Cascading Style Sheets (CSS)

Styling the Web

Luigi De Russis

Photo by Marcus Ganahl on Unsplash

Politecnico di Torino

Goal

- Styling web content
    - Definitions of the Cascading Style Sheets (CSS)
    - Basic concepts
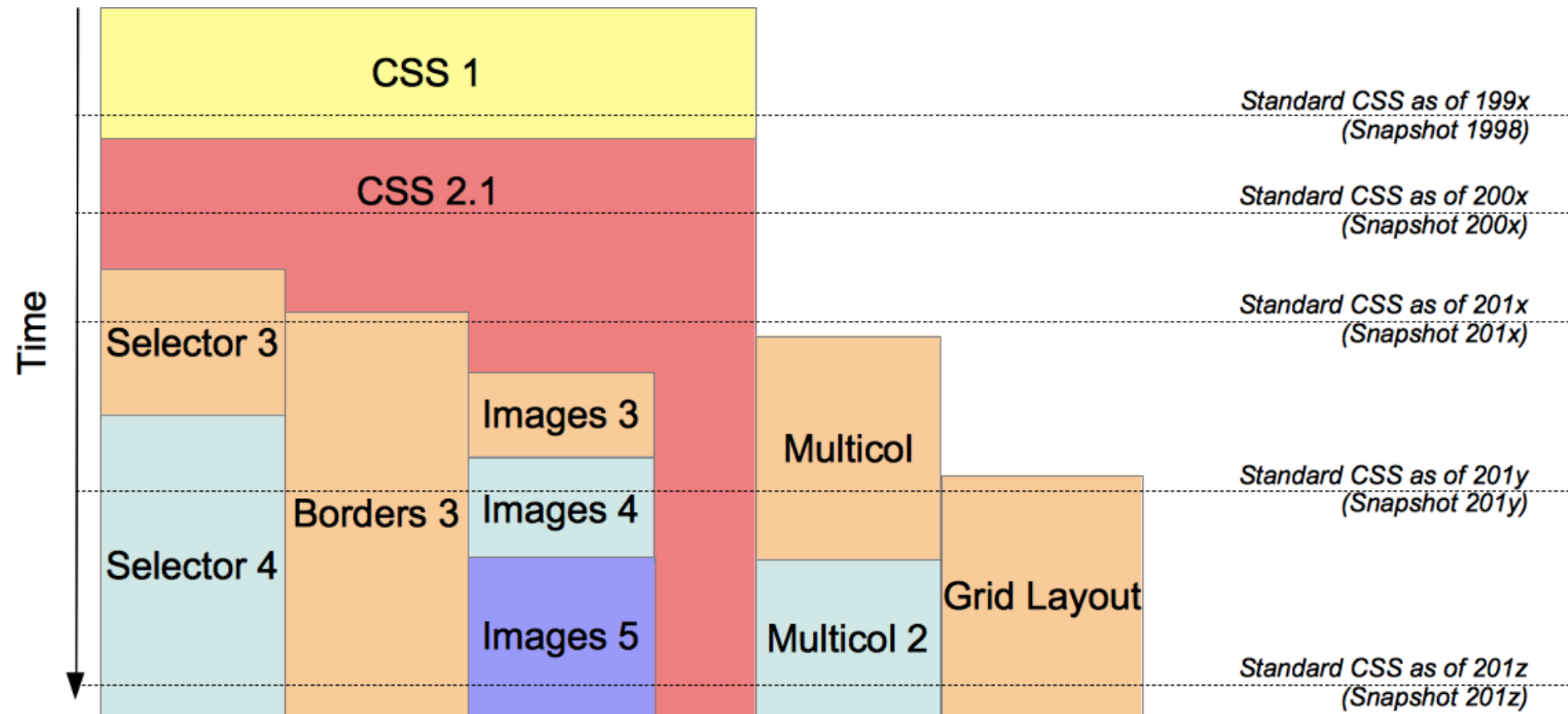   - How to use CSS in a web page

Outline

- CSS syntax
- CSS selectors
- CSS cascading
- CSS box model
- CSS positioning schemes

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

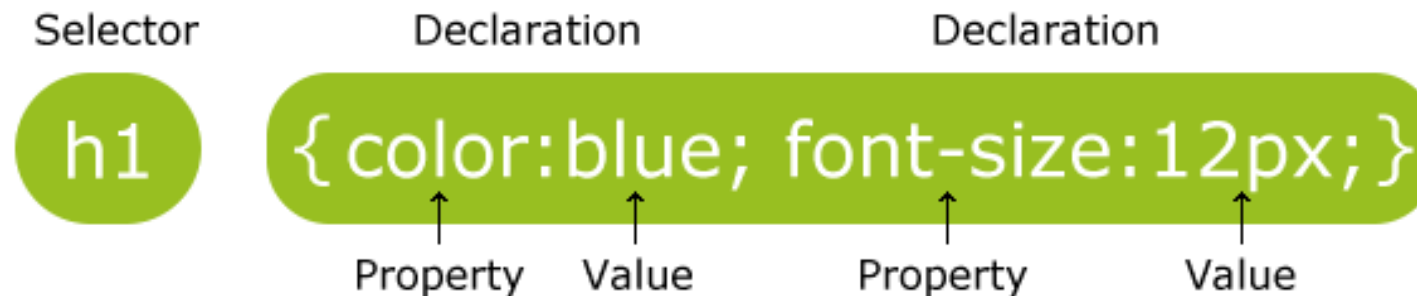https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3
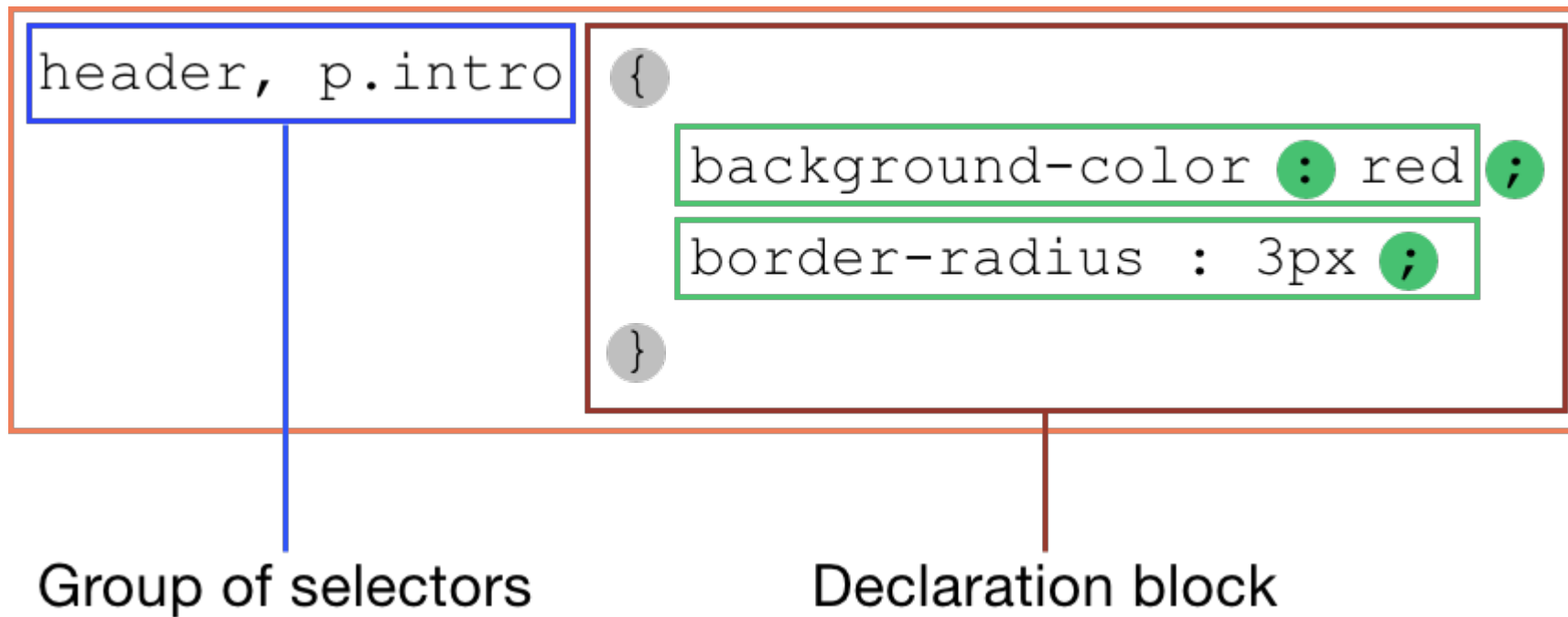
Cascading Style Sheets

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

Selector   Declaration   Declaration

h1   { color:blue; font-size:12px;}

Property   Value   Property   Value

General syntax


```
header, p.intro { 
    background-color : red ;
    border-radius : 3px ;
}
```

Group of selectors

Declaration block

HTML nested structure
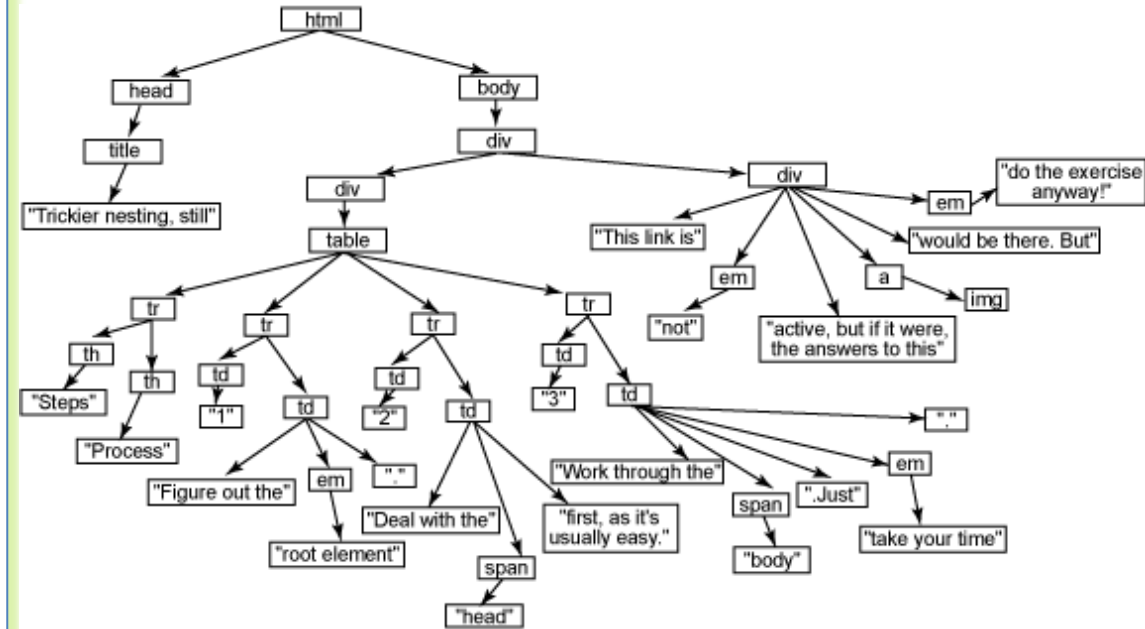
```
<html lang="en">
<head>
  <title>Trickier nesting, still</title>
</head>
<body>
  <div>
    <div>
      <table>
        <tr><th>Steps</th><th>Processes</th></tr>
        <tr><td>1</td><td>Figure out the <em>root element</em>.</td></tr>
        <tr><td>2</td><td>Deal with the <span>head</span> first 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>
```
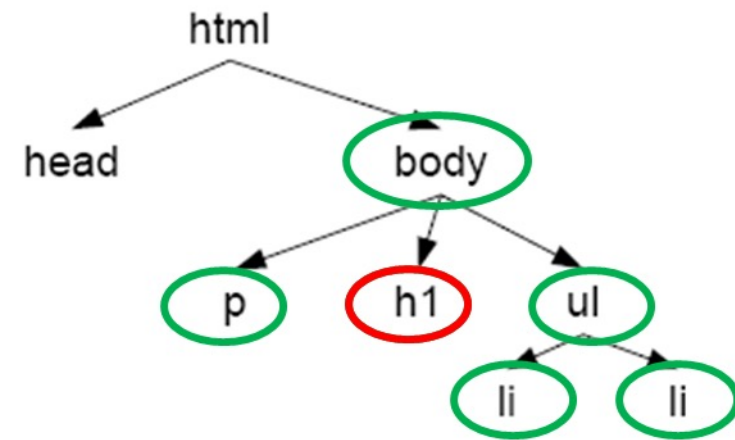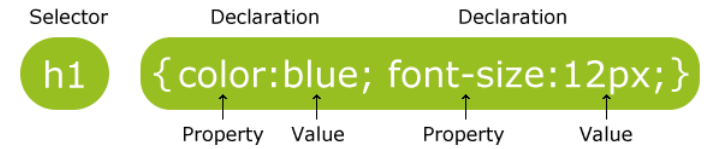
Tree structure and inheritance

- HTML documents are trees

- Styles are inherited along trees

- When two rules are in conflict the most specific wins
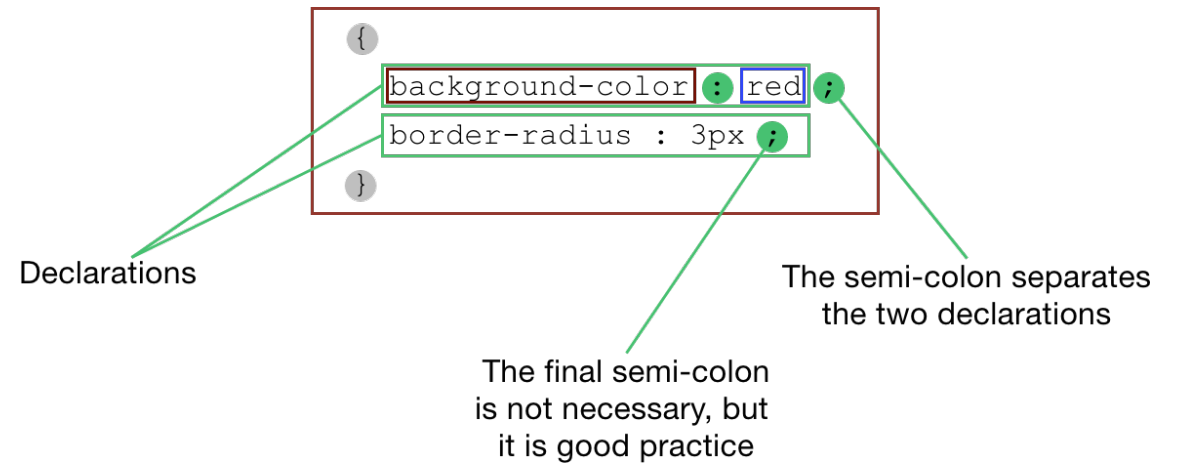
- Example
  - body {color: green}
  - h1 {color: red}

Declaration Syntax

Selector — h1

Declaration, Declaration — {color:blue; font-size:12px;}

Property, Value, Property, Value

A CSS declaration :

background-color : red

Property
The COLON separates the two entities
Value

{
background-color : red ;
border-radius : 3px ;
}

Declarations

The final semi-colon is not necessary, but it is good practice

The semi-colon separates the two declarations
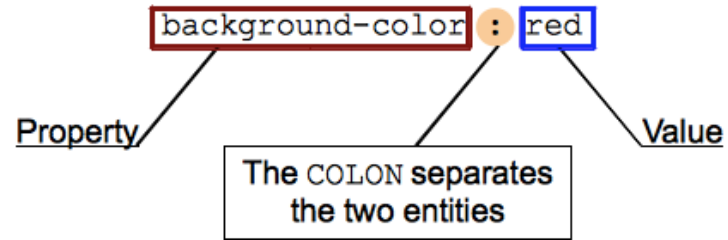
CSS properties (200+)

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

align-content align-items align-self all animation animation-delay animation-direction animation-duration animation-fill-mode animation-iteration-count animation-name animation-play-state animation-timing-function backface-visibility background background-attachment background-blend-mode background-clip background-color background-image background-origin background-position background-repeat background-size border border-bottom border-bottom-color border-bottom-left-radius border-bottom-right-radius border-bottom-style border-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

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:

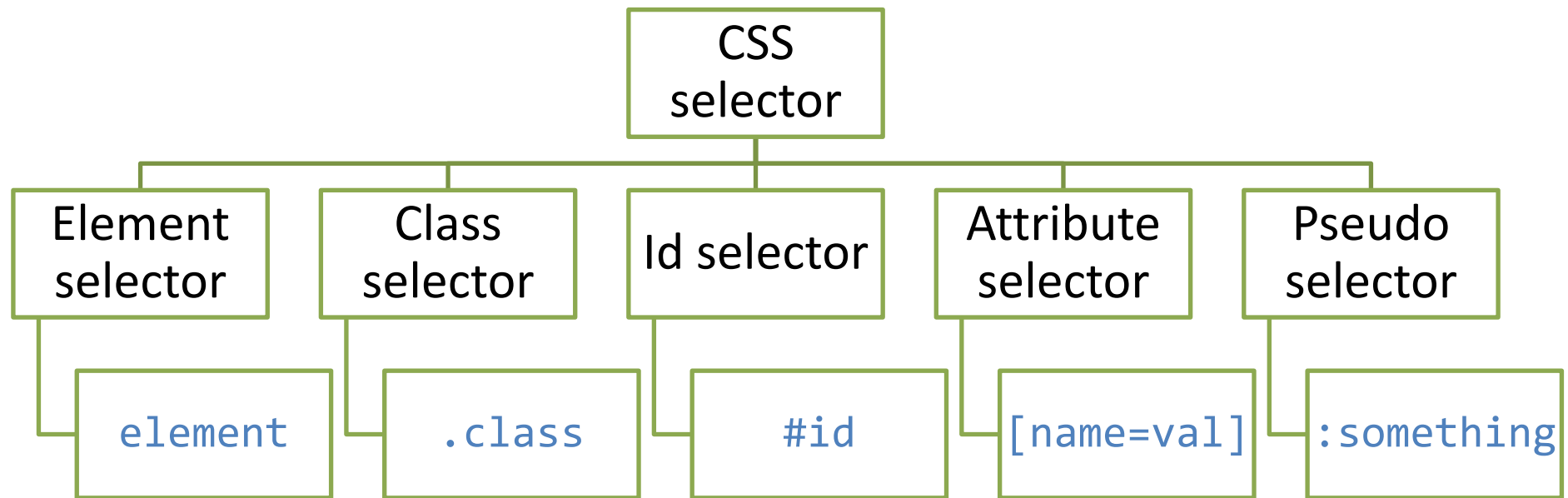| Unit	Description
em	Relative to the font-size of the element. 2em means 2 times the size of the font of the current element
rem	Relative to font-size of the root element (<html>) of the HTML page (Root EM)
vw	Relative to 1% of the width of the viewport, i.e., the browser window size
vh	Relative to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

- 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"

```
                              CSS
                            selector

  Element      Class                    Attribute     Pseudo
  selector    selector    Id selector   selector      selector

   element      .class        #id       [name=val]   :something
```
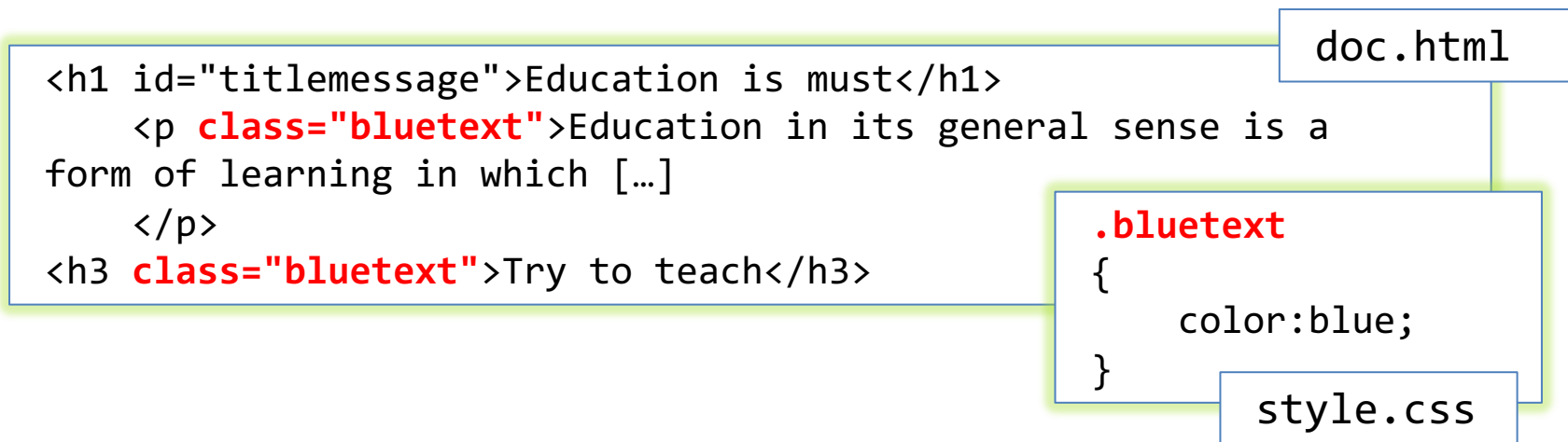
Element selector

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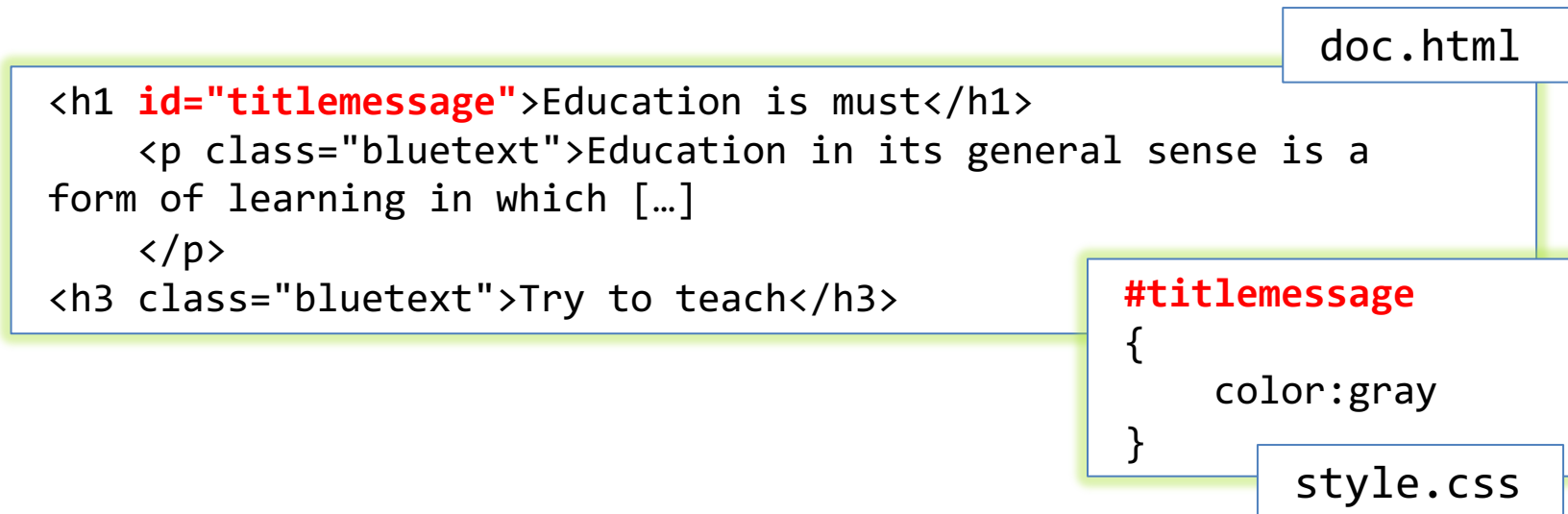
```
h1
{
    color:red;
}
```

style.css

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

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

doc.html

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

style.css

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

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]`
  $\rightarrow$ may be combined
- `S1, S2` $\rightarrow$ S1 union S2
- `S1 S2` $\rightarrow$ S2 nested within S1
- `S1 > S2` $\rightarrow$ S2 if a child of S1
- `S1 + S2` $\rightarrow$ S2 if it comes after a S1
- `S1 ~ S2` $\rightarrow$ 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 <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 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;}
```

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
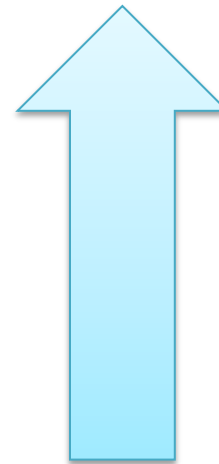
External style

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

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

style.css

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

Internal style

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

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

Inline style

- `<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:*	*if rules have* **the same** *specificity:*
more general	declared earlier
⬇	⬇
more specific	declared later

⬇

Inline Styles

⬇

Author !important Styles

⬇

User !important Styles
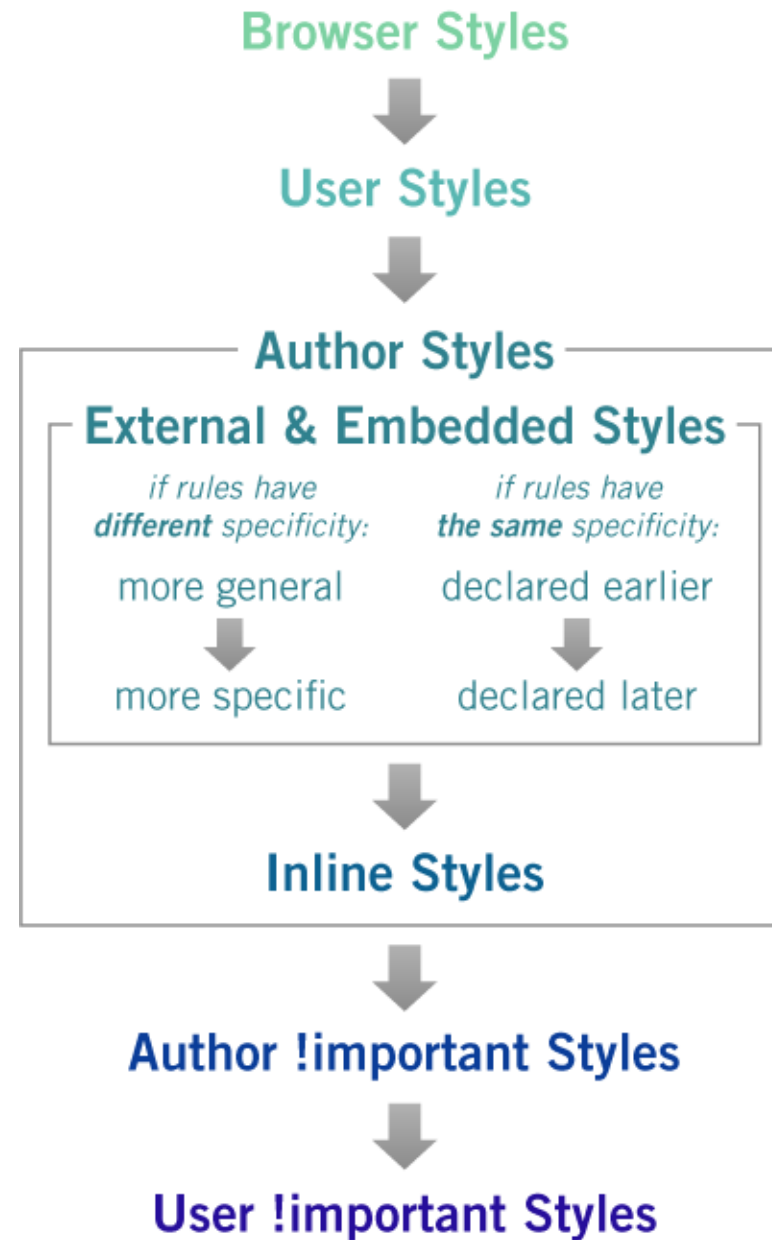
CSS specificity

- 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)

The definitive guide to CSS styling order

Includes CSS stylings for SVG

Styling methods

Inheritance

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

Inline attributes (for SVG)

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

Style sheet

Embedded / External styles

Specificity

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.
```

Tags (eg: p, div)

Ordering

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
```

Classes (eg: .MyClass)

Ordering

Within specificity, ordering rules still applies.

```
#myID { color: red; }
#myID { color: blue; }

Blue will be applied.
```

ID: (eg: #myID)

Ordering

```
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
```

Inline styles

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

Style sheet

Inline styles

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

vecta.io

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
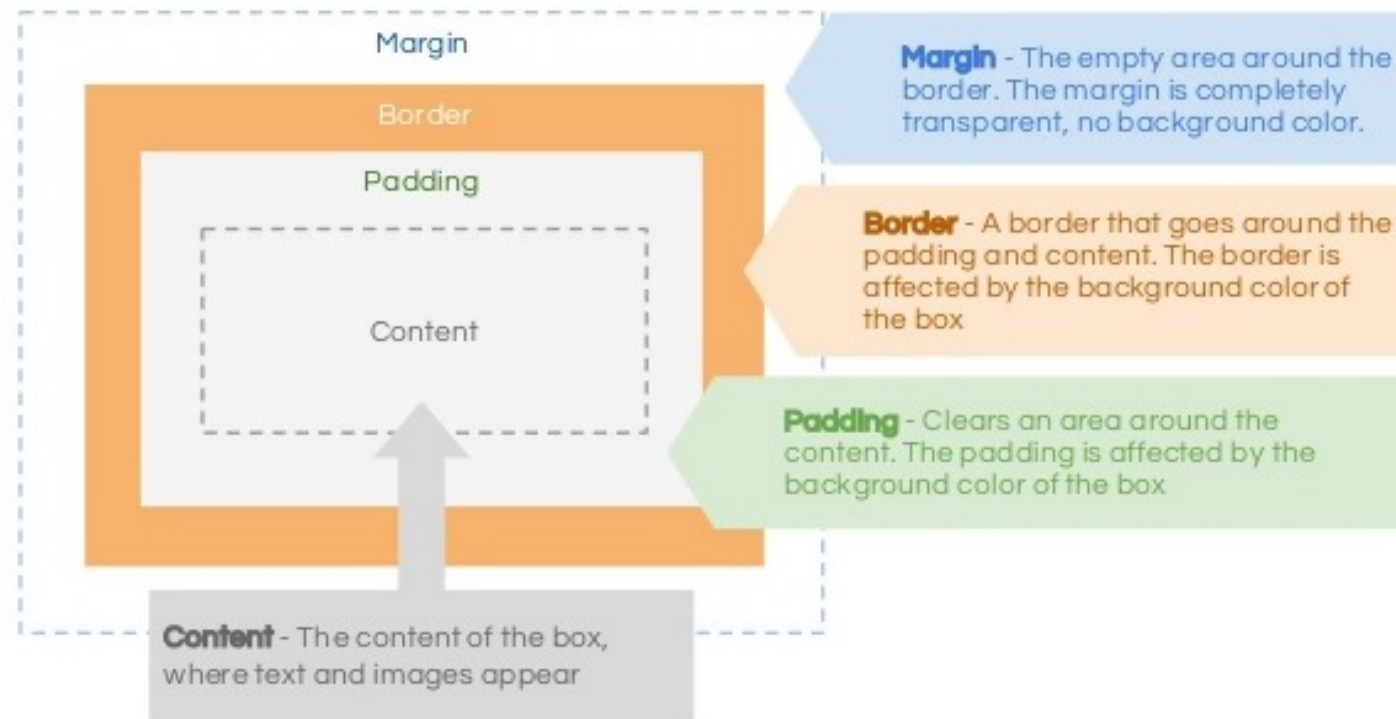
The box model

- 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
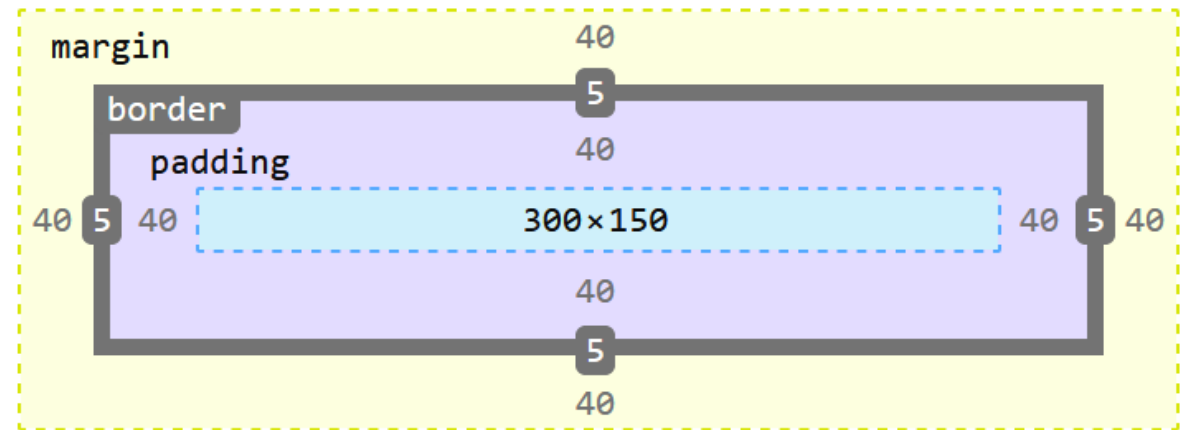
Cascading Style Sheets

CSS POSITIONING SCHEMES

Positioning schemes

- **Static**: normal flow (default)

- **Relative**: offset relative to the block position in the normal flow

- **Absolute**: the box position is determined by the top, left, right, bottom properties, relative to the containing block

- **Fixed**: fixed with respect to some reference point (the viewport)
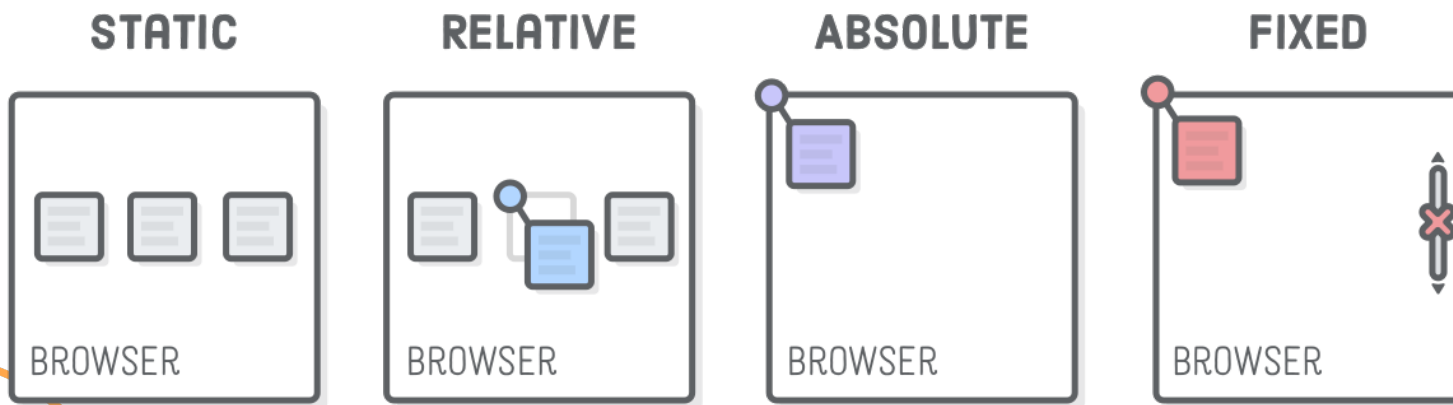

```
.item {
    position: static | relative
| absolute | fixed;
    left: 20px;
    top: 20px;
}
```

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

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;
}
```
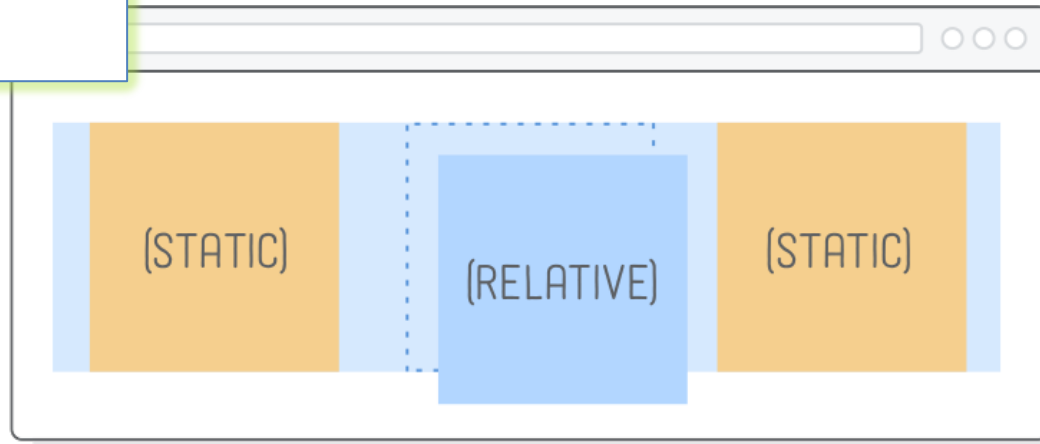
BROWSER

RELATIVE POSITIONING

(STATIC) (RELATIVE) (STATIC)

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;
}
```
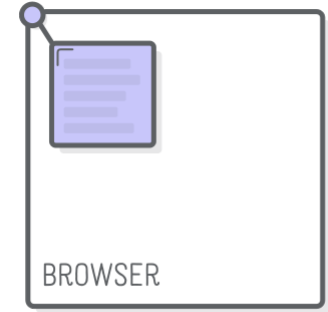

BROWSER

ABSOLUTE POSITIONING
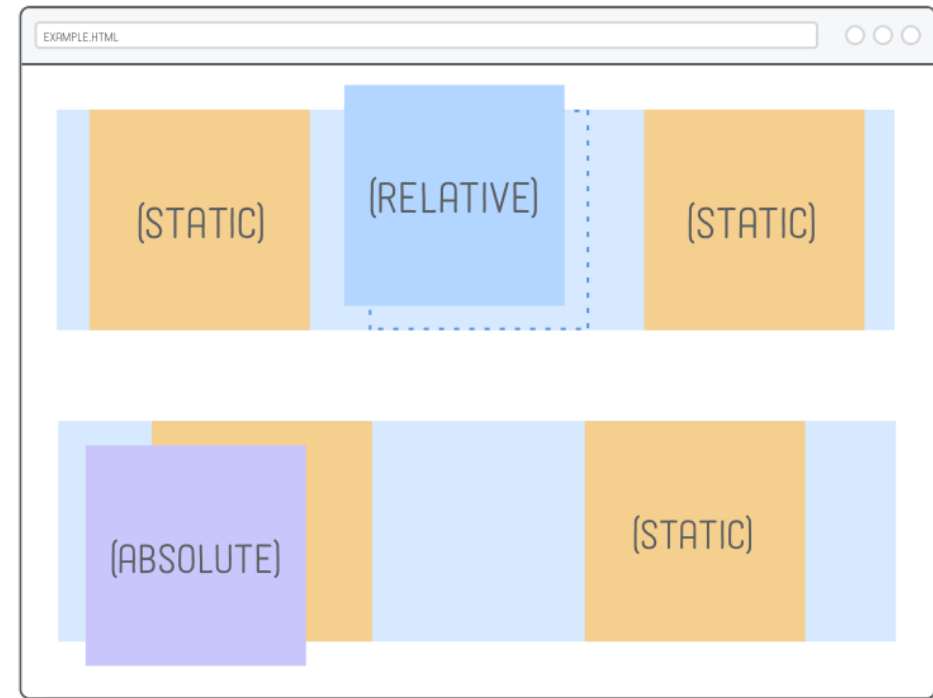
Fixed positioning

- 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
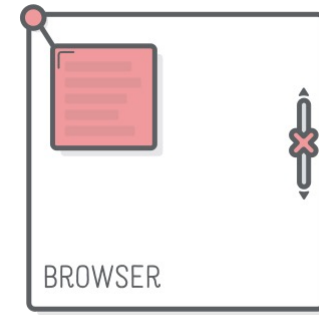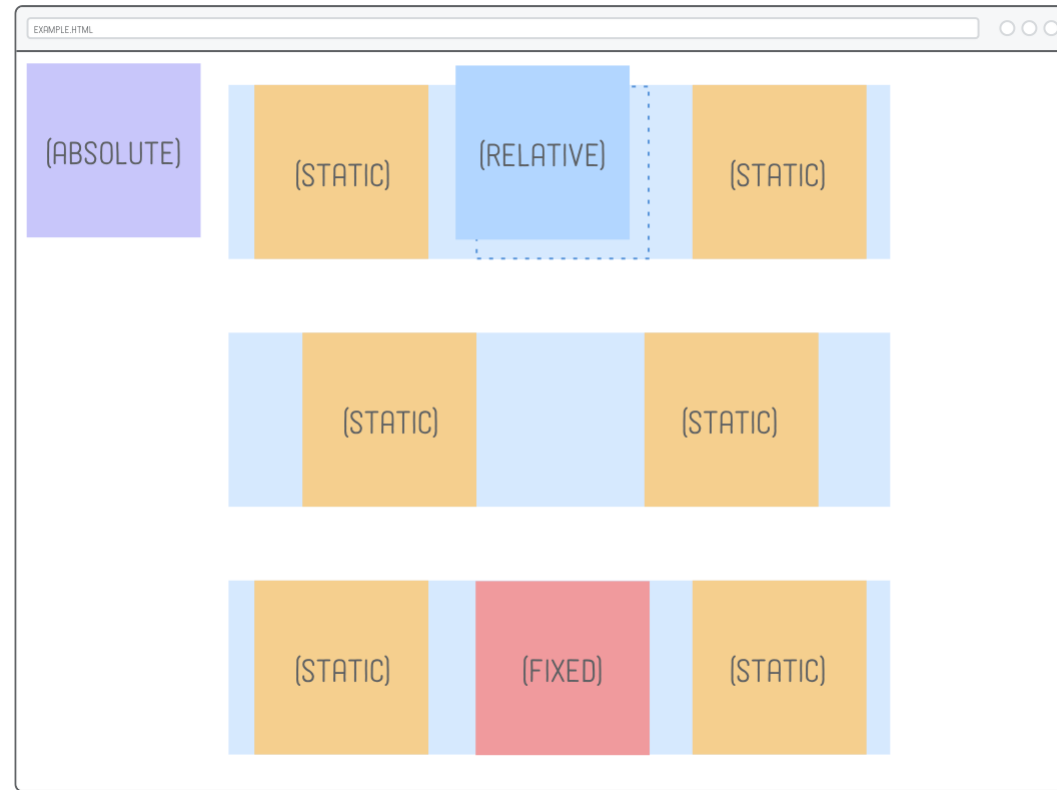
z-index

- 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)

Page Layout Methods

flex

grid

Easier to use

float

Layout approaches

box model

display:
inline, block
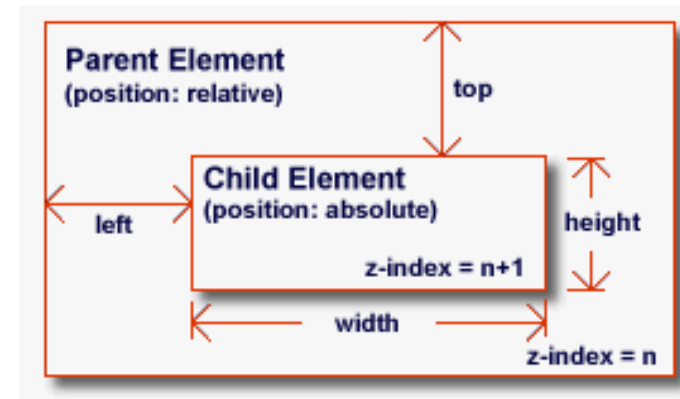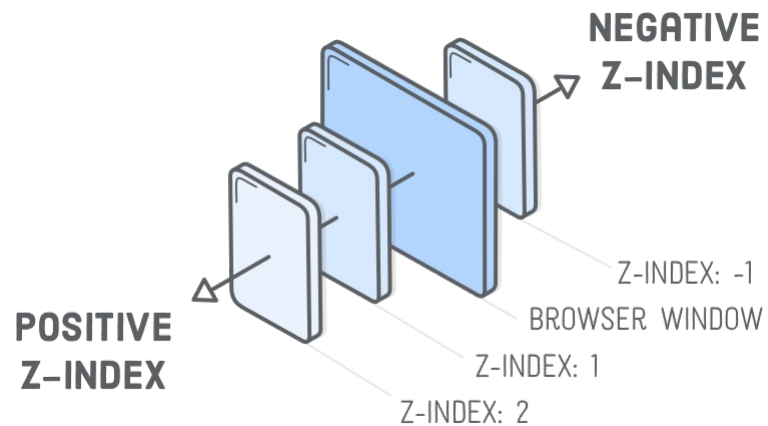
Basic concepts
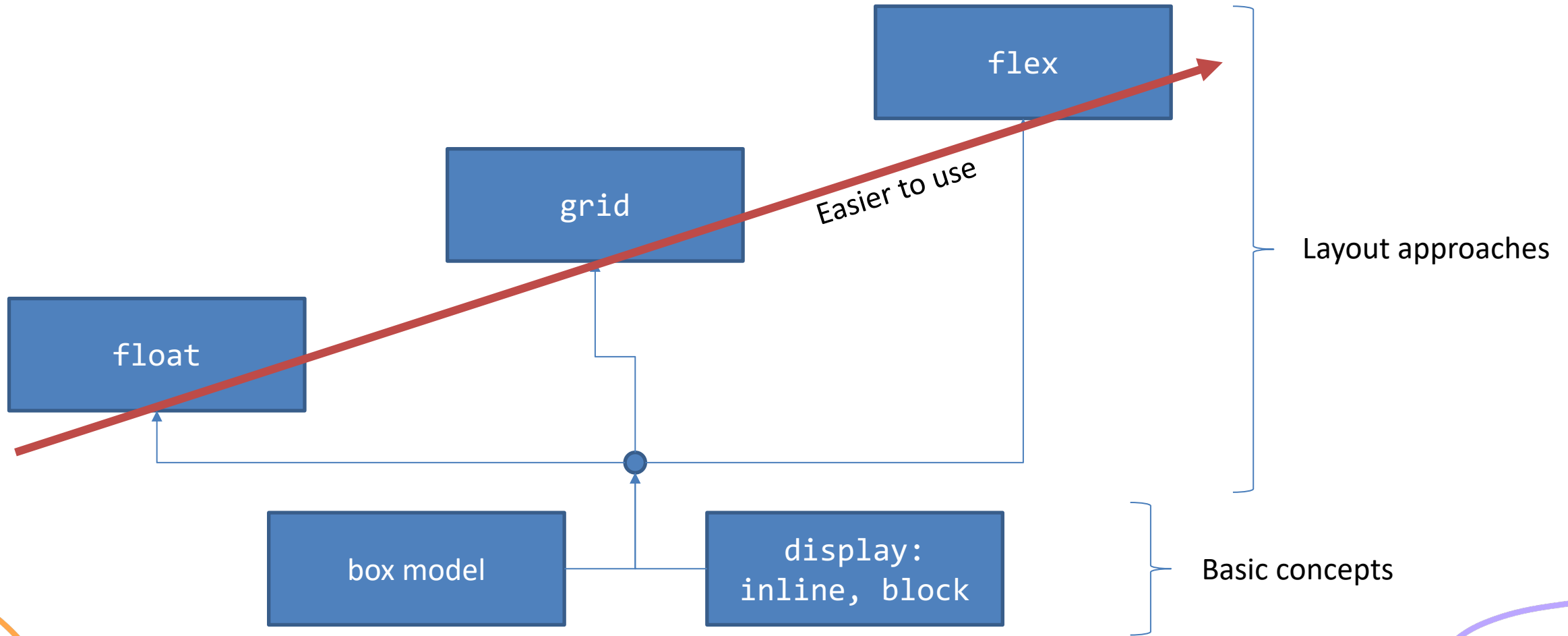
Page Layouts

FLOAT

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;
}
```
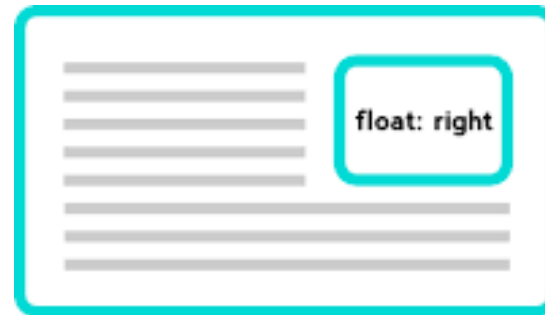
Floating

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 2 hidden under Box 1

Box 3
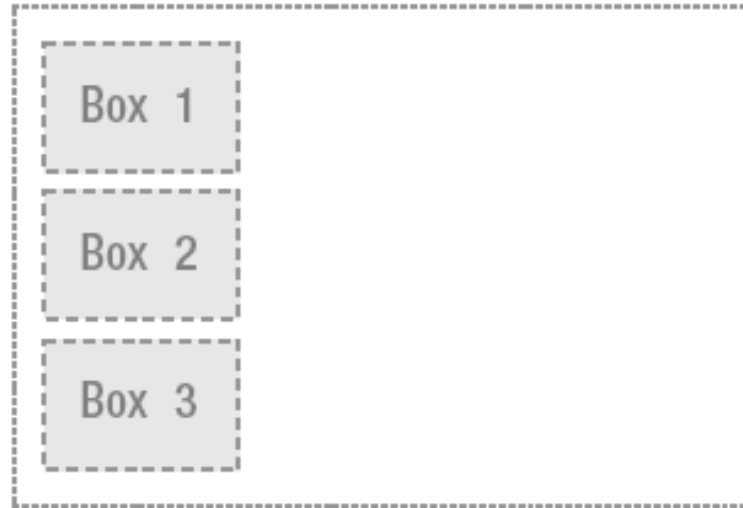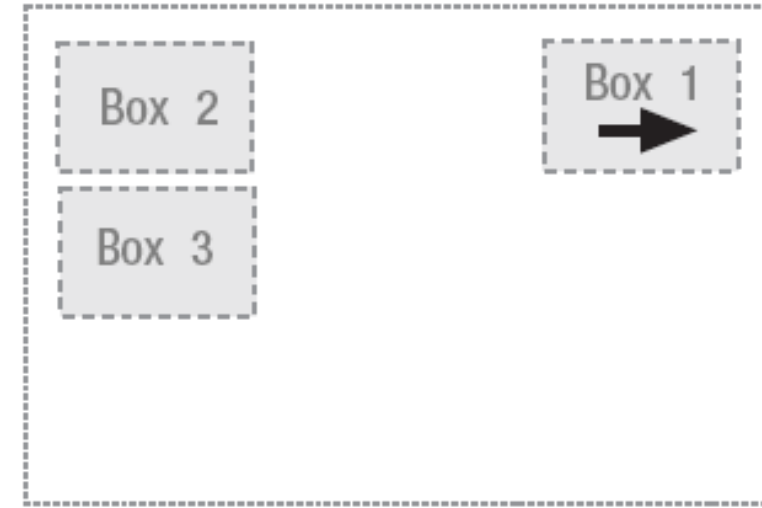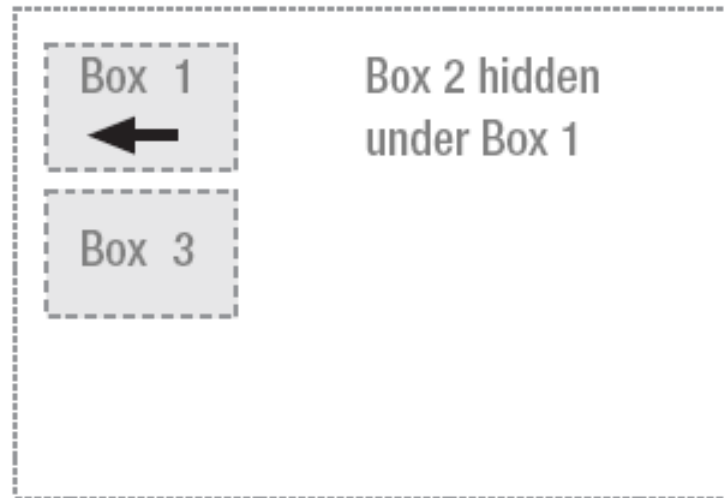
All three boxes floated left
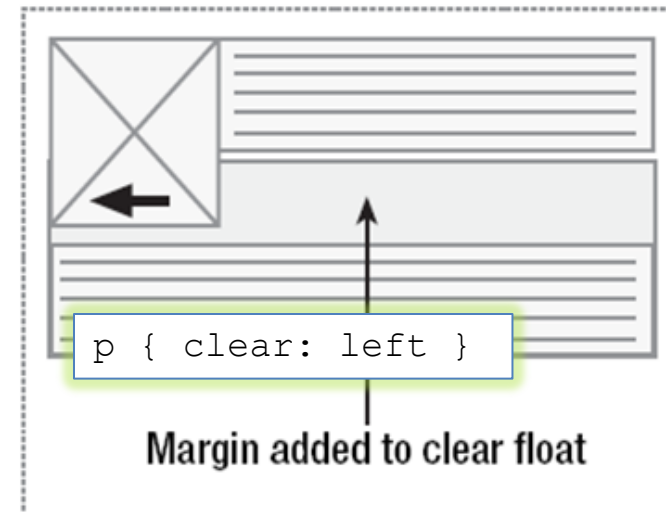
Box 1 ←   Box 2 ←   Box 3 ←
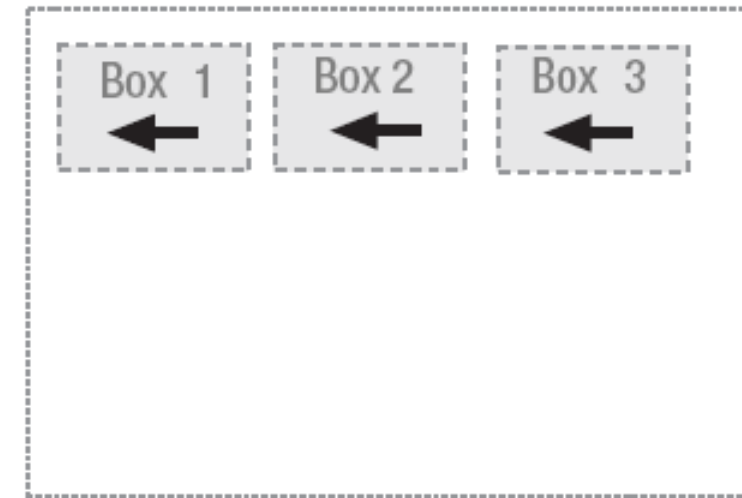
Clearing Floats

- "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 is like forcing a box back into the default vertical flow of the page
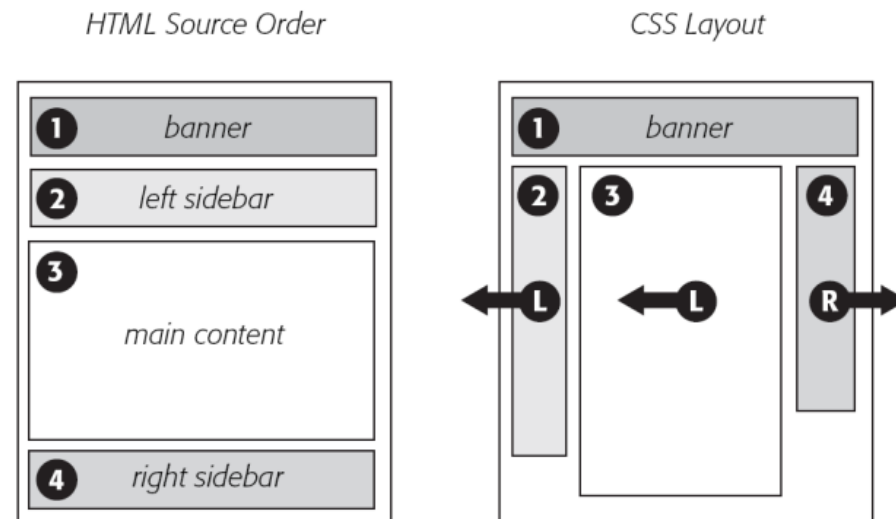
CLEARING WITH CHILD ELEMENT

CLEAR: BOTH;
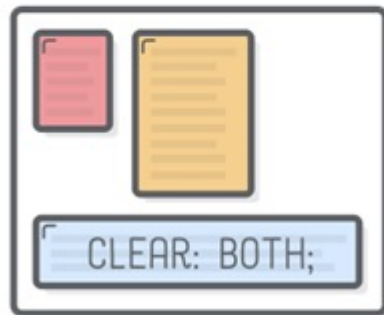
Second paragraph cleared

`p { clear: left }`

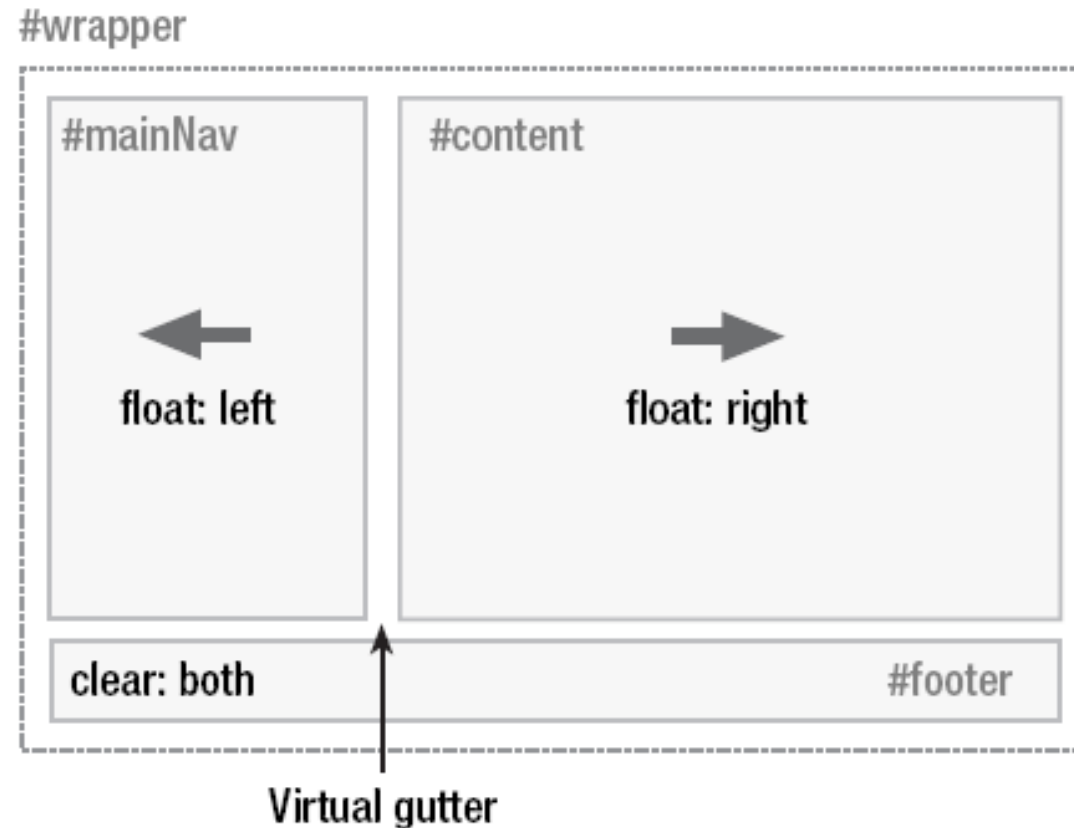Margin added to clear float

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

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>
```
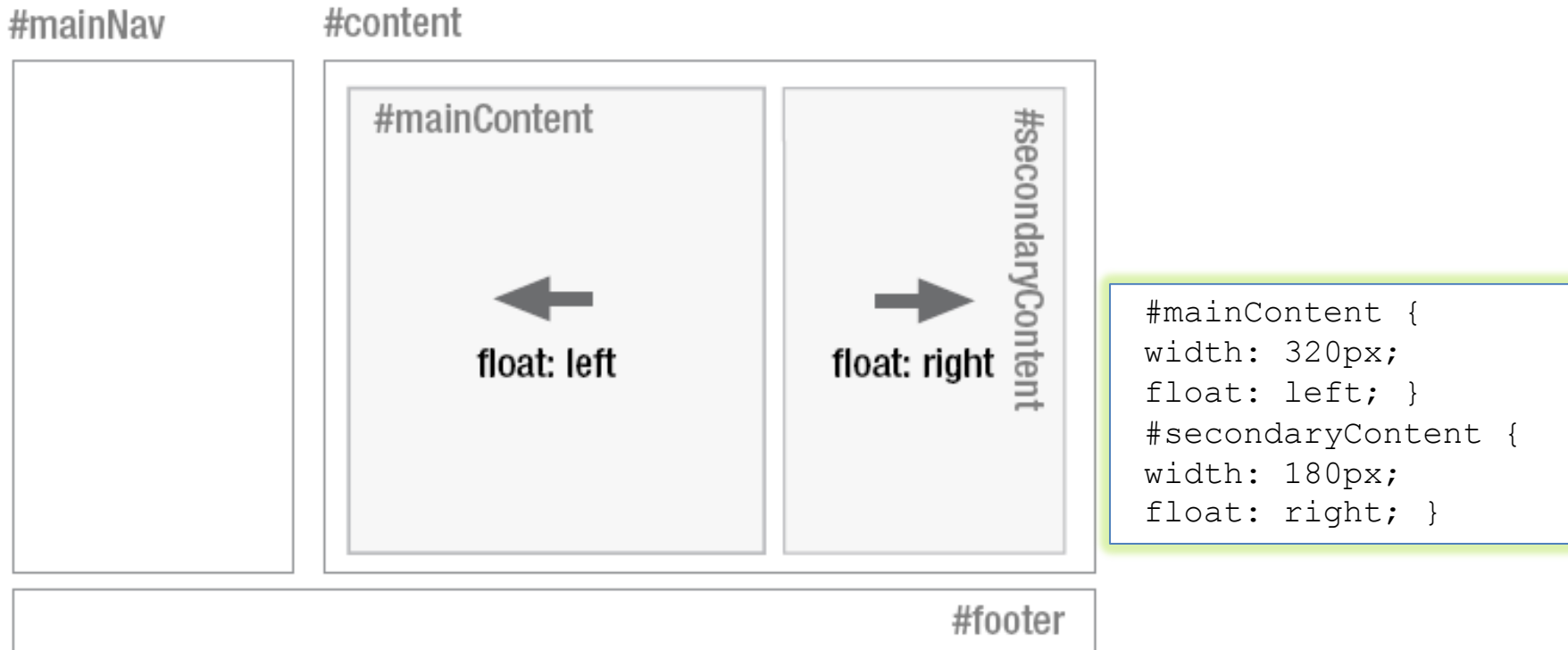
Two-column Floated Layout

```
#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;
}
```
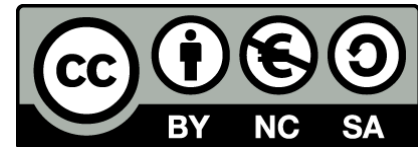
https://blog.html.it/layoutgala/index.html

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; }
```

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/

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/