<WA1/>

2020

Cascading Style Sheets

Styling the Web

Enrico Masala

Fulvio Corno

Slides adapted from Laura Farinetti

Goal

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

Cascading Style Sheets

PART 1: CSS STRUCTURE AND LANGUAGE

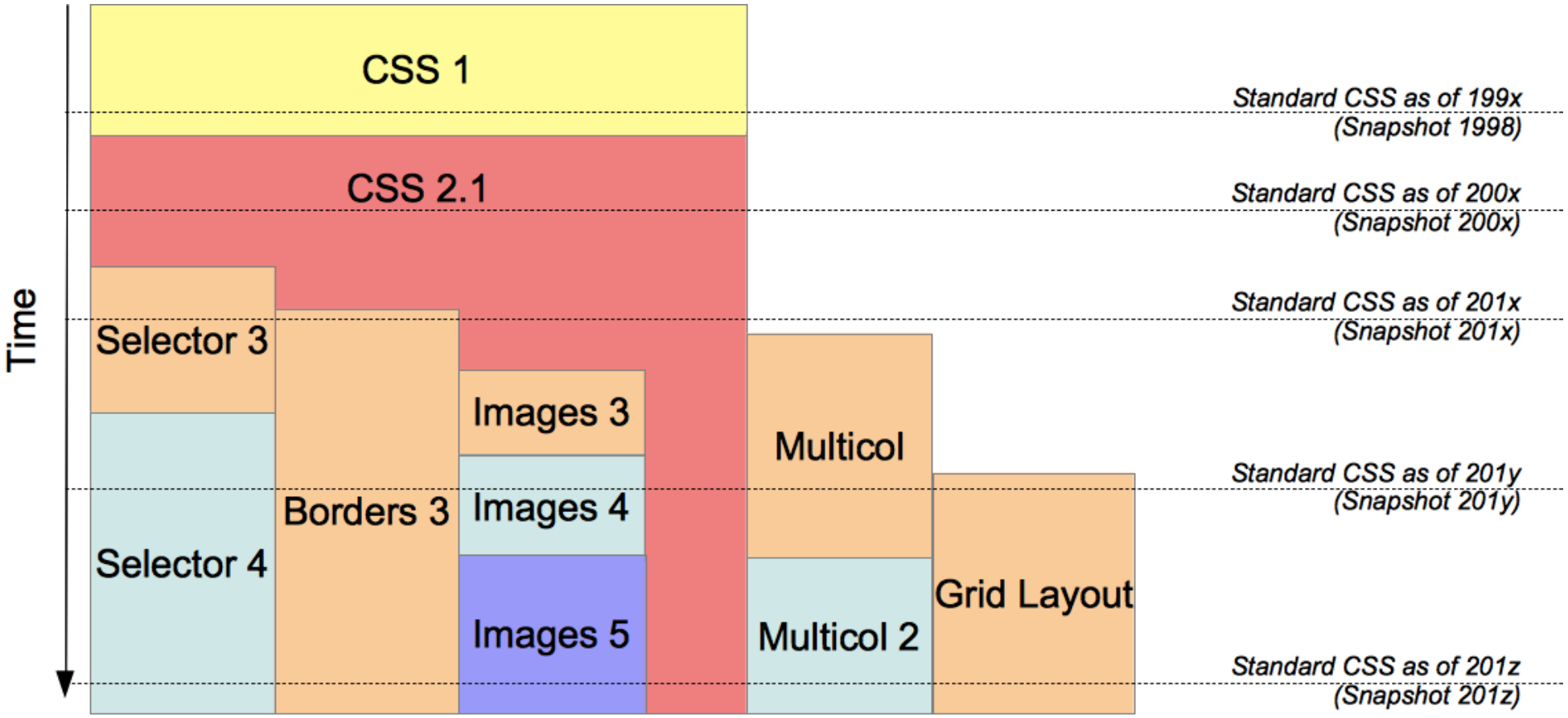
CSS3

- Introduction
- CSS syntax
- CSS selectors
- CSS cascading
- CSS box model
- CSS positioning schemes
- Page layout with float and grid
- CSS Flexbox
- Responsive layout
- The Bootstrap Library

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



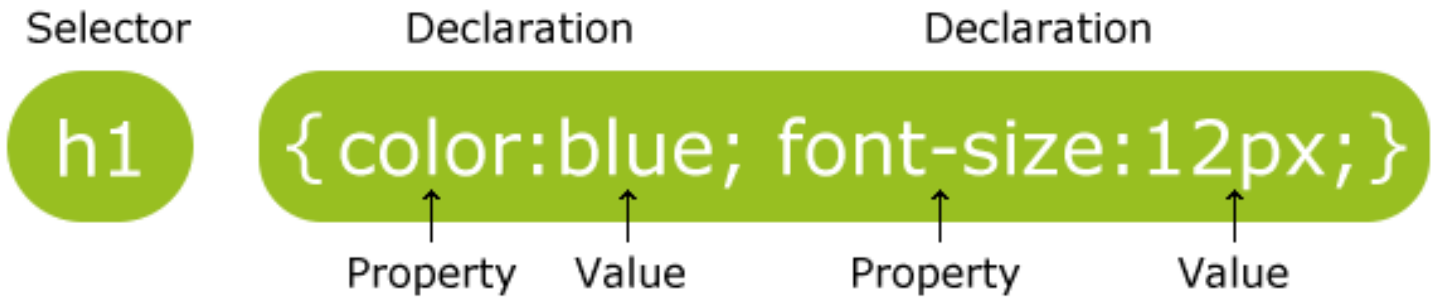
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3

Cascading Style Sheets

CSS SYNTAX

CSS Syntax

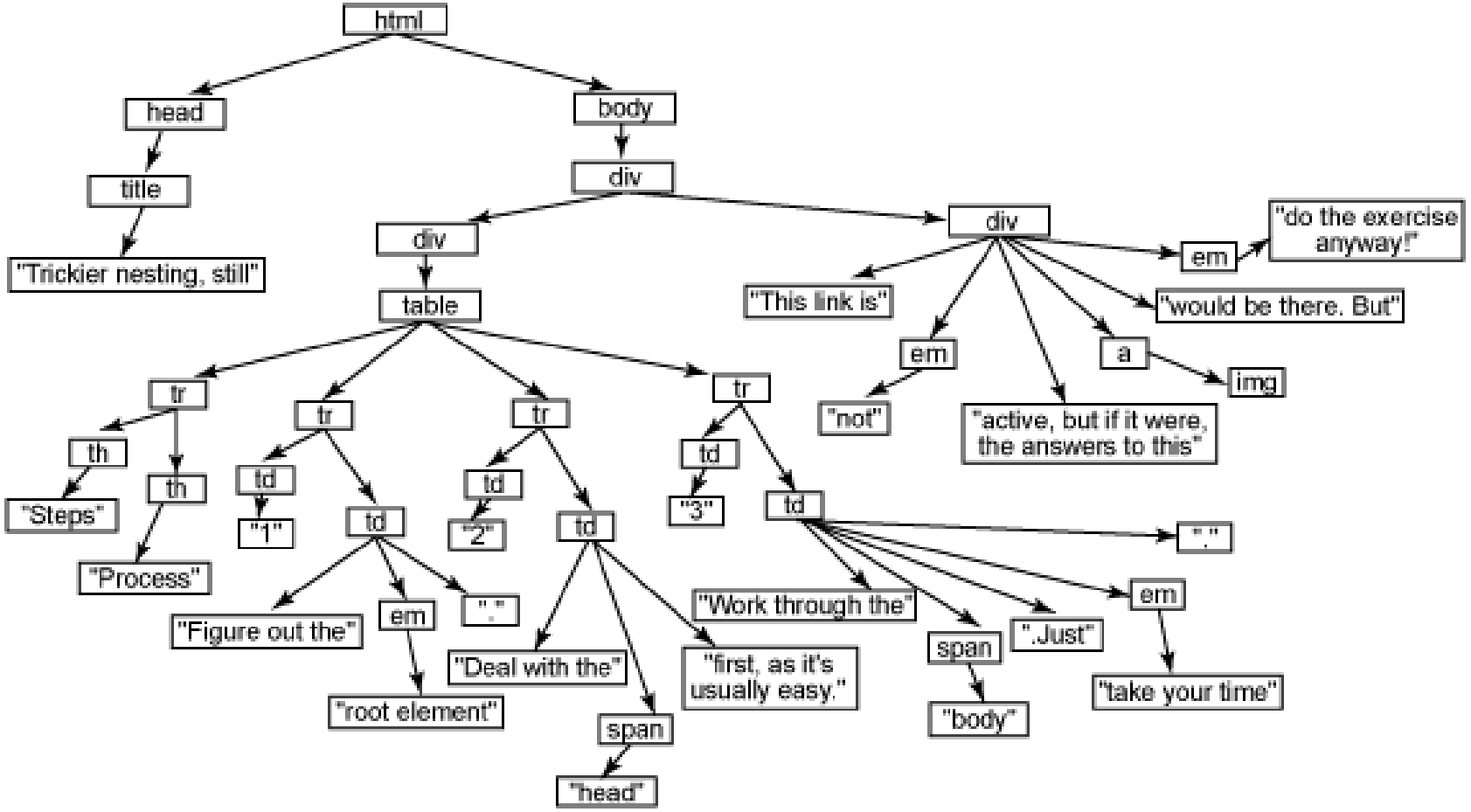
- CSS is based on **rules**
- A rule is a statement about one [stylistic] aspect of one or more HTML element
  - **Selector** + **Declaration**(s)
- A style sheet is a set of one or more rules that apply to an HTML document



HTML structure

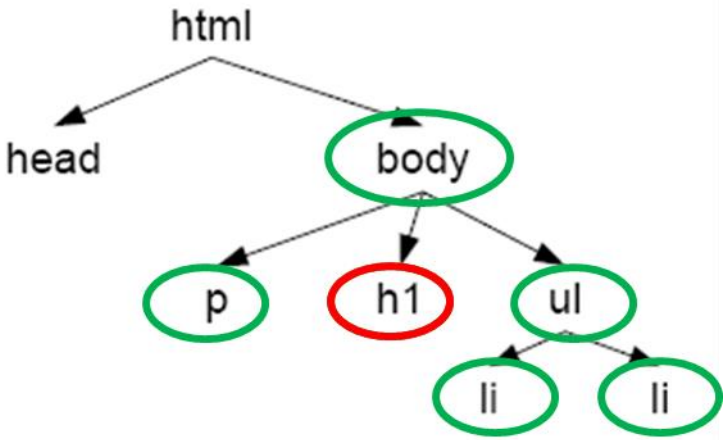
```
<html lang="en">
<head>
    <title>Trickier nesting, still</title>
</head>
<body>
    <div>
        <div>
            <table>
                <tr><th>Steps</th><th>Processes</th></tr>
                <tr><td>1</td><td>Figure out the <em>root element</em>.</td></tr>
                <tr><td>2</td><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>
```

Parsed HTML: tree structure

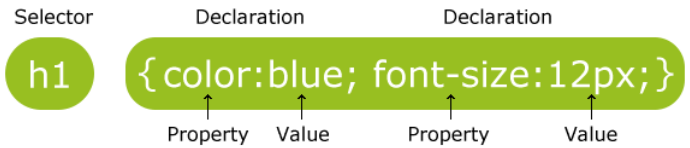


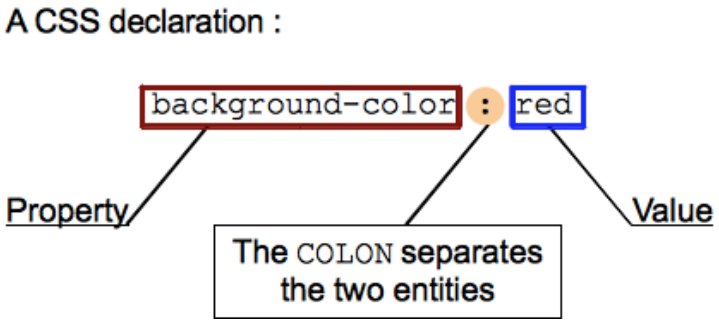
Tree structure and inheritance

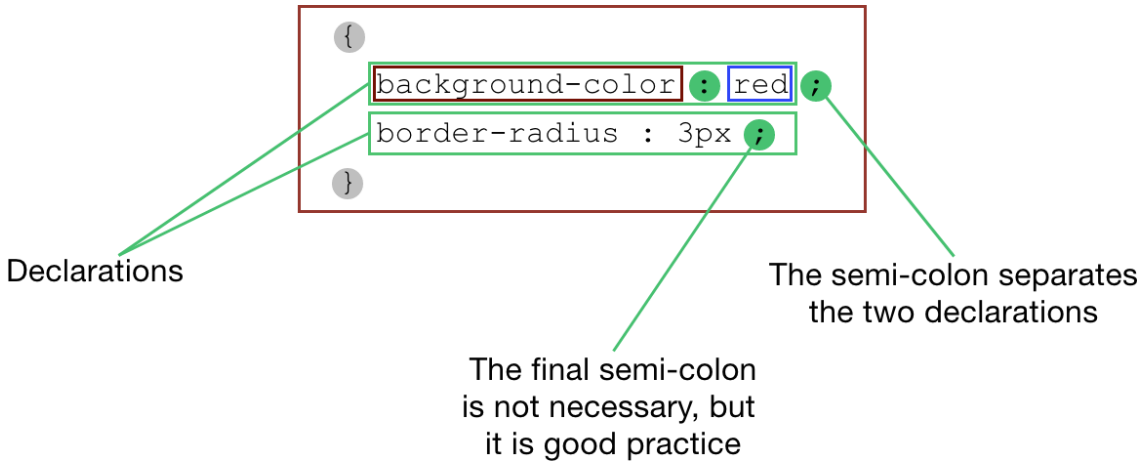
- HTML documents are trees
- Styles are inherited along trees
- When two rules are in conflict the most specific wins
- Example
  - body {color: green}
  - h1 {color: red}



Declaration Syntax







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:
  - Doea 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-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

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

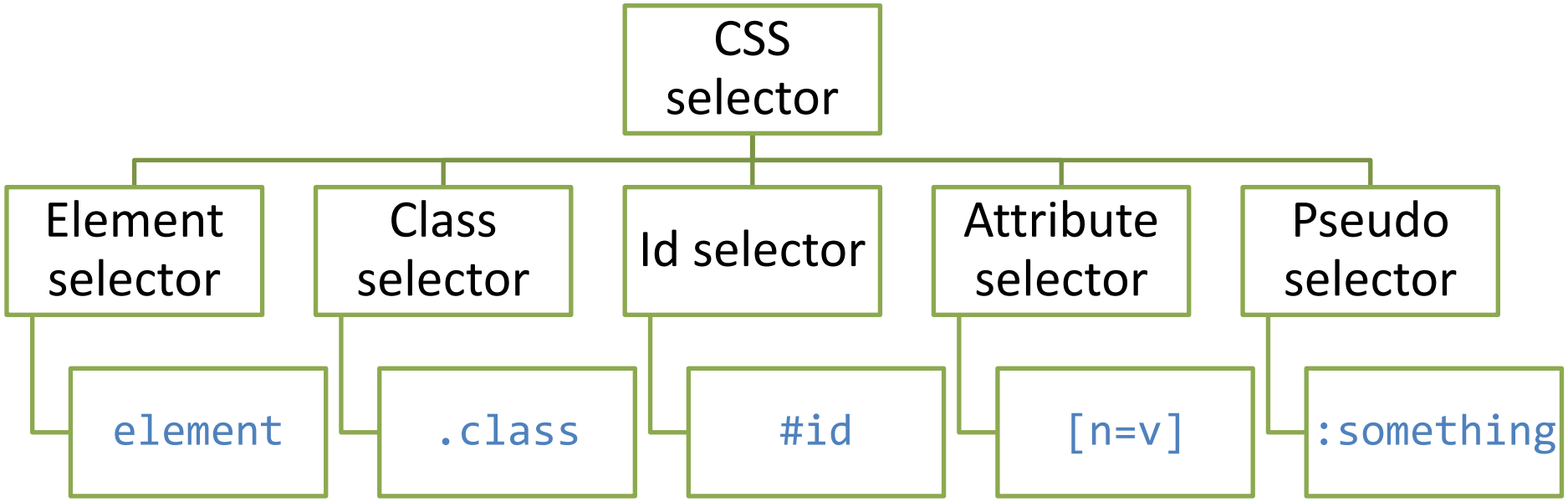
https://www.tutorialrepublic.com/css-reference/css3-properties.php

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”



Element selector

- Used to apply the same style to all instances of a specific element in a document
- You choose an element with its tag name and apply the style on that
- Example: apply the color red to all h1 elements that appear in the document

style.css

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

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>
```

style.css

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

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>
```

style.css

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

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;}
```

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

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
- `S1 + S2` → S2 if it comes after a S1
- `S1 ~ S2` → S2 if it comes later than S1

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 elements and all <p> elements	1
element element	div p	Selects all <p> elements inside elements	1
element>element	div > p	Selects all <p> elements where the parent is a element	2
element+element	div + p	Selects all <p> elements that are placed immediately after elements	2
element1~element2	p ~ ul	Selects every element that are preceded by a <p> element	3

http://www.w3schools.com/cssref/css_selectors.asp

Meaningful HTML

- Meaningful elements
  - h1, h2, ...
  - ul, ol, dl
  - strong and em
  - blockquote, cite
  - abbr, acronym, code
  - fieldset, legend, label
  - caption, thead, tbody, tfoot
- Id and class names
  - Allow to give extra meaning
- Div and span
  - Add structure to document

"BLOCK BOX"

"INLINE BOXES"

Div element

- Stands for "division"
- Used to group **block-level** elements
  - Divides a document into meaningful areas
- Use only if necessary and not redundant

```
<div id="mainNav">
  <ul>
    <li>Home</li>
    <li>About Us</li>
    <li>Contact</li>
  </ul>
</div>
```

```
<ul id="mainNav">
  <li>Home</li>
  <li>About Us</li>
  <li>Contact</li>
</ul>
```

```
<nav>
  <menu>
    <li>Home</li>
    <li>About Us</li>
    <li>Contact</li>
  </menu>
</nav>
```

Span element

- Used to group or identify **inline-level** elements

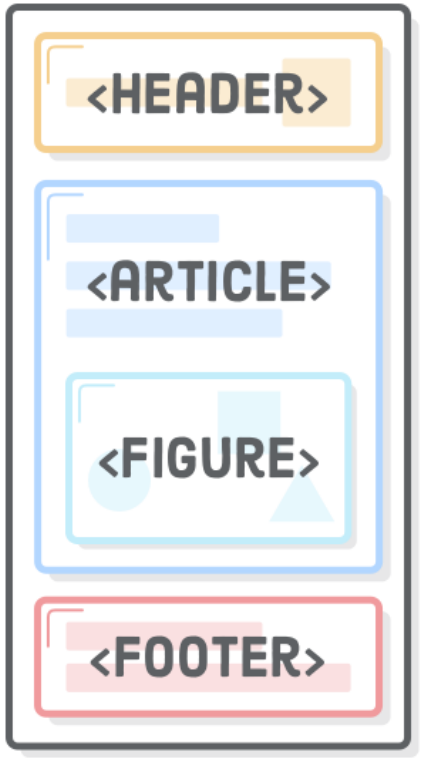

```
<h2>Where's Durstan?</h2>
<p>Published on
     <span class="date">March 22nd, 2005</span>
by <span class="author">Andy Budd</span></p>
```

```
<h2>Where's Durstan?</h2>
<p>Published on
     <time>March 22nd, 2005</time>
by <span class="author">Andy Budd</span></p>
```

Meaningful HTML

AMBIGUOUS STRUCTURE

(AKA "<DIV> SOUP")



IDENTIFIABLE SECTIONS

(AKA "SEMANTIC MARKUP")

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;}
```

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;
}
```

- 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

```
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

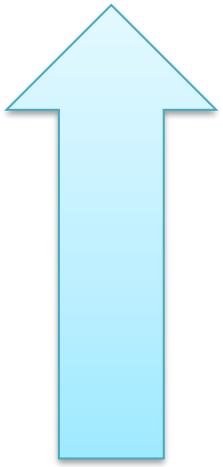
Selector	Example	Example description	CSS
:read-only	input:read-only	Selects input elements with the "readonly" attribute specified	3
:read-write	input:read-write	Selects input elements with the "readonly" attribute NOT specified	3
:required	input:required	Selects input elements with the "required" attribute 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

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)
  - Internal Style (usually in the HTML head section)
  - External Style
  - Browser Default Style



External style

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

style.css

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

```
<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

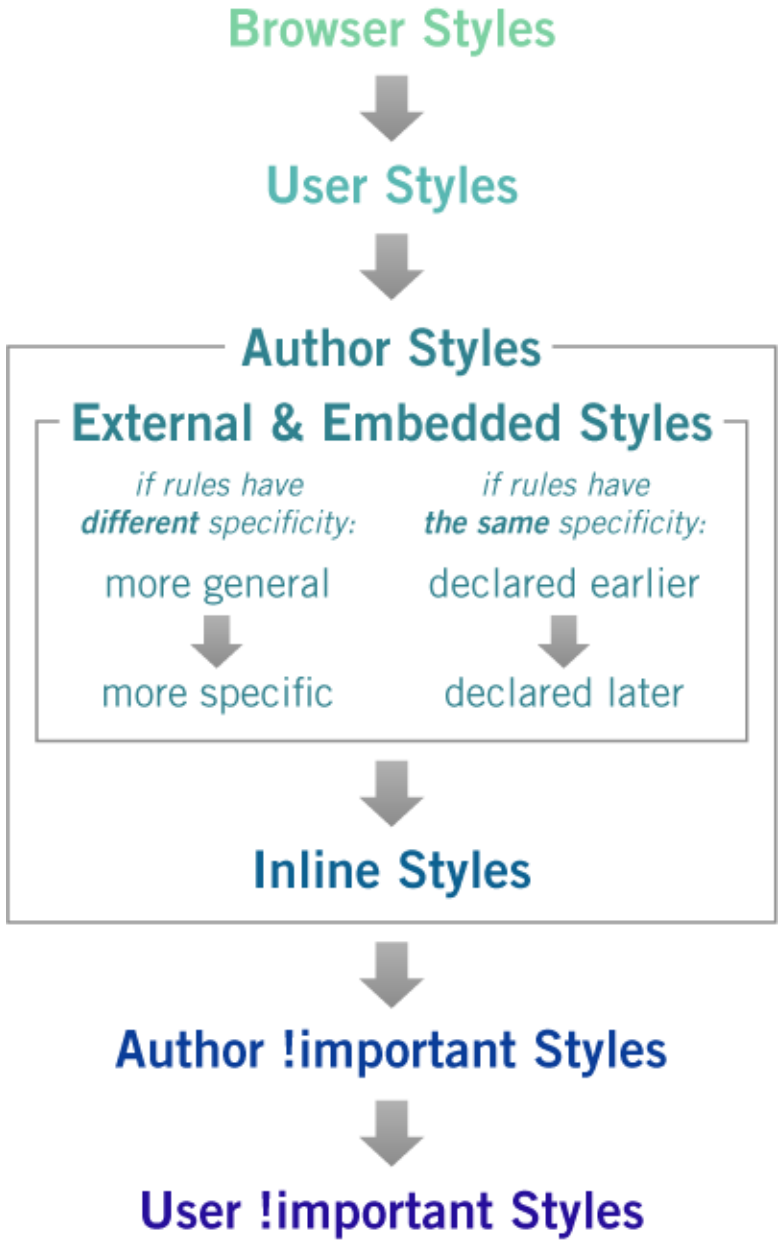
- `<style>` attribute within an HTML element
- Last resort, just for local (very local) changes

```
<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”

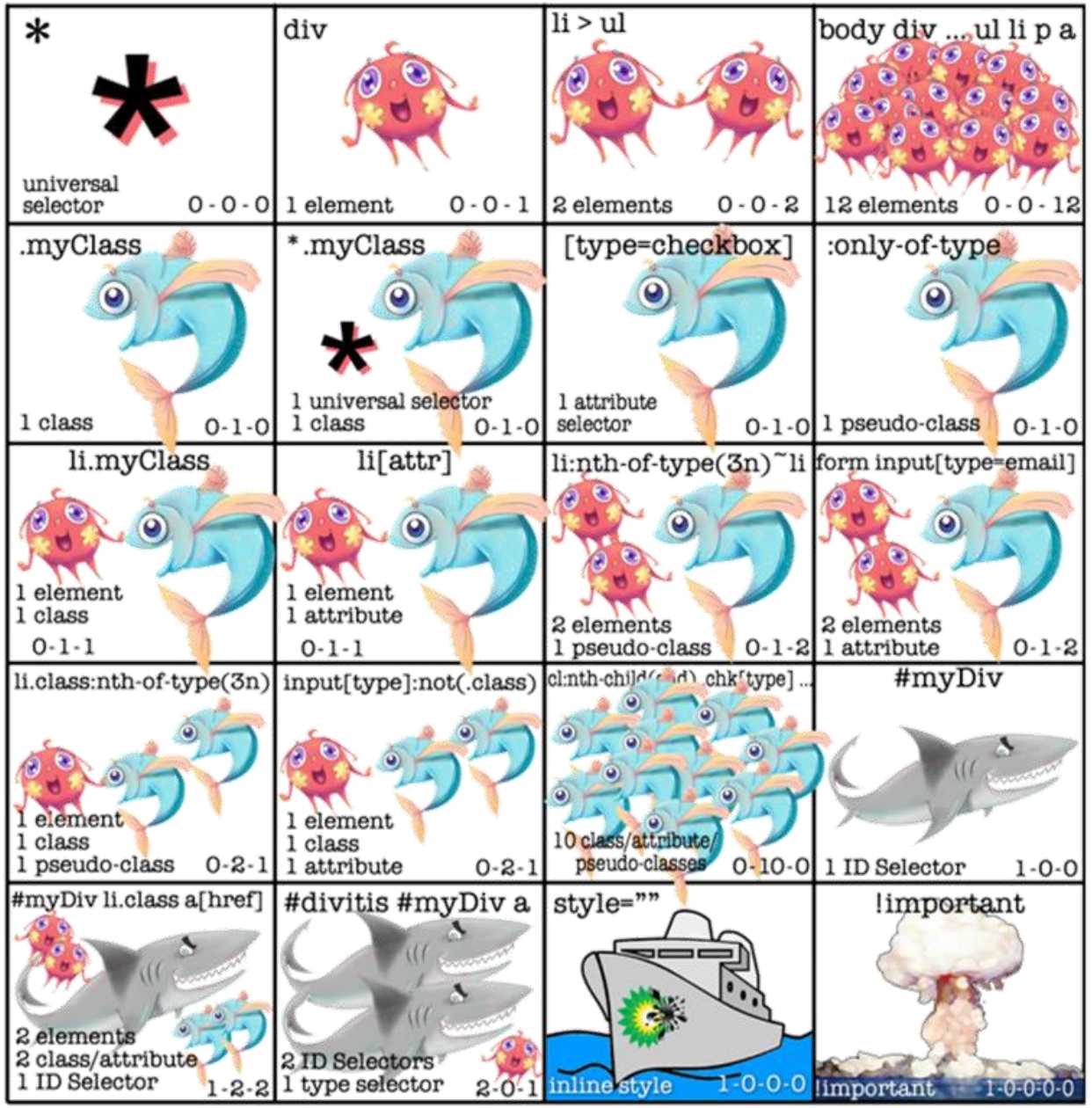
```
h1 {
  color:red !important
}
```



CSS specificity

- Specificity determines which CSS rule is applied by the browser
- Every selector has its place in the specificity hierarchy

1. Type selectors (h1) and pseudo-elements (::before)
2. Class selectors (.cls), attributes selector ([n=v]) and pseudo-classes (:hover)
3. ID selectors (#menu)



X-0-0: The number of ID selectors
0-Y-0: The number of class selectors, attributes selectors, and pseudo-classes
0-0-Z: The number of element (a.k.a. type) selectors and pseudo-elements
*, +, >, ~ : Universal selector and combinators do not increase specificity
:not(x): Negation selector has no value. Argument increases specificity

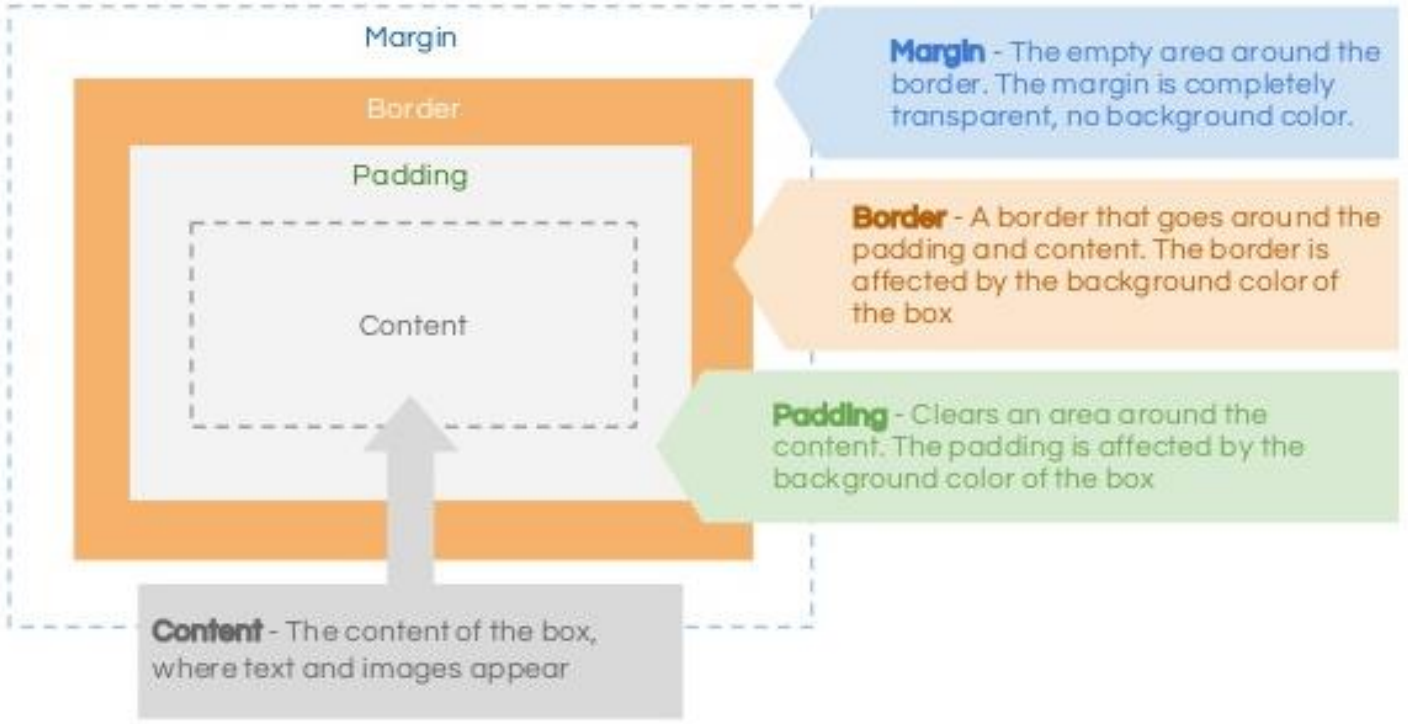
https://specifishity.com/

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

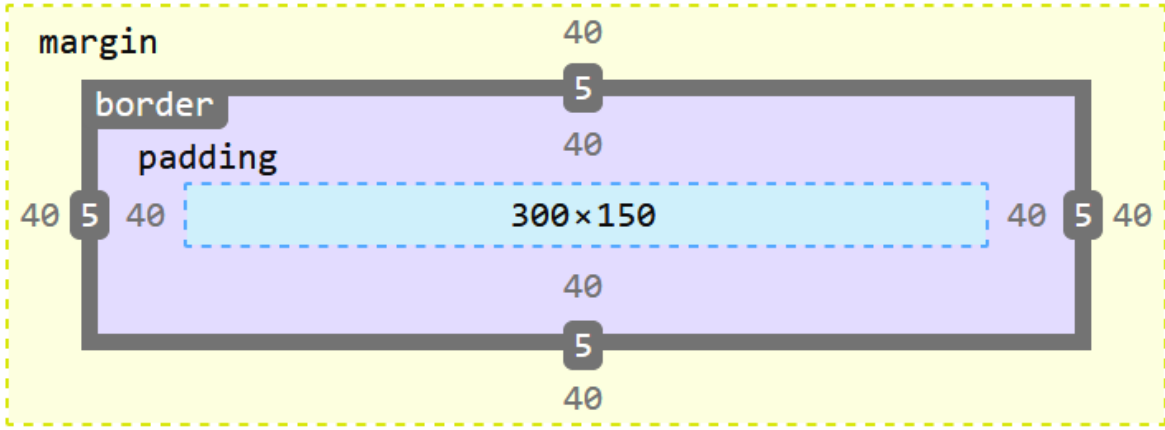


The box model

- 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

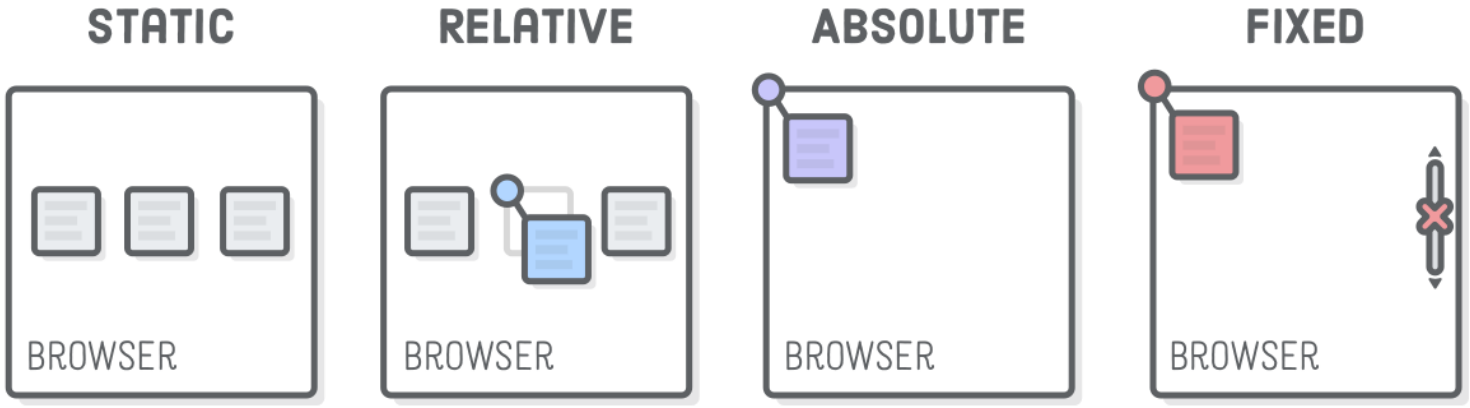


Cascading Style Sheets

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)

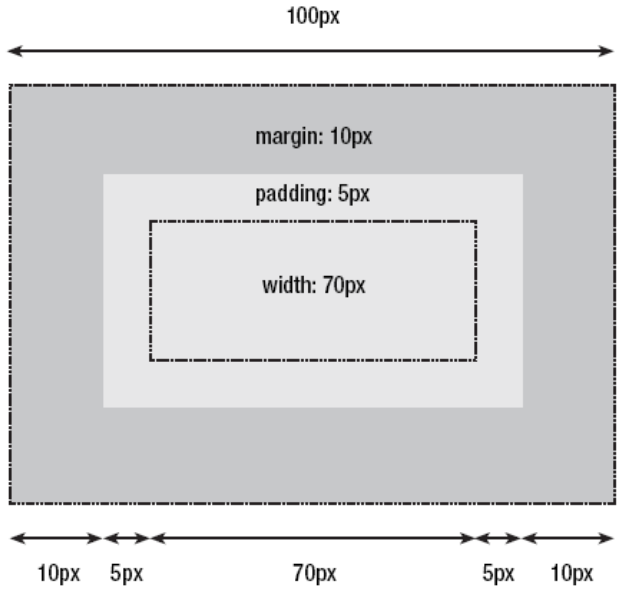


https://internetingishard.com/html-and-css/advanced-positioning/

Normal flow (static)

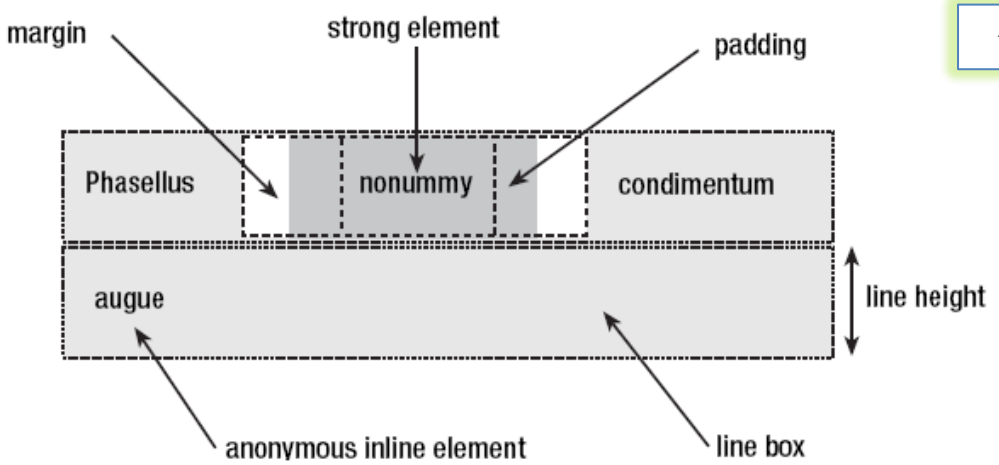
- Unless specified, all boxes start life being positioned in the normal flow
  - The position of an element's box in the normal flow is dictated by that element's position in the HTML code
- **Block-level** boxes will appear vertically one after the other
  - The vertical distance between boxes is calculated by the boxes' vertical margins

```
<div> … </div>
```



Normal flow (static)

- **Inline boxes** are laid out in a line horizontally
  - Horizontal spacing can be adjusted using padding, borders, and margins
  - Vertical padding, borders, and margins will have no effect on the height of an inline box

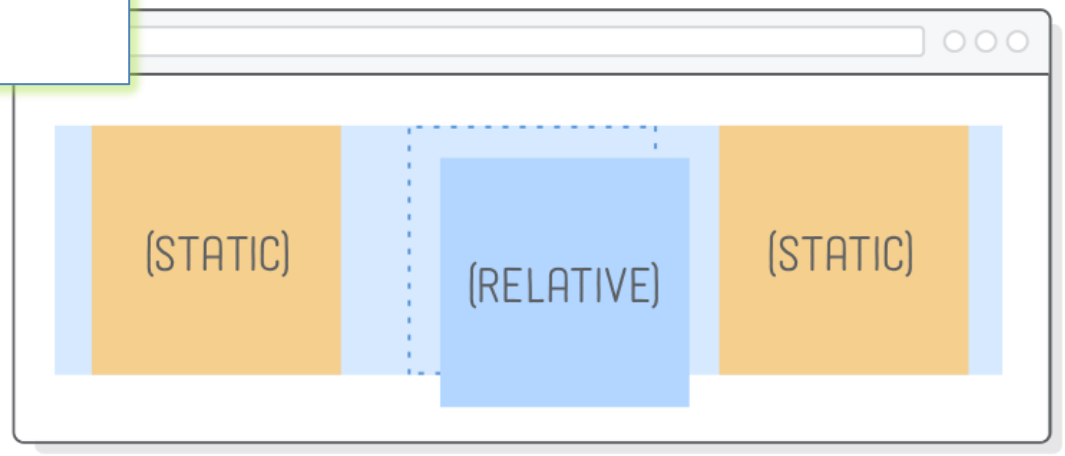



```
<span> … </span>
```

Relative positioning

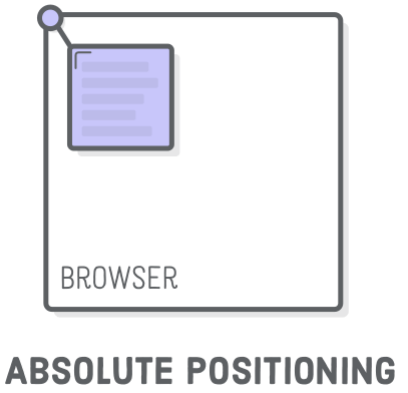
- One element can be shifted “relative” to its normal flow position by setting a vertical and/or horizontal offset

```
.item-relative {
  position: relative;
  left: 20px;
  top: 20px;
}
```



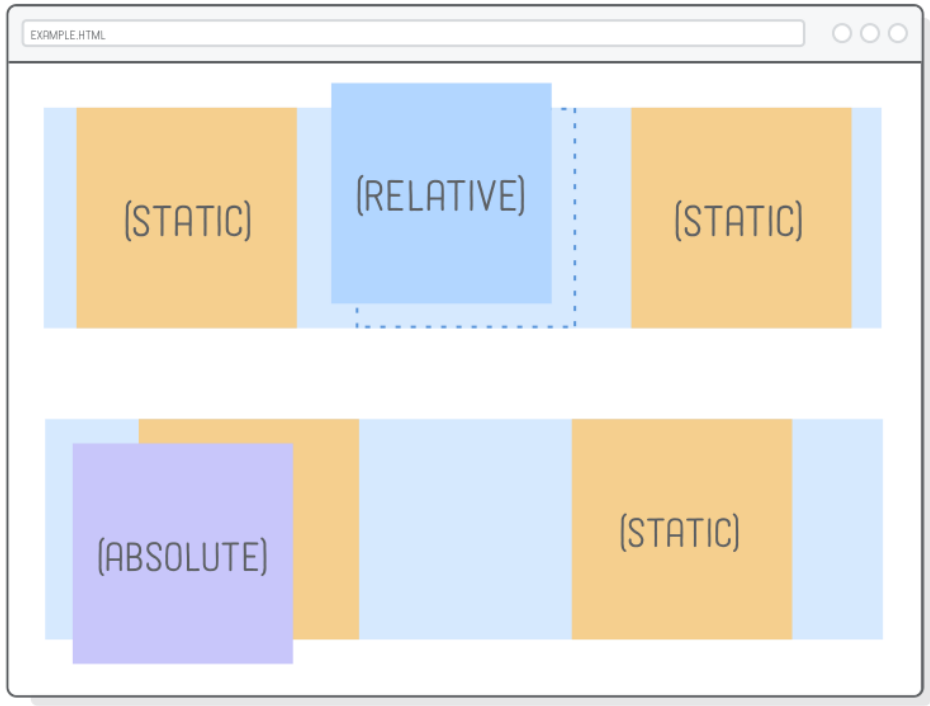
RELATIVE POSITIONING

Absolute positioning

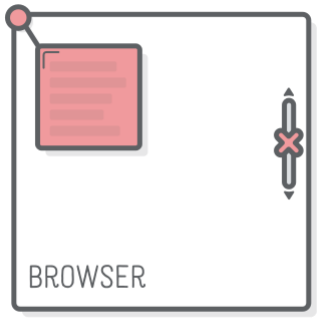


- Takes the element out of the flow of the document, thus taking up no space
- Other elements in the normal flow of the document will act as though the absolutely positioned element was never there

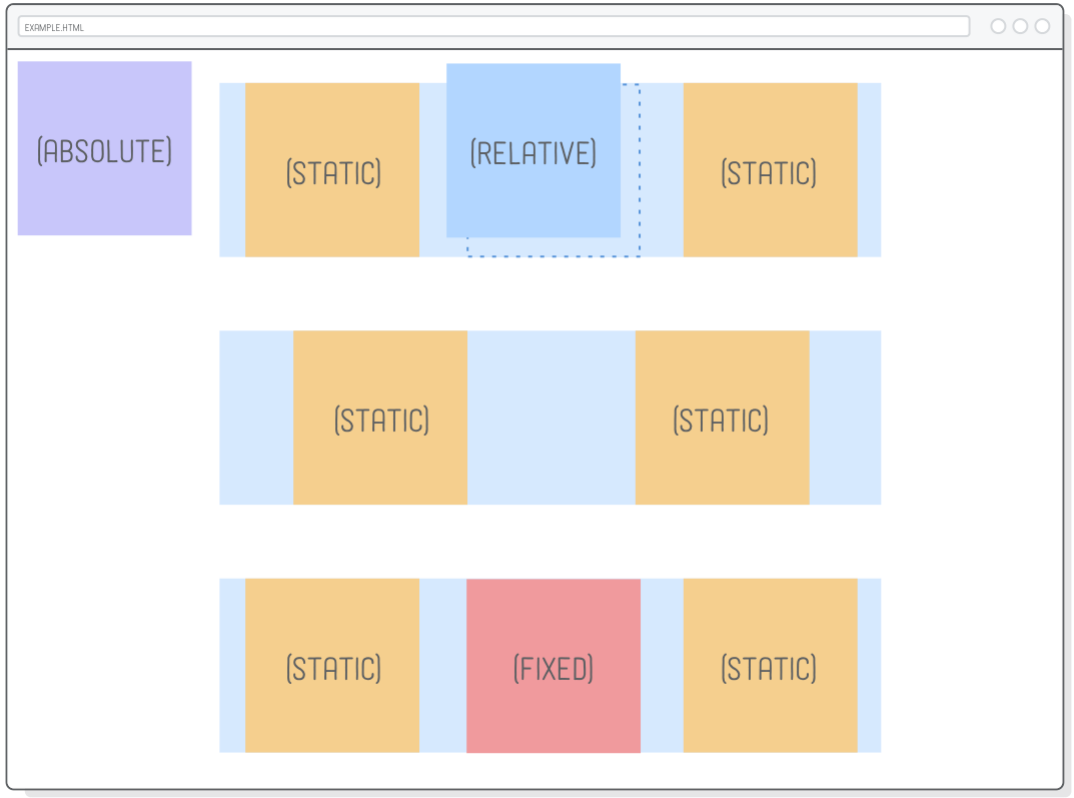
```
.item-absolute {
  position: absolute;
  left: 30px;
  top: 350px;
}
```



Fixed positioning

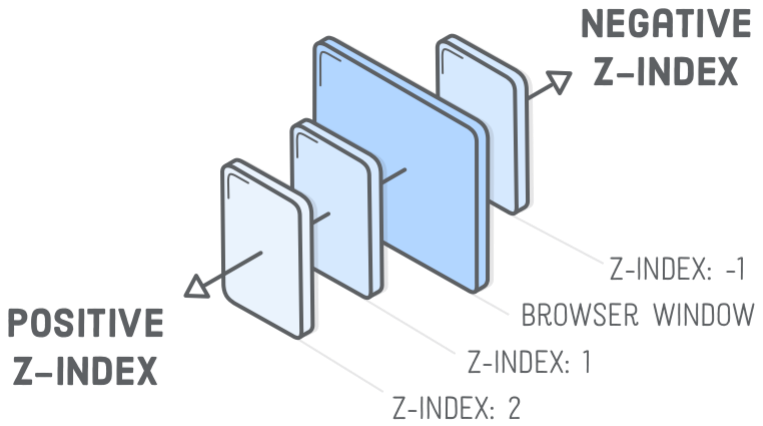


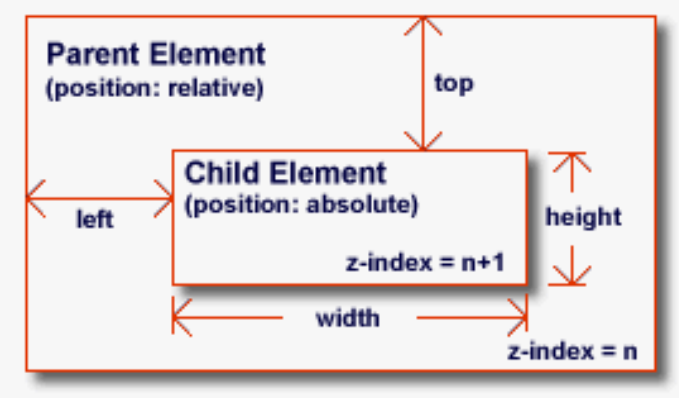
- Has a lot in common with absolute positioning: the element is removed from the normal flow of the page, and the coordinate system is relative to the entire browser window
- The key difference is that fixed elements don't scroll with the rest of the page



Fixed positioning and z-index

- Allows to create elements that always stay at the same position in the window
- 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)





Floating

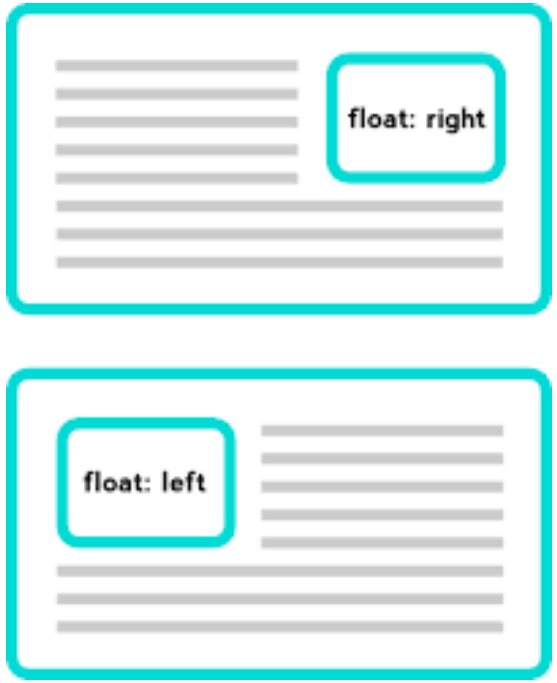
- The CSS **`float`** property gives control over the horizontal position of an element

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

Floating

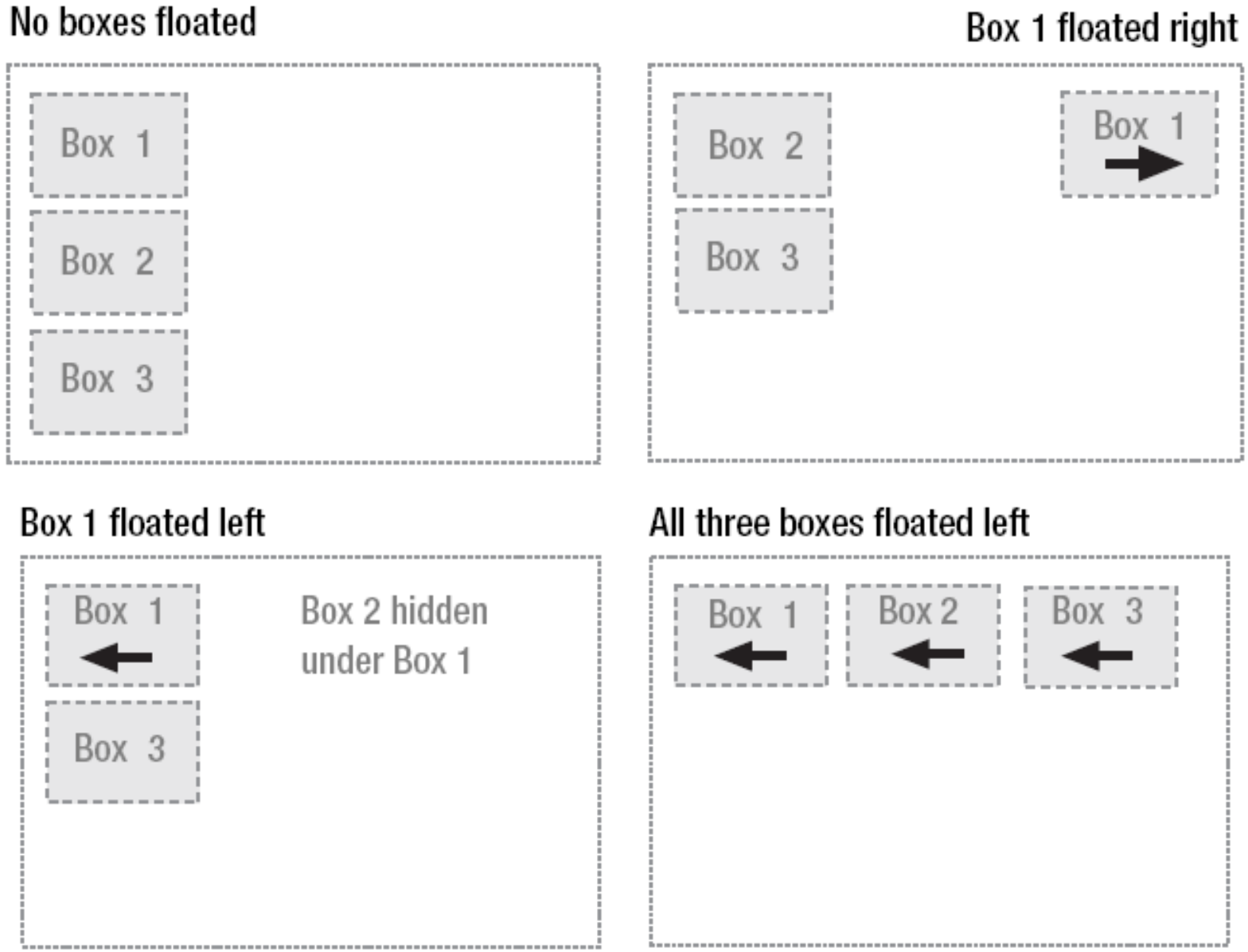
- A floated box can either be shifted to the left or the right until its outer edge touches the edge of its containing box, or another floated box
- Often used for images and when working with layouts

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



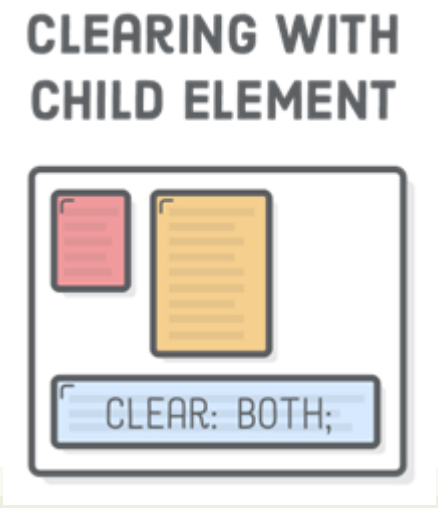
http://www.w3schools.com/Css/css_float.asp

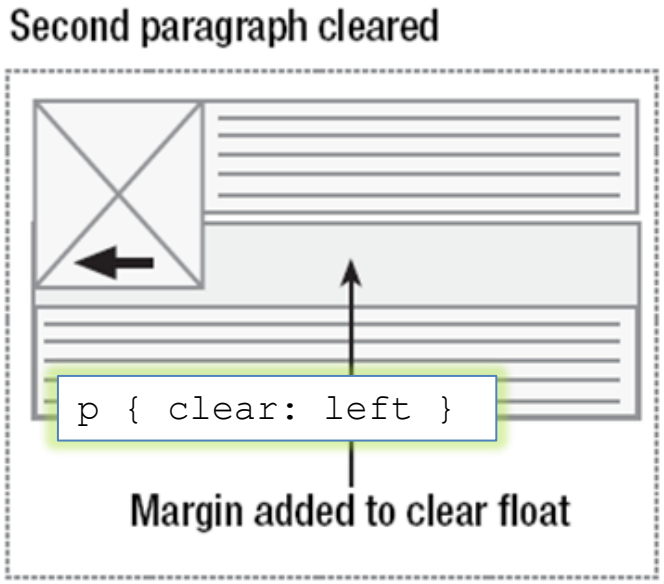
Floating


No boxes floated
Box 1
Box 2
Box 3
Box 1 floated right
Box 2
Box 3
Box 1
Box 1 floated left
Box 1
Box 3
Box 2 hidden
under Box 1
All three boxes floated left
Box 1
Box 2
Box 3

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





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

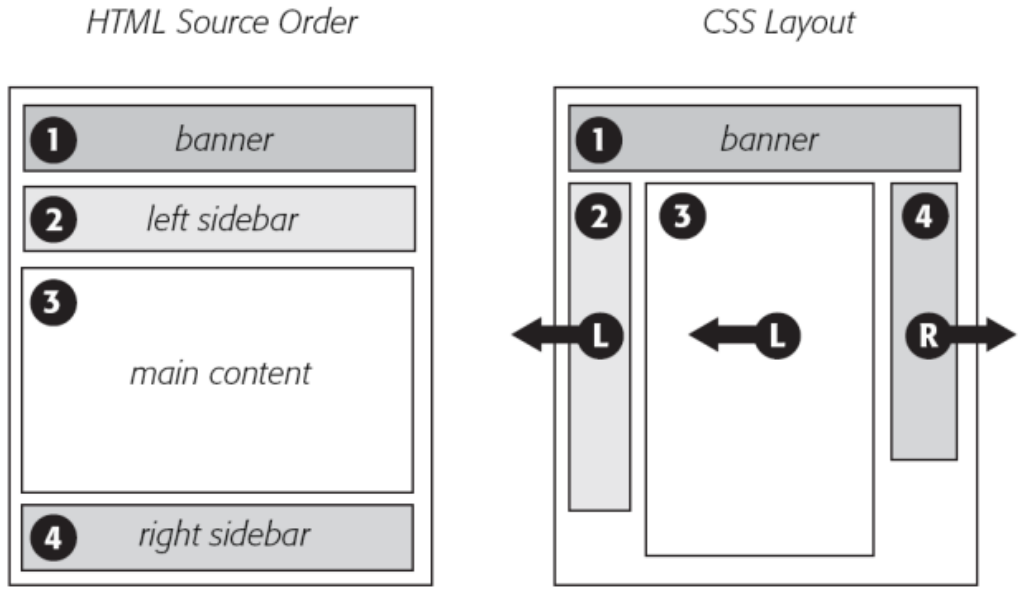
PART 2: CSS FOR PAGE LAYOUT

Cascading Style Sheets

PAGE LAYOUT WITH FLOATS AND GRIDS

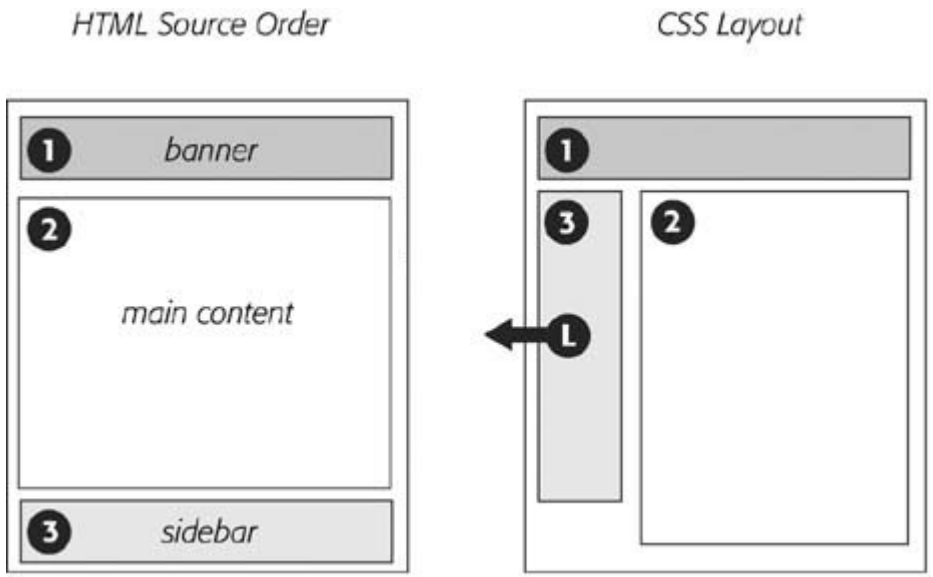
Float-based layouts

- 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

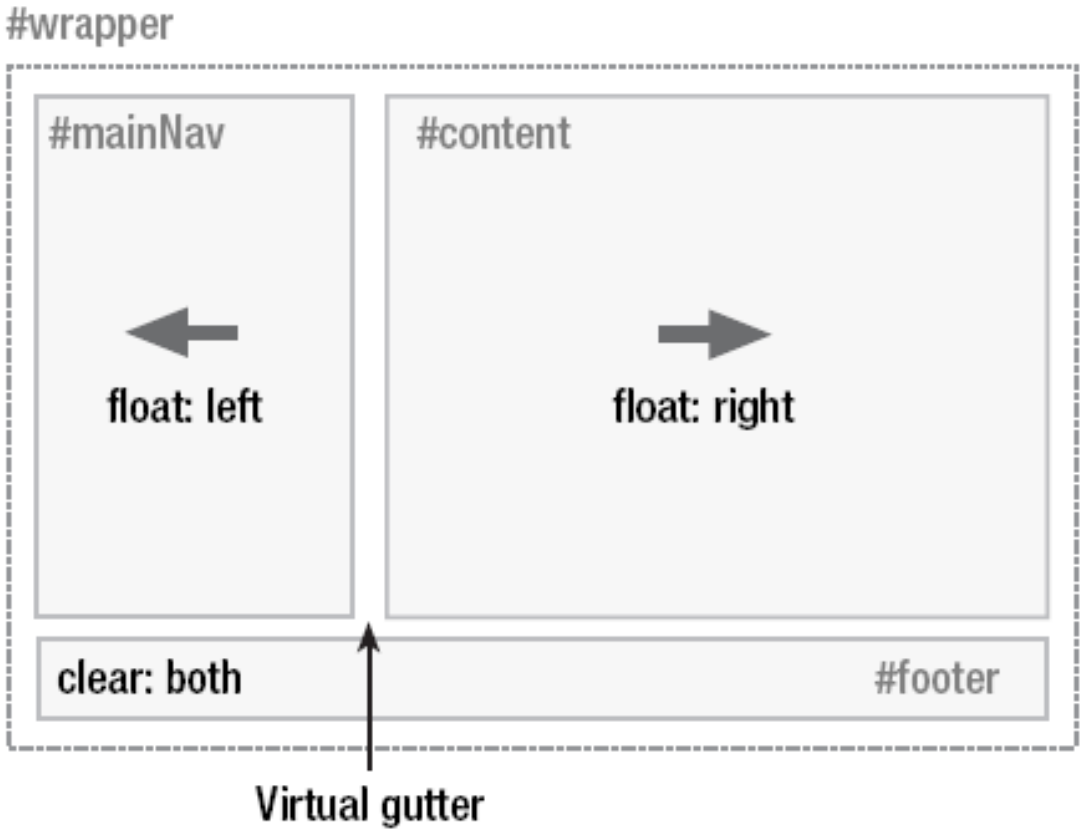
- HTML framework
  - Main navigation on the left side of the page
  - Content on the right
- For accessibility reasons the content area is above the navigation in the source
  - The main content is the most important element in the page and it so should come first in the document
  - There is no point forcing screen-reader users to read through a potentially long list of links before they get to the content



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>
```



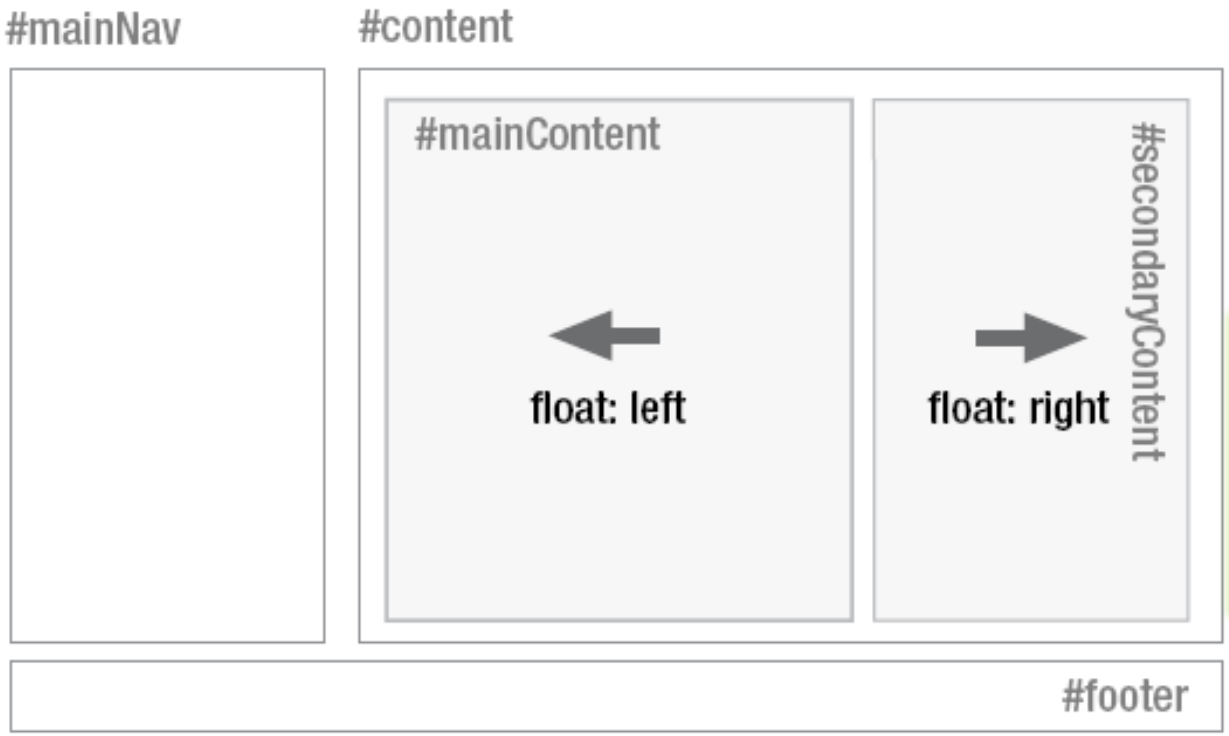
Two-column floated layout

```
#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

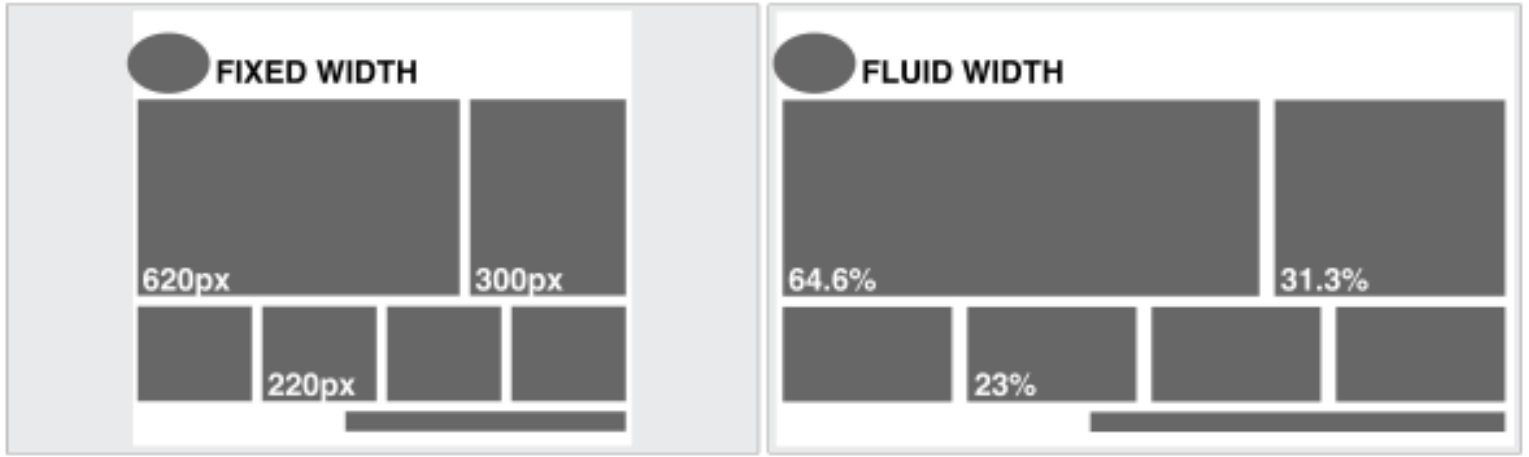



```
#mainContent {
width: 320px;
float: left; }
#secondaryContent {
width: 180px;
float: right; }
```

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

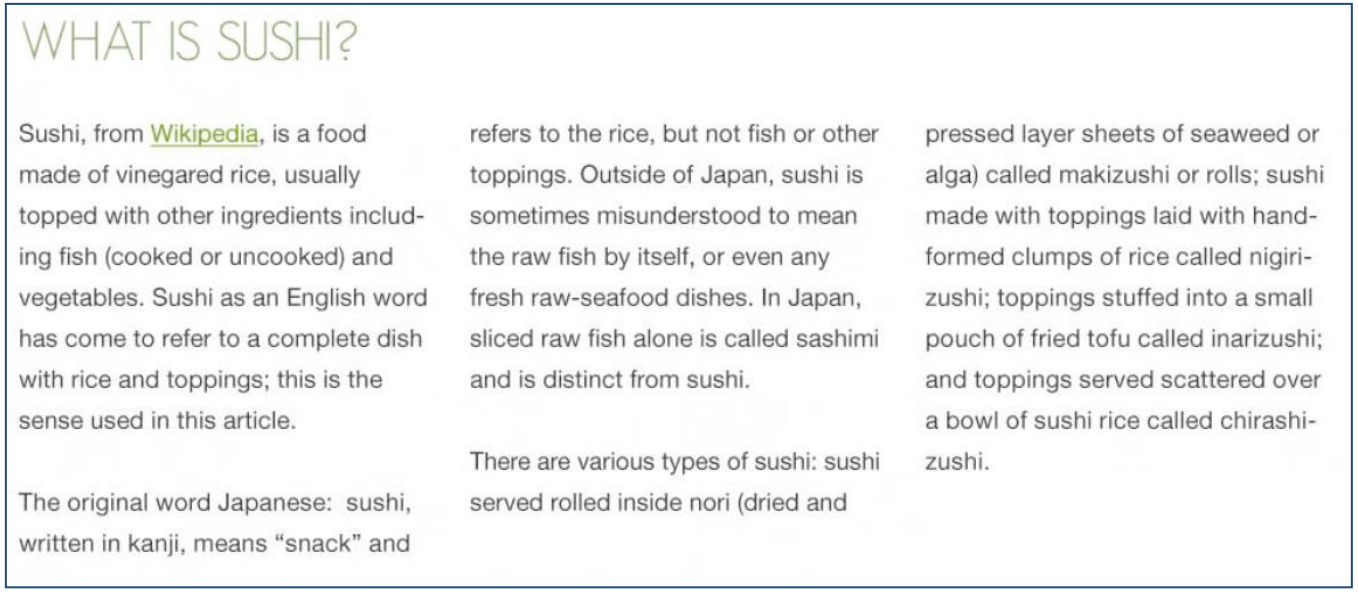
Fixed-width, liquid, and elastic layout

- Different ways of defining column widths
  - Absolute size vs percentage size
- Different behavior: pros and cons



Multi-column layout

- Novelty from CSS3
- Allows to get multi-column layouts without having to use multiple divs



WHAT IS SUSHI?

Sushi, from Wikipedia, is a food made of vinegared rice, usually topped with other ingredients including fish (cooked or uncooked) and vegetables. Sushi as an English word has come to refer to a complete dish with rice and toppings; this is the sense used in this article.

The original word Japanese: sushi, written in kanji, means "snack" and refers to the rice, but not fish or other toppings. Outside of Japan, sushi is sometimes misunderstood to mean the raw fish by itself, or even any fresh raw-seafood dishes. In Japan, sliced raw fish alone is called sashimi and is distinct from sushi.

There are various types of sushi: sushi served rolled inside nori (dried and pressed layer sheets of seaweed or alga) called makizushi or rolls; sushi made with toppings laid with hand-formed clumps of rice called nigiri-zushi; toppings stuffed into a small pouch of fried tofu called inarizushi; and toppings served scattered over a bowl of sushi rice called chirashi-zushi.

```
.entry-content {
  column-count: 2;
  column-gap: 30px; }
```

```
.entry-content {
  column-width: 270px;
  column-gap: 30px; }
```

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.

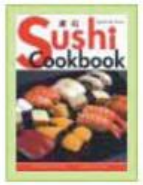
WHAT IS SUSHI?

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.

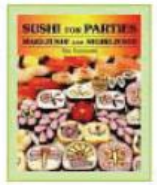
Technically, the word `sushi' refers to the rice, but colloquially, the term is used to describe a finger-size piece of raw fish or shellfish on a bed of vinegared rice or simply the consumption or raw fish in the Japanese style (while sushi is not solely a Japanese invention, these days, the Japanese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

Advanced layout: grid

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

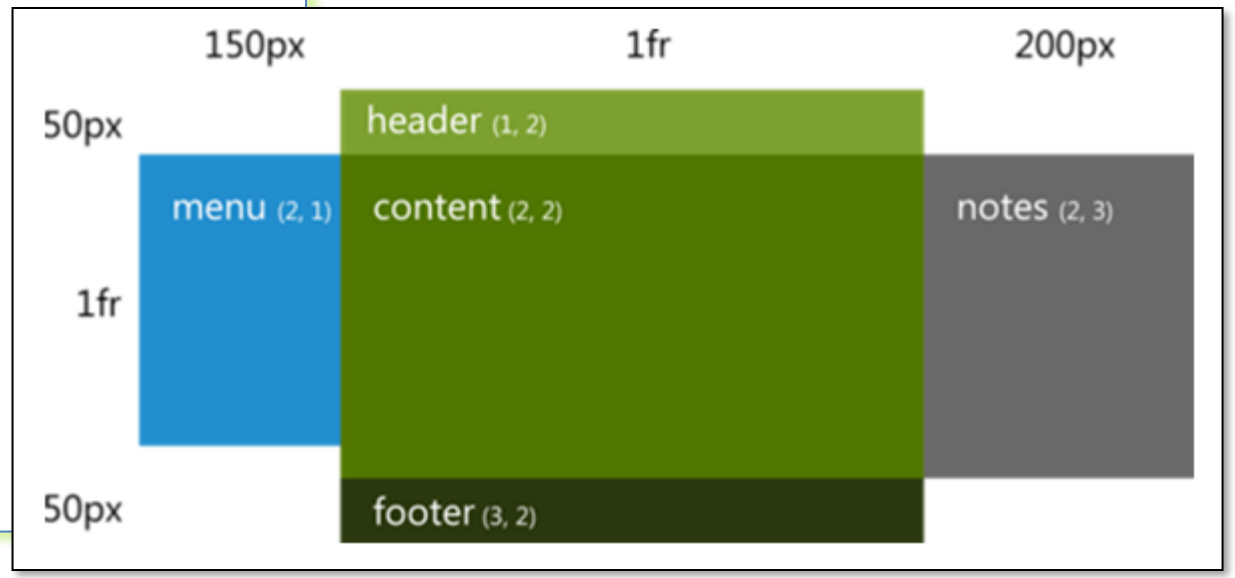
Example

- Classic three-column layout

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

Example

```
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; }
```



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

Cascading Style Sheets

CSS FLEXBOX

Flexbox

- Alternative to floats for defining the overall appearance of a web page
  - Floats allow to horizontally position boxes
  - Flexbox gives complete control over the alignment, direction, order, and size of boxes
- Floats were originally intended for the magazine-style layouts

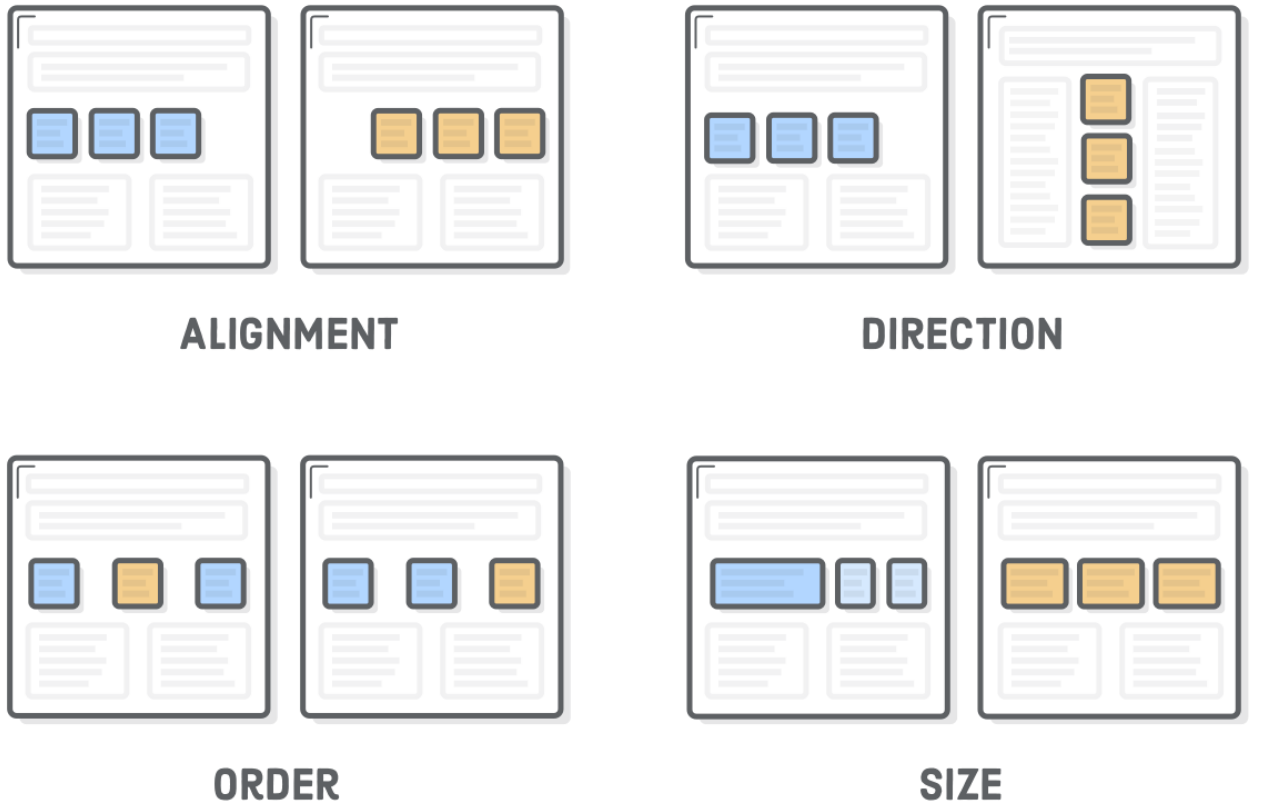
FLOATS

(MAGAZINE-STYLE LAYOUTS)

FLEXBOX

(OVERALL PAGE STRUCTURE)

Flexbox



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

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

"FLEX CONTAINER"

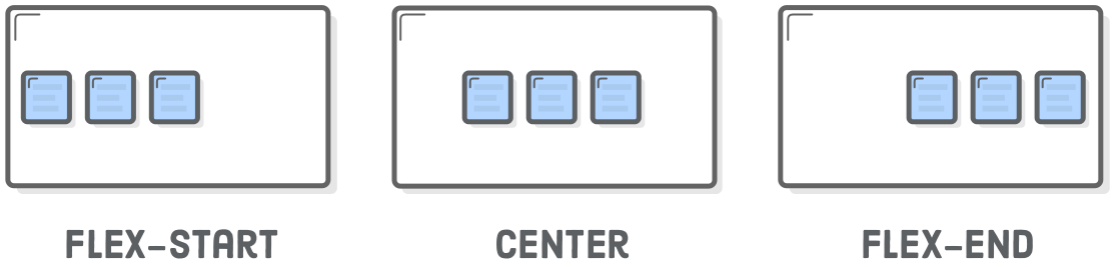
"FLEX ITEMS"

Horizontal alignment

- 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




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

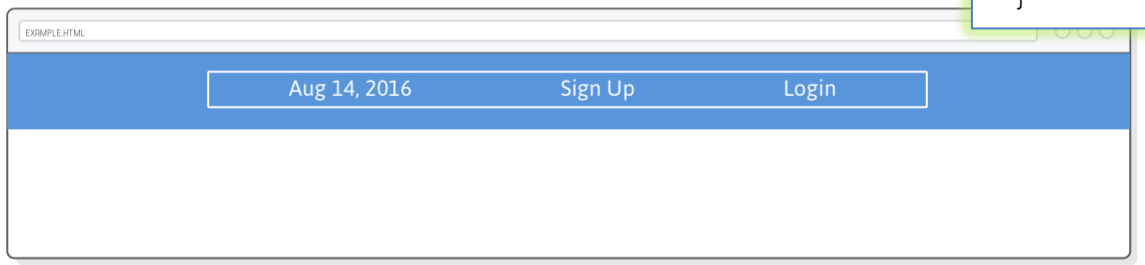
Distribution

- The justify-content property also lets you distribute items equally inside a container

SPACE-AROUND

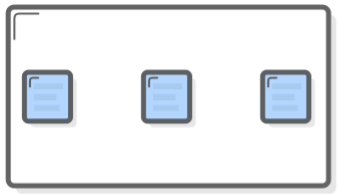
SPACE-BETWEEN

```
.menu {
  border: 1px solid #fff;
  width: 900px; display: flex;
  justify-content: space-around;
}
```



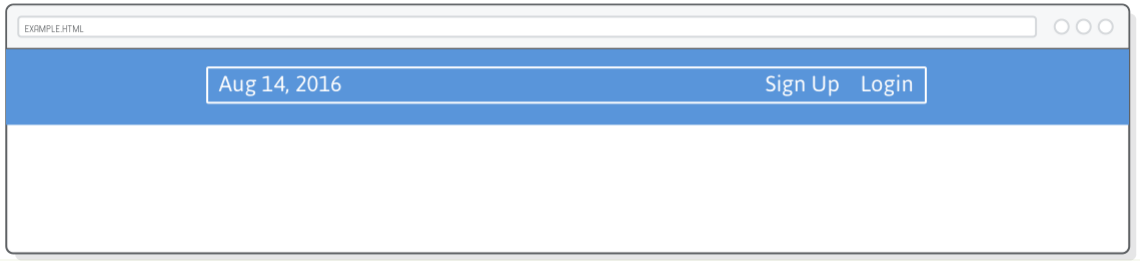
Grouping

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



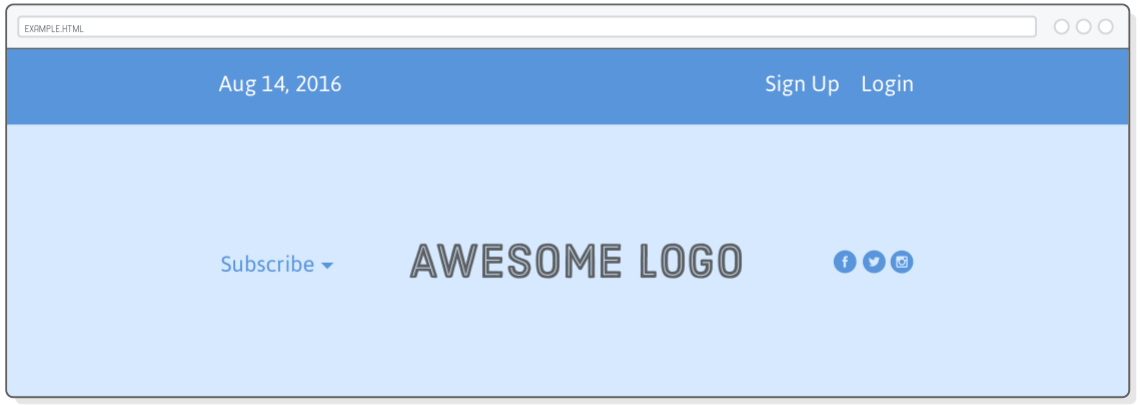
NO GROUPING
(3 FLEX ITEMS)

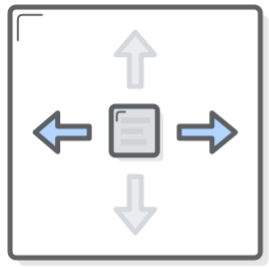
GROUPED ITEMS
(2 FLEX ITEMS)



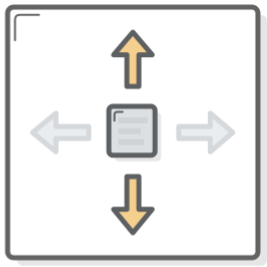
Vertical alignment

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





JUSTIFY-CONTENT

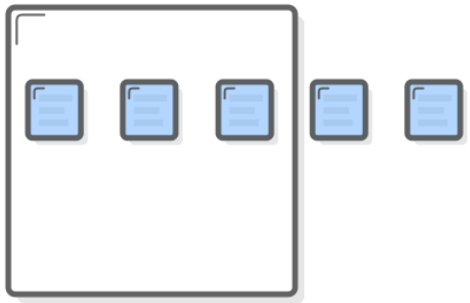


ALIGN-ITEMS

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

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;
}
```

Direction

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

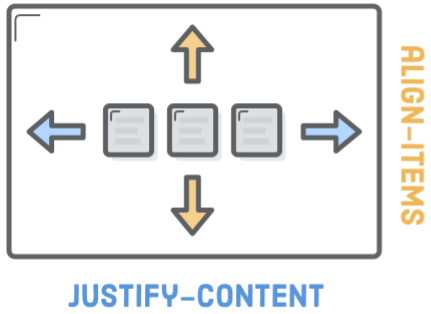
ROW

FLEX-DIRECTION: ROW;

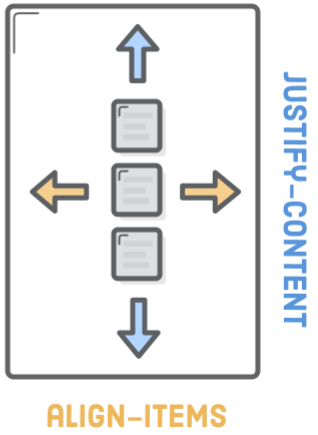
COLUMN

FLEX-DIRECTION: COLUMN;

FLEX-DIRECTION: ROW;

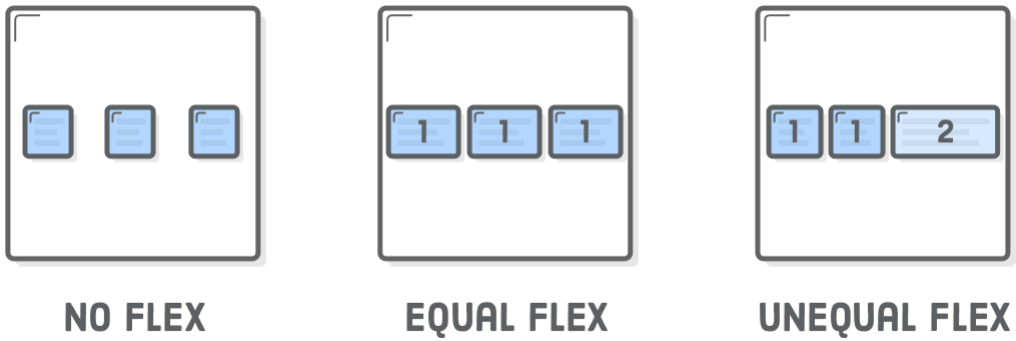


FLEX-DIRECTION: COLUMN;



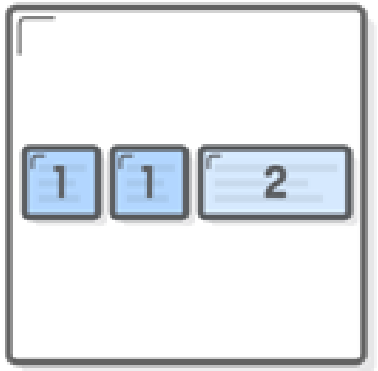
Flexible items

- 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>
```

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

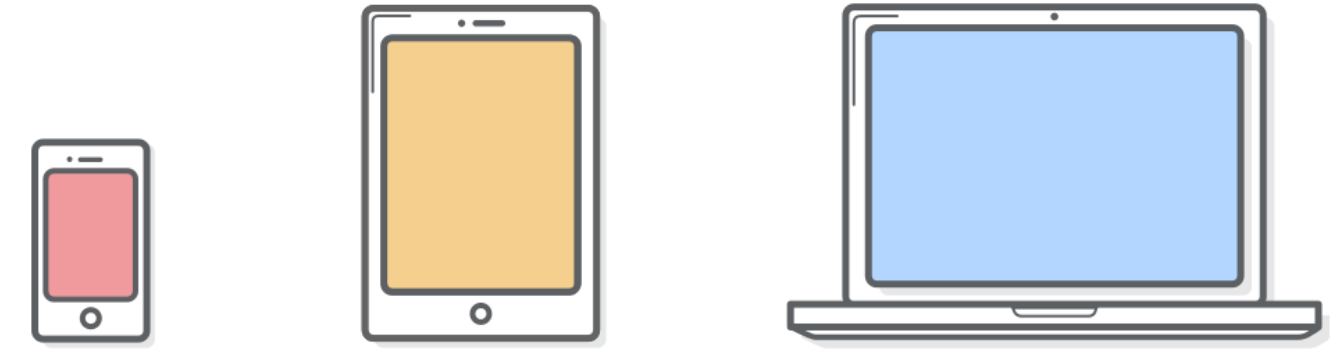
- Interneting is hard flexbox tutorial
  - https://internetingishard.com/html-and-css/flexbox/
- A complete guide to flexbox
  - https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- W3schools
  - https://www.w3schools.com/css/css3_flexbox.asp
- Flexbox, guida pratica
  - http://www.html.it/guide/flexbox-guida-pratica/

Cascading Style Sheets

RESPONSIVE LAYOUT

Responsive design

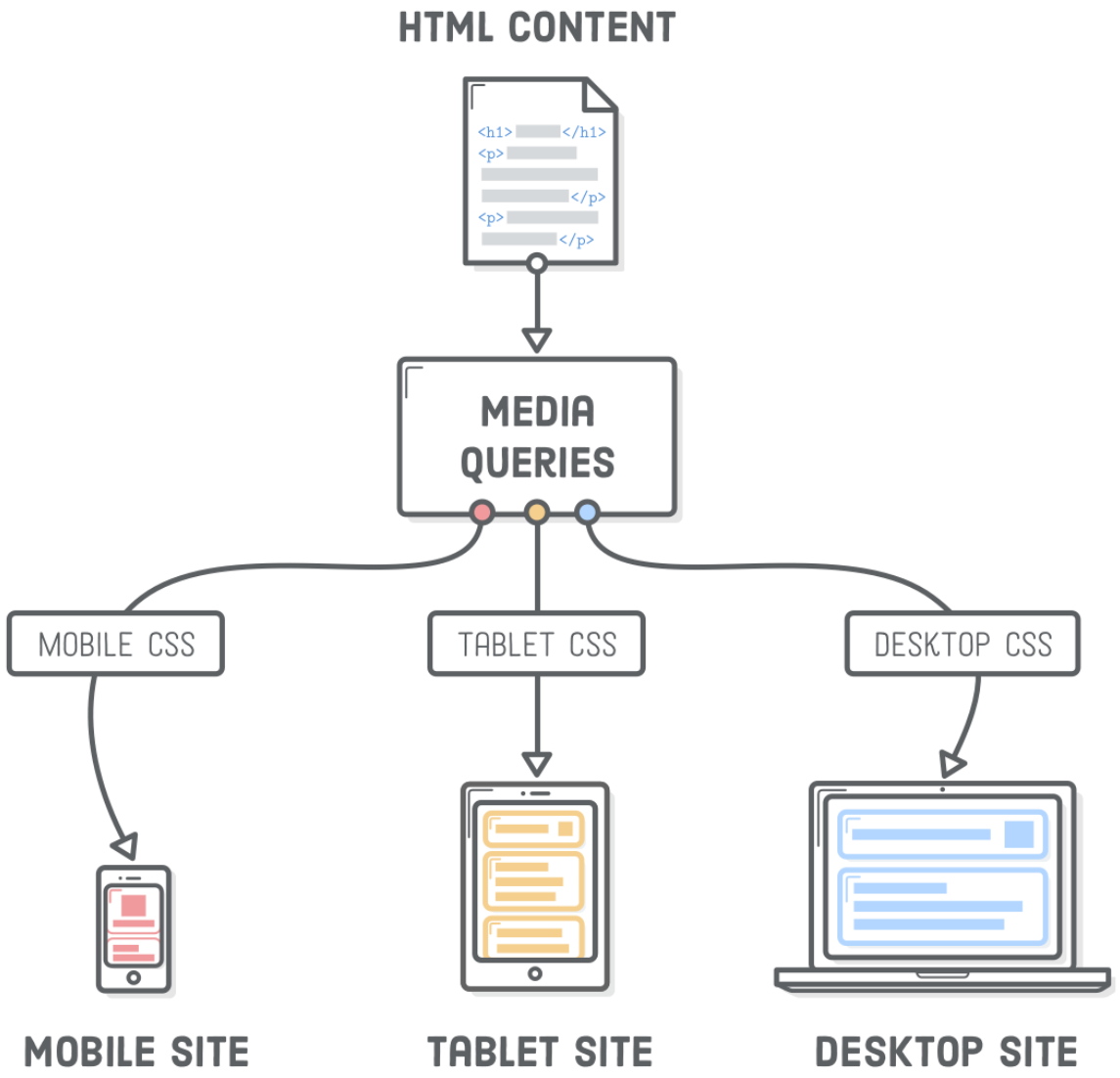
- Display well in everything from widescreen monitors to mobile phones
- Approach to web design to eliminate the distinction between the mobile-friendly version of your website and its desktop counterpart



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

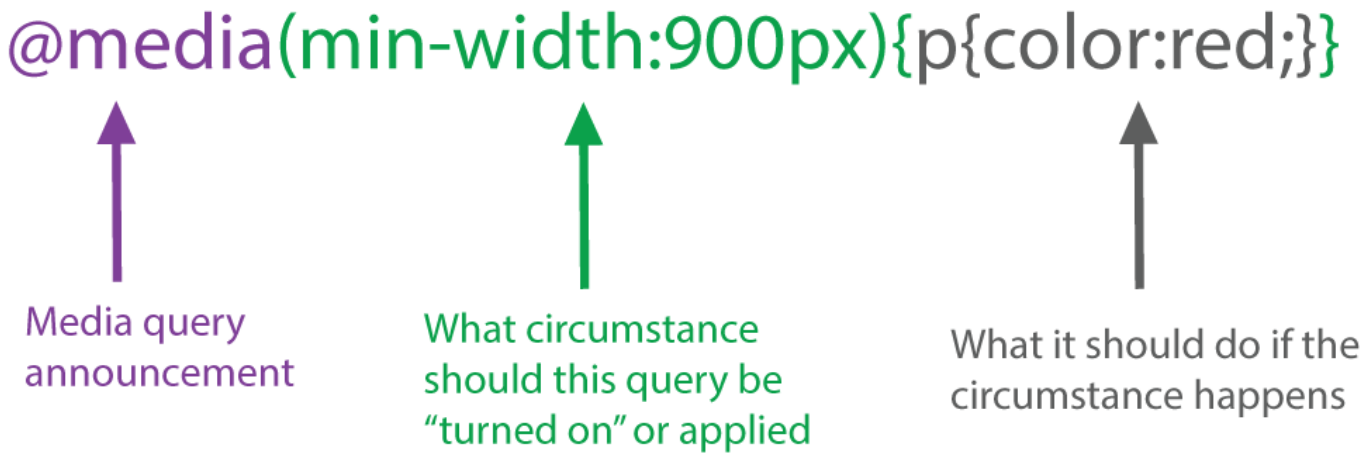
Responsive design

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



Media queries

- Example: if the screen is wider than 900 pixels then the color of the text should be red

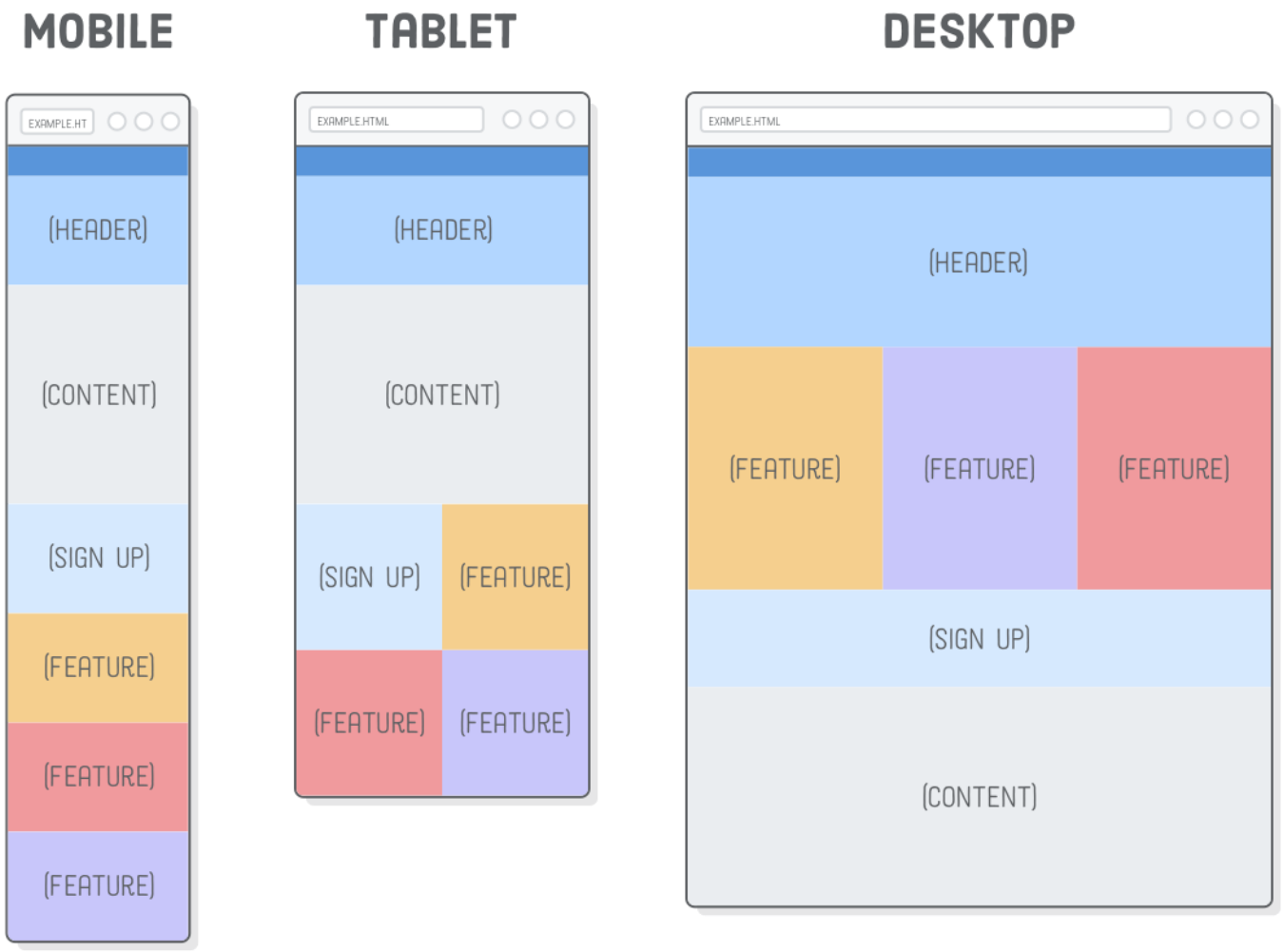


Media queries

http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

- Media queries look at the capability of the device, and can be used to check many things, such as
  - Width and height of the viewport
  - Width and height of the device
  - Orientation (is the tablet/phone in landscape or portrait mode?)
  - Resolution
  - … and much more
- List of supported media features

Layout for responsive design



Breakpoints

- Use device resolution:
  - the smart phone (usually the iPhone at 320px and 480px)
  - the tablet (usually the iPad at 768px and 1024px)
  - anything above 1024px
- Mobile first: the best practice is to design for your smallest viewport first… as you expand that view there will come a point at which the design looks terrible… this is where you add a breakpoint

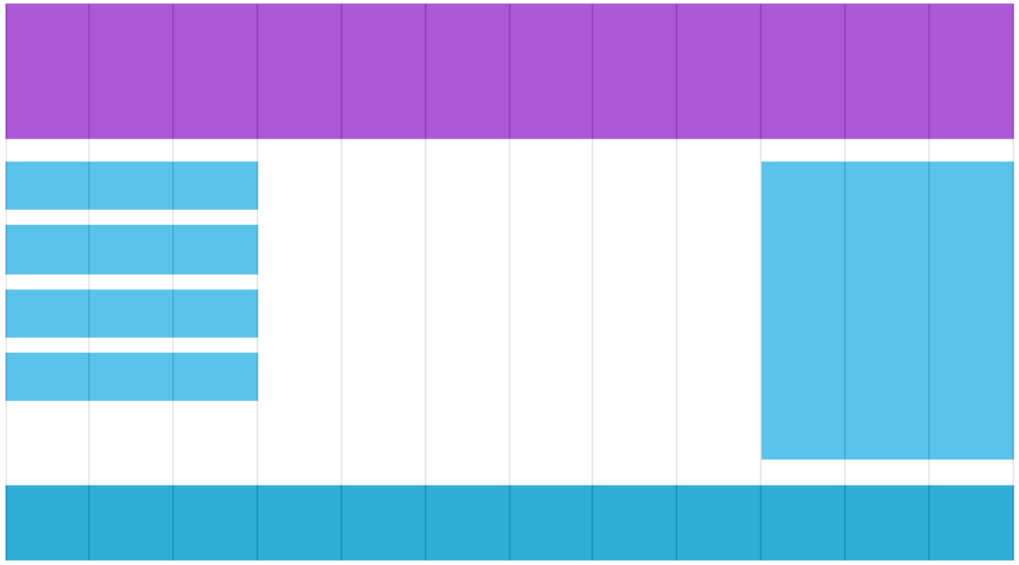
Example: device breakpoints

```
/* Smartphones (portrait and landscape) ----------- */
@media only screen and (min-device-width : 320px) and
(max-device-width : 480px)
{ /* Styles */ }

/* Smartphones (landscape) ----------- */
@media only screen and (min-width : 321px)
{ /* Styles */ }

/* Smartphones (portrait) ----------- */
@media only screen and (max-width : 320px)
{ /* Styles */ }

/* iPads (portrait and landscape) ----------- */
@media only screen and (min-device-width : 768px) and
(max-device-width : 1024px)
{ /* Styles */ }

...
```

https://responsivedesign.is/develop/browser-feature-support/media-queries-for-common-device-breakpoints/

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



Cascading Style Sheets

THE BOOTSTRAP FRAMEWORK

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/
  - https://getbootstrap.com/docs/4.4/

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

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/v4.4.1/bootstrap-4.4.1-dist.zip
  - npm install bootstrap

Bootstrap CDN

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
```

```
<script src="https://code.jquery.com/jquery-3.4.1.slim.min.js"
integrity="sha384-J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoRSJoZ+n"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
```

Page structure

- Start with **Basic template**
  - https://getbootstrap.com/docs/4.4/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/4.4/examples/
- Remember to include your page content in a
  - `<div class="container">`

Starter Template

Copy

```
<!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>
```

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

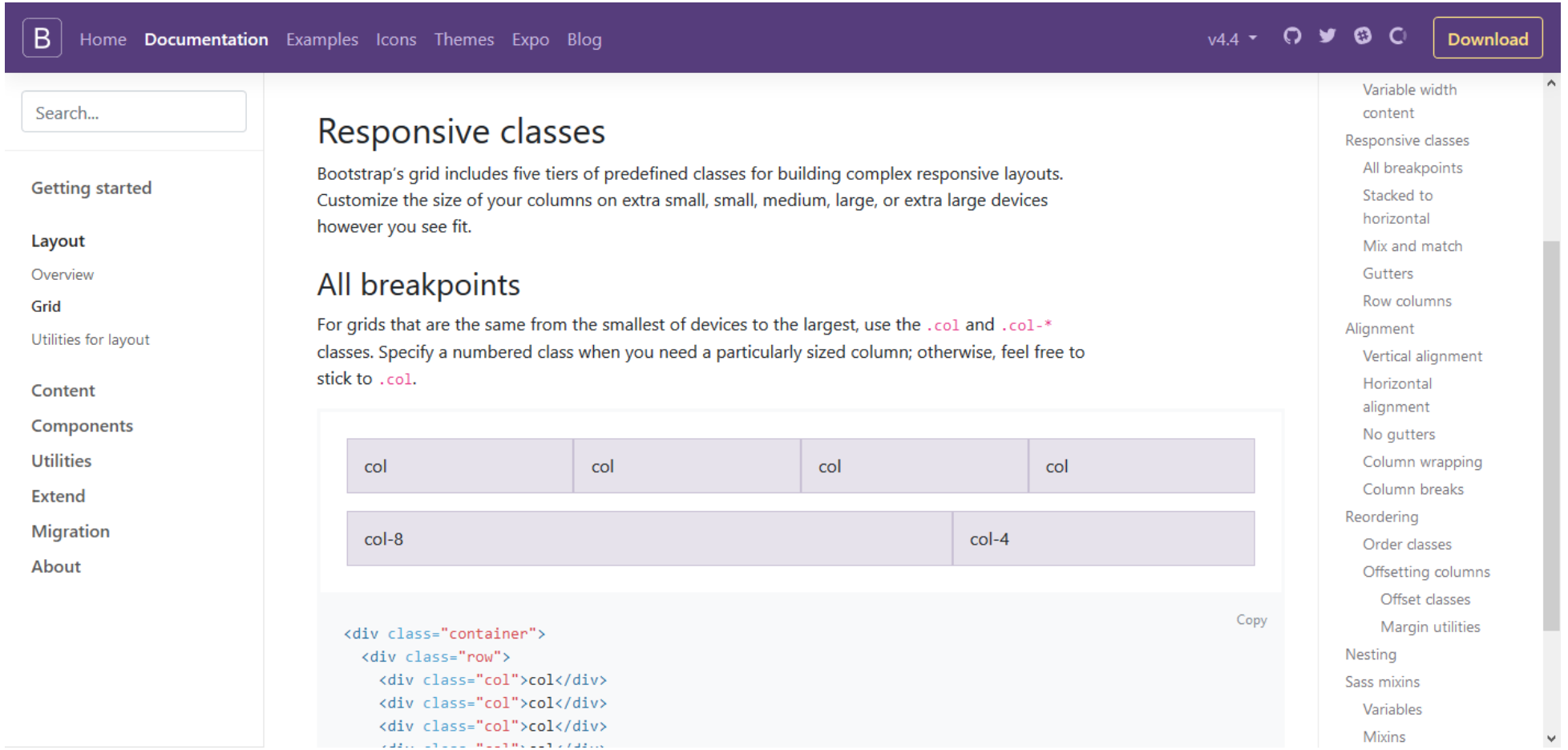
- Based on flexbox
- Uses a 12-column virtual grid
- The container is divided in .row blocks
- Each Row contains columns .col
  - By default, all columns have equal width
  - May specify spanning over more columns (.col-4)
  - May specify a breakpoint for that column to be visualized (.col-sm or .col-sm-4)
- 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/4.4/layout/grid/



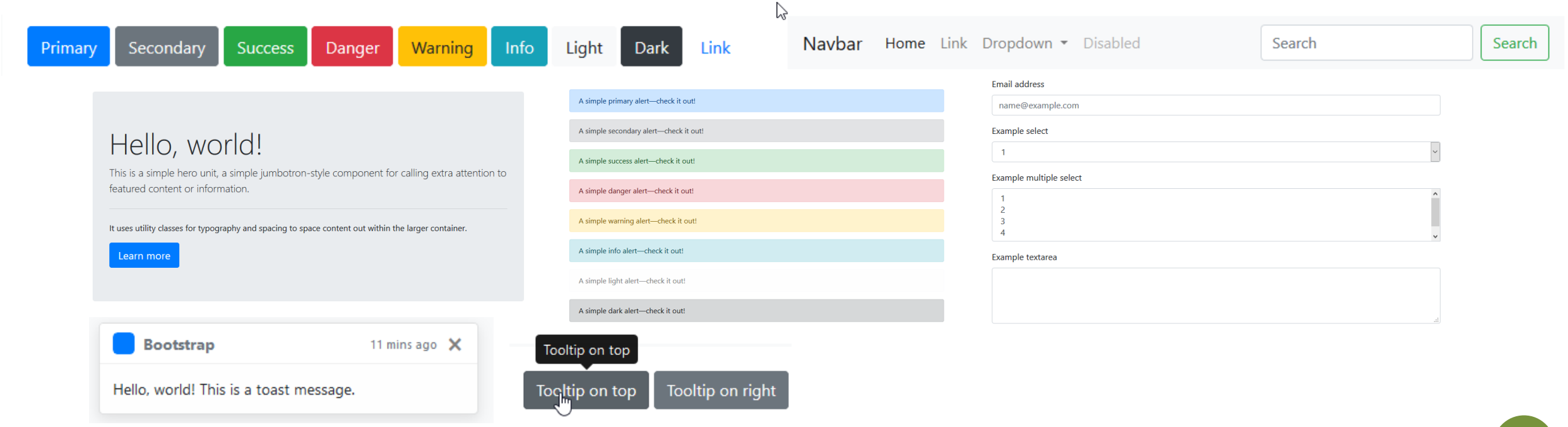
Page content

- 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/4.4/content/reboot/

Components

- Ready-to-use classes for creating common UI elements
- See: https://getbootstrap.com/docs/4.4/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/