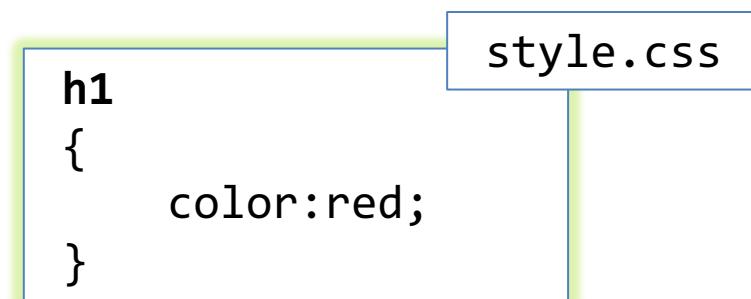
# CSS selectors

- Patterns used to select the element(s) you want to style
- Three main types of selectors plus “pseudo-selectors”



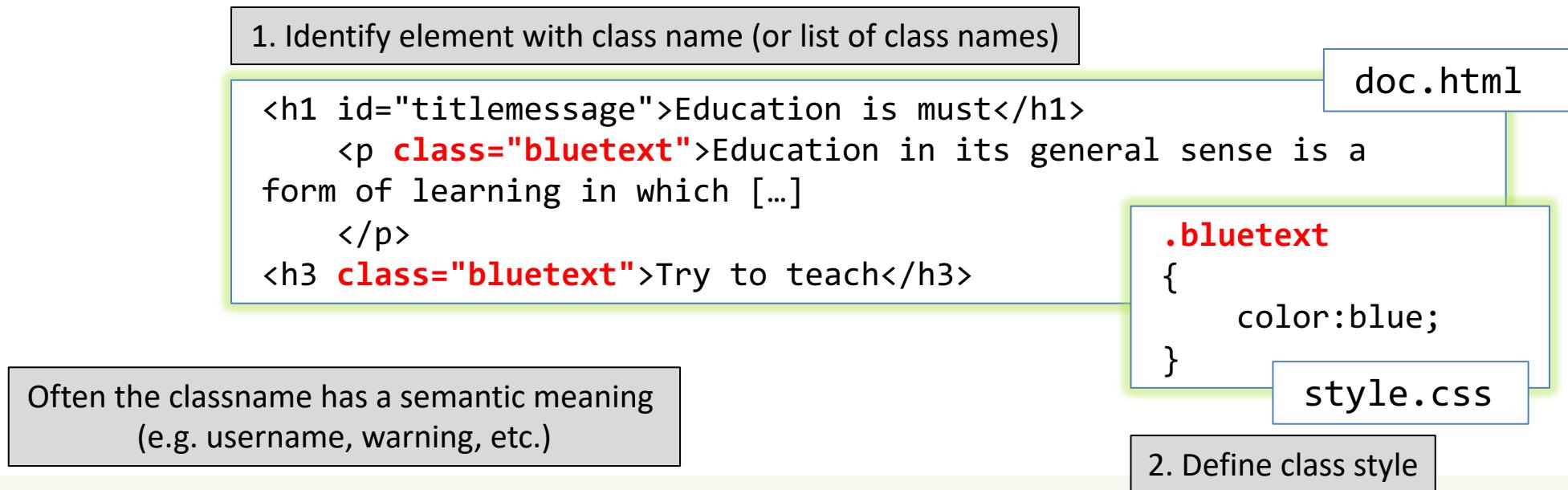
# Element selector

- Used to apply the same style to all instances of a specific element in a document
- Example: apply the color red to all h1 elements that appear in the document



# Class selector

- Used to apply the same style to all elements belonging to a specific (defined) class
- Applies a specific style to a set of related elements, identified by class



# Id selector

- Used to apply a style to a specific element in a document
- You can select a specific element by its (declared) id and apply a style to that (and possibly its children)

1. Identify element with ID name (UNIQUE)

```
<h1 id="titlemessage">Education is must</h1>
  <p class="bluetext">Education in its general sense is a
form of learning in which [...]
  </p>
<h3 class="bluetext">Try to teach</h3>
```

doc.html

```
#titlemessage
{
  color:gray
}
```

2. Define ID style

style.css

# Attribute selectors

Selector	Example	Example description	CSS
[attribute]	[target]	Selects all elements with a target attribute	2
[attribute=value]	[target=_blank]	Selects all elements with target="_blank"	2
[attribute~=value]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"	2
[attribute =value]	[lang =en]	Selects all elements with a lang attribute value starting with "en"	2
[attribute^=value]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"	3
[attribute\$=value]	a[href\$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"	3
[attribute*=value]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"	3

# Pseudo class selector

- Used to style an element based on something other than the structure of the document
  - E.g., the status of a form element or link

```
/* makes all unvisited links blue */
a:link {color:blue;}
/* makes all visited links green */
a:visited {color:green;}
/* makes links red when hovered or activated */
a:hover, a:active {color:red;}
/* makes table rows red when hovered over */
tr:hover {background-color: red;}
/* makes input elements yellow when focus is applied */
input:focus {background-color:yellow;}
```

# Combining selectors

Very useful for web scraping  
see developer console selectors & \$0

- `element.class#id[n=v]` `a.bluetext#textmessage[href$="pdf"]:visited`  
→ may be combined
- `S1, S2` → S1 union S2 [or]
- `S1 S2` → S2 nested within S1 [descendant]
- `S1 > S2` → S2 if a direct children of S1
- `S1 + S2` → S2 if it comes after a S1 [next/adjacent sibling]
- `S1 ~ S2` → S2 if it comes later than S1 [subsequent sibling]

[https://developer.mozilla.org/en-US/docs/Learn/CSS/Building\\_blocks/Selectors/Combinators](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Combinators)

# CSS selectors

Selector	Example	Example description	CSS
.class	.intro	Selects all elements with class="intro"	1
#id	#firstname	Selects the element with id="firstname"	1
*	*	Selects all elements	2
element	p	Selects all <p> elements	1
element,element	div, p	Selects all <div> elements and all <p> elements	1
element element	div p	Selects all <p> elements inside <div> elements	1
element>element	div > p	Selects all <p> elements where the parent is a <div> element	2
element+element	div + p	Selects all <p> elements that are placed immediately after <div> elements	2
element1~element2	p ~ ul	Selects every <ul> element that are preceded by a <p> element	3

[http://www.w3schools.com/cssref/css\\_selectors.asp](http://www.w3schools.com/cssref/css_selectors.asp)

# Display property

- Allows to control element visualization (block or inline)
- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way

```
li {display:inline;}
```

Overrides li “block”, horizontal list

```
span {display:block;}
```

[http://www.w3schools.com/Css/css\\_display\\_visibility.asp](http://www.w3schools.com/Css/css_display_visibility.asp)

# Display and visibility properties

- The property `display` allows to hide an element, too
  - The element will be hidden, the page will be displayed as if the element is **not there**

```
h1.hidden {  
    display: none;  
}
```

- The property `visibility` also can hide an element, but the element will still take up the same space as before
  - The element will be hidden, but **still affects the layout**

All the elements  
around do not move to  
fit the empty space

```
h1.hidden {  
    visibility: hidden;  
}
```

# CSS pseudo-class selectors

Selector	Example	Example description	CSS
:active	a:active	Selects the active link	1
::after	p::after	Insert something after the content of each <p> element	2
::before	p::before	Insert something before the content of each <p> element	2
:checked	input:checked	Selects every checked <input> element	3
:disabled	input:disabled	Selects every disabled <input> element	3
:empty	p:empty	Selects every <p> element that has no children (including text nodes)	3
:enabled	input:enabled	Selects every enabled <input> element	3
:first-child	p:first-child	Selects every <p> element that is the first child of its parent	2
::first-letter	p::first-letter	Selects the first letter of every <p> element	1
::first-line	p::first-line	Selects the first line of every <p> element	1

Selector	Example	Example description	CSS
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent	3
:focus	input:focus	Selects the input element which has focus	2
:hover	a:hover	Selects links on mouse over	1
:in-range	input:in-range	Selects input elements with a value within a specified range	3
:invalid	input:invalid	Selects all input elements with an invalid value	3
:lang( <i>language</i> )	p:lang(it)	Selects every <p> element with a lang attribute equal to "it" (Italian)	2
:last-child	p:last-child	Selects every <p> element that is the last child of its parent	3
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent	3
:link	a:link	Selects all unvisited links	1

# CSS pseudo-class selectors

Selector	Example	Example description	CSS
:not(selector)	:not(p)	Selects every element that is not a <p> element	3
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent	3
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child	3
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child	3
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent	3
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent	3
:only-child	p:only-child	Selects every <p> element that is the only child of its parent	3
:optional	input:optional	Selects input elements with no "required" attribute	3
:out-of-range	input:out-of-range	Selects input elements with a value outside a specified range	3

Selector	Example	Example description	CSS
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified	3
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified	3
:required	input:required	Selects input elements with the "required" attribute specified	3
:root	:root	Selects the document's root element	3
::selection	::selection	Selects the portion of an element that is selected by a user	
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)	3
:valid	input:valid	Selects all input elements with a valid value	3
:visited	a:visited	Selects all visited links	1



Baseline 2023

NEWLY AVAILABLE

:has() selector

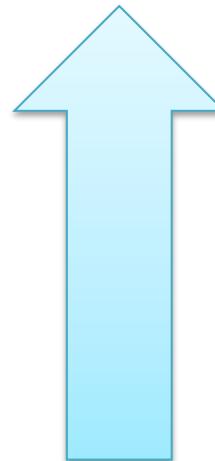
<https://developer.mozilla.org/en-US/docs/Web/CSS/:has>

Cascading Style Sheets

# **CSS CASCADING**

# Cascading Style Sheets

- The term “cascading” means that a document can include more than one style sheet
- In this case, visualization follows **priority rules**
  - Inline Style (inside HTML tag) not recommended
  - Internal Style (usually in the HTML head section)
  - External Style
  - Browser Default Style



**priority**

# External style

preferred: modularity

- Link to an external style sheet using the `<link>` element

```
h1 { font-size:17px;  
      font-family:verdana; color:green; }  
h2 { font-size:18px;  
      font-family:arial; color:red; }
```

style.css

```
<head>  
  <link rel=stylesheet type="text/css"  
        href="style.css">  
</head>  
<body>  
  <h1>Questo testo e' di colore verde, e utilizza il  
      font verdana a 17 pixel</h1>  
  <h2>Questo testo e' di colore rosso, e utilizza il  
      font arial a 18 pixel</h2>  
</body>
```

# Internal style

- <style> element inside the document header
- Not recommended – prefer external styles

```
<head>
  <style type="text/css">
    h1 { font-size:17px; font-family:verdana;
          color:green; }
    h2 { font-size:18px; font-family:arial;
          color:red; }
  </style>
</head>
```

# Inline style

not recommended

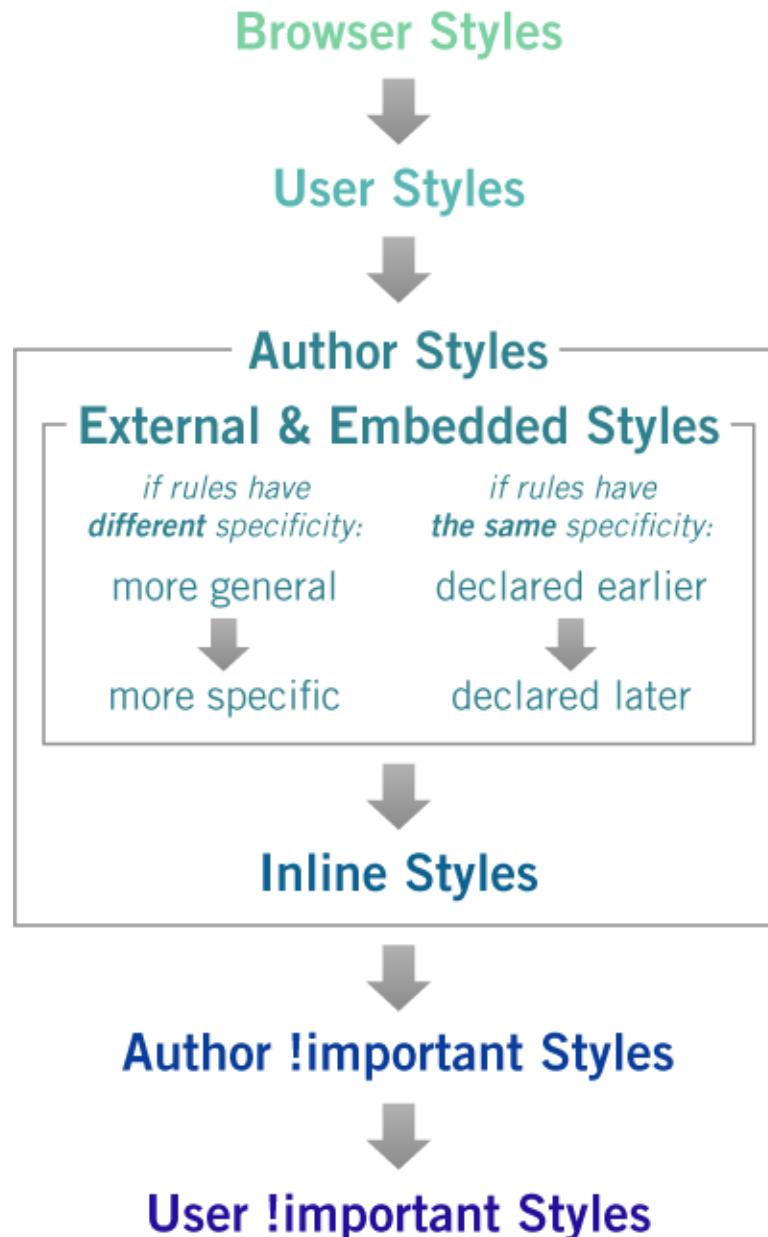
- <style> attribute within an HTML element
- Last resort, just for local (very local) changes

```
<h1 style="font-size:17px;  
font-family:verdana; color:green; "> Questo  
testo e' di colore verde, e utilizza il  
font verdana a 17 pixel </h1>
```

# Priority rules

- Rules can be marked as “important”

```
h1 {  
    color:red !important  
}
```



# The definitive guide to CSS styling order

Includes CSS stylings for SVG

Ordering, selectors or specificity and important keyword does not apply to SVG inline attributes

CSS codes to the right or bottom has higher priority and will be applied.

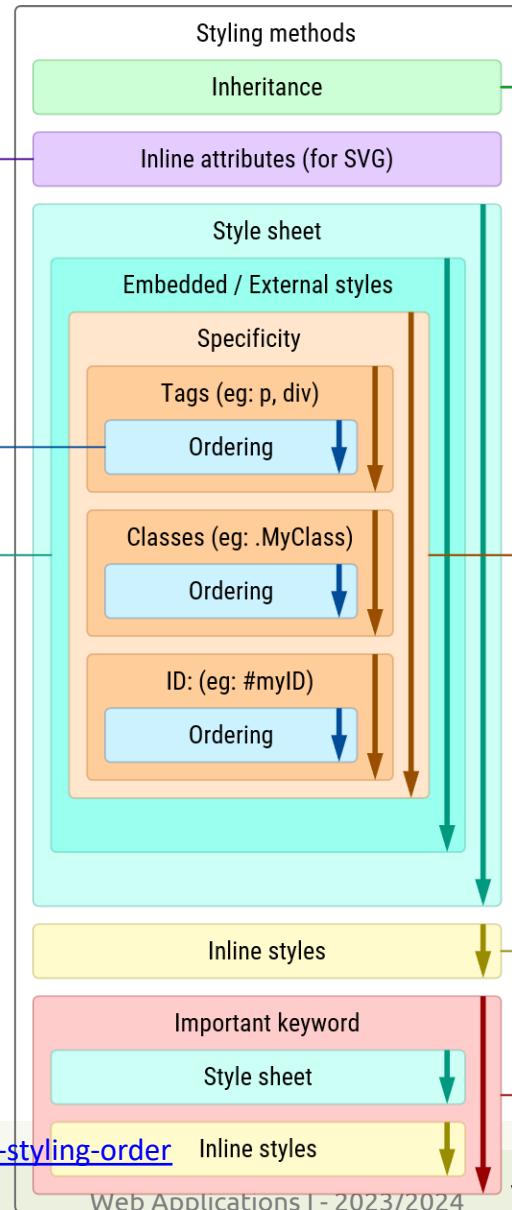
```
p { color: red; color: blue; }  
Blue will be applied
```

```
p {  
  color: red;  
  color: blue;  
}  
Blue will be applied
```

```
p { color: red; }  
p { color: blue; }  
Blue will be applied
```

For each style, ordering rules continue to apply, from left to right and top to bottom.

```
<style>  
#myID { color: red }  
</style>  
<style>  
#myID { color: blue }  
</style>  
Blue will be applied
```



Styling inherited from nearest parent element

Child styling (if exist) has higher priority even though inherited parent styling contains important keyword

Inherited styles has the lowest priority among styling methods

Specificity has higher priority than ordering, with tags, classes and ID in ascending priority.

```
#myID { color: green; }  
.MyClass { color: blue; }  
p { color: red; }
```

Element will be styled with green because ID specificity has the highest priority, superceding ordering rules.

Within specificity, ordering rules still applies.

```
#myID { color: red; }  
#myID { color: blue; }  
Blue will be applied.
```

Inline styles has higher priority than style sheets, and within inline styles, ordering rules applies.

```
<p style="color: red" style="color: blue">  
Blue will be applied
```

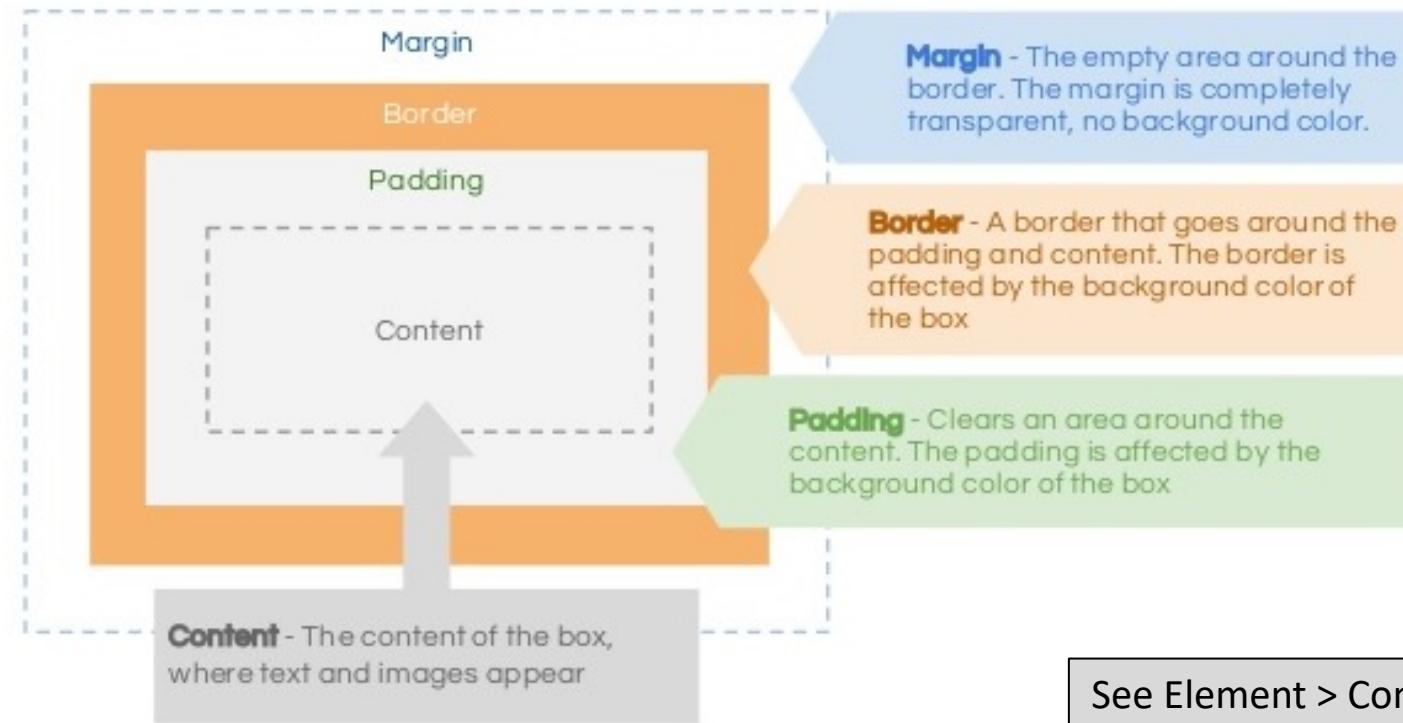
Important keyword in inline styles has higher priority than the same keyword in style sheets.

Cascading Style Sheets

# **CSS BOX MODEL**

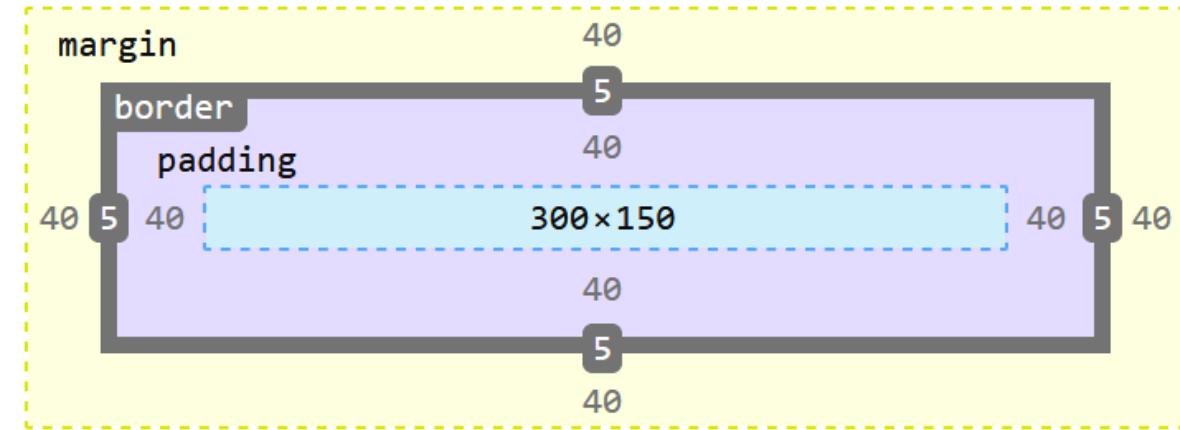
# The box model

- One of the cornerstones of CSS
- Every element on the page is considered to be a rectangular box

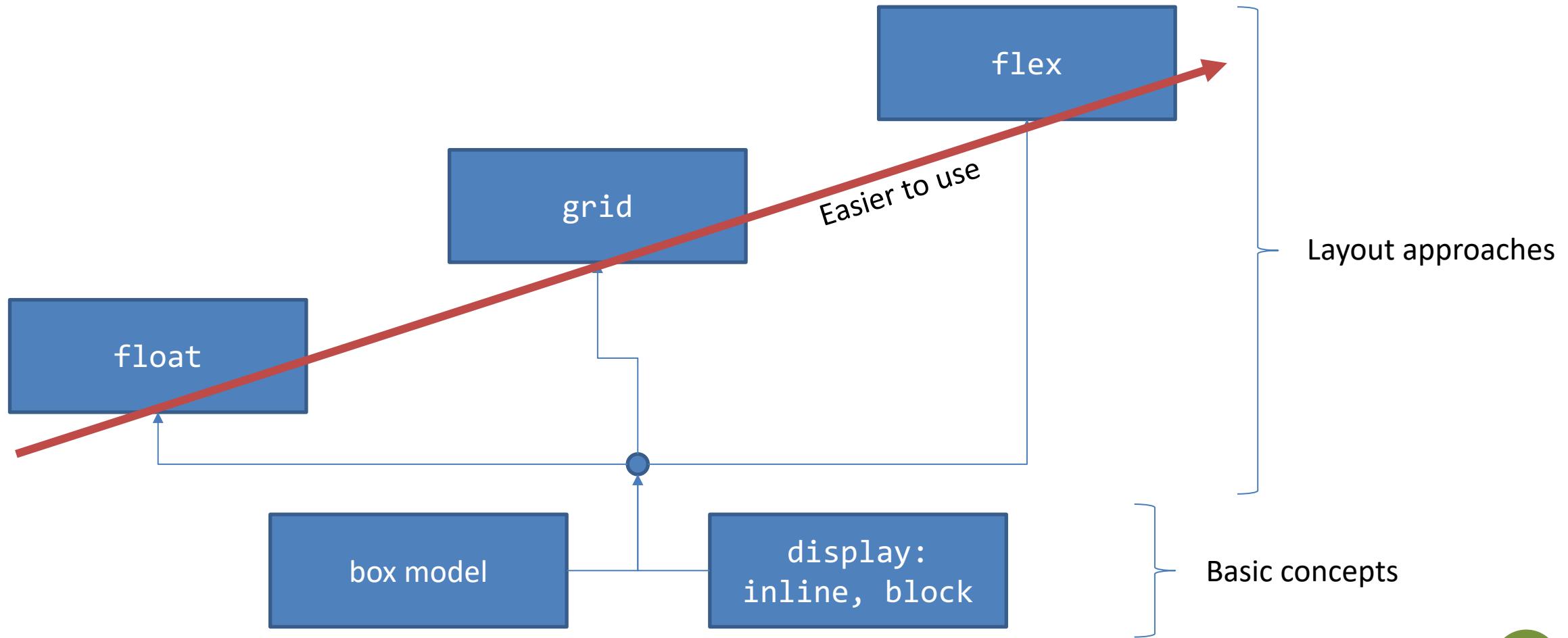


# The box model

- Total element width =  
width + left padding + right padding +  
left border + right border + left margin  
+ right margin
- Total element height =  
height + top padding +  
bottom padding + top border +  
bottom border + top margin + bottom  
margin
- You can set any of these  
properties, independently



# Page Layout methods

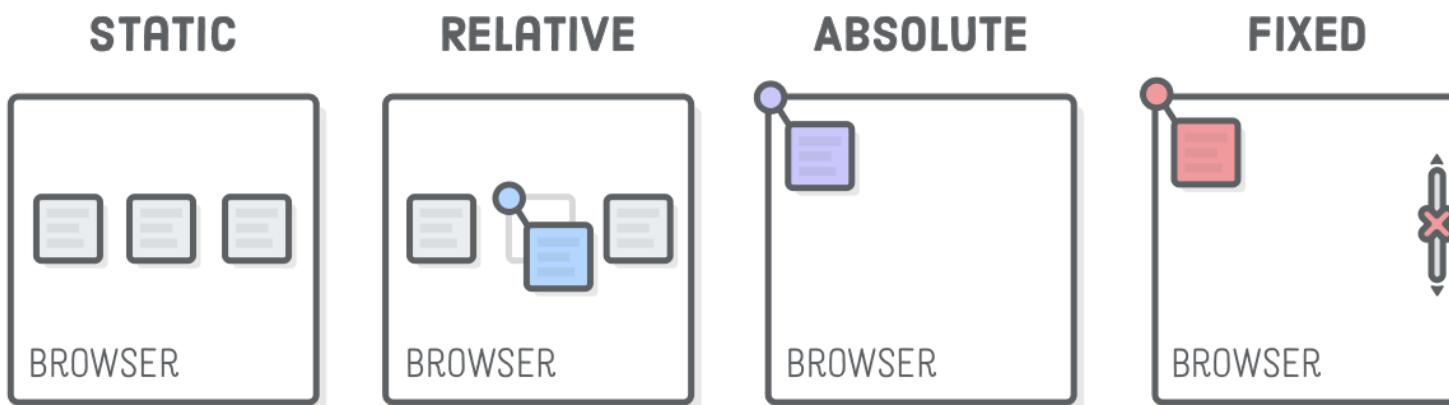


Cascading Style Sheets

# **CSS POSITIONING SCHEMES**

# Positioning schemes

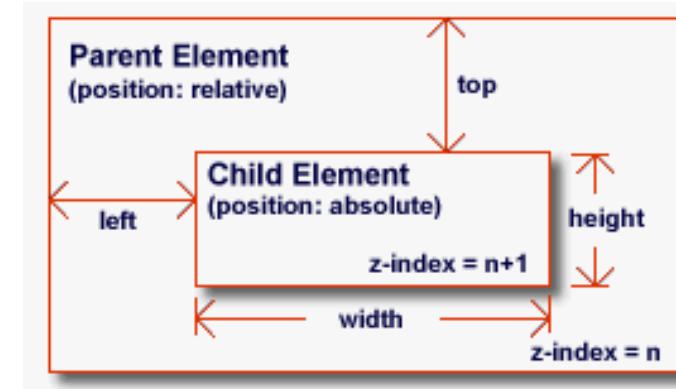
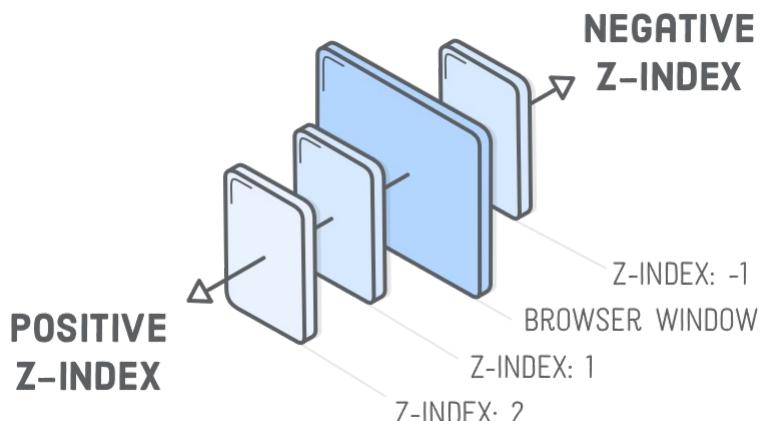
- **Static:** normal flow
- **Relative:** offset relative to the block position in the normal flow
- **Absolute:** the box position is determined by the top, left, right, bottom properties, relative to the containing block
- **Fixed:** fixed with respect to some reference point (the viewport)



```
.item {  
  position: static | relative;  
  absolute | fixed;  
  left: 20px;  
  top: 20px;  
}
```

# z-index

- In case of overlaps the z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others)



Cascading Style Sheets

# LAYOUT WITH FLOATS

# Floating

- The CSS **float** property gives control over the horizontal position of an element



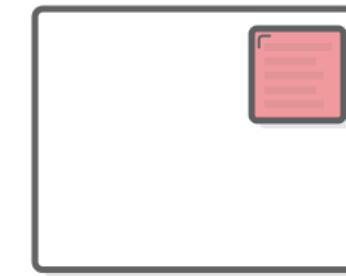
**LEFT ALIGN**

FLOAT: LEFT;



**CENTER ALIGN**

MARGIN: 0 AUTO;



**RIGHT ALIGN**

FLOAT: RIGHT;

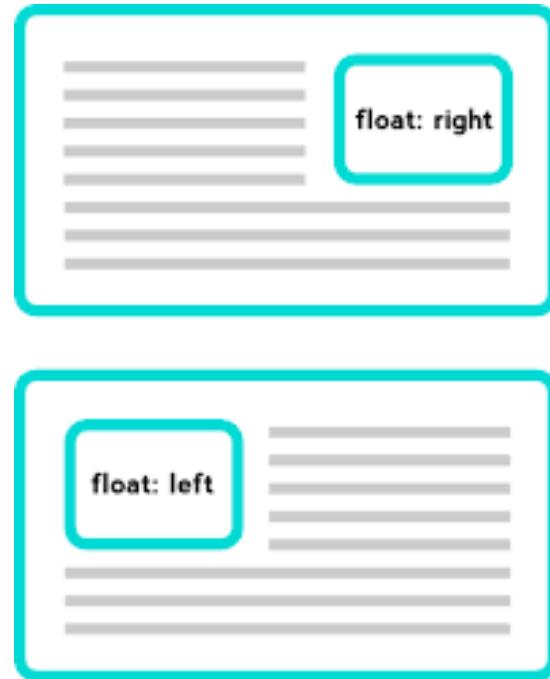
<https://internetingishard.com/html-and-css/floats/>

# Floating

- A floated box can either be shifted to the left or the right until its outer edge touches the edge of its containing box, or another floated box
- **Often used for images** and when working with layouts

```
img  
{  
    float: right;  
}
```

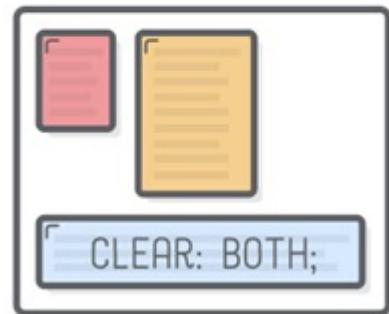
[http://www.w3schools.com/Css/css\\_float.asp](http://www.w3schools.com/Css/css_float.asp)



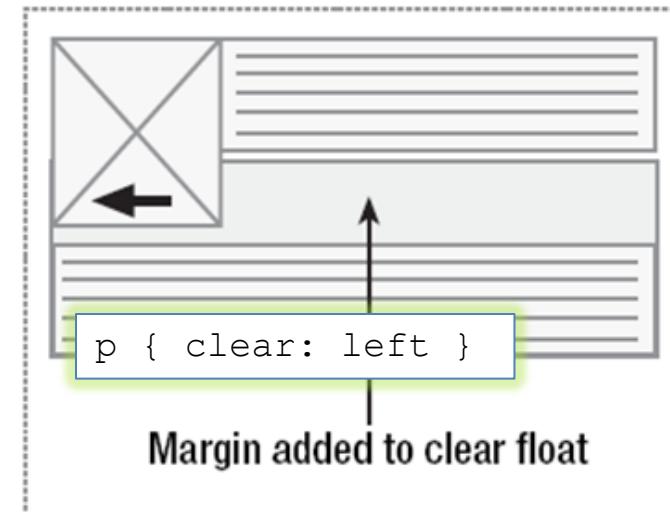
# Clearing floats

- “Clearing” a float: tell a block to ignore any floats that appear before it
  - Instead of flowing around, a cleared element appears after any floats
  - It’s like forcing a box back into the default vertical flow of the page

## CLEARING WITH CHILD ELEMENT

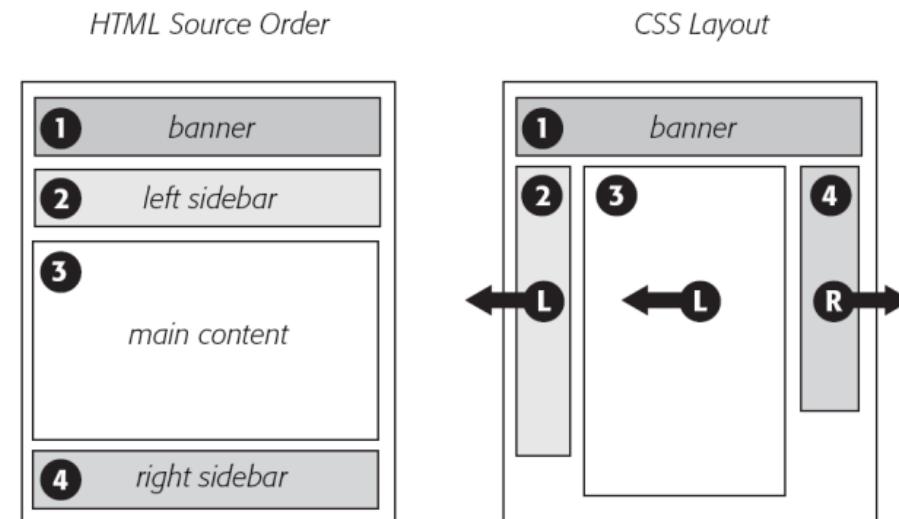


Second paragraph cleared



# Float-based layouts

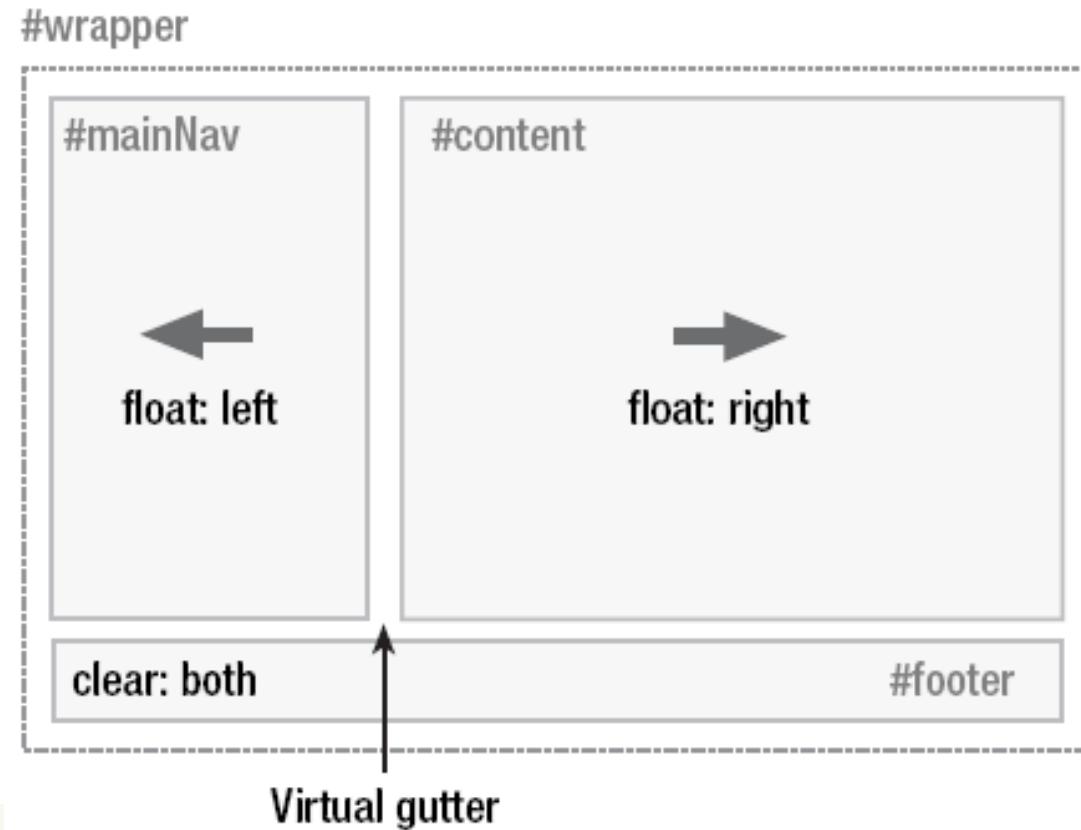
- Set the width of the elements you want to position, and then float them left or right
  - Two-column floated layout
  - Three-column floated layout



# Two-column floated layout

- Create a virtual gutter by floating one element left and one element right

```
<div id="wrapper">  
  <div id="branding">  
    ...  
  </div>  
  <div id="content">  
    ...  
  </div>  
  <div id="mainNav">  
    ...  
  </div>  
  <div id="footer">  
    ...  
  </div>  
</div>
```



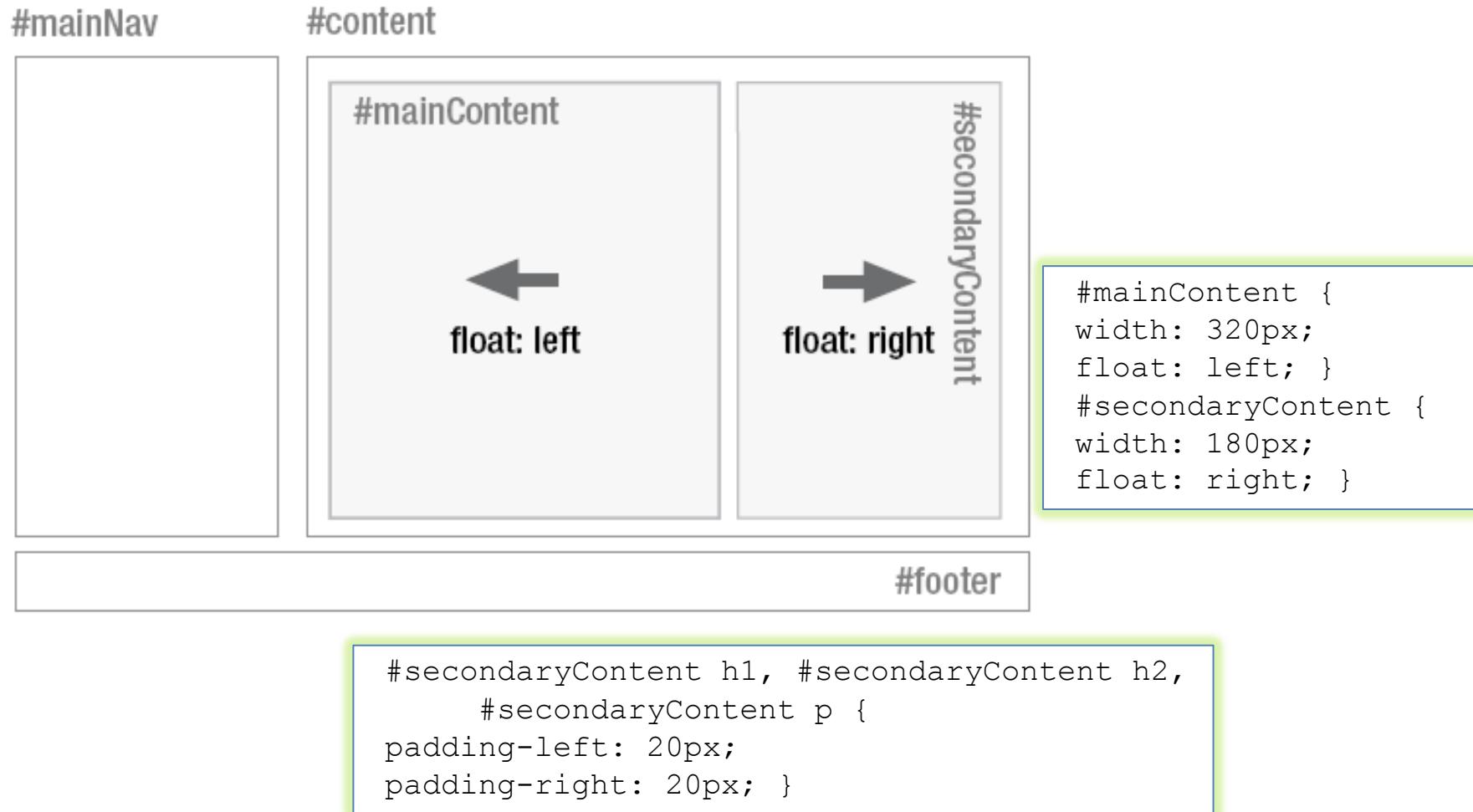
# Two-column floated layout

```
#content {  
width: 520px;  
float: right;  
}  
#mainNav {  
width: 180px;  
float: left;  
}  
#footer {  
clear: both;  
}
```

```
#mainNav {  
padding-top: 20px;  
padding-bottom: 20px;  
}  
#mainNav li {  
padding-left: 20px;  
padding-right: 20px;  
}  
#content h1, #content h2,  
#content p {  
padding-right: 20px;  
}
```

<https://blog.html.it/layoutgala/index.html>

# Three-column floated layout



# References for CSS box model and positioning

- Learn CSS layout
  - <http://learnlayout.com/>
- Floatutorial
  - <http://css.maxdesign.com.au/floatutorial/>
- All about floats
  - <https://css-tricks.com/all-about-floats/>

Cascading Style Sheets

# PAGE LAYOUT WITH GRIDS

# Advanced layout: grid

## Maki-zushi



The rice and seaweed rolls with fish and/or vegetables. There are also more specific terms for the rolls depending on the style.

## Nigiri-zushi



The little fingers of rice topped with wasabi and a filet of raw or cooked fish or shellfish. Generally the most common form of sushi you will see.

## Temaki-zushi



Also called a hand-roll. Cones of sushi rice, fish and vegetables wrapped in seaweed. It is very similar to maki.

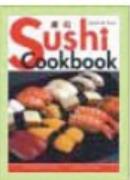
## WHAT IS SUSHI?

Beginning as a method of preserving fish centuries ago, sushi has evolved into an artful, unique dining experience. In its earliest form, dried fish was placed between two pieces of vinegared rice as a way of making it last. The nori (seaweed) was added later as a way to keep one's fingers from getting sticky.

## Sashimi

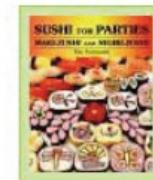


Sashimi is raw fish served sliced, but as-is. That means no rice bed or roll, but it is often served alongside daikon and/or shiso. This is my favorite style as you really get the flavor of the fish..



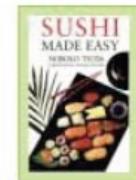
### QUICK & EASY SUSHI COOKBOOK

This book has great pictures, however it is not as complete as Sushi Made Easy.



### SUSHI FOR PARTIES: MAKI-ZUSHI AND NIGIRI-ZUSHI

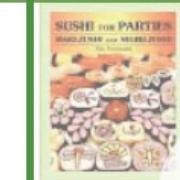
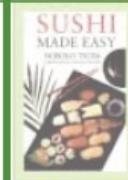
This book also has great pictures, with advanced maki (cut roll) making techniques.



### SUSHI MADE EASY

A very decent all-around book for the money.

# Advanced layout: grid

<p><b>Maki-zushi</b></p>  <p>The rice and seaweed rolls with fish and/or vegetables. There are also more specific terms for the rolls depending on the style.</p> <p><b>a</b></p>	<p><b>a</b></p>	<p><b>Nigiri-zushi</b></p>  <p>The little fingers of rice topped with wasabi and a filet of raw or cooked fish or shellfish. Generally the most common form of sushi you will see.</p>	<p><b>Temaki-zushi</b></p>  <p>Also called a hand-roll. Cones of sushi rice, fish and vegetables wrapped in seaweed. It is very similar to maki.</p> <p><b>d</b></p>
<p><b>WHAT IS SUSHI?</b></p>  <p>Beginning as a method of preserving fish centuries ago, sushi has evolved into an artful, unique dining experience. In its earliest form, dried fish was placed between two pieces of vinegared rice as a way of making it last. The nori (seaweed) was added later as a way to keep one's fingers from getting sticky.</p> <p><b>e</b></p>	<p><b>e</b></p>	<p>Technically, the word 'sushi' refers to the rice, but colloquially, the term is used to describe a finger-sized piece of raw fish or shellfish on a bed of vinegared rice or simply the consumption of raw fish in the Japanese style (while sushi is a solely a Japanese invention, these days, the Japanese style is considered the de facto serving standard).</p> <p><b>g</b></p>	
<p><b>Sashimi</b></p>  <p>Sashimi is raw fish served sliced, but as-is. That means no rice bed or roll, but it is often served alongside daikon and/or shiso. This is my favorite style as you really get the flavor of the fish..</p> <p><b>i</b></p>	 <p><b>j</b></p> <p><b>QUICK &amp; EASY SUSHI COOKBOOK</b> This book has great pictures, however it is not as complete as Sushi Made Easy.</p>	 <p><b>k</b></p> <p><b>SUSHI FOR PARTIES AND NIGIRI-ZUSHI</b> This book also has great pictures, with advanced maki (cut roll) making techniques.</p>	 <p><b>l</b></p> <p><b>SUSHI MADE EASY</b> A very decent all-around book for the money.</p>

# Advanced layout: grid

- It is possible to define a grid in which content can flow or be placed, or that remain empty
- There are 3 ways to define a grid
  - Explicit grid: defined with ‘`grid-columns`’ and ‘`grid-rows`’ properties
  - Natural grid: automatically created by elements with a natural grid structure (multi-column elements and tables)
  - Default grid: all other block elements define a single-cell grid

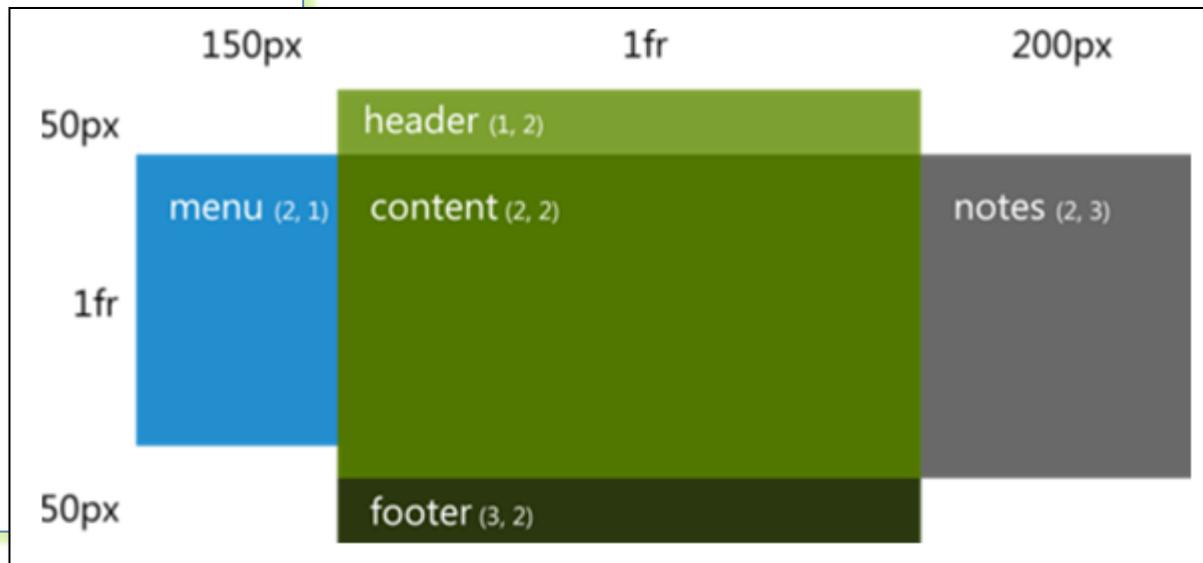
# Example

```
section {  
  display: grid;  
  grid-columns: 150px 1fr 200px;  
  grid-rows: 50px 1fr 50px; }  
section header {  
  grid-column: 2;  
  grid-row: 1; }  
section nav {  
  grid-column: 1;  
  grid-row: 2; }  
section article {  
  grid-column: 2;  
  grid-row: 2; }  
section aside {  
  grid-column: 3;  
  grid-row: 2; }  
section footer {  
  grid-column: 2;  
  grid-row: 3; }
```

```
<section>  
  <header>Title</header>  
  <nav>Menu</nav>  
  <article>Content</article>  
  <aside>Notes</aside>  
  <footer>Footer</footer>  
</section>
```

- **fr = fraction values**
  - new unit applicable to grid-rows and grid-columns properties

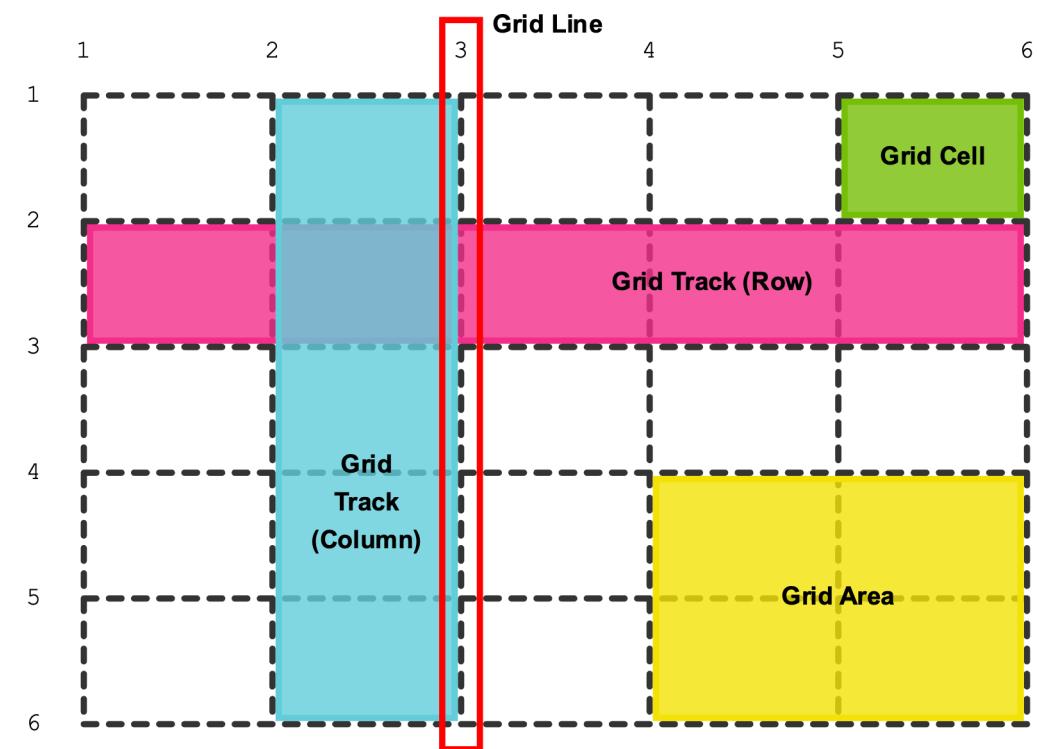
1fr is for 1 part of the available space



Span syntax (see next reference)

# Suggested reference for Grid Layout

- <https://webkit.org/blog/7434/css-grid-layout-a-new-layout-module-for-the-web/>



Cascading Style Sheets

# **CSS FLEXBOX**

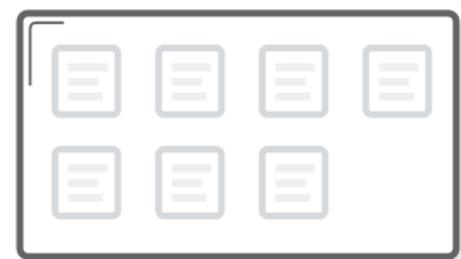
Suggested for labs and exams

# Flexbox

- Flexbox uses two types of boxes
  - Flex containers: group a set of flex items and define how they're positioned
  - Flex items
- Every HTML element that's a **direct** child of a flex container is an item

To turn one HTML elements into a flex container:

```
div { display: flex ; }
```



“FLEX CONTAINER”



“FLEX ITEMS”

<https://internetingishard.netlify.app/html-and-css/flexbox/>

# Flexbox



ALIGNMENT



DIRECTION



ORDER



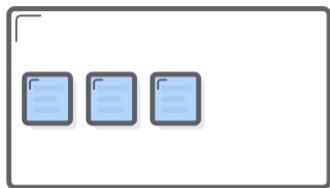
SIZE

<https://internetingishard.com/html-and-css/flexbox/>

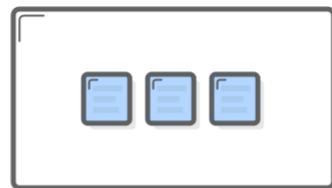
[https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)

# Horizontal alignment

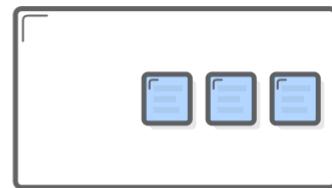
- “justify-content” property defines the horizontal alignment of its items
  - center, flex-start, flex-end
  - space-around, space-between



FLEX-START



CENTER

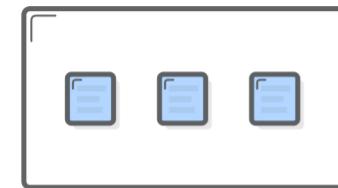


FLEX-END

```
<div class='container'>
  <div class='item'>A</div>
  <div class='item'>B</div>
  <div class='item'>C</div>
</div>
```

```
.menu-container {
  /* ... */
  display: flex;
  justify-content: center;
}
```

div behaves as inline



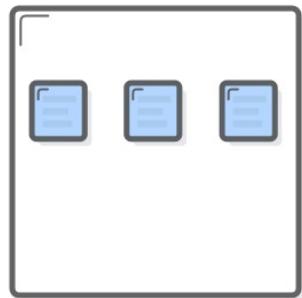
SPACE-AROUND



SPACE-BETWEEN

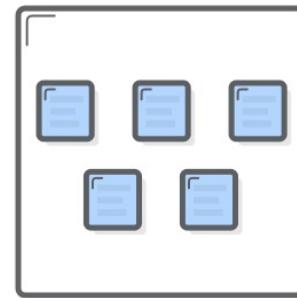
# Wrapping

- The **flex-wrap** property creates a grid
  - Then, you can change alignment, direction, order, and size of items



**NO WRAPPING**

FLEX-WRAP: NOWRAP;



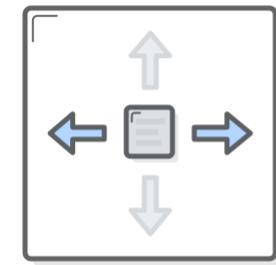
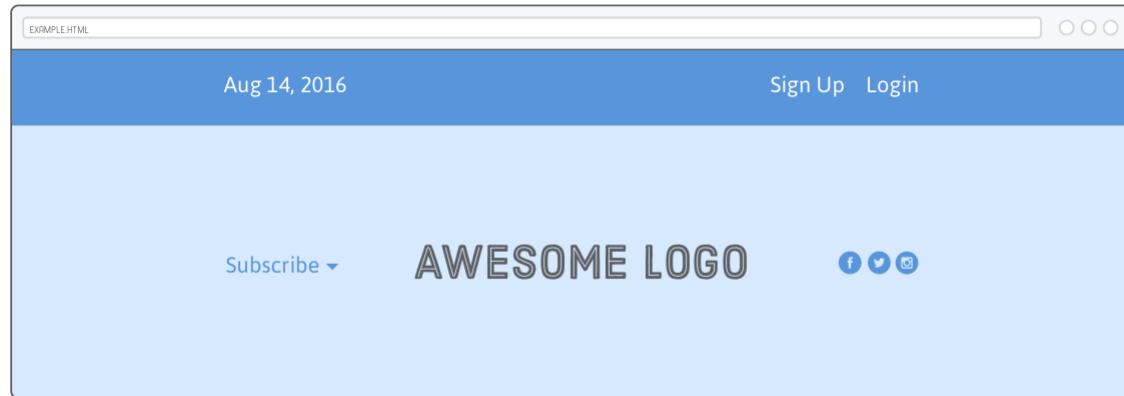
**WITH WRAPPING**

FLEX-WRAP: WRAP;

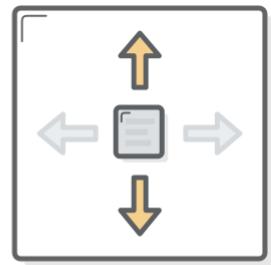
```
.photo-grid {  
    width: 900px;  
    display: flex;  
    justify-content: center;  
    flex-wrap: wrap;  
}
```

# Vertical alignment

- Flex containers can also define the vertical alignment of their items



JUSTIFY-CONTENT

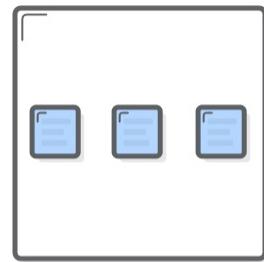


ALIGN-ITEMS

```
.header {  
    width: 900px;  
    height: 300px;  
    display: flex;  
    justify-content: space-between;  
    align-items: center;  
}
```

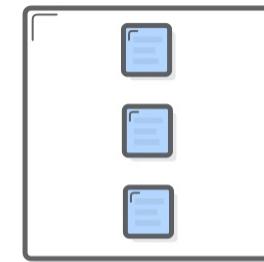
# Direction

- Refers to whether a container renders its items horizontally or vertically



**ROW**

FLEX-DIRECTION: ROW;

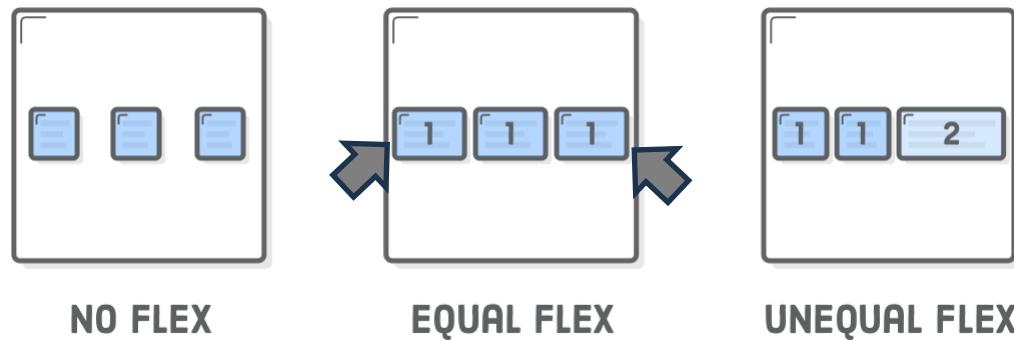


**COLUMN**

FLEX-DIRECTION: COLUMN;

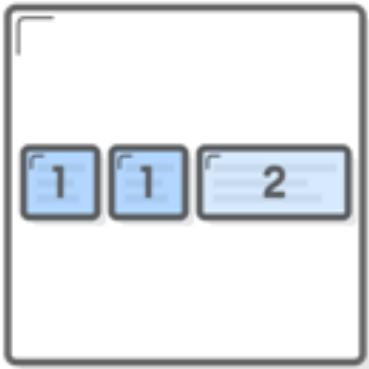
# Flexible items

- Flex items are flexible: they can shrink and stretch **to match the width of their containers**
- The **flex** property defines the width of individual items in a flex container
  - a **weight** that tells the flex container how to distribute extra space to each item
  - E.g., an item with a flex value of 2 will grow twice as fast as items with the default value of 1



# Flexible items

## Example



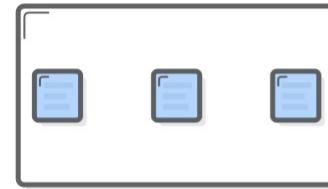
```
.footer {  
    display: flex;  
    justify-content: space-between;  
}  
  
.footer-item {  
    border: 1px solid #fff;  
    background-color: #D6E9FE;  
    height: 200px;  
    flex: 1; }  
  
.footer-three { flex: 2; }
```

```
<div class='footer'>  
    <div class='footer-item footer-one'></div>  
    <div class='footer-item footer-two'></div>  
    <div class='footer-item footer-three'></div>  
</div>
```

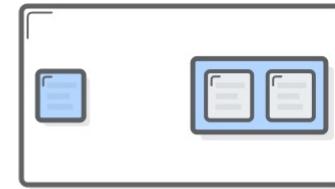


# Grouping

- Flex containers only know how to position elements that are **one level deep** (i.e., their child elements)
  - You can **group flex items using <div>**



**NO GROUPING**  
(3 FLEX ITEMS)



**GROUPED ITEMS**  
(2 FLEX ITEMS)

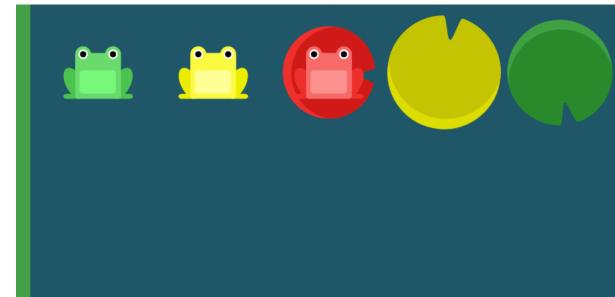


# Summary of CSS flexbox

- `display: flex` to create a flex container
- `justify-content` to define the horizontal alignment of items
- `align-items` to define the vertical alignment of items
- `flex-direction` if you need columns instead of rows
- `row-reverse` or `column-reverse` values to flip item order
- `order` to customize the order of individual elements
- `align-self` to vertically align individual items
- `flex` to create flexible boxes that can stretch and shrink

# References for CSS flexbox

- Interneting is hard flexbox tutorial
  - <https://internetingishard.com/html-and-css/flexbox/>
- A complete guide to flexbox
  - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
- W3schools
  - [https://www.w3schools.com/css/css3\\_flexbox.asp](https://www.w3schools.com/css/css3_flexbox.asp)
- Flexbox, practical guide (in Italian)
  - <http://www.html.it/guide/flexbox-guida-pratica/>
- Flexbox Froggy (a game-like tutorial)
  - <http://flexboxfroggy.com/>



Cascading Style Sheets

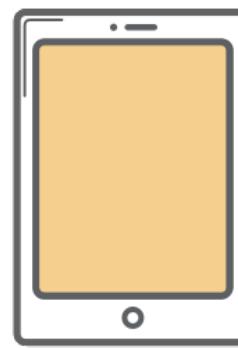
# **RESPONSIVE LAYOUT**

# Responsive design

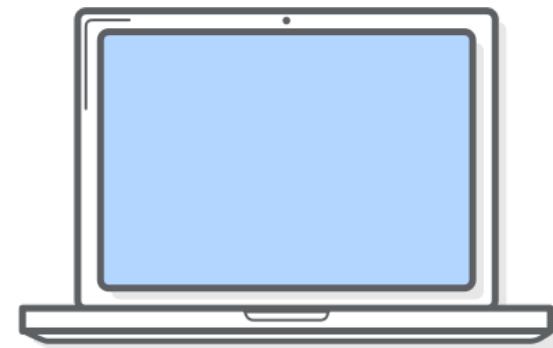
- Display well in everything from widescreen monitors to mobile phones
- Approach to web design to eliminate the distinction between the mobile-friendly version of your website and its desktop counterpart



MOBILE



TABLET



DESKTOP

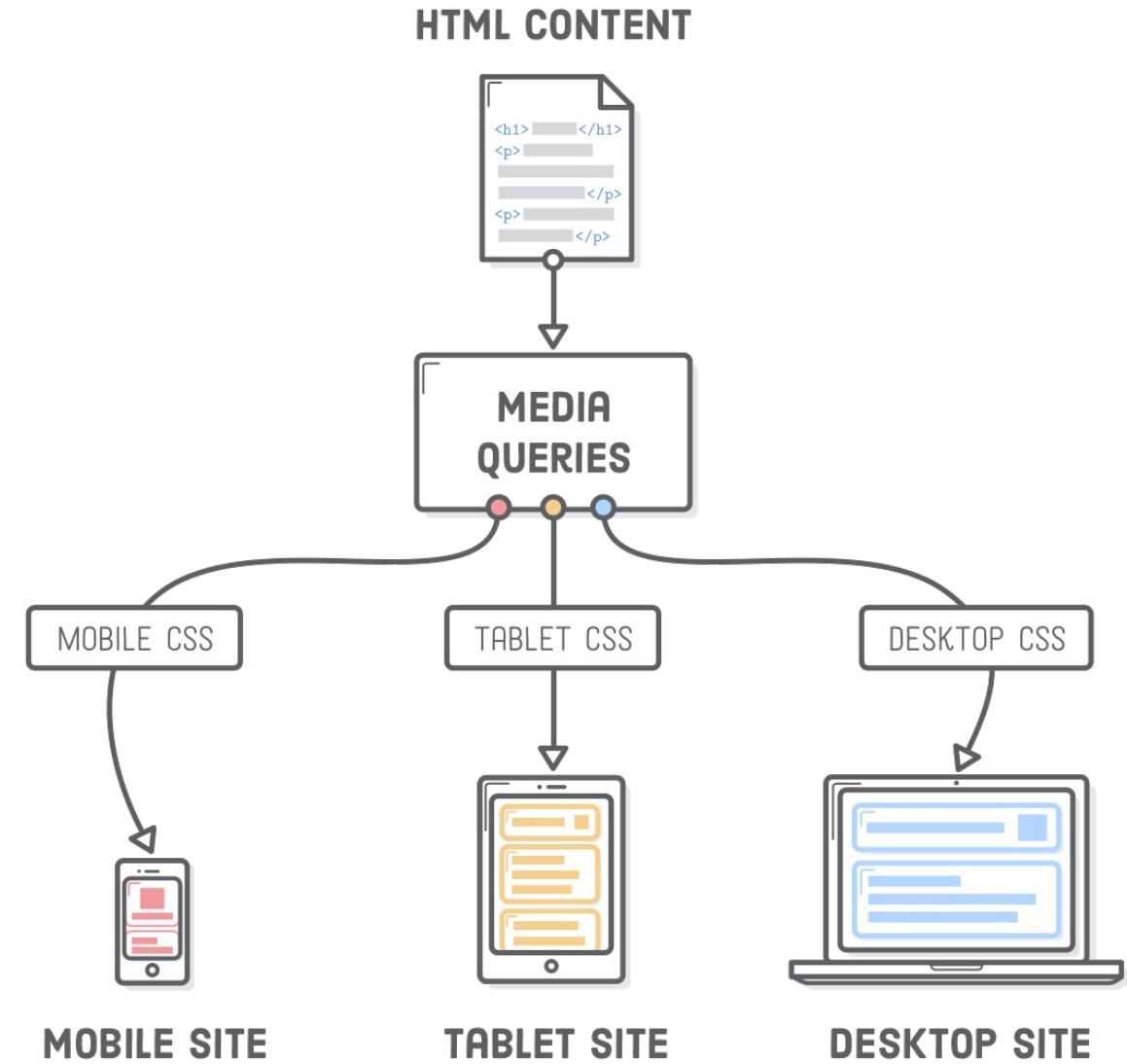
<https://internetingishard.com/html-and-css/responsive-design/>

# Responsive design

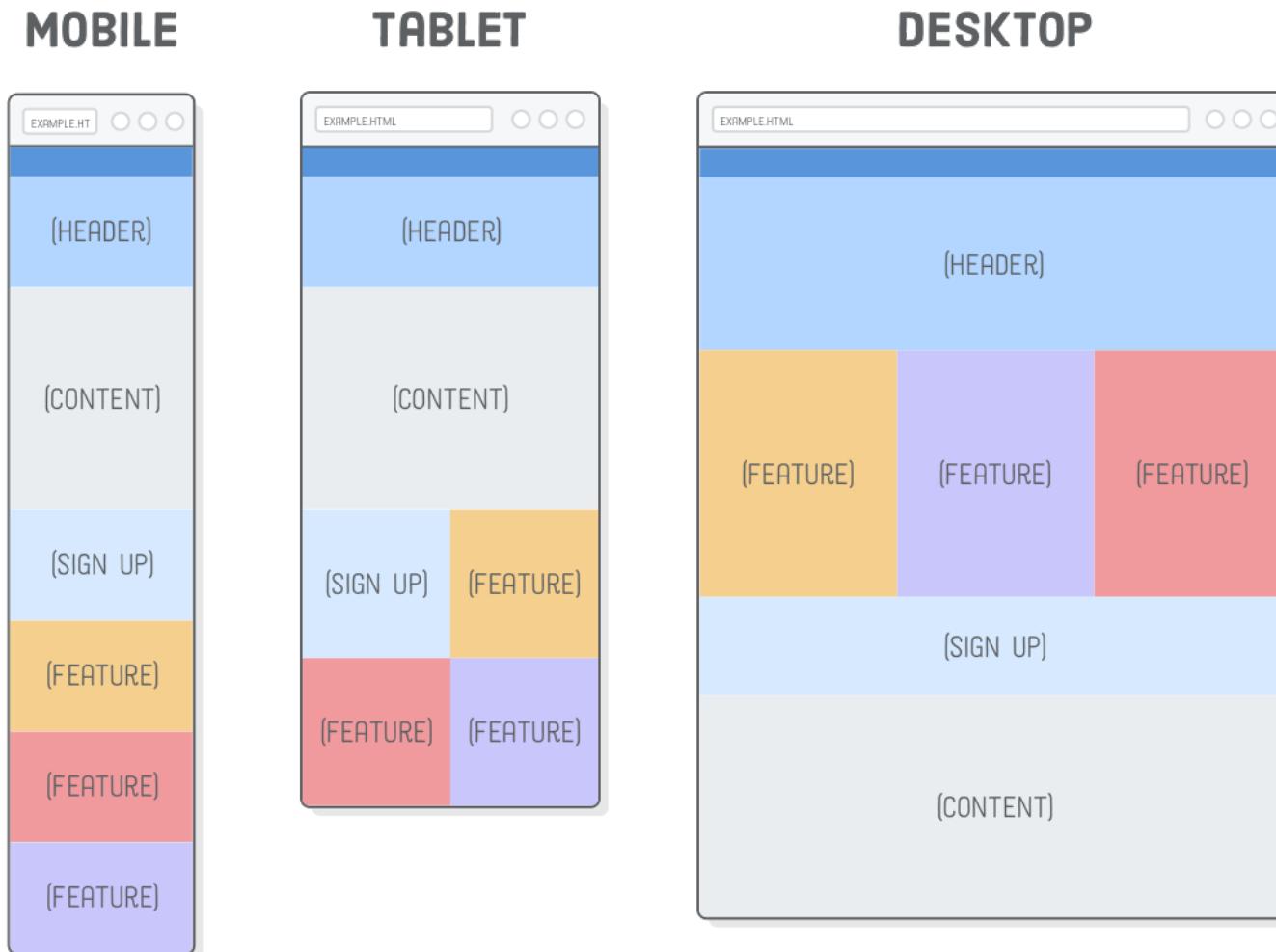
- Responsive design is accomplished through CSS “media queries”
  - A way to conditionally apply CSS rules

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

↑  
Media query announcement  
↑  
What circumstance should this query be “turned on” or applied  
↑  
What it should do if the circumstance happens



# Layout for responsive design



# CSS Frameworks

- Set of templates to simplify web development
- Example: Bootstrap
  - Open Source CSS (and Javascript) framework
  - Allows applying “modern” styles with sensible and nice-looking defaults
    - Many ready-to-use UI elements (e.g., buttons, menus, tabs, collapsible items, ecc.)
  - Takes care of cross-browser issues
  - Simplified layout model
  - Developed by Twitter
    - <https://getbootstrap.com/docs/5.2/getting-started/introduction/>

# Bootstrap - intro

- Front-end framework that provides a collection of HTML, CSS and (optionally) JavaScript-based design templates that enable developers to build responsive, mobile-first websites with ease.
- Class Nomenclature
  - Base (common styles)
    - class="accordion", class="btn"
  - + Modifier (additional styles)
    - class="accordion accordion-flush", class="btn btn-primary"

Layout  
Content  
Forms  
Components  
Helpers/Utilities

# Bootstrap - quickstart

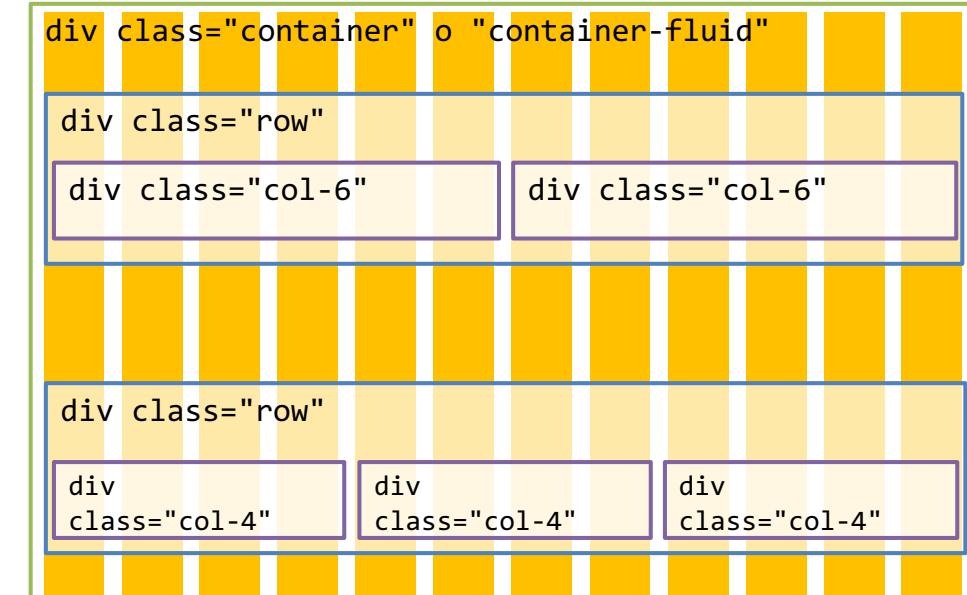
- Include additional CSS and JS
  - <link rel="stylesheet" href="...">
  - <script src="..."></script>

<https://getbootstrap.com/docs/5.3/getting-started/introduction/>

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-KDRAoPfGtBQ4qZB+WEVxL9W7X8l1Tnyv1qAABq3E4J17HnOOGiJYdIuHw==" crossorigin="anonymous"></script>
  </body>
</html>
```

# Bootstrap – grid-system

- Many web pages are based on a **grid-system**, i.e., the page is divided into columns
- Bootstrap **grid-system** has 12 columns
  - Applied to element with class “container” that contains
  - Rows as class “row”
  - Columns as class “col-Y”
    - The sum of Ys is 12



# Bootstrap – spacing

- A wide range of shorthand class names for margin, padding, and gap
- Notation: {property}{sides}-{size}
  - Properties: **m** for margin, **p** for padding
  - Sides: t for top, b for bottom, s (start) for left LTR (right RTL), e (end), x both start-end, y both top-bottom, or *blank* for all 4 sides
  - Sizes: 0-5 or auto for 0, 0.25rem, 0.5rem, 1rem, 1.5rem, 3rem and margin-auto

# Bootstrap – table

- Customize tables with table-\* classes
  - e.g. table table-striped / table-hover, ecc.

## Hoverable rows

Add `.table-hover` to enable a hover state on table rows within a `<tbody>`.

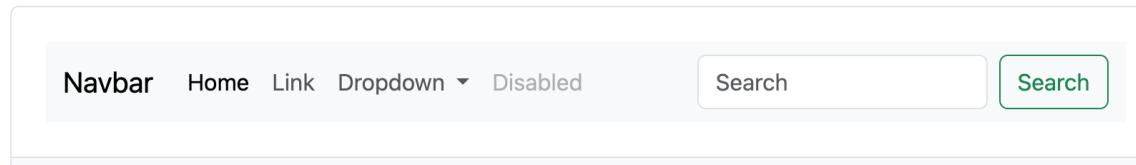
#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

```
<table class="table table-hover">
  ...
</table>
```



# Bootstrap – Components

- Component: enhanced HTML element
  - With style – custom CSS
  - With behavior – custom JS



Accordion Item #1

This is the first item's accordion body. It is shown by default, until the collapse plugin adds the appropriate classes that we use to style each element. These classes control the overall appearance, as well as the showing and hiding via CSS transitions. You can modify any of this with custom CSS or overriding our default variables. It's also worth noting that just about any HTML can go within the `.accordion-body`, though the transition does limit overflow.

Accordion Item #2

Accordion Item #3

A screenshot of an Accordion component. It consists of three items: "Accordion Item #1" (which is expanded, showing its content), "Accordion Item #2" (which is collapsed, indicated by a downward arrow icon), and "Accordion Item #3" (which is also collapsed, indicated by a downward arrow icon). Each item has a blue header and a white body containing text.

See Examples > Header

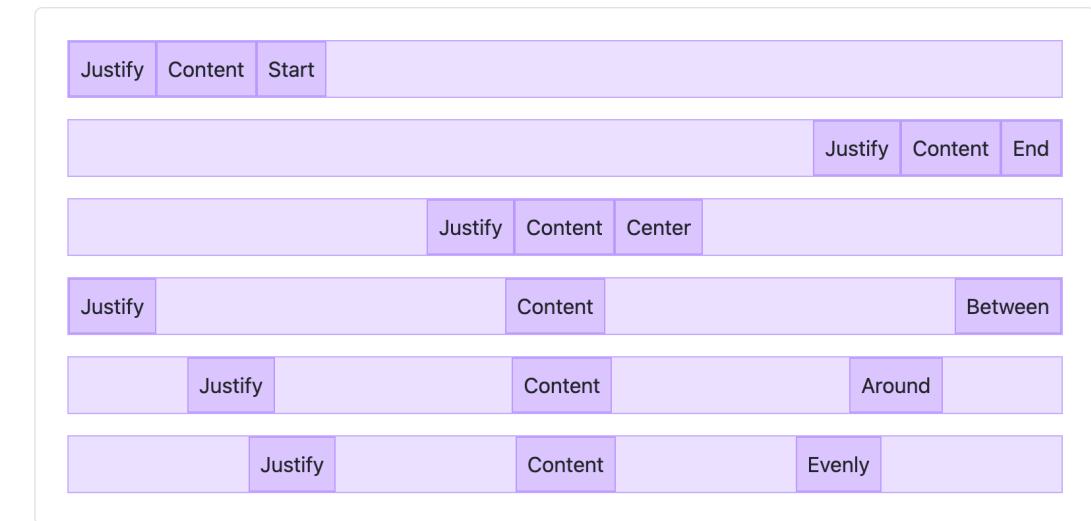
# Bootstrap – Utilities / Helpers

- Classes to quickly style elements without using any CSS code
- E.g. enable flex behaviors
  - Apply *display utilities (d-flex class)* to create a flexbox container and transform direct children elements into flex items.

```
<div class="d-flex justify-content-start">
  <div class="p-2">Justify</div>
  <div class="p-2">Content</div>
  <div class="p-2">Start</div>
</div>
```

## Justify content

Use `justify-content` utilities on flexbox containers to change the alignment of flex items on the main axis (the x-axis to start, y-axis if `flex-direction: column`). Choose from `start` (browser default), `end`, `center`, `between`, `around`, or `evenly`.



# Bootstrap – icons

- Import as fonts  
`bootstrap-icons.min.css`

## CDN

Include the icon fonts stylesheet—in your website `<head>` or via `@import` in CSS—from jsDelivr and get started in seconds. [See icon font docs](#) for examples.

```
<link rel="stylesheet" href="https://
```

## Icon font

Icon fonts with classes for every icon are also included for Bootstrap Icons. Include the icon web fonts in your page via CSS, then reference the class names as needed in your HTML (e.g., `<i class="bi-alarm-clock"></i>`).

Use `font-size` and `color` to change the icon appearance.

The image shows two code snippets side-by-side, each consisting of a small icon followed by its corresponding HTML code. The first snippet shows a small grey alarm clock icon with the code `<i class="bi-alarm"></i>`. The second snippet shows a larger blue alarm clock icon with the code `<i class="bi-alarm" style="font-size: 2rem; color: cornflowerblue;"></i>`. Both snippets include a small edit icon at the end.

```
<i class="bi-alarm"></i>
```

```
<i class="bi-alarm" style="font-size: 2rem; color: cornflowerblue;"></i>
```

# Importing CSS Frameworks: Security Issues

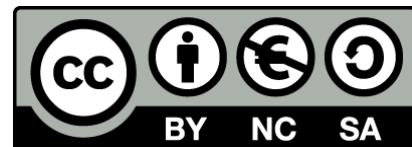
- CSS frameworks often comprise both CSS and Javascript code
  - Same considerations apply as for code packages: security, availability, etc.
- CSS rules and Javascript code are loaded and run into the web application
  - Make sure they are either served by a trusted web server or
  - Loaded from an external source (CDN) but verified for integrity before execution

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
      integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pN1yT2bRjXh0JMhjY6hW+ALEwIH"
      rel="stylesheet" crossorigin="anonymous">
```

- NOTE: This is exactly the same issue as seen for the JS packages

# References

- CSS First Steps:  
[https://developer.mozilla.org/en-US/docs/Learn/CSS/First\\_steps](https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps)
- Grid and Flexbox:  
<https://ishadeed.com/article/grid-layout-flexbox-components/>



# License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- <https://creativecommons.org/licenses/by-nc-sa/4.0/>