FRESNO STATE
Discovery. Diversity. Distinction.

Web Programming (CSci 130)

Department of Computer Science
College of Science and Mathematics
California State University Fresno
H. Cecotti

Learning outcomes

- More commands and examples with CSS
  - **A large number of commands**
    - Impossible to see everything during the class
      - Need to try, test, during lab times, and of course at home
    - To get the concepts of the syntax and what can be achieved
      - Do not hesitate to be bold and search for new presentations, new functionalities
      - Some positions are just in relation to the presentation (UI/UX emphasis).
  - More selectors
  - **CSS animations**

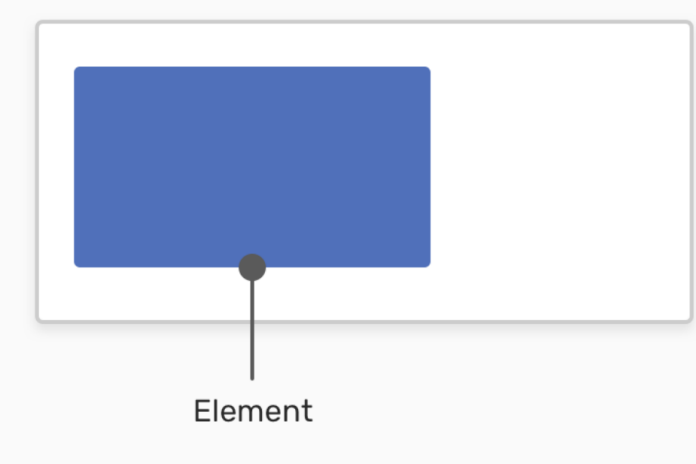
Positioning

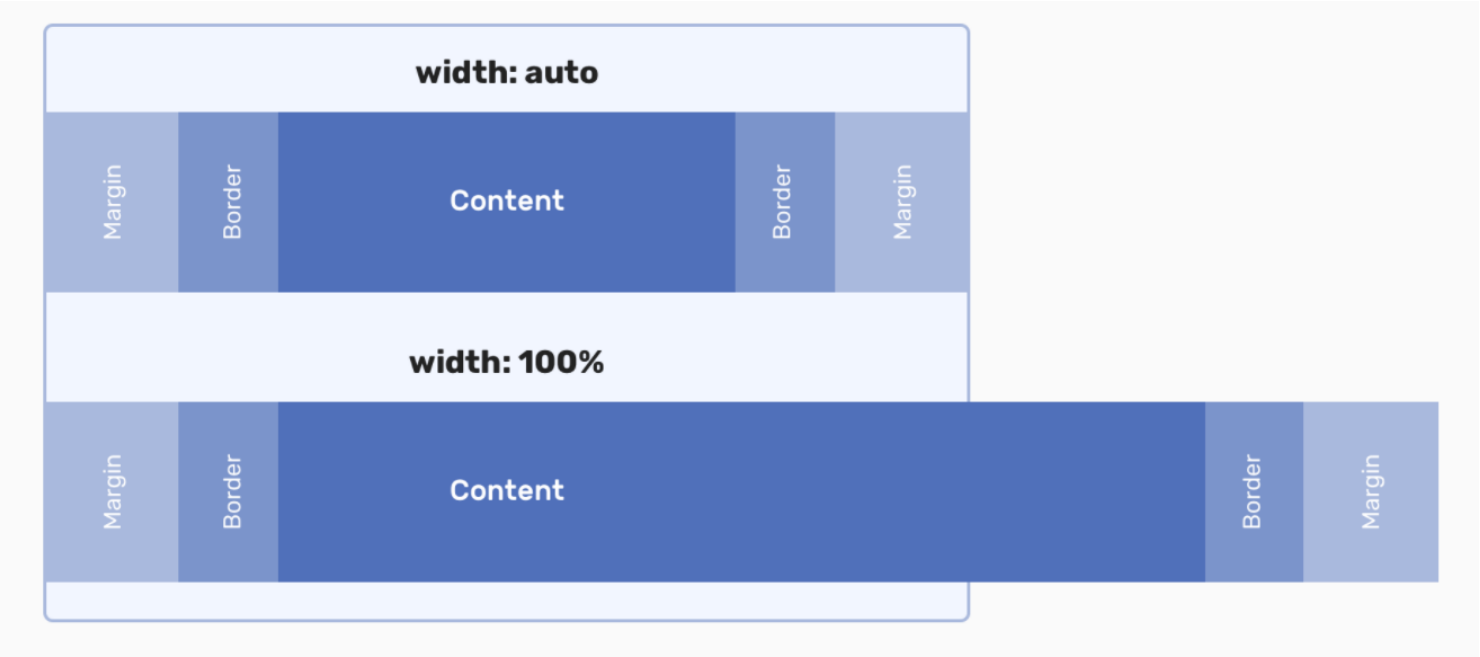
- **Definitions**
  - Static: the default value, all elements are in order as they appear in the document.
  - Relative: the element is positioned relative to its normal position.
  - Absolute: the element is positioned absolutely to its first positioned parent.
  - Fixed: the element is positioned related to the browser window.
  - Sticky: the element is positioned based on the user's scroll position.
    - Useful for menu, if you want it always in the top
- Reading with examples
  - https://www.w3schools.com/css/css_positioning.asp

Auto

- auto: automatically adjusted
  - ➢ used for properties like margin, positioning, height, width
  - ➢ /!\ Warning: the use of auto keyword varies from a property to another
- The initial width of block-level elements like or <p> is auto, which makes them take the full horizontal space of their containing block (parent block).
  - ➢ → When an element has auto as a value for width, it can have margin, padding, and border **without** becoming bigger than its parent element
- Example to center an element
  - ➢ margin: 0 auto;
  - ➢ margin: 25px 50px;
    - ○ top and bottom margins are 25px + right and left margins are 50px

Auto



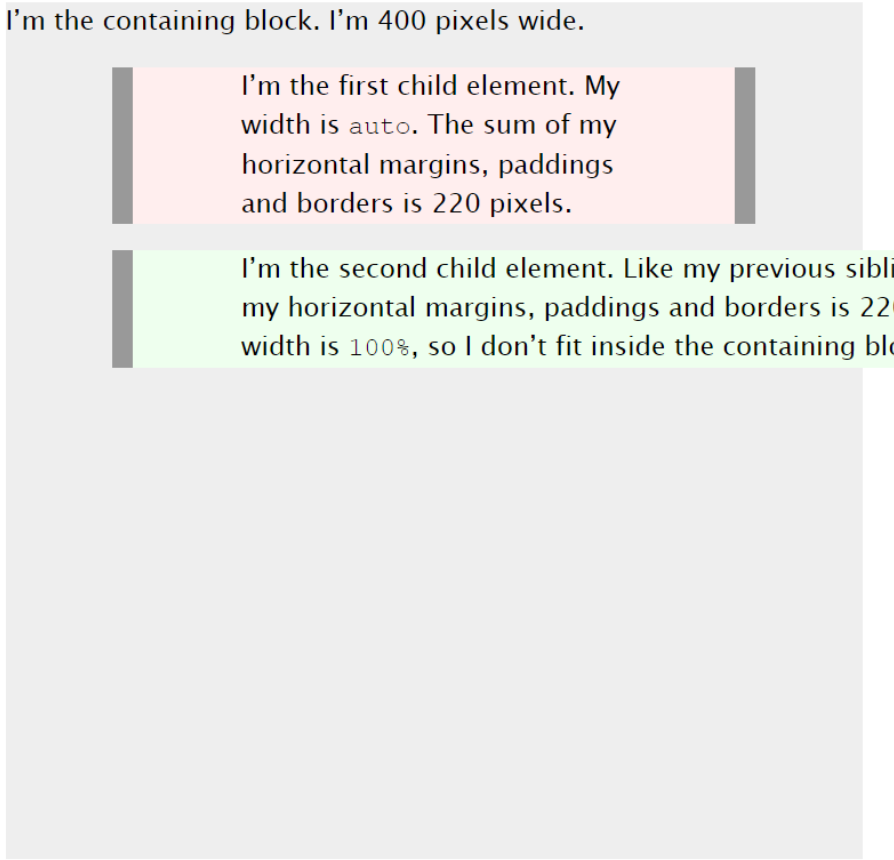


To center the blue element

```
.element {
    margin-left: auto;
    margin-right: auto;
}
```

Auto

```
#body {
    width:70em;
    max-width:100%;
    margin:0 auto;
}
#container {
    width:400px;
    height:400px;
    background:#eee;
}
#child-1 {
    width:auto;
    margin:0 50px;
    padding:0 50px;
    border:10px solid #999;
    border-width:0 10px;
    background:#fee;
}
#child-2 {
    width:100%;
    margin:0 50px;
    padding:0 50px;
    border:10px solid #999;
    border-width:0 10px;
    background:#efe;
}
```



Gradient

- Linear
  - Parameters

Gradient Style	Values	Notes
linear-gradient([origin,] color [stop], color [stop] [, color [stop]]*) for example: **linear-gradient(bottom left, #fff, #f00 30%, #000)** results in a gradient, originating from the bottom left corner of the box and ending at the top right corner; it begins with white, which becomes red 30% of the way across the gradient, which then ends at black	**origin** specifies the corner of the box and can be a combination of **top**, **left**, **bottom**, **right**, and **center** keywords or percentage values (originating from the top left) relative to the size of the box the first **color** value refers to the color at the start of the gradient; the last **color** value refers to the end of the gradient; you can have any number of colors in the gradient **stop** specifies the location of the color in the gradient and can be a length or a percentage relative to the length of the entire gradient	by default, linear gradients originate from the top center of the box by default, the browser attempts to distribute colors evenly across the gradient if only two colors are specified, the default stops are 0% and 100%

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 200px;
    background-color: red; /* for old browsers that dont support gradients */
    background-image: linear-gradient(#EE0000,#0000CC); /* standard syntax (must be last) */
</style>
</head>
<body>
<h2>Linear Gradient: Top to Bottom</h2>
<p>This linear gradient starts at the top. It starts red and finishes blue:</p>
<div id="grad1"></div>
<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support gradients.</p>
</body>
</html>
```

Gradient

- Radial
  - Parameters

radial-gradient([origin,] [shape-or-size-or-both,] color [stop], color [stop] [, color [stop]]*)

for example:

radial-gradient(30% 30%, circle closest-corner, #fff, #000)

results in a gradient originating 30% of the way from the top left corner of the box, radiating out as a circle until it reaches the nearest corner of the box; it starts with white in the center of the circle, and ends with black at the outer edge

origin specifies the corner of the box and can be a combination of **top**, **left**, **bottom**, **right**, and **center** keywords or percentage values (originating from the top left) relative to the size of the box

shape can be specified as **circle** or **ellipse** by default; the shape fills the dimensions of the box (so is an ellipse unless the box is square)

size can be a keyword: **closest-side**, **closest-corner**, **farthest-side**, **farthest-corner**, **contain**, **cover**

size can also explicitly set the dimensions of a radial gradient using a length value (or two length values, if you want to set the horizontal and vertical lengths separately)

the first **color** value refers to the color at the start of the gradient; the last **color** value refers to the end of the gradient; you can have any number of colors in the gradient

stop specifies the location of the color in the gradient and can be a length or a percentage relative to the length of the entire gradient

by default, radial gradients originate from the center of the box

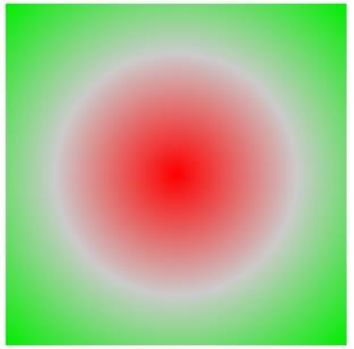
by default, the size keyword is set to **contain**

by default, the browser attempts to distribute colors evenly across the gradient

if only two colors are specified, the default stops are 0% and 100%

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 300px;
    width: 300px;
    background-image: radial-gradient(#ff0000, #cccccc, #00ee00);
}
</style>
</head>
<body>
<h1>Radial Gradient - Evenly Spaced Color Stops</h1>
<div id="grad1"></div>
<p><strong>Note:</strong> Internet Explorer 9 and earlier versions do not support gradients.</p>
</body>
</html>
```



More selectors

- HTML tags are blocks within a “Tree”
  - Access children, parent, children in a particular order (first, last)
    - First letter, last letter, first paragraph, last paragraph: it can be meaningful for documents
    - Try to factorize your presentation as much as possible to minimize changes
- See the excel file on Canvas with the list of functions.

  - Questions?
  - Examples?

- Work on this excel sheet to be prepared for the **first Midterm**
  - Syntax of CSS3

Span vs. Div

- HTML: all about the semantic
  - , : no semantic but very useful
    - To group together a piece of HTML and add some information relative to this element.
    - It is done with the ID or Class attributes
- **Span**: in-line
  - For a small piece of HTML inside a line (like inside a paragraph)
- **Div**: block-line
  - Like the <hr> line break to show there are different “blocks”
- See example on Canvas:
  - class_html5_css3_c.html
  - The use of div and different selectors

CSS3 and the mouse

- Main functions for the links

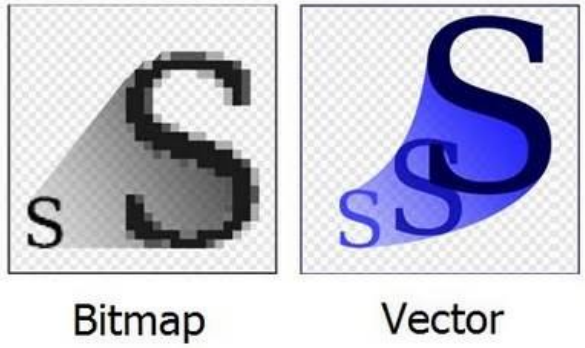
```
/* unvisited link */
a:link {
    color: green;
}
/* visited link */
a:visited {
    color: green;
}
/* mouse over link */
a:hover {
    color: red;
}
/* selected link */
a:active {
    color: yellow;
}
```

Menus

- **Examples on Canvas**
  - Different types of menu
    - class_html5_css3_d.html
  - Menu in the top and bottom
    - class_html5_css3_e.html
  - Resize the window with the media properties
    - class_html5_css3_f.html

SVG

- **SVG** (Scalable Vector Graphics) specification
  - Open Standard
    - developed by the World Wide Web Consortium (W3C) since 1999
  - 3 types of graphics objects
    - Vector graphic shape (e.g., outlines with straight lines and curves)
    - Bitmap image
    - Text
  - Features
    - Small file sizes that compress well
    - Scales to **any** size without losing clarity (except very tiny)
    - Looks great on retina displays
    - Design control like interactivity and filters
  - Images **and** their behaviors
    - defined in XML files → you can create and edit **SVG** images with
      - Any text **editor**
      - **Drawing** software



SVG

- Example
  - <svg> tag
  - <g> tag : to group elements

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
     <g>
       <line x1="10" y1="10" x2="85" y2="10"
           style="stroke: #006600;"/>

       <rect x="10" y="20" height="50" width="75"
           style="stroke: #006600; fill: #006600"/>
       <text x="10" y="90" style="stroke: #660000; fill: #660000">
         Text grouped with shapes</text>
     </g>

</svg>
```

Position: x,y
Size: height, width
Rounded corner: rx, ry

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

     <rect x="10" y="10" height="50" width="50"
           rx="5" ry="5"
           style="stroke:#006600; fill: #00cc00"/>
     <rect x="70" y="10" height="50" width="50"
           rx="10" ry="10"
           style="stroke:#006600; fill: #00cc00"/>
     <rect x="130" y="10" height="50" width="50"
           rx="15" ry="15"
           style="stroke:#006600; fill: #00cc00"/>
</svg>
```

SVG

- CSS examples

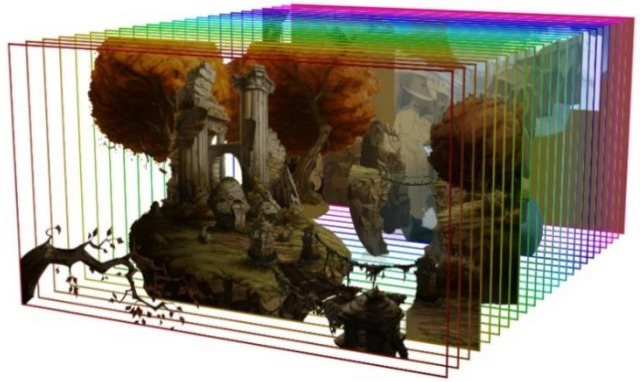
```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>My SVG example</h1>

<svg width="400" height="110">
  <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:3;stroke:rgb(0,0,0)" />
</svg>

<svg height="100" width="100">
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />
</svg>

<svg height="140" width="500">
  <ellipse cx="200" cy="80" rx="100" ry="50"
  style="fill:yellow;stroke:purple;stroke-width:2" />
</svg>

<svg height="210" width="500">
  <line x1="0" y1="0" x2="200" y2="200" style="stroke:rgb(255,0,0);stroke-width:2" />
</svg>

<svg height="210" width="500">
  <polygon points="200,10 250,190 160,210" style="fill:lime;stroke:purple;stroke-width:1" />
</svg>

<svg height="200" width="500">
  <polyline points="20,20 40,25 60,40 80,120 120,140 200,180"
  style="fill:none;stroke:black;stroke-width:3" />
</svg>

</body>
</html>
```

Animation

- **Some techniques to consider when doing animations**
  - ➢ Optical illusions: **illusion of movement**
    - Beta (β) movement
      - Described by Max Wertheimer (1912)
      - A **series** of **static** images on a screen creates the illusion of a smooth animation
      - The static images do **not** physically change but it gives the appearance of motion
        - because they change faster than the eye can see
    - Phi (φ) phenomenon
      - Perceiving a series of still images, when viewed in rapid succession, as continuous motion
    - The φ and β : an apparent movement caused by:
      - luminous impulses in sequence (φ)
      - luminous stationary impulses (β)

Animation

- **Parallax scrolling**: to give a depth illusion
  - Main principle
    - Layers in the **background** move **slower** than in the foreground during a left/right scrolling
  - Used in:
    - Video Games
      - Shadow of the Beast (1989, Psygnosis)
    - Animated movies
      - Disney (multiplan camera)
  - Approach
    - Layers with transparency
    - Sprites
  - How to use it
    - Vertically
    - Horizontally



Shadow of the Beast

Animation

- **Parallax scrolling**
  - ➢ Creation of layers → Animation
    - Background
      - Mountains
      - Clouds
    - Foreground
      - Vegetation layer
      - Ground layer
  - ➢ Popular technique only with HTML5 + CSS3
- The **parallax animation** effect is created by 2 or more layers of an interface moving at different speeds or in different directions in order to produce an impression of depth.
- → Parallax scrolling enhanced certain aspects of the user experience
- **Warning**
  - ➢ **Too much movement, especially of text, can be dizzying**

CSS animation

- 2D and 3D geometric transformations
  - Main functions
    - translate()
    - rotate()
    - scale()
    - skewX()
    - skewY()
    - matrix()
- Lab time
  - Try with a little square:

```
div {
    width: 50px;
    height: 50px;
    margin: 50px;
    background-color: red;
    border: 2px solid black;
}
```

```
/* Rotation */
div {
    -ms-transform: rotate(20deg); /* IE 9 */
    -webkit-transform: rotate(20deg); /* Safari */
    transform: rotate(20deg);
}
/* Scale */
div {
    -ms-transform: scale(2, 3); /* IE 9 */
    -webkit-transform: scale(2, 3); /* Safari */
    transform: scale(2, 3);
}
/* SkewX */
div {
    -ms-transform: skewX(20deg); /* IE 9 */
    -webkit-transform: skewX(20deg); /* Safari */
    transform: skewX(20deg);
}
/* SkewY */
div {
    -ms-transform: skewY(20deg); /* IE 9 */
    -webkit-transform: skewY(20deg); /* Safari */
    transform: skewY(20deg);
}
/* Skew */
div {
    -ms-transform: skew(20deg, 10deg); /* IE 9 */
    -webkit-transform: skew(20deg, 10deg); /* Safari */
    transform: skew(20deg, 10deg);
}
```

CSS animation

- Transitions
  - Parameters: CSS property on which the effect is applied: duration of the effect
- Examples
  - 1) A 4s font size transition with a 2s delay between the time the user mouses over the element and the beginning of the animation effect
  - 2) The box below combines transitions for: width, height, background-color, transform.

```
#delay {
    font-size: 14px;
    transition-property: font-size;
    transition-duration: 4s;
    transition-delay: 2s;
}

#delay:hover {
    font-size: 36px;
}
```

Example 1

```
.box {
    border-style: solid;
    border-width: 1px;
    display: block;
    width: 100px;
    height: 100px;
    background-color: #0000FF;
    -webkit-transition: width 2s, height 2s, background-color 2s, -webkit-transform 2s;
    transition: width 2s, height 2s, background-color 2s, transform 2s;
}

.box:hover {
    background-color: #FFCCCC;
    width: 200px;
    height: 200px;
    -webkit-transform: rotate(180deg);
    transform: rotate(180deg);
}
```

Example 2

CSS animation example

- Transitions
  - https://www.w3schools.com/css/tryit.asp?filename=trycss3_transition2
- Zoom in-out
  - Scale
- Lab time
  - By using gimp to edit images

    Create a webpage with the picture of a face

    When you put the mouse on the eyes,

    the eyes get bigger
    - Use the alpha channel to manage transparency
    - Work on the position of the different elements
      - The face (image1)
      - The eyes (image2 and image3)

```
<!DOCTYPE html>
<html>
<head>

<style>
.outer-div
{
    height: 300px;
    overflow: hidden;
}
.inner-div
{
    height: 100%;
    width: 100%;
    background-size: cover;
    background-position: center;
    transition: all 0.5s ease;
    background-image: url('mybackground.jpg');
}
.inner-div:hover
{
    transform: scale(1.2);
}
</style>
</head>

<body>
<div class="outer-div"><div class="inner-div"></div></div>
</body>
</html>
```

CSS animation

- **More examples**
  - Examples on Canvas:
    - class_html5_css3_anim_1.html
  - Examples for geometric transformations:
    - class_html5_css3_anim_2.html

Selectors

- .

Pattern	Meaning	CSS3?	Selector Type
*	any element		Universal selector
E	an element of type E		Type selector
E[foo]	an E element with a "foo" attribute		Attribute selector
E[foo="bar"]	an E element whose "foo" attribute value is exactly equal to "bar" (quotes are optional)		Attribute selector
E[foo~="bar"]	an E element whose "foo" attribute value is a list of whitespace-separated values, one of which is exactly equal to "bar" (quotes are optional)		Attribute selector
E[foo^="bar"]	an E element whose "foo" attribute value begins with "bar" (quotes are optional)	Y	Attribute selector
E[foo$="bar"]	an E element whose "foo" attribute value ends with "bar" (quotes are optional)	Y	Attribute selector
E[foo*="bar"]	an E element whose "foo" attribute value contains "bar" somewhere within it (quotes are optional)	Y	Attribute selector
E[foo|="en"]	an E element whose "foo" attribute has a hyphen-separated list of values beginning (from the left) with "en" (quotes are optional)		Attribute selector

Pattern	Meaning	CSS3?	Selector Type
E:root	an E element, root of the document	Y	Structural pseudo-class
E:nth-child(n)	an E element, the *n*th child of its parent	Y	Structural pseudo-class
E:nth-last-child(n)	an E element, the *n*th child of its parent, counting from the last one	Y	Structural pseudo-class
E:nth-of-type(n)	an E element, the *n*th sibling of its type	Y	Structural pseudo-class
E:nth-last-of-type(n)	an E element, the *n*th sibling of its type, counting from the last one	Y	Structural pseudo-class
E:first-child	an E element, first child of its parent		Structural pseudo-class
E:last-child	an E element, last child of its parent	Y	Structural pseudo-class
E:first-of-type	an E element, first sibling of its type	Y	Structural pseudo-class
E:last-of-type	an E element, last sibling of its type	Y	Structural pseudo-class
E:only-child	an E element, only child of its parent	Y	Structural pseudo-class

Selectors

- .

Pattern	Meaning	CSS3?	Selector Type
`E:only-of-type`	an E element, only sibling of its type	Y	Structural pseudo-class
`E:empty`	an E element that has no children (including text nodes)	Y	Structural pseudo-class
`E:link, E:visited`	an E element being the source anchor of a hyperlink of which the target is not yet visited (`:link`) or already visited (`:visited`)		Link pseudo-classes
`E:focus, E:hover, E:active`	an E element during certain user actions		User action pseudo-classes
`E:target`	an E element being the target of the referring URI	Y	Target pseudo-class
`E:lang(fr)`	an element of type E in language "fr"		`:lang()` pseudo-class
`E:enabled, E:disabled`	a user interface element E that is enabled or disabled	Y	UI element states pseudo-classes
`E:checked`	a user interface element E that is checked (for instance a radio button or checkbox)	Y	UI element states pseudo-classes

Pattern	Meaning	CSS3?	Selector Type
`E::first-line`	the first formatted line of an E element		`::first-line` pseudo-element
`E::first-letter`	the first formatted letter of an E element		`::first-letter` pseudo-element
`E::before`	generated content before an E element		`::before` pseudo-element
`E::after`	generated content after an E element		`::after` pseudo-element
`E.warning`	an E element that has a class of "warning"		Class selector
`E#myid`	an E element with an ID equal to "myid"		ID selector
`E:not(s)`	an E element that does not match simple selector `s` (for example, `input:not(.warning)`)	Y	Negation pseudo-class
`E F`	an F element descendant of an E element		Descendant combinator
`E > F`	an F element child of an E element		Child combinator
`E + F`	an F element immediately preceded by an E element		Adjacent sibling combinator
`E ~ F`	an F element preceded by an E element	Y	General sibling combinator

Conclusion

- **Many effects are possible**:
  - To focus first on the semantic and basic HTML code
    - Well structured, well defined
    - Well identified objects
  - To create the style sheets
    - Think about the different media
    - The style
    - Use animation appropriately
- Explore further
  - https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/SVG_and_CSS
  - Reading:
    - Book: HTML5 and CSS3, Castro & Hyslop
    - Check the examples given on Canvas
- Next Lab
  - Topic: Practice with Animations using CSS3 only (*no JS yet*)