

HTML5 & CSS3

A chance to Do things Differently

Eng. Niveen Nasr El-Den
iTi

Day 3



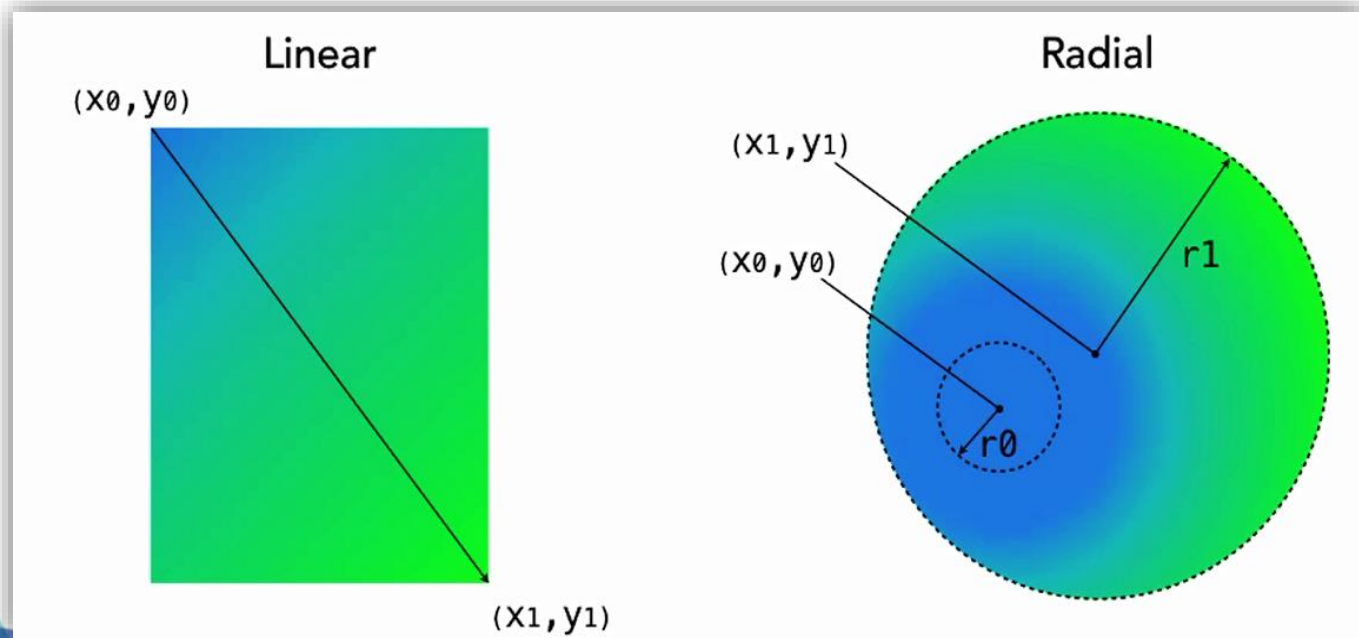
Canvas

Gradient

- Gradient can be used to fill rectangles, circles, lines, text, etc.. it can be used anywhere a stroke or fill is used
- Two types of gradient

▷ Linear Gradient

▷ Radial Gradient



Gradient

- Linear Gradient

- `createLinearGradient(startX, startY, endX, endY);`

- Radial Gradient

- `createRadialGradient(startX, startY, startRadius, endX, endY, endRadius);`

- Note:

- Add color stops to create color transitions using `addColorStop(offset, color);`
 - It can be called multiple times to change a gradient
 - Its offset value between 0.0 and 1.0

Dealing with Image

- To draw an image on canvas area we use
 - ▷ *drawImage(imgObj, x, y [, width, height])*
 - *imgObj* defines image required to be displayed, it must be created first and wait for being loaded before instantiating *drawImage()*.
 - *x,y* defines top left corner of the image relative to the top left corner of the canvas (0,0)
 - *width, height* define width, height of the displayed image
 - ▷ Note:
 - Construct your image object using “new Image()”

<https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/drawImage>

Transformation

- Transformation affects all drawing operations that come after it
- 3 basic transformation
 - ▷ Translate
 - ▷ Scale
 - ▷ Rotate
- Transformation is additive
- Its good using **save()** & **restore()** for the context state

Scaling, Rotating & Translating

- `scale(x, y)`
 - resize current drawing either bigger or smaller
- `rotate(angle)`
 - rotate the current context around the origin within the canvas area
- `translate(x, y)`
 - move current context within the canvas area into a different point

Saving & Restoring Canvas State

- Every canvas object contains a stack of drawing states.
- The canvas state can store:
 - ▷ strokeStyle
 - ▷ fillStyle
 - ▷ font
 - ▷ globalAlpha
 - ▷ lineWidth
 - ▷ lineCap
 - ▷ lineJoin
 - ▷ miterLimit
 - ▷ The current transformation matrix (rotation, scaling, translation)
 - ▷ shadowOffsetX
 - ▷ shadowOffsetY
 - ▷ shadowBlur
 - ▷ shadowColor
 - ▷ globalCompositeOperation
 - ▷ textAlign
 - ▷ textBaseline
 - ▷ The current clipping region

References

- <http://slides.html5rocks.com>
- <http://www.tutorialspoint.com>
- <http://www.html5canvastutorials.com/tutorials/html5-canvas-element/>
- <http://okeschool.com/tutorial/549/canvas/canvas-basics/canvas-introduction.html>
- <http://cheatsheetworld.com/programming/html5-canvas-cheat-sheet/>
- [http://www.kirupa.com/canvas/canvas transformations.htm](http://www.kirupa.com/canvas/canvas_transformations.htm)

CSS3

Other Selectors & New Properties

Positioning Styles

- Elements can be positioned as:

- ▷ position:relative

- How to position an element relative to its normal position.

- ▷ position:absolute

- How to position an element using an absolute value.

- ▷ position:fixed

- How to position an element using fixed value

- ▷ position:static

- The default position of an element

- ▷ position:sticky

Positioning Styles

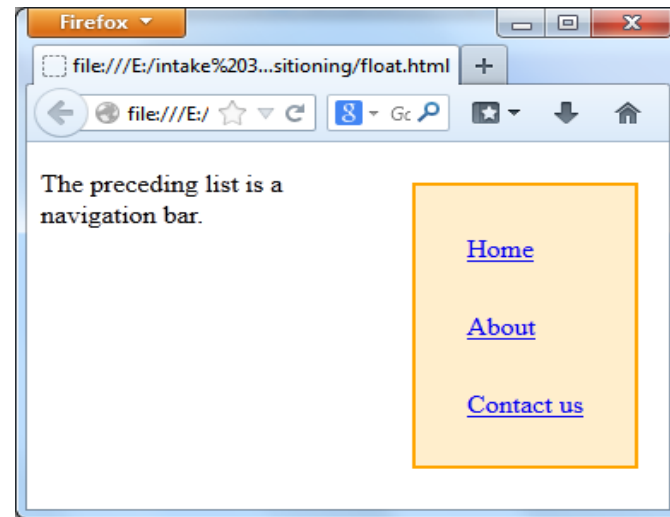
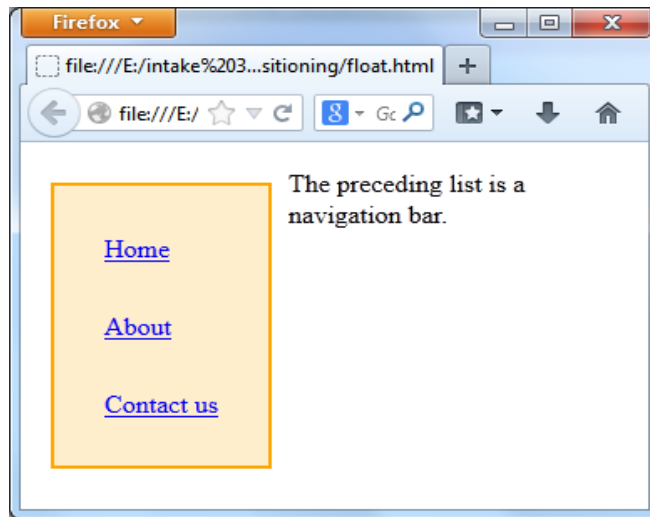
(offset properties)

Property	Effects
left: <i>n</i>	Sets the left edge of the element relative to its container element; <i>n</i> is a <i>string</i> measurement unit, e.g., 100px.
top: <i>n</i>	Sets the top edge of the element relative to its container element; <i>n</i> is a <i>string</i> measurement unit, e.g., 100px
pixelLeft	Sets the left edge of the element relative to its container element; <i>n</i> is <i>numeric</i> for use in calculations, e.g., 100.
pixelTop	Sets the top edge of the element relative to its container element; <i>n</i> is <i>numeric</i> for use in calculations, e.g., 100.

Example!

float and clear

- Declares whether a box should float to the left or right of other content, or whether it should not be floated at all.



Example!

visibility

- The visibility property determines if an element is visible or not.
- Hidden element pre-allocates its space on the page if not positioned absolute
- The general format is:
visibility:hidden | visible

display

- The **display** property differs from the visibility property in that it does *not* reserve space for hidden items
- Some possible values for display:
 1. none
 2. block
 3. inline
 4. inline-block
- The general format is :

display:block | inline | none

OR

object.style.display="block | inline | none"

Example!

z-index

- The z-index property is used to place an element "behind" another element.
- z-index only affects elements that have a **position** value other than static
- Default z-index is 0.
- The higher number the higher priority. z-index: -1 has lower priority.
- The general format is:

z-index:n

OR

object.style.zIndex=n

Example!

Dynamic sizing

Property	Effects
<code>width: <i>value</i></code>	Sets the width of the element; <i>n</i> is a string measurement, either in pixels or percentages.
<code>height: <i>n</i></code>	Sets the height of the element; <i>n</i> is a string measurement, either in pixels or percentages.
<code>pixelWidth</code>	Sets the width of the element; <i>n</i> is numeric for use in calculations.
<code>pixelHeight</code>	Sets the height of the element; <i>n</i> is numeric for use in calculations.

overflow

- Specifies if content of a block-level **element** should be **clipped** when it is **larger** than the **parent** element.

- The general format is :

overflow:visible | hidden | scroll

OR

object.style.overflow="visible | hidden | scroll "

Example!

CSS Selectors

- Several types of selectors are defined for use when implementing Style Sheets:
 - ✓ Simple Basic Selectors
 - ✓ Attribute selectors
 - ✓ Combinators
 - ✓ Pseudo-Classes
 - ✓ Pseudo-Elements
- A selector can contain a chain of one or more simple selectors separated by combinators, optionally followed by attribute selectors, ID selectors, or pseudo-classes. but it can contain only one pseudo-element, which must be appended to the last simple selector in the chain

Pseudo-Classes Selector

- A pseudo-class is similar to a class in HTML, but it's not specified explicitly in the markup.
- pseudo-class selectors
 - ✓ Dynamic
 - ✓ Link / Target
 - ✓ UI Element
 - ✓ Structural

Pseudo-Classes Selector

- Dynamic pseudo-class selectors
 - ▷ **:active**
 - matches any element that's being activated by the user → the "pressed" state of a button-style link
 - ▷ **:hover**
 - matches elements that are being designated by a pointing device. i.e. when the mouse cursor rolls over a link, that link is in it's hover state and this will select it.
 - ▷ **:focus**
 - matches any element that's currently in focus

Pseudo-Classes Selector

- Link / Target pseudo-class selector
 - ▷ **:link**
 - matches link elements that are **unvisited**
 - ▷ **:visited**
 - matches link elements that have been **visited**
 - ▷ **:target**
 - matches an element that's the target of a fragment identifier in the document's URI

Pseudo-Classes Selector

- UI element pseudo-class selectors
 - ▷ **:enabled**
 - matches user interface elements that are enabled
 - ▷ **:disabled**
 - matches user interface elements that are disabled
 - ▷ **:checked**
 - matches elements like checkboxes or radio buttons that are checked.
 - ▷ **:valid**
 - ▷ **:required**
 - ▷ etc.

Pseudo-Classes Selector

- Structural (Position-Number based) pseudo-class selectors
 - ▷ :first-child
 - ▷ :last-child
 - ▷ :only-child
 - ▷ :nth-child(n)
 - ▷ :nth-last-child(n)
 - ▷ :first-of-type
 - ▷ :last-of-type
 - ▷ :only-of-type
 - ▷ :nth-of-type(n)
 - ▷ :nth-last-of-type(n)
 - ▷ etc.

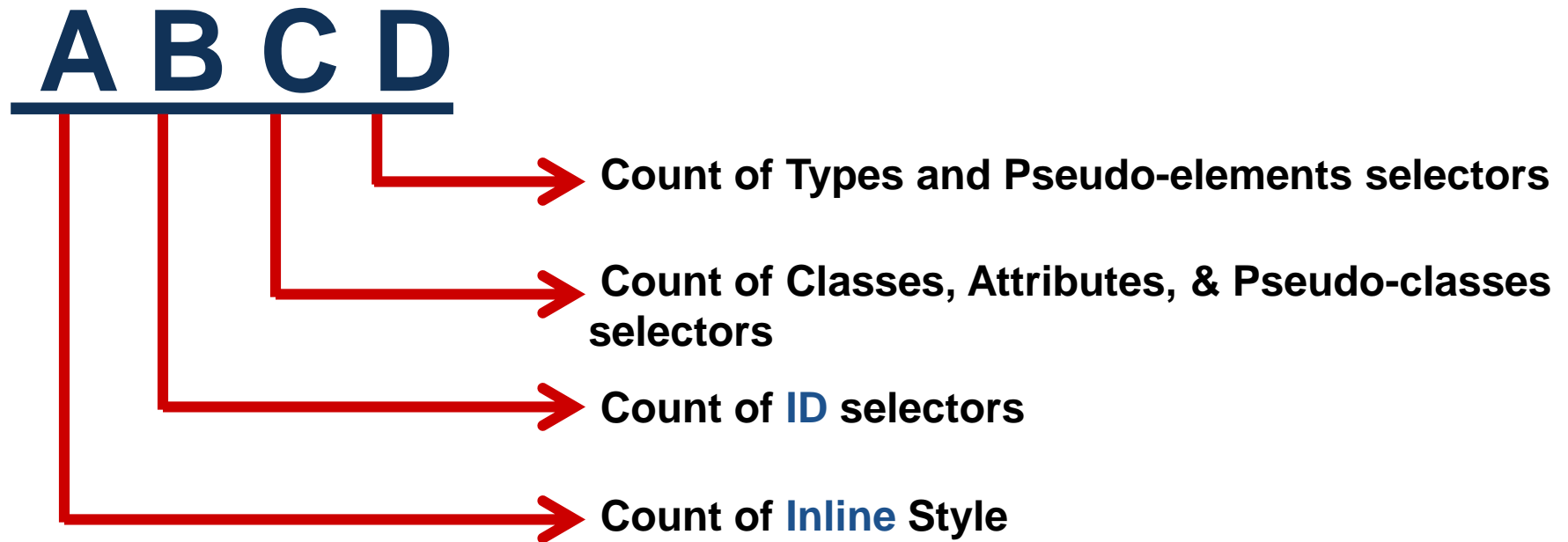
Pseudo-Element Selector

- Pseudo-elements match virtual elements that don't exist explicitly in the document tree.
- In CSS1 and CSS2, pseudo-elements start with a colon (:). In CSS3, pseudo-elements start with a double colon (::), which differentiates them from pseudo-classes.
- **:first-letter**
 - ▷ represents the first character of the first line of text within an element
- **:first-line**
 - ▷ represents the first formatted line of text
- **:before**
 - ▷ specifies content to be inserted before another element
- **:after**
 - ▷ specifies content to be inserted after another element
- **::selection**
 - ▷ represents a part of the document that's been highlighted by the user

<https://oioiam.github.io/entities/>

<https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>

Specificity



Specificity

Example

body#home div#warning p.message { color: red; }

Inline Style	IDs	Classes, Attributes, and Pseudo-classes	Element Types and Pseudo-elements

Specificity

Example

ul#nav li.active a { color: red; }

Inline Style	IDs	Classes, Attributes, and Pseudo-classes	Element Types and Pseudo-elements

Specificity

Example

```
#footer *:not(nav) li{ color: red; }
```

Inline Style	IDs	Classes, Attributes, and Pseudo-classes	Element Types and Pseudo-elements

Note:

The **:not()** sort-of-pseudo-class adds no specificity by itself, only what's inside the parents is added to specificity value.

Specificity

Example

body#home div#warning p.message { color: red; }

p.message { color: green; }

#home #warning p.message { color: yellow; }

#warning p.message { color: white; }

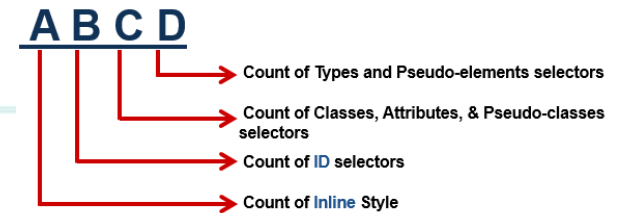
body#home div#warning p.message { color: blue; }

p { color: teal; }

*** body#home>div#warning p.message** { color: red; }

#warning p { color: black; }

Specificity



Selector	A	B	C	D
body#home div#warning p.message	0	2	1	3
* body#home>div#warning p.message	0	2	1	3
body#home div#warning p.message	0	2	1	3
#home #warning p.message	0	2	1	1
#warning p.message	0	1	1	1
#warning p	0	1	0	1
p.message	0	0	1	1
p	0	0	0	1

Specificity Important Notes

- The universal selector (*) has no specificity value
- Pseudo-elements (e.g. :first-line) get 0,0,0,1 unlike their pseudo-class which get 0,0,1,0
- The pseudo-class :not() adds no specificity by itself, only what's inside it's parentheses
- The **!important** value appended a CSS property value is an *automatic win*.

New Properties

New Properties

- **@rule**
- **Animation**
- **Transition**
- **Transformation (2D,3D)**
- **...etc.**

Opacity

- Specifies the transparency of an element
- Opacity has a default initial value of 1
 - ▷ Range: 0.0 (invisible) to 1.0 (solid)
- Not inherited, but a child element less transparent than the parent.

Example!

Shadowing

- Text Shadow

<http://www.cssmatic.com/>

- Box Shadow

<https://cssgenerator.org/box-shadow-css-generator.html>

- The box-shadow property allows designers to easily implement multiple drop shadows (outer or inner) on box elements, specifying values for color, size, blur and offset.

Example!

Vendor Extension Prefixes

<https://autoprefixer.github.io/>

Prefix	Organization
-moz-	Mozilla Foundation
-ms-	Microsoft
-o-	Opera Software
-webkit-	Safari and Chrome


@rule

- At-rules are instructions or directives to the CSS parser. They can be used for a variety of purposes.
 - ▷ @charset
 - ▷ @import
 - ▷ @media
 - ▷ @page
 - ▷ @font-face
 - ▷ @namespace
 - ▷ @keyframe

Font Style

@font-face & Different Formats

<http://fontsquirrel.com/>
<https://www.dafont.com/>

ttf	✓	✓	✓	✓	✓
otf	✓	✓	✓	✓	✓
svg	✓		✓		✓
eot				✓	
woff	✓	✓	✓	✓	✓
					

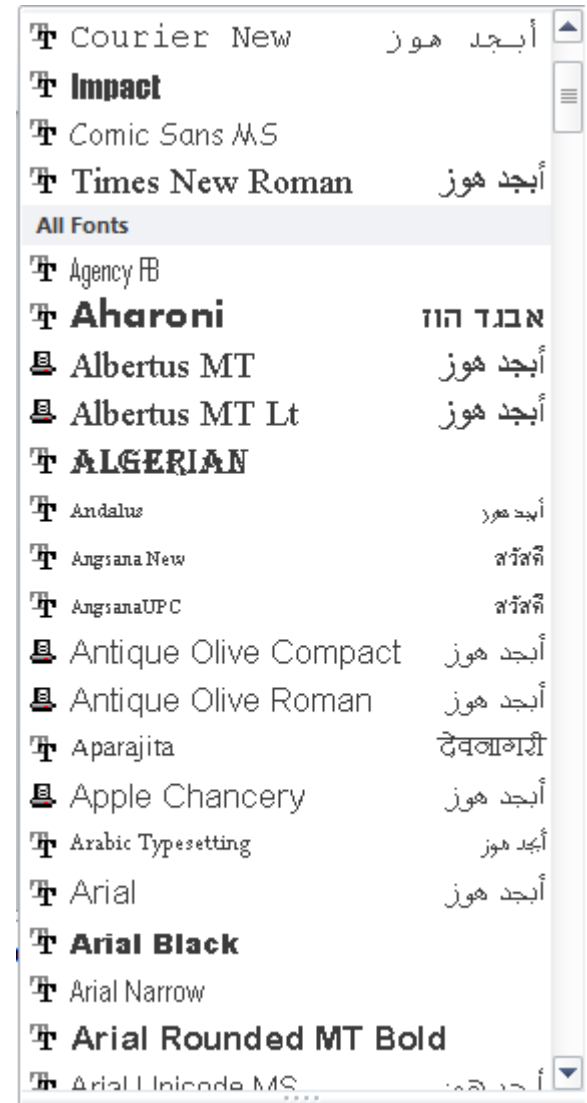
@font-face: allows specifying custom fonts

Example!

<https://fonts.google.com/>

Font Collections

- Serif
 - Time New Roman
- Sans-serif
 - Arial
- Cursive
 - Comic sans
- Fantasy
 - **Impact**
- Monospace
 - Courier New



Animation

Property	Description
@keyframes	Specifies the animation
animation	A shorthand property for all the animation properties below, except the animation-play-state property
animation-name	Specifies a name for the @keyframes animation
animation-duration	Specifies how many seconds an animation takes to complete one cycle
animation-timing-function	Specifies the speed curve of the animation (linear ease ease-in ease-out ease-in-out..)
animation-delay	Specifies when the animation will start
animation-iteration-count	Specifies the number of times an animation should be played
animation-direction	Specifies whether or not the animation should play in reverse on alternate cycles (normal alternate..)

Transform

- Applies a 2D or 3D transformation to an element
 - ▷ rotate, scale, skew, translate.. etc.
 - ▷ i.e. `scale(x,y)`, `scale3d(x,y,z)`, `scaleX(x)`, `scaleY(y)`, `scaleZ(z)`.. etc.

Example!

Transform

Property	Description
transform	Applies a 2D or 3D transformation to an element rotate, scale, skew, translate.. etc. i.e. scale(x,y), scale3d(x,y,z), scaleX(x), scaleY(y), scaleZ(z).. etc.
transform-origin	Allows you to change the position on transformed elements <i>x-axis y-axis z-axis</i> ;

Example!

Transition

- Allows property changes in CSS values to occur smoothly over a specified duration.

Property	Description
transition	A shorthand property for setting the four transition properties
transition-property	Specifies the name of the CSS property the transition effect is for
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-timing-function	Specifies the speed curve of the transition effect
transition-delay	Specifies when the transition effect will start

Example!

References

- <http://css-tricks.com>
- <http://css.maxdesign.com.au/selectutorial>
- <https://developer.mozilla.org/en-US/docs/Web/API/Window/getComputedStyle>
- <http://code.tutsplus.com/>
- <http://www.sitepoint.com>
- <http://www.css3.info/>
- <https://www.html5rocks.com/en/tutorials/filters/understanding-css/>
- <https://css-tricks.com/examples/ShapesOfCSS/>

Assignment