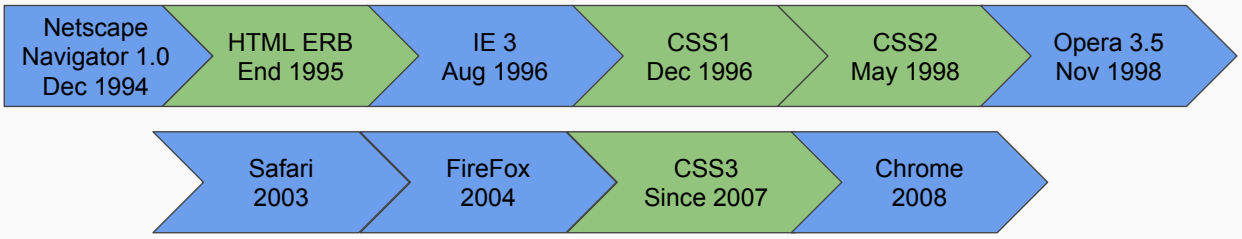
CSS

History of CSS



CSS Evolution (1)

- CSS1: the origin
  - Select element by **name**, **id**, or **class**
- CSS2
  - Web Fonts (embedded fonts in a web document)
  - Rules can select element by **attributes** or **parent-child** relationship
  - Allow **multiple classes** for one element
  - Position elements in 3D (`z-index` property)
  - Generated content (pseudoelements `::before`, `::after`)
  - Element visibility (`visible`, `hidden`, none)
  - Improved table styling (row-group, column-group, caption, ...)

CSS Evolution (2)

- CSS3
  - Border with rounded corners
  - Border images
  - Improved background properties
  - Transparent colors (RGBA: RGB-*Alpha*, HSLA: Hue-Saturation-Lightness-*Alpha*)
  - Gradient background (linear & radial)
  - Shadow (text & box)
  - @font-face selector
  - 2D & 3D transformations (rotate, scale, translate), animations
  - Flexbox

Styling in CSS

- Cascading Stylesheet
- Styles are defined using a set of rules
- Each rule
  - begins with a selector to select the element(s) onto which the rule is applied
  - Specify a group of properties to apply to the element(s)

```
selectorA {
    property1: value;
    property2: value;
}

selectorB {
    property1: value;
    property2: value;
}
```



```
p {
  margin: 4px;
  color: white;
  background-color: black;
}
```

```
#sidebar {
  background-color: gray;
}
```

```
.active {
  font-size: 120%;
}
```

- Selector: paragraphs
- Properties
  - 4-pixel margin
  - White text on black background

- Selector: element with id “sidebar”

- Selector: element with class “active”

Defining Styles

- Internal Stylesheet: **rules** written in the same file as the HTML doc
  - Use `<style> </style>` in header
- External Stylesheet: **rules** written in a file separate from the HTML doc
  - Use `<link rel="stylesheet" href="...." type="text/css">` in header
- ~~Inline Style: properties written as the `style` attribute of an element~~ (**not recommended**)
  - `<p style="color: red">.....</p>`

Grouping Selectors

```
p {
   margin: 8px;
   font-style: bold;
}

li {
   margin: 8px;
   font-style: bold;
}

td {
   margin: 8px;
   font-style: bold;
}
```

```
/* same style for these elements */
p, li, td {
   margin: 8px;
   font-style: bold;
}
```

Vertical Margin Collapse

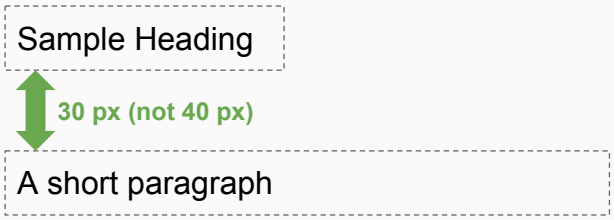
```
/* in mystyle.css */
h1 {
  margin-bottom: 30px;
}

p {
  margin-top: 10px;
}
```

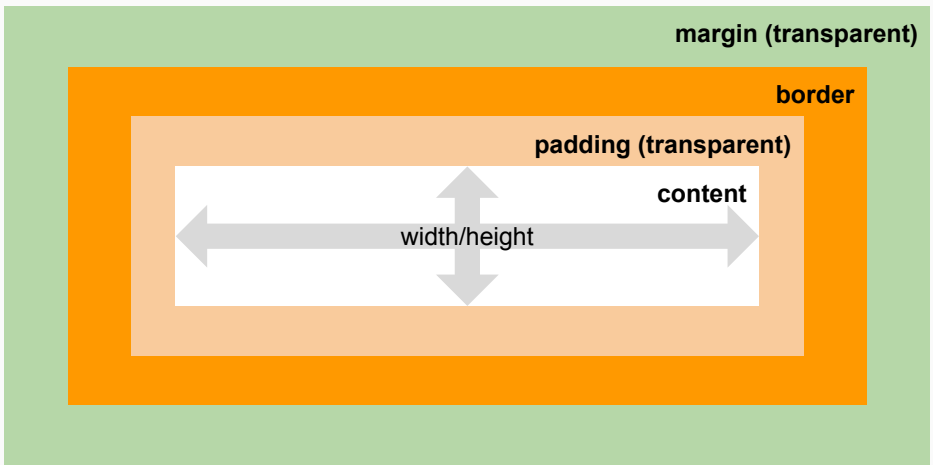
```
<!-- in mypage.html →

<h1>Sample Heading</h1>
<p>A short paragraph</p>
```

Top and bottom margins of two (vertically) adjacent **block elements** are collapsed into one margin (the largest of the two)



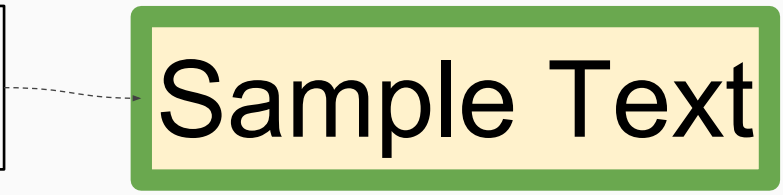
CSS Box Model



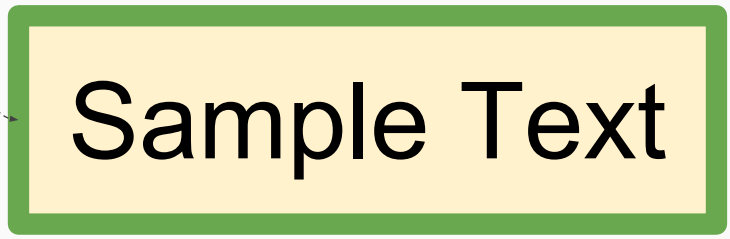
`background-color` paints the content, padding, and **border**

CSS Box Model: Padding

```
span {
   padding: 4px;
   border: 12px solid green;
   background: beige;
}
```



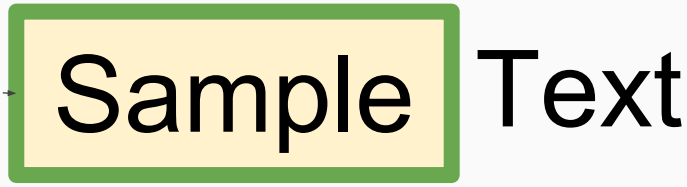

```
span {
   padding: 16px;
   border: 12px solid green;
   background: beige;
}
```



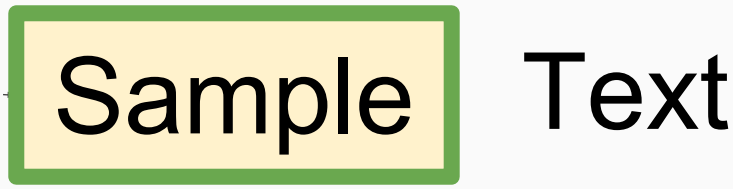

```
<span>Sample Text</span>
```


CSS Box Model: Margin

```
span {
  margin-right: 2px;
  border: 6px solid green;
  background: beige;
}
```




```
span {
  margin-right: 8px;
  border: 6px solid green;
  background: beige;
}
```




```
<span>Sample</span> Text
```

CSS Colors

Named Colors (140 standard names)

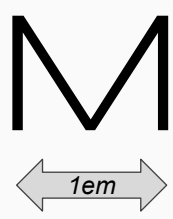
AntiqueWhite	BlueViolet	BurlyWood	CadetBlue
Coral	Crimson	DarkBlue	DarkGoldenRod
DarkGreen	DarkOrange	DarkRed	DarkSeaGreen
DodgerBlue	ForestGreen	Fuchsia	Gold
HotPink	IndianRed	Khaki	Lavender
LawnGreen	LightBlue	LightSalmon	LightSteelBlue
MediumAquaMarine	NavajoWhite	Olive	OliveDrab
Orange	OrangeRed	PaleGreen	YellowGreen

CSS Colors

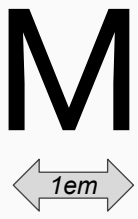
- By RGB (0-255 per color)
  - rgb(155,138,73) or CSS3 rgba(155, 138, 73, 0.7)
- By Hex String (00-FF per color)
  - #C55 or #FCA9
  - #9B8A49 or CSS3 #9B8A493F
- By HSL (CSS3)
  - Hue angles: 0=red, 120=green, 240=blue, 360=red
  - hsl(120, 80%, 90%)
  - hsla(120, 80%, 90%. 0.7)
- Color Picker (by Brandon Mathis)

Font Size: 1em

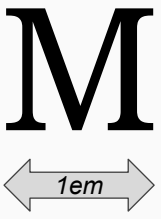
Font: Syncopate



Font: Roboto



Font: Droid Serif



1em: the width of uppercase M in the **current** font (traditional interpretation)
1em: the width of the **current** font (modern typography interpretation)

Layout

(Block vs. Inline) & `display`

- Block-level elements: start on a new line and **take up the available full-width**
- Inline elements: do not start on a new line and **take up as much width** as necessary
- Change default behavior using **`display`**
  - `display: inline` ⇒ changes a block element to an inline element
  - `display: block` ⇒ changes an inline element to a block element
  - `display: none` ⇒ hide the element
- Examples

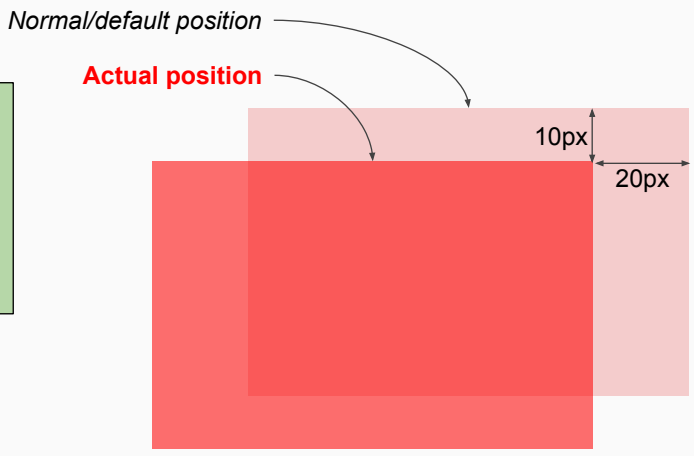
Element Positioning / Layout

- `position:static` ⇒ element is placed in its “normal position” (*based on the current page flow*)
- `position: relative` ⇒ relative to its normal position
- `position: fixed` ⇒ relative to the **viewport**
- `position: absolute` ⇒ relative to the **nearest positioned ancestor**
- The last three properties are used together with positioning keywords:
  - `top`, `right`, `bottom`, `left`
- Examples

Relative Positioning

```
#mybox {
    background-color: red;

    position: relative;
    top: 10px;
    right: 20px;
}
```

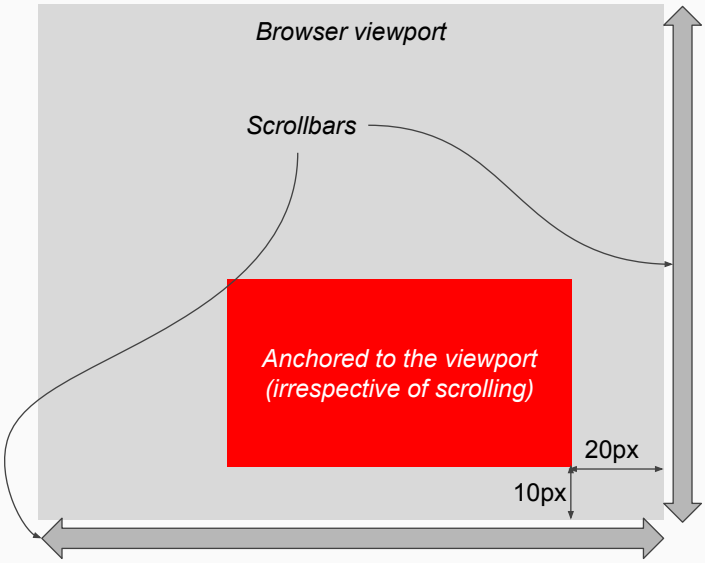


Watch out for overflow!

Fixed Positioning (Unaffected by viewport scrolls)

```
#mybox {
    background-color: red;

    position: fixed;
    bottom: 10px;
    right: 20px;
}
```



Absolute Positioning (w.r.t ancestor)

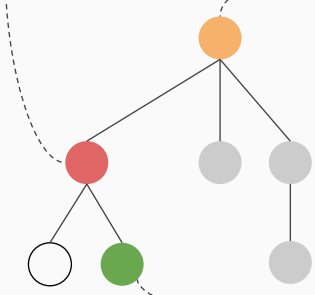
```
#myredbox {
  background: red;
  /* no position */
}
```

```
#mytopbox {
  background: lightsalmon;

  position: relative;
}
```

```
#mygreenbox {
  background-color: green;

  position: absolute;
  bottom: 10px;
  right: 20px;
}
```



Nearest Containing Ancestor
(with position: relative/fixed/absolute)

Anchored to ancestor

20px

10px

More CSS Selectors

Selectors: SQL vs CSS

SQL:

```
SELECT column(s) FROM table(s) WHERE condition;
```

CSS:

```
SELECT element(s) FROM DOMTREE WHERE condition;
```

CSS Rule (**if ⇒ then**)

```
selector {
   prop1 : value1;
   prop2 : value2;
   prop3 : value3;
   .
   .
   .
}
```

```
if the element _________  {
   apply value1 to prop1;
   apply value2 to prop2;
   apply value3 to prop3;
   .
   .
   .
}
```


CSS “if conditions”

If the element

- is-a certain tag/type
- has a certain attribute (regardless of its value)
- has a certain attribute with a specific value
- belongs to a specific class
- is a descendant of another element
- is a sibling of another element

CSS Selectors Capabilities

- CSS selectors provide a powerful mechanism to “query” your elements and apply properties when certain **conditions** are met
- A large portion of **UI update logic can be factored out of your code** and delegated to CSS
  - UI updates can be implemented by manipulating **CSS classes** of your elements
  - Refrain from directly updating style-related properties (color, size, margin, …) from code
- **Why class? Why not attribute?**

Selector Examples

`*` { prop : val }	Apply the style to ALL elements
`p` { prop : val }	Apply the style to paragraphs
`.author` { prop : val }	Apply the style to elements with .author class
`p.author` { prop : val }	Apply the style only to paragraph with .author class
`[title]` { prop : val }	Apply the style to any elements with title attribute
`p[title]` { prop : val }	Apply the style to paragraphs with title attribute
`:empty` { prop : val }	Apply the style to empty elements (have no child)
`p:empty` { prop : val }	Apply the style to empty paragraphs
`:not(p)` { prop : val }	Apply the style to ALL elements except paragraphs

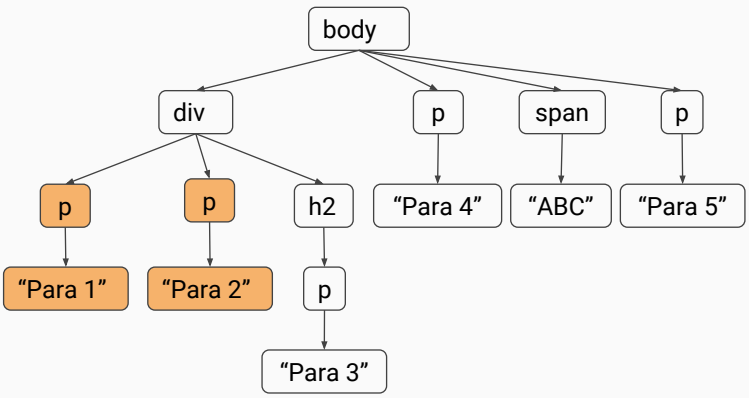
Combinator Selectors (CSS2)

Types	Selector	Apply Rules to
Immediate Child	`div > p { ...rules...}`	paragraphs which are an **immediate child** of div
Descendant	`div p { ...rules...}`	paragraphs somewhere inside div (immediate children included)
Adjacent Sibling	`div + p { ...rules...}`	sibling paragraph immediately following a div
General Sibling	`div ~ p { ...rules...}`	sibling paragraphs which follow a div (immediately following sibling included)

Examples

Child (Immediate Descendant) Selector

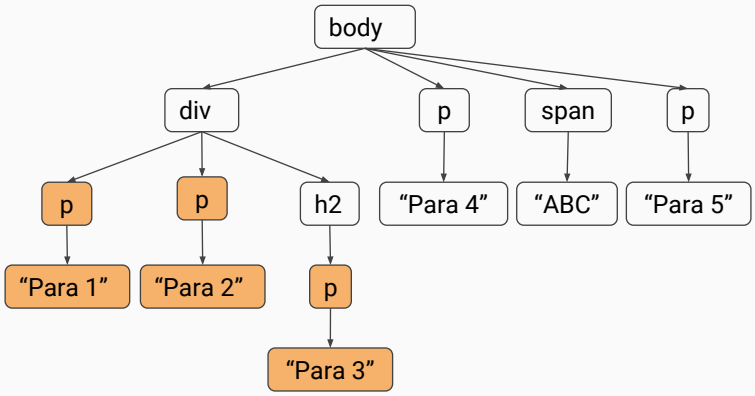
```
<body>
  <div>
    <p>Para 1</p>
    <p>Para 2</p>
    <h2><p>Para 3</p></h2>
  </div>
  <p>Para 4</p>
  <span>ABC</span>
  <p>Para 5</p>
</body>
```




```
/* apply to paragraphs which are an immediate child of a div */
div > p {
  background-color: orange;
}
```

Descendant Selector

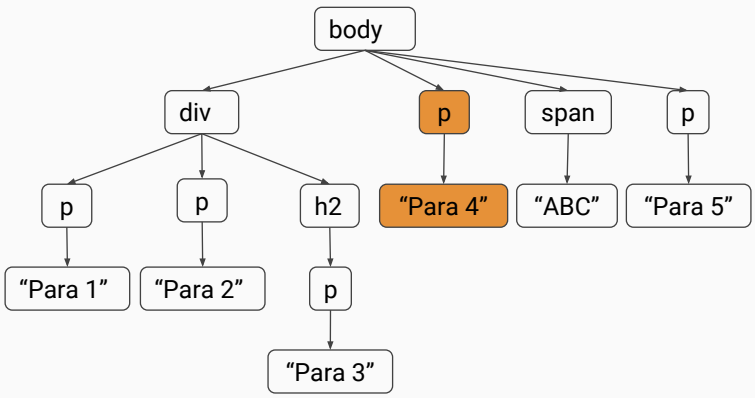
```
<body>
  <div>
    <p>Para 1</p>
    <p>Para 2</p>
    <h2><p>Para 3</p></h2>
  </div>
  <p>Para 4</p>
  <span>ABC</span>
  <p>Para 5</p>
</body>
```




```
/* apply to paragraphs which are a descendant of a div */
div  p {
  background-color: orange;
}
```


Immediate Sibling Selector

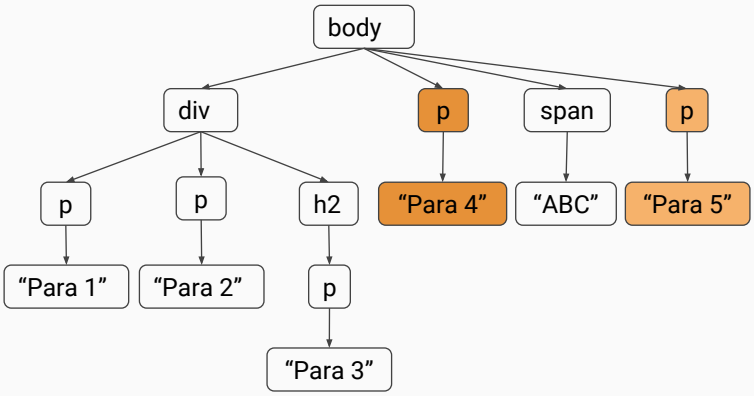
```
<body>
  <div>
    <p>Para 1</p>
    <p>Para 2</p>
    <h2><p>Para 3</p></h2>
  </div>
  <p>Para 4</p>
  <span>ABC</span>
  <p>Para 5</p>
</body>
```




```
/* apply to paragraphs which are an immediate sibling following a div */
div + p {
  background-color: orange;
}
```

General Siblings Selector

```
<body>
  <div>
    <p>Para 1</p>
    <p>Para 2</p>
    <h2><p>Para 3</p></h2>
  </div>
  <p>Para 4</p>
  <span>ABC</span>
  <p>Para 5</p>
</body>
```




```
/* apply to paragraphs which are a (younger) sibling of a div */
div ~ p {
  background-color: orange;
}
```


Selector :pseudo-classes

- Links (:link, :visited, :hover, :active)
- Input (:checked, :disabled, :enabled, :focus, :in-range, :out-of-range, :invalid, :valid, :optional, :required, :read-only, :read-write)
- Child order (:first-child, :last-child, :nth-child, :nth-last-child, :only-child)
- Of-Type order (:first-of-type, :last-of-type, :nth-of-type, :nth-last-of-type, :only-of-type)

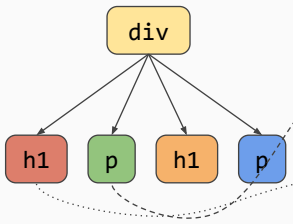
:first-child vs. :first-of-type

```
<div>
  <h1>First Heading</h1>
  <p>One paragraph</p>

  <h1>Second Heading</h1>
  <p>A bit longer paragraph</p>
</div>
```

*The **first** "daugher" in a family may be the **third** "child"*

```
div p:first-child {
  /* no matching element */
}
```




```
div p:first-of-type {
  /* applies only to "One paragraph" */
}
```

```
div h1:first-child {
  /* applies only to "First Heading" */
}
```


CSS3 :nth-child()

- :nth-child(odd): select child #1, #3, #5, …
- :nth-child(even): select child #2, #4, #6, …
- :nth-child(3n+1): select child #1, #4, #7, #10, ...

Attribute Selectors

- Objective: select elements with a particular attribute
- Selectors
  - `[attr]` ⇒ select elements that have attribute `attr` (regardless of its value)
  - `[attr=val]` ⇒ select elements whose attribute `attr` is set to "`val`"
  - `[attr~=val]` ⇒ select elements whose attribute `attr` **contains** "`val`" (whole word)
  - `[attr*=val]` ⇒ select elements whose attribute `attr` **contains** "`val`" (partial word)
  - `[attr|=val]` ⇒ select elements whose attribute `attr` **starts with** "`val`" (whole word)
  - `[attr^=val]` ⇒ select elements whose attribute `attr` **starts with** "`val`" (partial word)
  - `[attr$=val]` ⇒ select elements whose attribute `attr` **ends with** "`val`" (partial word)

CSS Pseudo Classes Example

```
/* stylesheet: white on green */
h1.active {
  color: white;
  background-color: green;
}

p.active:hover { font-weight: bold }
```

```
<!-- HTML doc -->

<h1>First Heading</h1>
<p class="active">This text is
terse.</p>

<h1 class="active">Second Heading</h1>
<p>This text is slightly longer than
the previous one.</p>
```

bold when mouse is over text

First Heading
This text is terse.

Second Heading
This text is slightly longer than the previous one.

rendered on browser

Pseudo Elements

- Objective: select certain part of an element
- Selectors
  - `::after` ⇒ insert some content **after** an element
  - `::before` ⇒ insert some content **before** an element
  - `::first-letter` ⇒ only the first letter of a **text**
  - `::first-line` ⇒ only the first line of a **text**
  - `::selection` ⇒ matches the portion of an element that is **selected by user**

Pseudo Classes

- Practical use: **select only elements in a particular "state"**
- Link states (`:link`, `:visited`, `:hover`, `:active`)
- Input states (`:checked`, `:disabled`, `:empty`, `:enabled`, `:focus`)
- Positional (`:first-child`, `:last-child`, `:nth-child()`, `:nth-last-child()`)

Pseudo Elements

- Practical use: **select only certain part of an element**
- Selectors ("`::`" in CSS3, "`:`" in CSS[1|2])
  - `::after`
  - `::before`
  - `::first-letter`
  - `::first-line`
  - `::selection`

References: Many more Mozilla Dev Network

Example of ::pseudo-element

```
p.warn::before {
  content: “WARNING: “;
  color: red;
}
```

WARNING: Beware of dogs

```
<p class=”warn”>Beware of dogs</p>
```


Quiz: What is the effect of the CSS Styles?

```
<!-- HTML -->
<div>
  <div>Parent</div>
  <ol>
    <li>Child 1</li>
    <li>Child 2</li>
    <li>Child 3</li>
  </ol>
</div>
```

```
/* CSS */
div > ol {
  display: none;
}

div:hover ol {
  display: block;
}
```

See it live on JSFiddle

More Selector Examples

`p.author[title] { prop : val }`	Apply the style to paragraphs with author class and title attribute
`div p span { prop : val }`	Apply the style to span which are a descendant of a paragraph which is a descendant of a div
`div p > span { prop : val }`	Apply the style to span which are a **child** of a paragraph which is a descendant of a div
`div p:first-of-type > span { prop : val }`	Apply the style to span which are a **child** of a first paragraph (*not necessarily the first child*) which is a descendant of a div