Lezione11-CSS

Thursday, March 18, 2021 9:47 AM

<WA1/>
<AW1/>
2021

Cascading Style Sheets

Styling the Web

Fulvio Corno
Luigi De Russis
Enrico Masala

Some slides adapted from Laura Farinetti

POLITECNICO DI TORINO

Goal

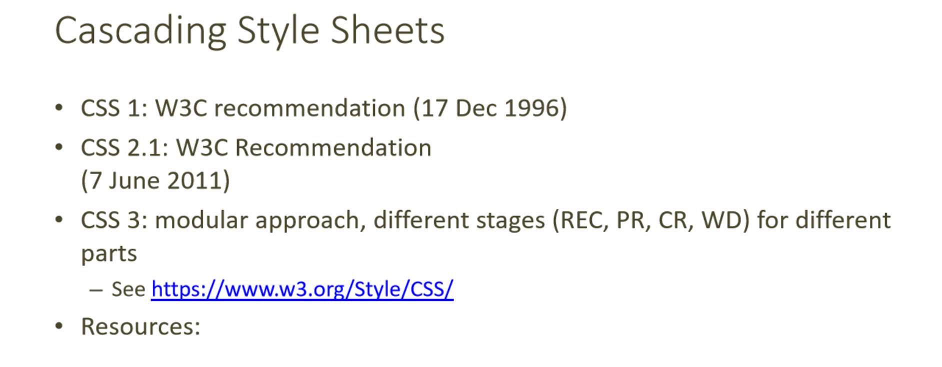
- Styling web content - CSS
- Advanced layout in web pages
- Responsive layouts
- Libraries

Much more complex than html (many corner cases, options, behaviours..)
We will use bootstrap library to semplify stuff instead of writing css by hand (long and many details browser dependant!)

Outline

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

Same rules may be applied to different elements.



Cascading Style Sheets

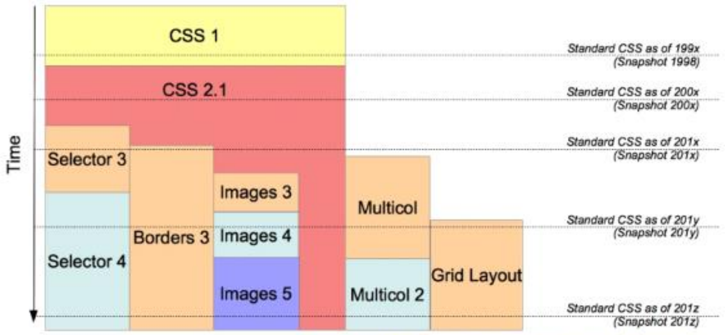
- CSS 1: W3C recommendation (17 Dec 1996)
- CSS 2.1: W3C Recommendation (7 June 2011)
- CSS 3: modular approach, different stages (REC, PR, CR, WD) for different parts
  - See https://www.w3.org/Style/CSS/
- Resources:

Same rules may be applied to different elements.

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


Time
CSS 1
CSS 2.1
Selector 3
Selector 4
Borders 3
Images 3
Images 4
Images 5
Multicol
Multicol 2
Grid Layout
Standard CSS as of 199x (Snapshot 1998)
Standard CSS as of 200x (Snapshot 200x)
Standard CSS as of 201x (Snapshot 201x)
Standard CSS as of 201y (Snapshot 201y)
Standard CSS as of 201z (Snapshot 201z)

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

New techniques added every year. Simple sintax, the properties are added, with various complex behaviours

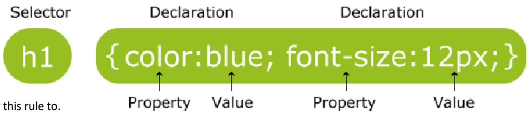
Cascading Style Sheets

CSS SYNTAX

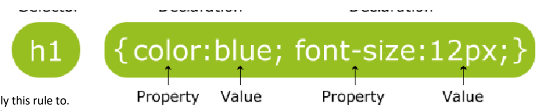
CSS Syntax

Only 1 type of instruction: rules

- 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



Selects 0,1,many elements to apply this rule to.

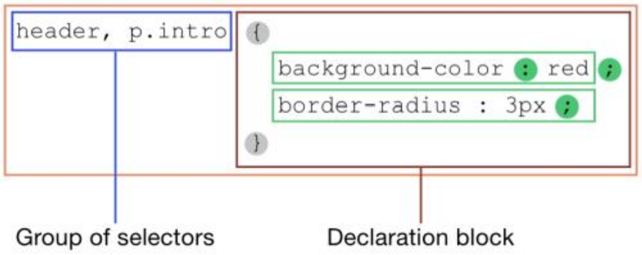


Selects 0,1,many elements to apply this rule to.
Different type of selector with different algorithms to
Select the various elements

Declaration is applied on the selected elements! Each element has MANY properties, each with a different meaning! We can customize them all! (f12, dom ,browser styles)
Very powerful! But difficult to understand all the properties and how they interact!

General syntax

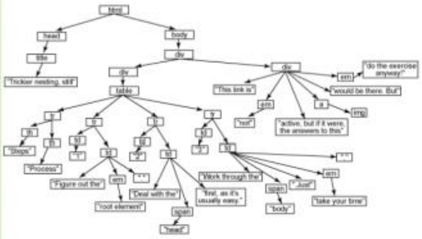
All elements called header AND all paragraphs which have class intro



Many rules influence also the children of that element! (cascade)
Some may not, especially if the child has a rule that overwrites the parent rule!

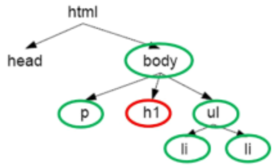
HTML nested structure

```
<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 it it were, the  answer to this
<a><img src="exercise.jpg"></a> would be there. But <em>do the exercise
anyway!</em>
    </div>
  </div>
</body>
</html>
```

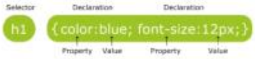


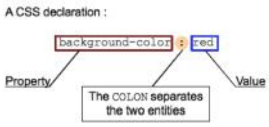
Tree structure and inheritance

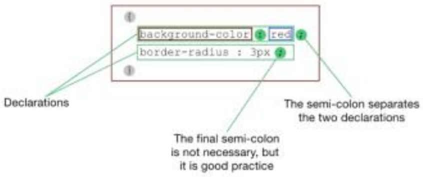
- HTML documents are trees
- Styles are inherited along trees
- When two rules are in conflict the most specific wins
- Example
  - body {color: green}
  - h1 {color: red}



Declaration Syntax






background-color
red
border-radius : 3px
Declarations
The semi-colon separates the two declarations
The final semi-colon is not necessary, but it is good practice

Too many!

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?

Width, eight, spacing, borders, margins, expressed in pixels, line length..

Left, center, right: strings with certain meanings

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-bottom-width 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

Border-right to specify the property of the right border of a block
Border-right-(color, style, width) is more specific than border-right!

https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

For each color element contains All possible values of a color
5 specifications for each property:
<meaning>: what it does
<values>: types of values that it contains
<initial value> : default value
<applies to> to which element I can apply that property
<inherited> : if father has that property, also children has it?

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

Different between text (paragraph formatting:alignment,..) and font (letters, characters formatting: font size, color, ..)

CSS Units: Most Used

- CSS has several different **units** for expressing a length
  - format: a number followed by a unit (e.g., 10px)
  - width, font-size, margin, padding, …
- 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)

CSS Units: Most Used

- 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 parent element

Em="m"

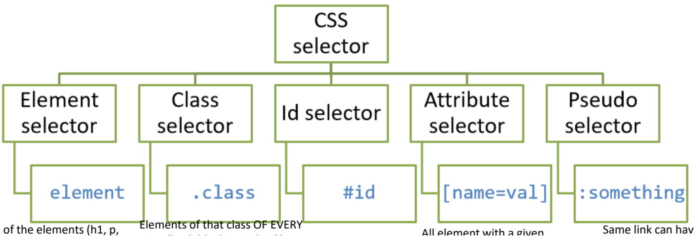
- Suggestions:
  - prefer relative units to absolute ones, when possible
  - rem is (nowadays) preferred to em

Cascading Style Sheets

CSS SELECTORS

CSS selectors

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

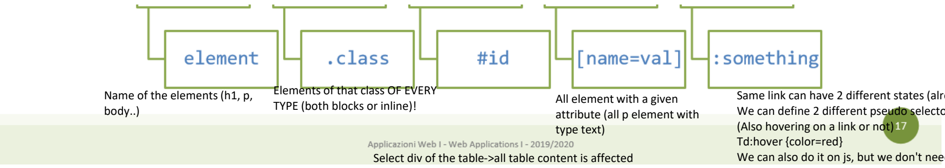


Name of the elements (h1, p,

Elements of that class OF EVERY

All element with a given

Same link can have 2 different states (already clicked or not)



Name of the elements (h1, p, body..)

Elements of that class OF EVERY TYPE (both blocks or inline)!

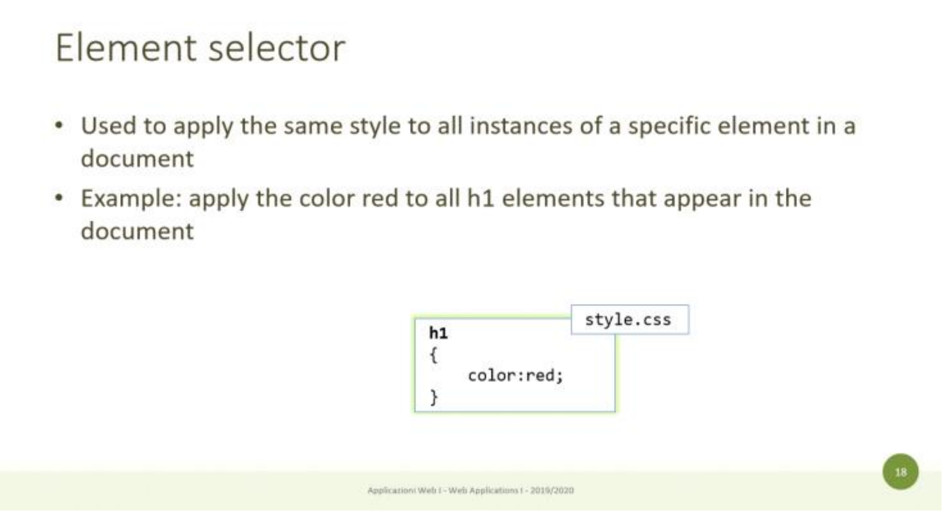
All element with a given attribute (all p element with type text)

Same link can have 2 different states (already clicked or not)
We can define 2 different pseudo selector!
(Also hovering on a link or not)
Td:hover {color=red}
We can also do it on js, but we don't need to since in css is FOR FREE (done by the browser)

Select div of the table->all table content is affected

Applicazioni Web I - Web Applications I - 2019/2020

17



Element selector

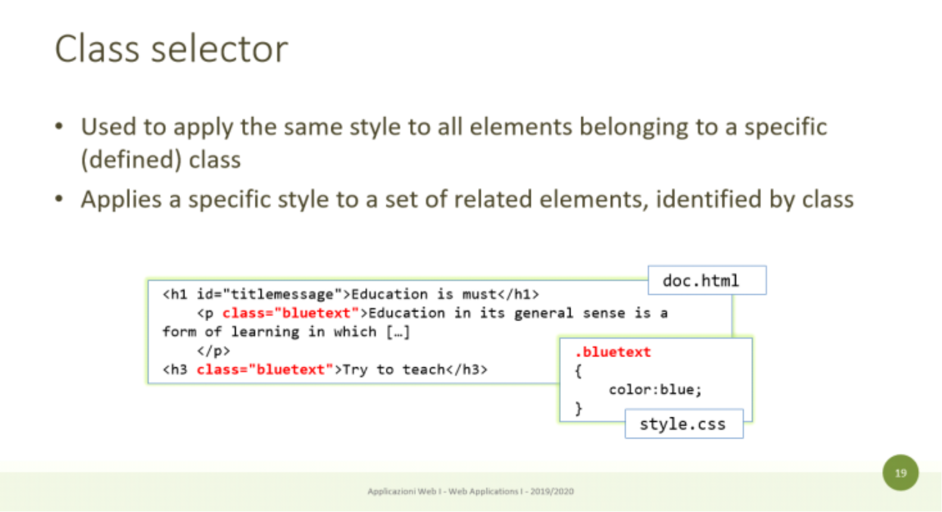
- Used to apply the same style to all instances of a specific element in a document
- Example: apply the color red to all h1 elements that appear in the document

style.css

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

Applicazioni Web I - Web Applications I - 2019/2020

18



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

doc.html

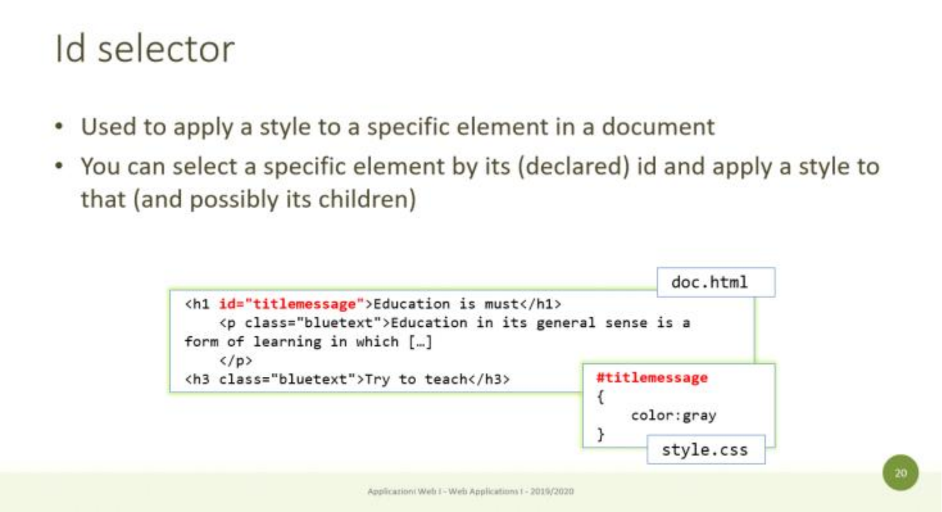
```
<h1 id="titlemessage">Education is must</h1>
    <p class="bluetext">Education in its general sense is a
form of learning in which […]
    </p>
<h3 class="bluetext">Try to teach</h3>
```

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

style.css

Applicazioni Web I - Web Applications I - 2019/2020

19



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)

doc.html

```
<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>
```

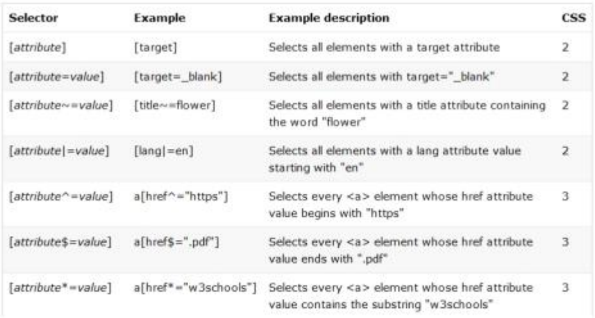
```
#titlemessage
{
    color:gray
}
```

style.css

Applicazioni Web I - Web Applications I - 2019/2020

20

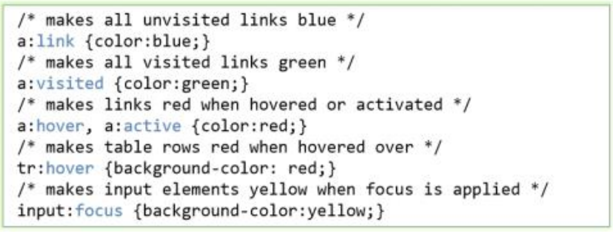
Attribute selectors



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



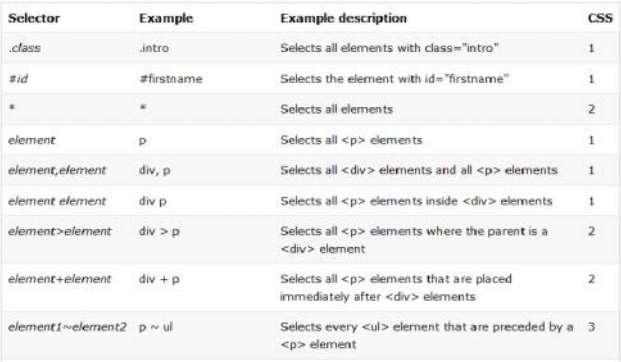
```
/* makes all unvisited links blue */
a:link {color:blue;}
/* makes all visited links green */
a:visited {color:green;}
/* makes links red when hovered or activated */
a:hover, a:active {color:red;}
/* makes table rows red when hovered over */
tr:hover {background-color: red;}
/* makes input elements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

Combining selectors

- `element.class#id[n=v]`
  → may be combined
- `S1, S2` → S1 union S2
- `S1 S2` → S2 nested within S1
- `S1 > S2` → S2 if a child of S1 Immediately
- `S1 + S2` → S2 if it comes after a S1
- `S1 ~ S2` → S2 if it comes later than S1

Header h1 {…}
(h1 inside an header) : select element by describing its position in the dom tree of the page!

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

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;}
```

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

We can change the default behaviour of the elements from block to inline or viceversa

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, and the page will be displayed as if the element is not there

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

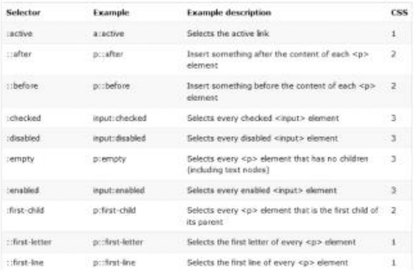
Easily make parts of the page disappear! (It is still there!)

- 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

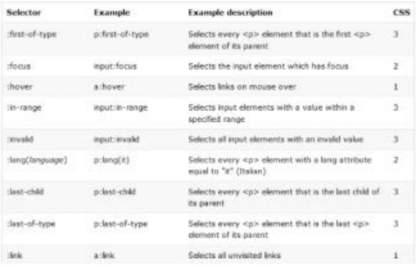
We can prepare placeholders to later insert real content when it's ready/needed

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

CSS pseudo-class selectors

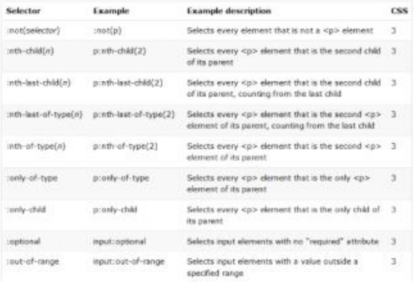


Selector	Example	Example description	CSS
:active	a:active	Selects the active link	1
::after	p::after	Insert something after the content of each <p> element	2
::before	p::before	Insert something before the content of each <p> element	2
:checked	input:checked	Selects every checked <input> element	3
:disabled	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

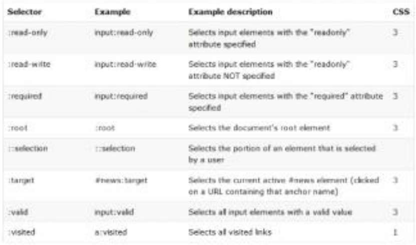


Selector	Example	Example description	CSS
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent	3
:focus	input:focus	Selects the input element which has focus	2
:hover	a:hover	Selects links on mouse over	1
:in-range	input:in-range	Selects input elements with a value within a specified range	3
:invalid	input:invalid	Selects all input elements with an invalid value	3
:lang(language)	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



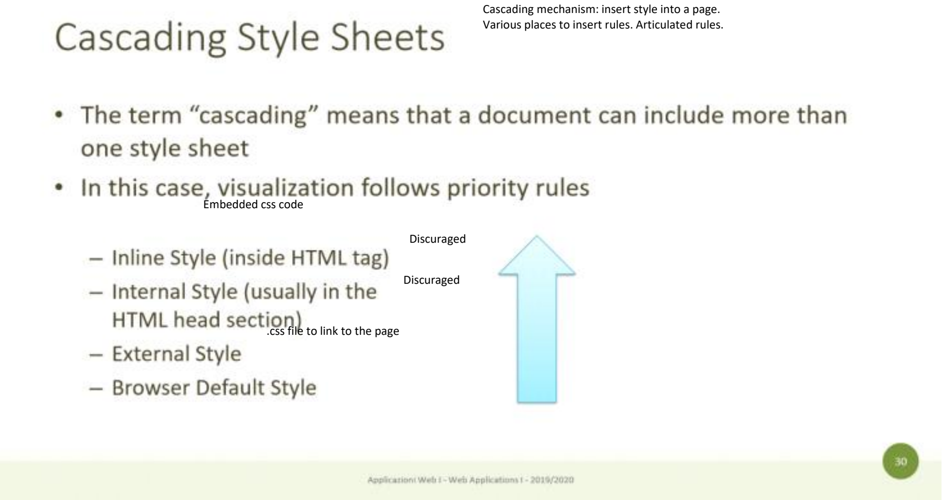
Selector	Example	Example description	CSS
:not(*selector*)	:not(p)	Selects every element that is not a <p> element	3
:nth-child(*n*)	p:nth-child(2)	Selects every <p> element that is the second child of its parent	3
:nth-last-child(*n*)	p:nth-last-child(2)	Selects every <p> element that is the second child 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-wite	input:read-wite	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

Cascading Style Sheets

CSS CASCADING



Cascading Style Sheets

Cascading mechanism: insert style into a page.
Various places to insert rules. Articulated rules.

- The term “cascading” means that a document can include more than one style sheet
- In this case, visualization follows priority rules

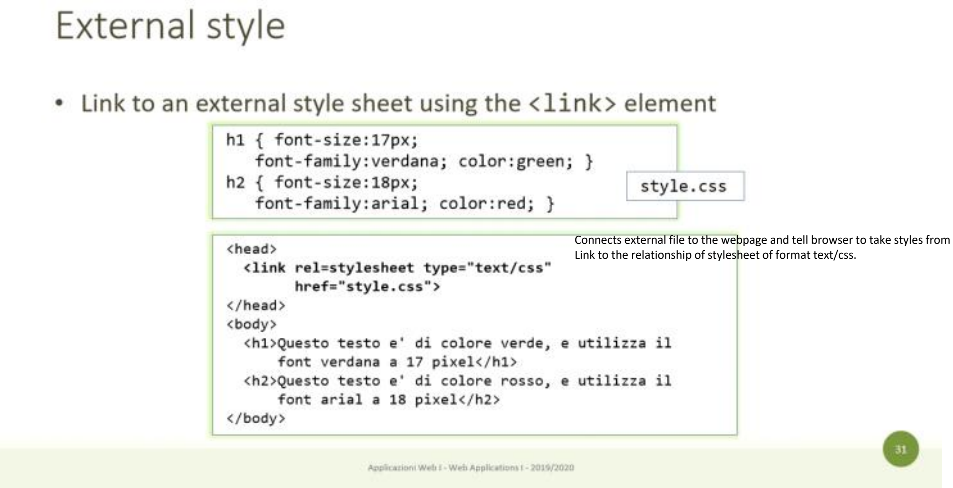
Embedded css code

  - Inline Style (inside HTML tag)

    Discuraged
  - Internal Style (usually in the HTML head section)

    Discuraged

    .css file to link to the page
  - External Style
  - Browser Default Style



External style

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

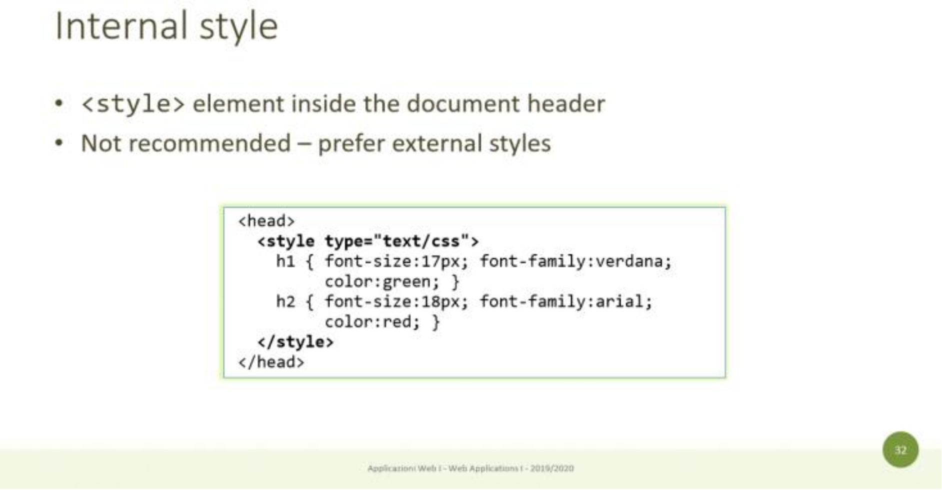
```
h1 { font-size:17px;
   font-family:verdana; color:green; }
h2 { font-size:18px;
   font-family:arial; color:red; }
```

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>
```

Connects external file to the webpage and tell browser to take styles from there!
Link to the relationship of stylesheet of format text/css.

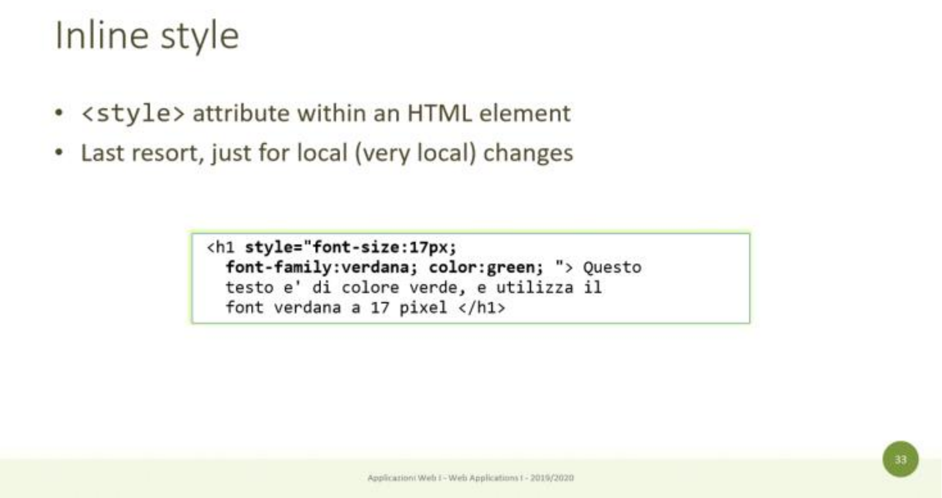
.css
Body{
Background color = …
}
Applies property background color=… to the elements selected by body



Internal style

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

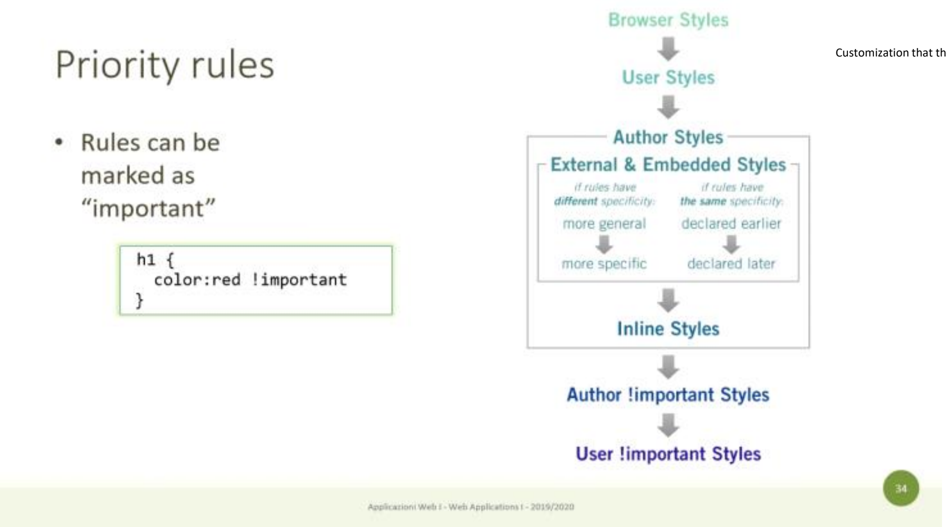
```
<head>
  <style type="text/css">
    h1 { font-size:17px; font-family:verdana;
         color:green; }
    h2 { font-size:18px; font-family:arial;
         color:red; }
  </style>
</head>
```



Inline style

- `<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
}
```

Browser Styles

User Styles

Author Styles

External & Embedded Styles

if rules have *different* specificity: more general → more specific

if rules have *the same* specificity: declared earlier → declared later

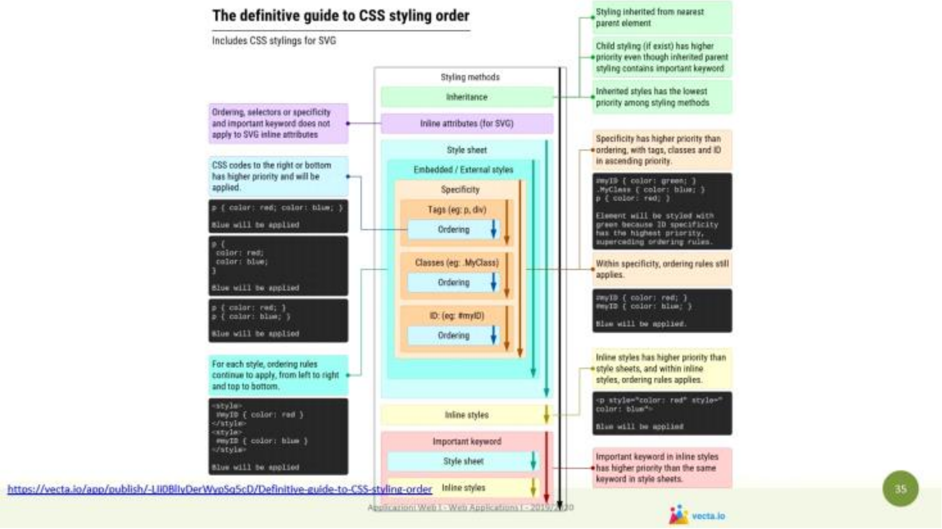
Inline Styles

Author !important Styles

User !important Styles

Customization that the user can do to their browser

Overwrite some behaviours



The definitive guide to CSS styling order

Includes CSS stylings for SVG

Styling methods

Inheritance

Inline attributes (for SVG)

Style sheet

Embedded / External styles

Specificity

Tags (eg: p, div)

Ordering

Classes (eg: .MyClass)

Ordering

ID: (eg: #myID)

Ordering

Inline styles

Important keyword

Style sheet

Inline styles

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.

vecta.io

https://vecta.io/app/publish/-LlI0BllyDerWypSq5cD/Definitive-guide-to-CSS-styling-order

We try to keep it sample!
Today we have a flavour, but then we'll use a library for the dirty work!

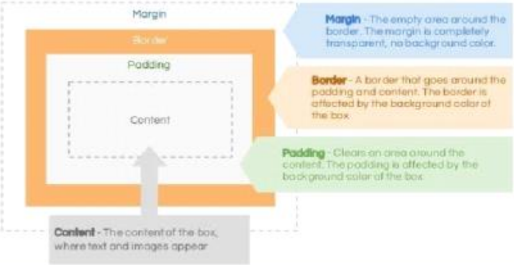
Cascading Style Sheets

CSS BOX MODEL

The box model

Reason on the space that each object occupies

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



Margin - The empty area around the border. The margin is completely transparent, no background color.

Border - A border that goes around the padding and content. The border is affected by the background color of the box

Padding - Clears an area around the content. The padding is affected by the background color of the box

Content - The content of the box, where text and images appear

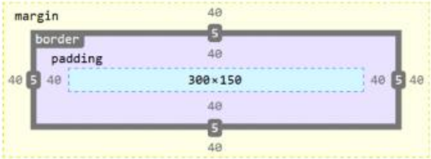
We start from the CONTENT ("a") (image) (table) … that occupies the Private space of the element
BORDER: of 0px or more (visible) (thin, thick) can change color
PADDING: space inside the element to avoid it touching the border (same color of the border)
MARGIN: space outside the border to avoid it touching other elements (same color of the outside)

Can be changed directly in the inspector, but these modification don't survive in the real css (useful for testing)

H1 has default margin of the browser, if we don't like it we redefine it!

The box model

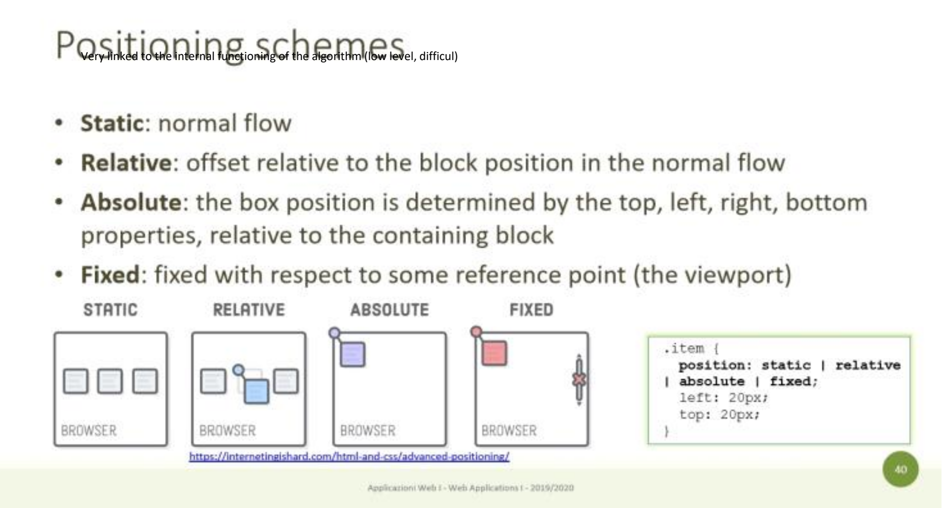
content
- Total element width =
  width + left padding + right padding + left border + right border + left margin + right margin
- Total element height =
  height + top padding + bottom padding + top border + bottom border + top margin + bottom margin
- You can set any of these properties, independently



Cascading Style Sheets

CSS POSITIONING SCHEMES

Long history of layouts of css



Positioning schemes

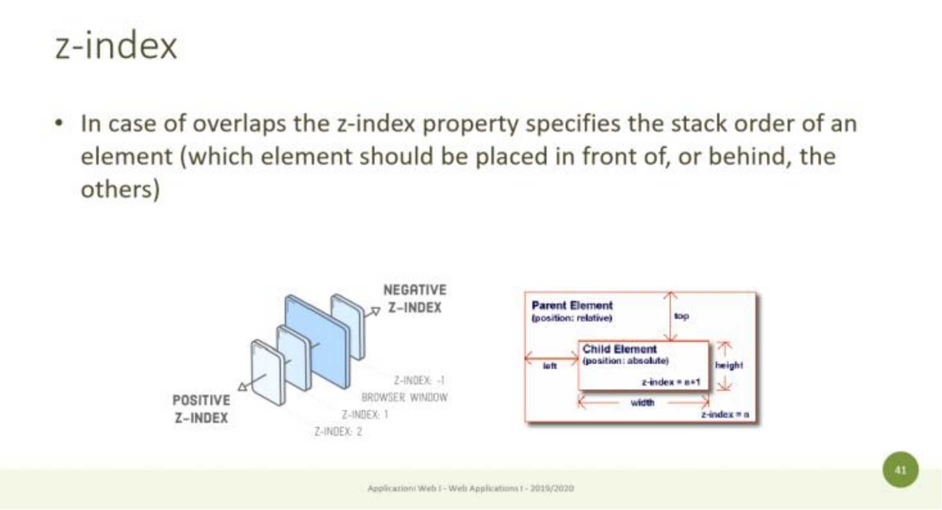
Very linked to the internal functioning of the algorithm (low level, difficul)

- **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)

STATIC
RELATIVE
ABSOLUTE
FIXED
BROWSER
BROWSER
BROWSER
BROWSER

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

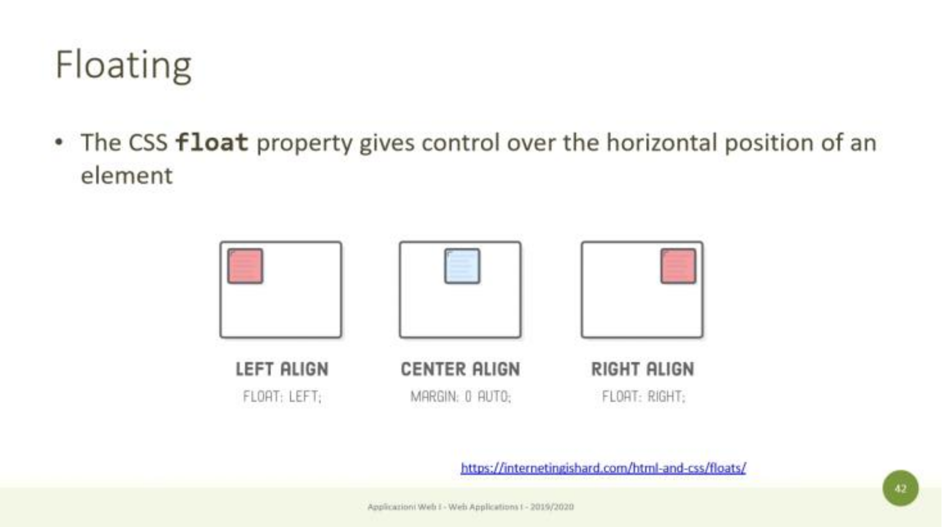
https://internetingishard.com/html-and-css/advanced-positioning/



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)

NEGATIVE
Z-INDEX
POSITIVE
Z-INDEX
Z-INDEX: -1
BROWSER WINDOW
Z-INDEX: 1
Z-INDEX: 2
Parent Element
(position: relative)
top
left
Child Element
(position: absolute)
z-index = n+1
height
width
z-index = n



Floating

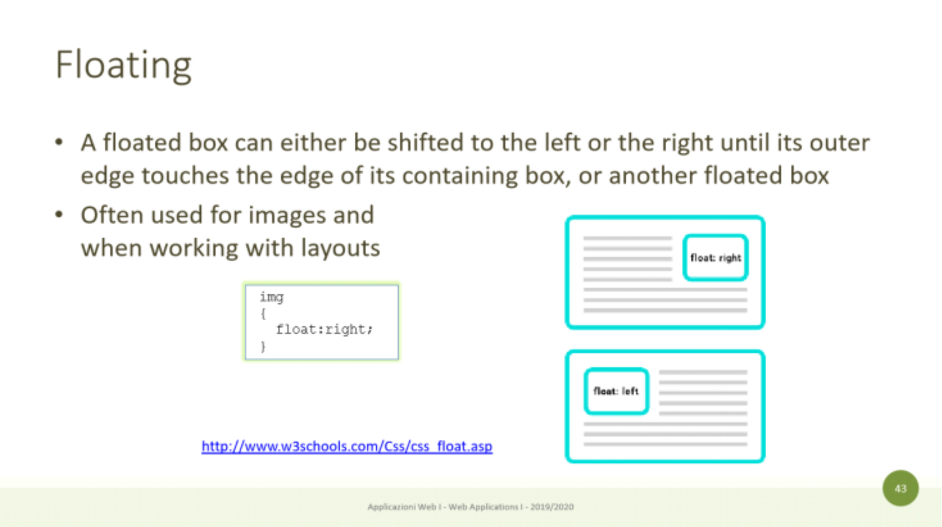
- The CSS **`float`** property gives control over the horizontal position of an element

LEFT ALIGN
FLOAT: LEFT;

CENTER ALIGN
MARGIN: 0 AUTO;

RIGHT ALIGN
FLOAT: RIGHT;

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



Floating

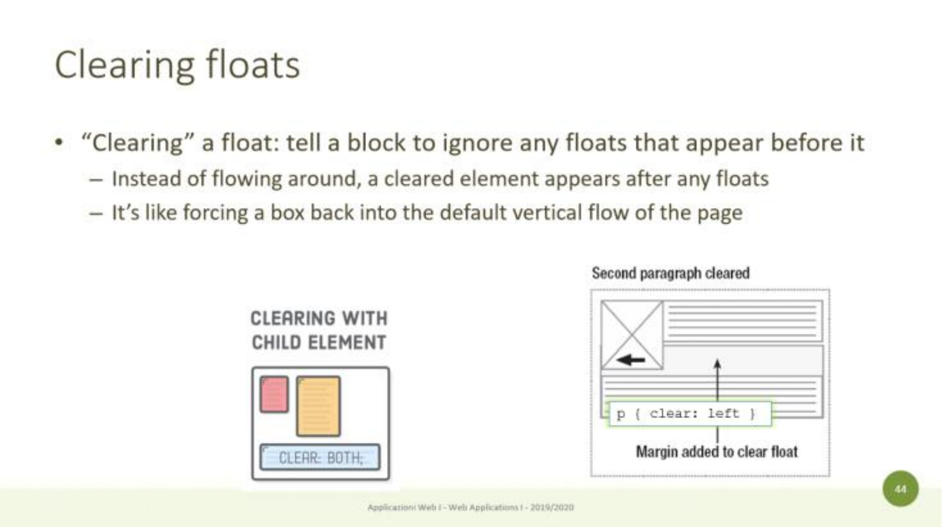
- A floated box can either be shifted to the left or the right until its outer edge touches the edge of its containing box, or another floated box
- Often used for images and when working with layouts

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

float: right

float: left

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

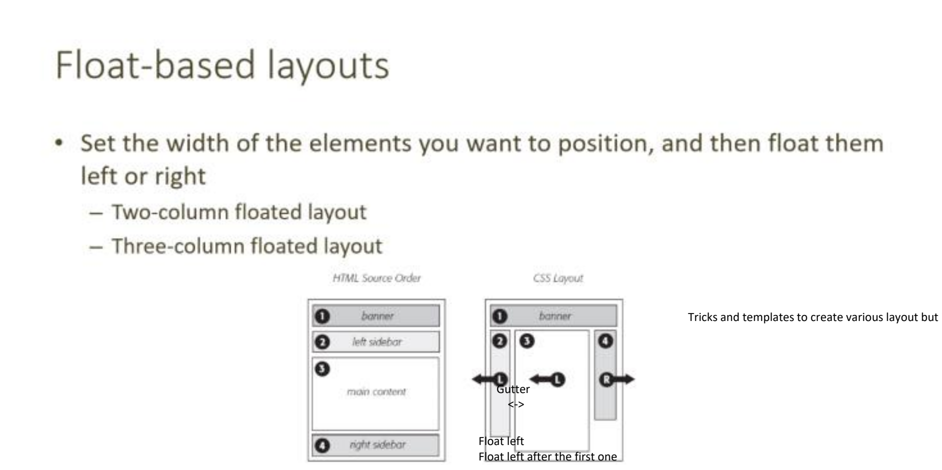
CLEARING WITH CHILD ELEMENT

CLEAR: BOTH;

Second paragraph cleared

p { clear: left }

Margin added to clear float



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

HTML Source Order

1 banner
2 left sidebar
3 main content
4 right sidebar

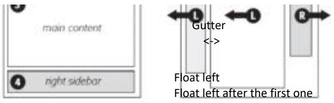
CSS Layout

1 banner
2 3 4
L L R

Gutter
<->

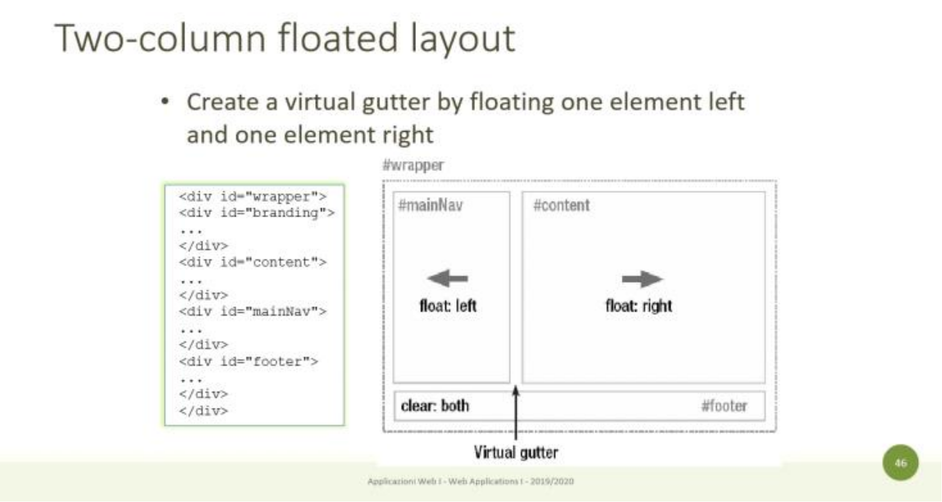
Float left
Float left after the first one

Tricks and templates to create various layout but it was very difficult!



Gutter
<->

Float left
Float left after the first one
Float right

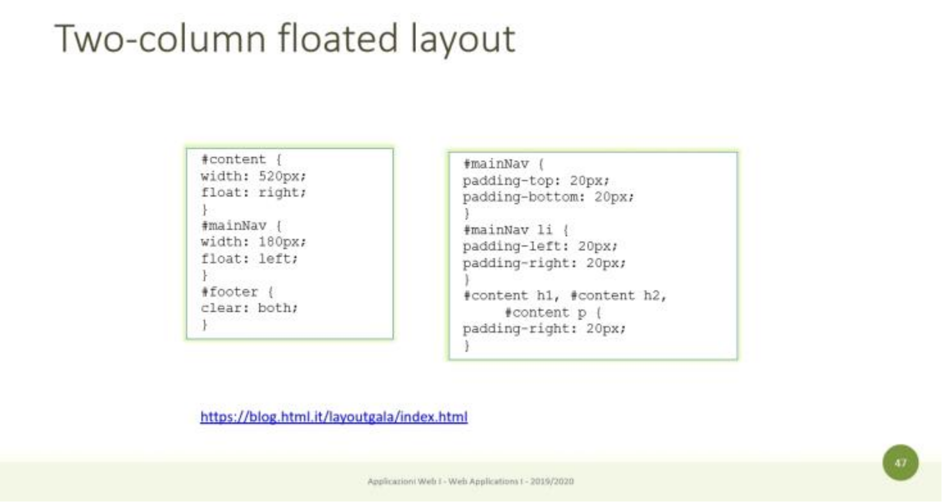


Two-column floated layout

- 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>
```

Complex layouts with blood and tears

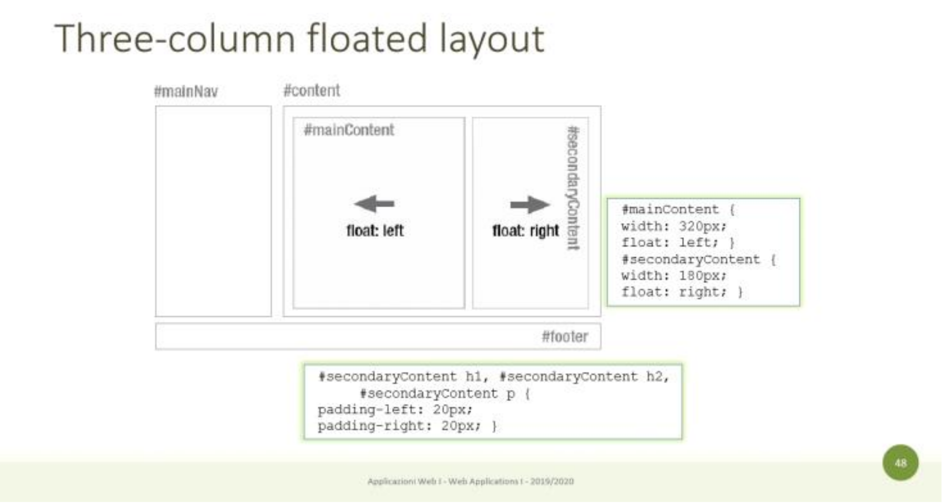


Two-column floated layout

```
#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

```
#mainContent {
width: 320px;
float: left; }
#secondaryContent {
width: 180px;
float: right; }
```

```
#secondaryContent h1, #secondaryContent h2,
     #secondaryContent p {
padding-left: 20px;
padding-right: 20px; }
```

References for CSS box model and positioning

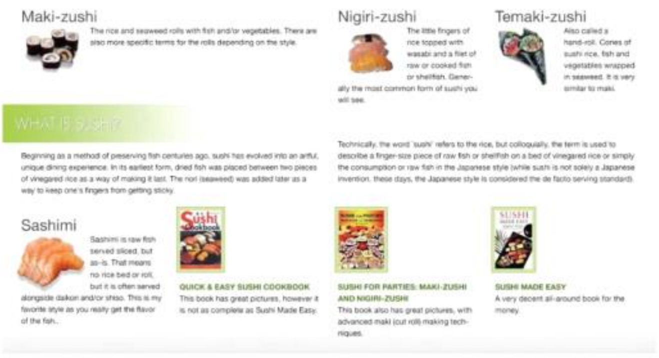
- Learn CSS layout
  - http://learnlayout.com/
- Floatutorial
  - http://css.maxdesign.com.au/floatutorial/
- All about floats
  - https://css-tricks.com/all-about-floats/

Cascading Style Sheets

PAGE LAYOUT WITH GRIDS

Now way easier!
We'll use these in the bootstrap library

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.

Technically, the word 'sushi' refers to the rice, but colloquially, the term is used to describe a finger-size piece of raw fish or shellfish on a bed of vinegared rice or simply the consumption or raw fish in the Japanese style (while sushi is not solely a Japanese invention, these days, the Japanese style is considered the de facto serving standard).

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

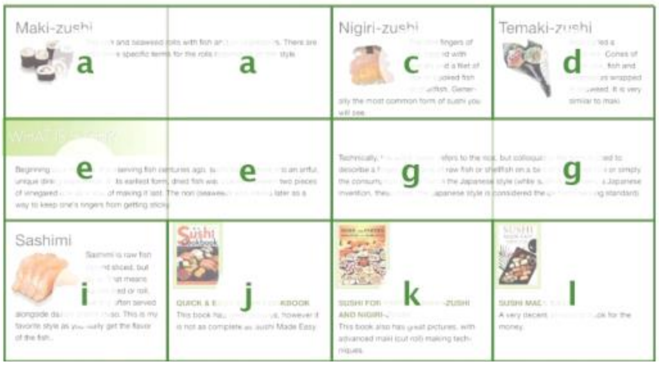
This book also has great pictures, with advanced maki (cut roll) making techniques.

SUSHI MADE EASY

A very decent all-around book for the money.

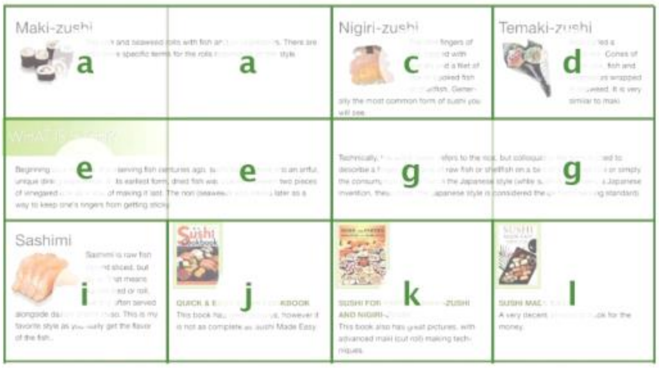
Elements can occupy different amount of cells.
We do not need to bother about pixels, higher level mechanism to decide what goes where.
We can have grids with width of the cells equal for each cell or customizable to make room for different types of elements. Easier to manage then floats, especially for problems about different heights or even variable heights!

Advanced layout: grid



We do not need to bother about pixels, higher level mechanism to decide what goes where.
We can have grids with width of the cells equal for each cell or customizable to make room for different types of elements. Easier to manage then floats, especially for problems about different heights or even variable heights!

Advanced layout: grid



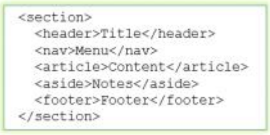
Just declare that one div is a grid, giving rows and columns. Elements position themself automatically depending on the size of the elements!

Advanced layout: grid

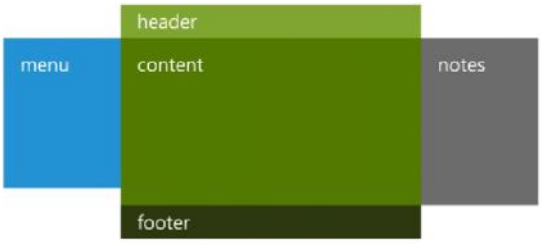
- It is possible to define a grid in which content can flow or be placed, or that remain empty
- There are 3 ways to define a grid
  - Explicit grid: defined with '`grid-columns`' and '`grid-rows`' properties
  - Natural grid: automatically created by elements with a natural grid structure (multi-column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

Example

- Classic three-column layout



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



?

Example

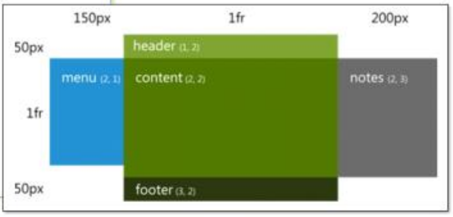
Left border

Right border

```
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; }
```

Head border

Foot border

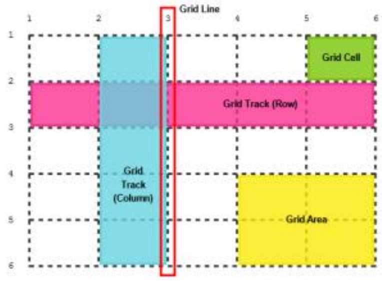


- fr = fraction values
  - new unit applicable to grid-rows and grid-columns properties

Suggested reference for Grid Layout

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

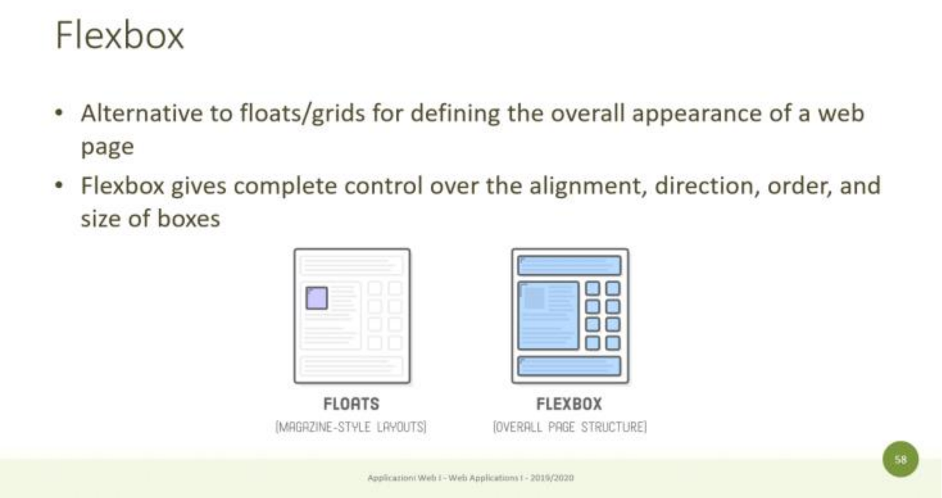
https://webkit.org/blog/7434/css-grid-layout-a-new-layout-module-for-the-web/



Cascading Style Sheets

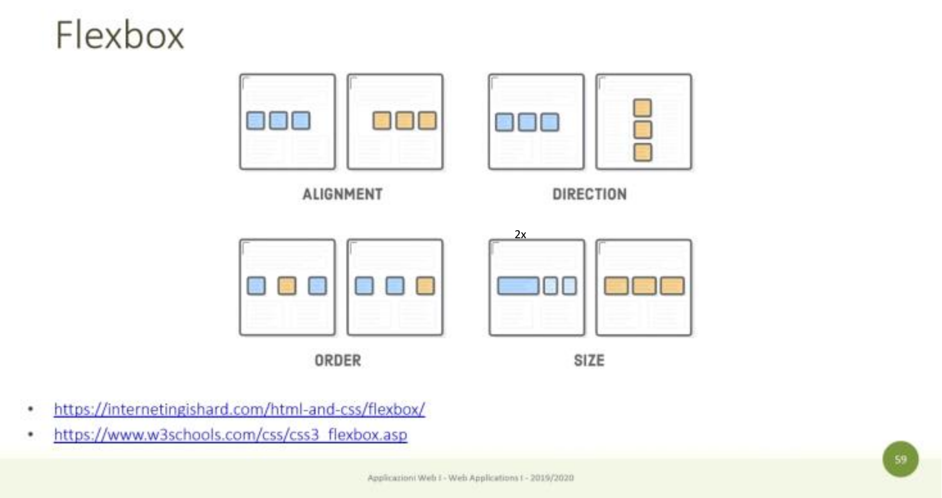
CSS FLEXBOX

Alternative to grid layout: flex layout: rules to position elements inside automatically



Flexbox

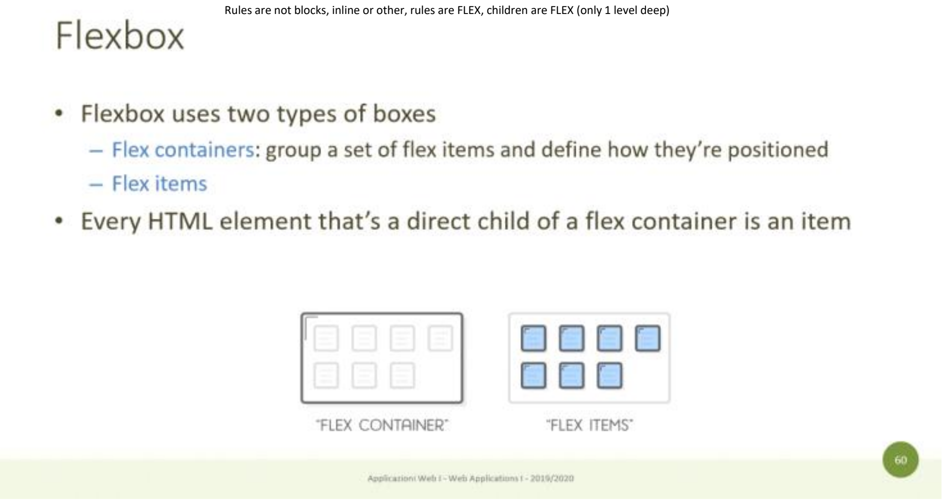
- Alternative to floats/grids for defining the overall appearance of a web page
- Flexbox gives complete control over the alignment, direction, order, and size of boxes



Flexbox

2x

- https://internetingishard.com/html-and-css/flexbox/
- https://www.w3schools.com/css/css3_flexbox.asp



Rules are not blocks, inline or other, rules are FLEX, children are FLEX (only 1 level deep)

Flexbox

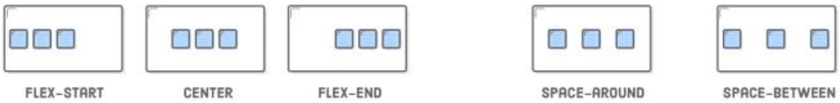
- Flexbox uses two types of boxes
  - Flex containers: group a set of flex items and define how they're positioned
  - Flex items
- Every HTML element that's a direct child of a flex container is an item

Horizontal alignment

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

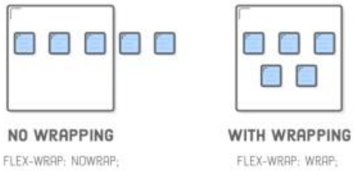
Display property change completely the layout algorithm
to apply to the elements inside!
The container does all the work!
While with grid we needed to position each element!

- To turn one HTML elements into a flex container:
  `{ display:flex ; }`
- "`justify-content`" property defines the horizontal alignment of its items
  - center, flex-start, flex-end
  - space-around, space-between



Wrapping

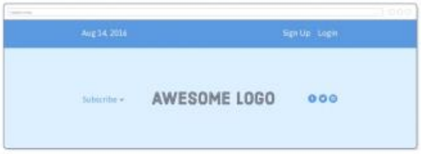
- The `flex-wrap` property creates a grid
  - Then, you can change alignment, direction, order, and size of items

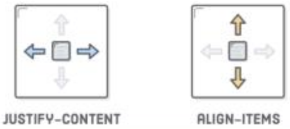


```
.photo-grid {
  width: 900px;
  display: flex;
  justify-content: center;
  flex-wrap: wrap;
}
```

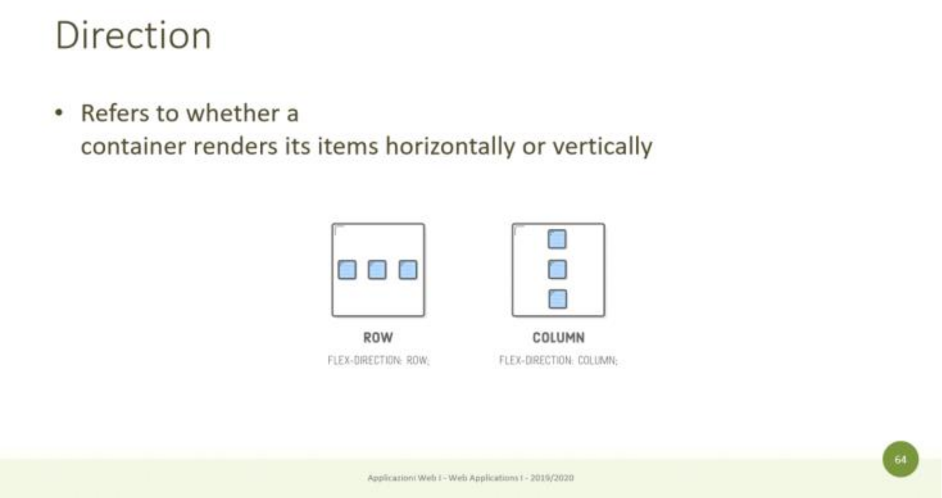
Vertical alignment

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





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

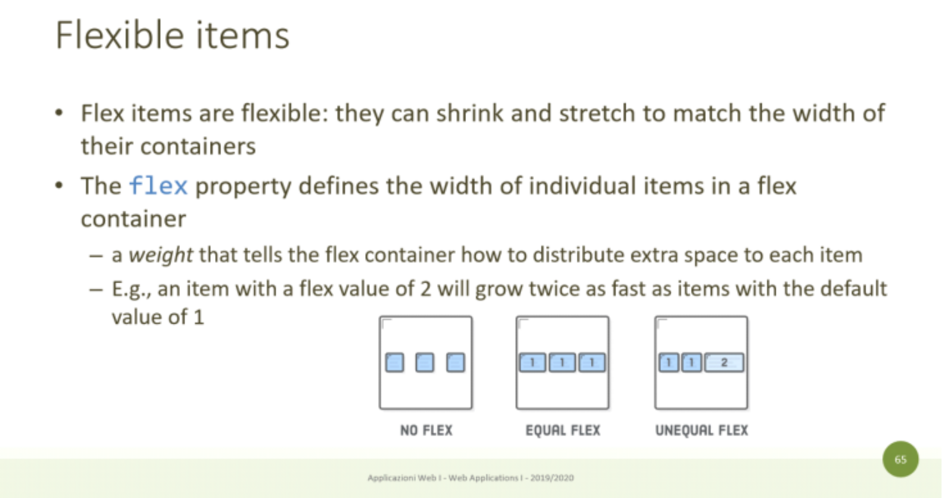


Direction

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

ROW
FLEX-DIRECTION: ROW;
COLUMN
FLEX-DIRECTION: COLUMN;

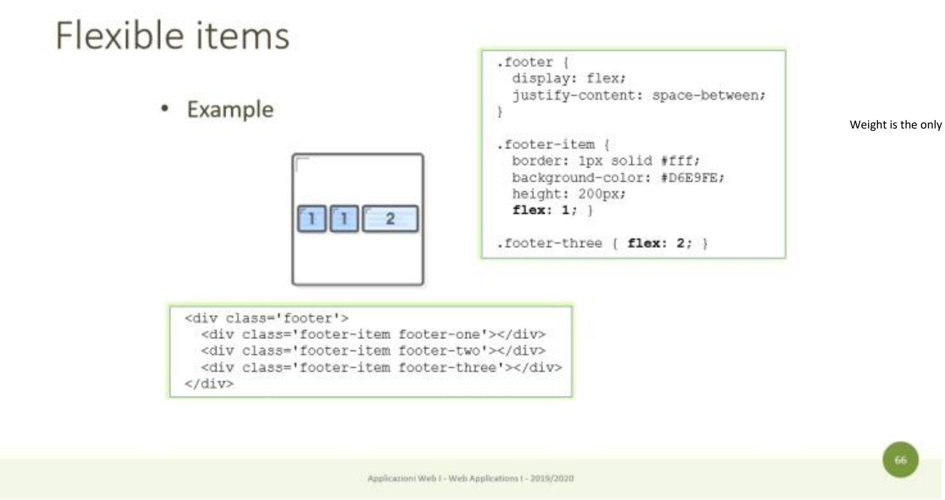
Size of the element is computed automatically by the algorithm!



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

1 1 1
1 1 2
NO FLEX
EQUAL FLEX
UNEQUAL FLEX



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; }
```

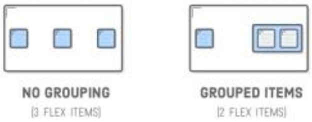
```
<div class='footer'>
  <div class='footer-item footer-one'></div>
  <div class='footer-item footer-two'></div>
  <div class='footer-item footer-three'></div>
</div>
```

Weight is the only property that we can give to the children

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>

Inside the internal flex, flex is not applied



Summary of CSS flexbox

Floats, grids, flex layouts possible!

- `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
- Flexbox, guida pratica
  - http://www.html.it/guide/flexbox-guida-pratica/
- Flexbox Froggy (a game-like tutorial)
  - http://flexboxfroggy.com/

Often we don't use those attribute directly but through the library

Cascading Style Sheets

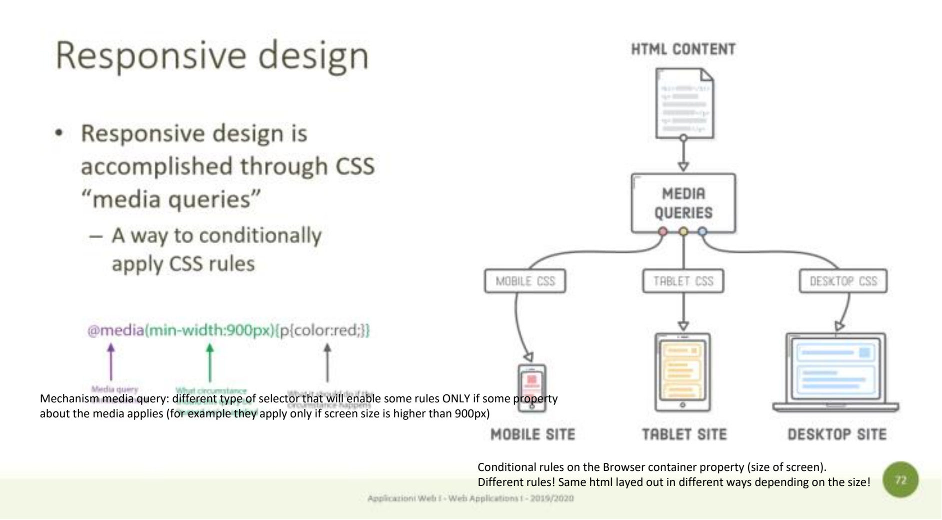
RESPONSIVE LAYOUT

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

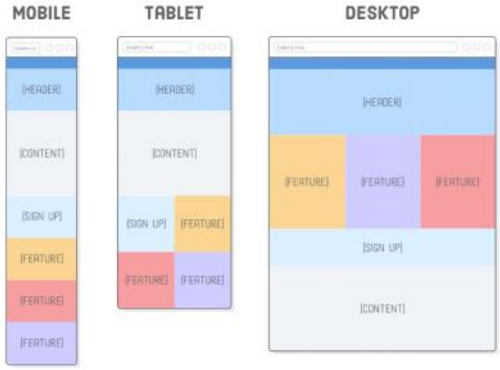
- Responsive design is accomplished through CSS "media queries"
  - A way to conditionally apply CSS rules

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

Mechanism media query: different type of selector that will enable some rules ONLY if some property about the media applies (for example they apply only if screen size is higher than 900px)

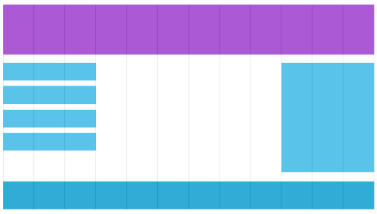
Conditional rules on the Browser container property (size of screen).
Different rules! Same html layed out in different ways depending on the size!

Layout for responsive design



Grid-view

- Many web pages are based on a grid-view, i.e. the page is divided into columns
- A responsive grid-view often has 12 columns, a total width of 100%, and will shrink and expand as you resize the browser window



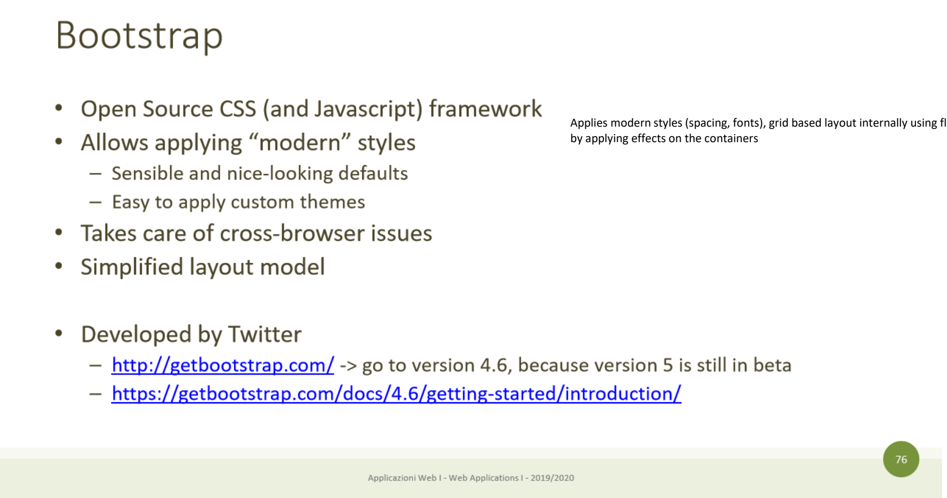
Compromise instead of doing by hand:
Media queries+grid.
We do not care about pixel but about big organization of the page!

3-4 main css frameworks

Cascading Style Sheets

THE BOOTSTRAP FRAMEWORK

Flex layout+media queries+many css classes already defined, that we can use and customize avoiding low level manual css settings.



Bootstrap

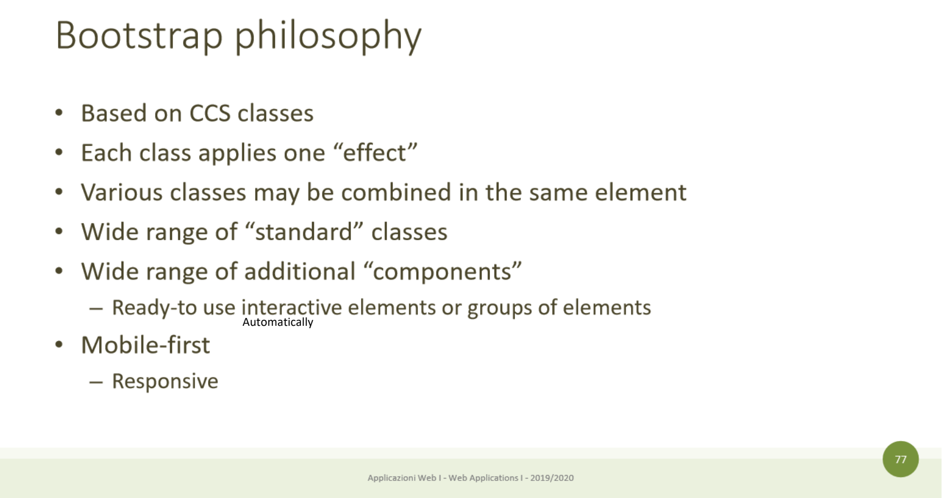
- Open Source CSS (and Javascript) framework
- Allows applying “modern” styles
  - Sensible and nice-looking defaults
  - Easy to apply custom themes
- Takes care of cross-browser issues
- Simplified layout model

- Developed by Twitter
  - http://getbootstrap.com/ -> go to version 4.6, because version 5 is still in beta
  - https://getbootstrap.com/docs/4.6/getting-started/introduction/

Applicazioni Web I - Web Applications I - 2019/2020

76

Applies modern styles (spacing, fonts), grid based layout internally using flexbox, responsive design, all by applying effects on the containers



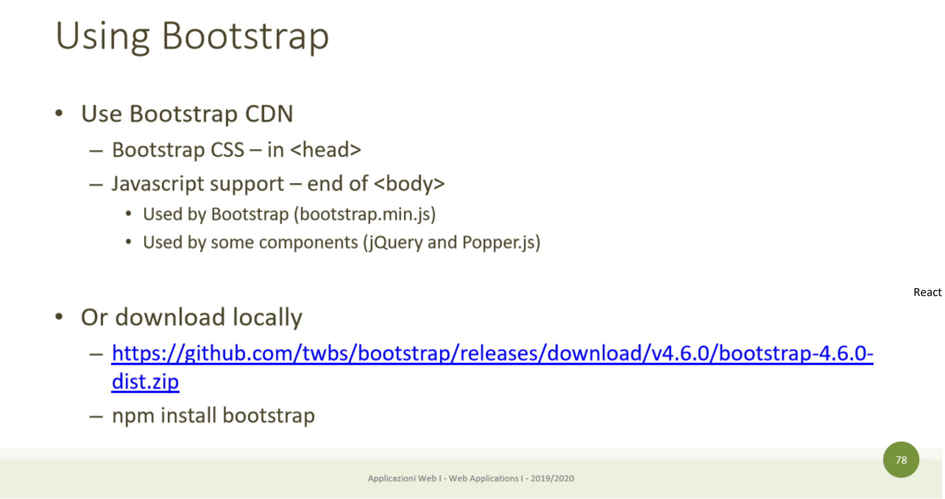
Bootstrap philosophy

- Based on CCS classes
- Each class applies one “effect”
- Various classes may be combined in the same element
- Wide range of “standard” classes
- Wide range of additional “components”
  - Ready-to use interactive elements or groups of elements
- Mobile-first
  - Responsive

Automatically

Applicazioni Web I - Web Applications I - 2019/2020

77



Using Bootstrap

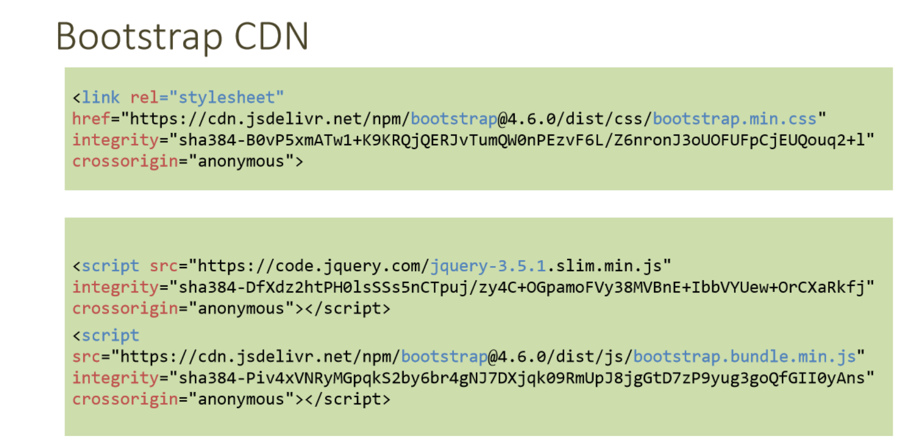
- Use Bootstrap CDN
  - Bootstrap CSS – in <head>
  - Javascript support – end of <body>
    - Used by Bootstrap (bootstrap.min.js)
    - Used by some components (jQuery and Popper.js)

- Or download locally
  - https://github.com/twbs/bootstrap/releases/download/v4.6.0/bootstrap-4.6.0-dist.zip
  - npm install bootstrap

Applicazioni Web I - Web Applications I - 2019/2020

78

React only support version 4.6 for now! (5.0 still beta)



Bootstrap CDN

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/css/bootstrap.min.css"
integrity="sha384-B0vP5xmATw1+K9KRQjQERJvTumQW0nPEzvF6L/Z6nronJ3oUOFUFpCjEUQouq2+l"
crossorigin="anonymous">
```

```
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns"
crossorigin="anonymous"></script>
```

Just a css file loaded from network. (or saved offline but online is still fast and more updated)
We also need 2 js modules because it uses them internally!

Already Fonts and Spacing is more Modern looking!

```
src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js"
integrity="sha384-Piv4xVNRyMGpqkS2by6br4gNJ7DXjqk09RmUpJ8jgGtD7zP9yug3goQfGII0yAns"
crossorigin="anonymous"></script>
```

Already Fonts and Spacing is more Modern looking!

We write classes on html that refers to the classes styled in the bootstrap library.

To give bootstrap full control on the page we need to put all elements into .container (gives fixed width depending on the size of the browser) or .container fluid (always occupy 100% of the width of the page)

Container defines the grid. New grid environment: new container

Responsive breakpoints: 5 different possible size of screen (xs, s, m, l, xl) (we decide the minimum size until we display grid as column instead of as a block)
Use grid to decide how much space each element will take on the screen (define row and col)
Depending on how many columns I decide to put, it splits evenly the available space (max 12, 6-6, 4,3, ...) or unevenly space (col-3) (col-9) (col-3/12) (col-9/12)

Page structure

- Start with **Basic template**
  - https://getbootstrap.com/docs/4.6/getting-started/introduction/#starter-template
  - Notice the “responsive viewport”
    `<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">`
- Or, choose from published **Examples**
  - https://getbootstrap.com/docs/4.6/examples/
- Remember to include your page content in a
  - `<div class="container">`

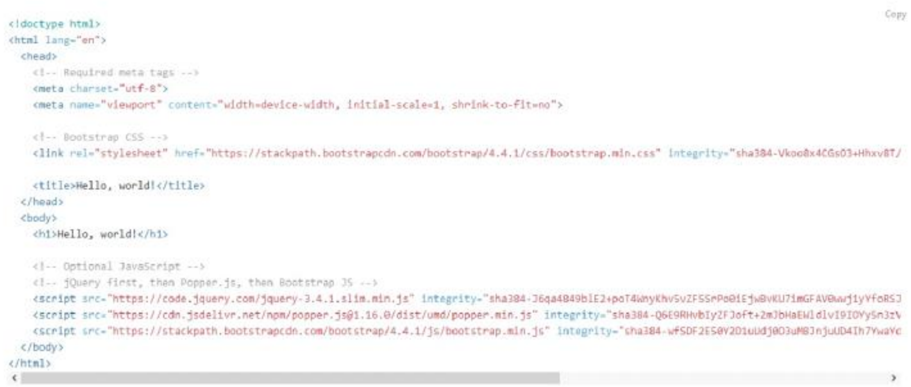
Study documentation to format what you need (names of classes to use)

Components: user interfaces already coded.
Consistent, clean, modern framework!
If we don't like it use extensions for it (icons, themes,..)

I can have personal css file to add custom rules to overwrite bootstrap BUT I won't need it because I just need to customize bootstrap (use btn btn-success -> semantic button -> to change color we just need to use a new bootstrap theme which changes the color of the btn -success).

N.B. From JS we can change the value of 1 class! 1 line of code to modify the whole page layout!
Library to avoid css complexity!

Starter Template



```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.4.1.slim.min.js" integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJ
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js" integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js" integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd
  </body>
</html>
```

Adding your style

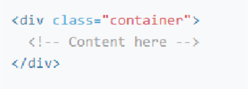
- Write your additional CSS rules in a file, and load it in the head section, after the bootstrap CSS

```
<head>
  ...
<link rel="stylesheet" href="... > // bootstrap css

<!-- Custom CSS for this document -->
<link rel="stylesheet" href="mystyle.css">
  ...
</head>
```

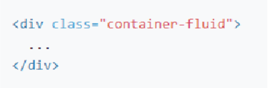
Containers (mandatory top-level element)

- All-in-one: responsive, fixed-width container: its max-width changes at each breakpoint



```
<div class="container">
  <!-- Content here -->
</div>
```

- Fluid: full width container, spanning the entire width of the viewport

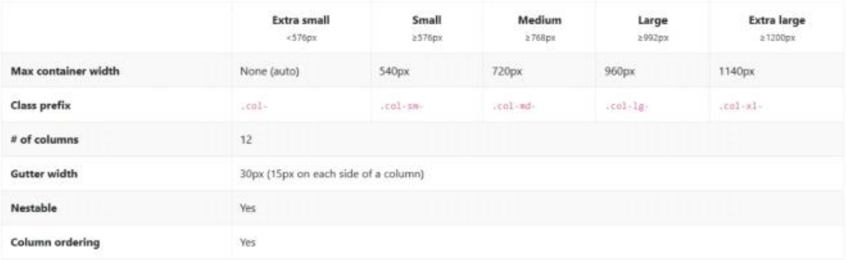


```
<div class="container-fluid">
  ...
</div>
```

Grid system

- Based on flexbox
- Uses a 12-column virtual grid
- The container is divided in `.row` blocks
- Each Row contains columns `.col`
  - By default, all columns have equal width
  - May specify spanning over more columns (`.col-4`)
  - May specify a breakpoint for that column to be visualized (`.col-sm` or `.col-sm-4`)
- 5 Breakpoints

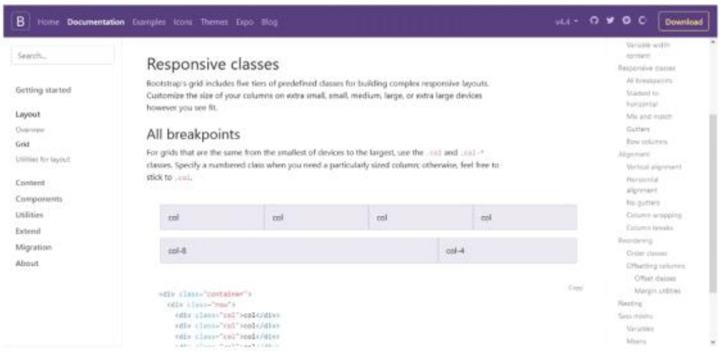
Breakpoints



	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	`.col-`	`.col-sm-`	`.col-md-`	`.col-lg-`	`.col-xl-`
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

Example layouts

https://getbootstrap.com/docs/4.6/layout/grid/



Responsive classes

Bootstrap's grid includes five tiers of predefined classes for building complex responsive layouts. Customize the size of your columns on extra small, small, medium, large, or extra large devices however you see fit.

All breakpoints

For grids that are the same from the smallest of devices to the largest, use the `.col` and `.col-*` classes. Specify a numbered class when you need a particularly sized column; otherwise, feel free to stick to `.col`.

col	col	col	col

col-8	col-4

```
<div class="container">
  <div class="row">
    <div class="col">col</div>
    <div class="col">col</div>
    <div class="col">col</div>
```

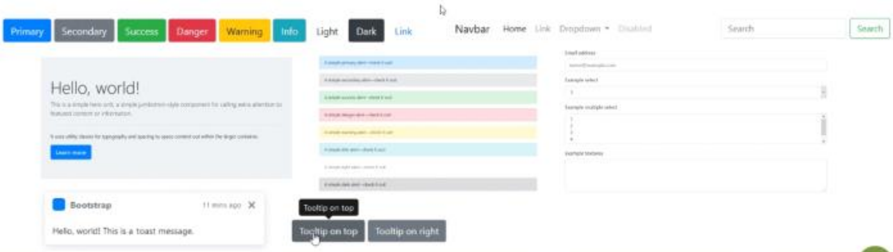
Page content

- Reboot – normalize and replace browser defaults
- Typography – heading and text styles
- Images and Figures – alignment and responsiveness
- Tables – rich visual styles, and responsive tables

- See: https://getbootstrap.com/docs/4.6/content/

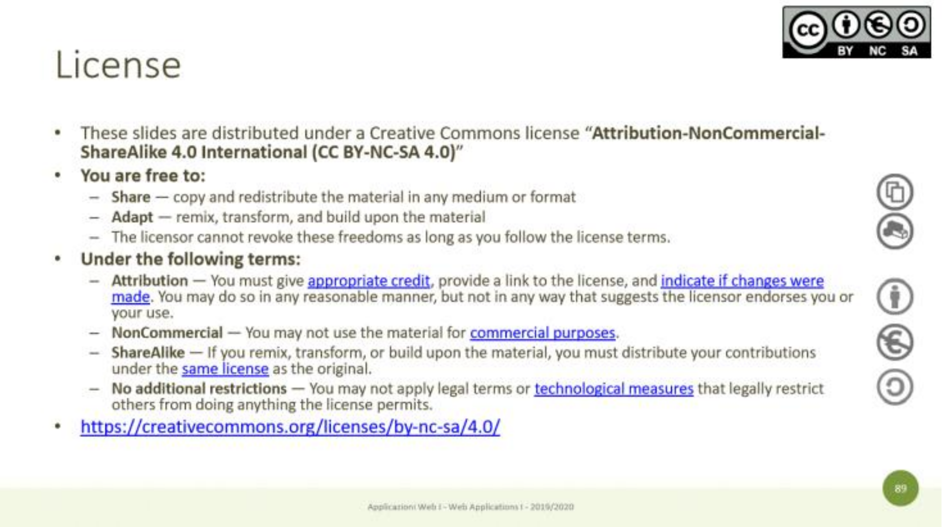
Components

- Ready-to-use classes for creating common UI elements
- See: https://getbootstrap.com/docs/4.6/components/



Hello, world!

Hello, world! This is a toast message.



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/