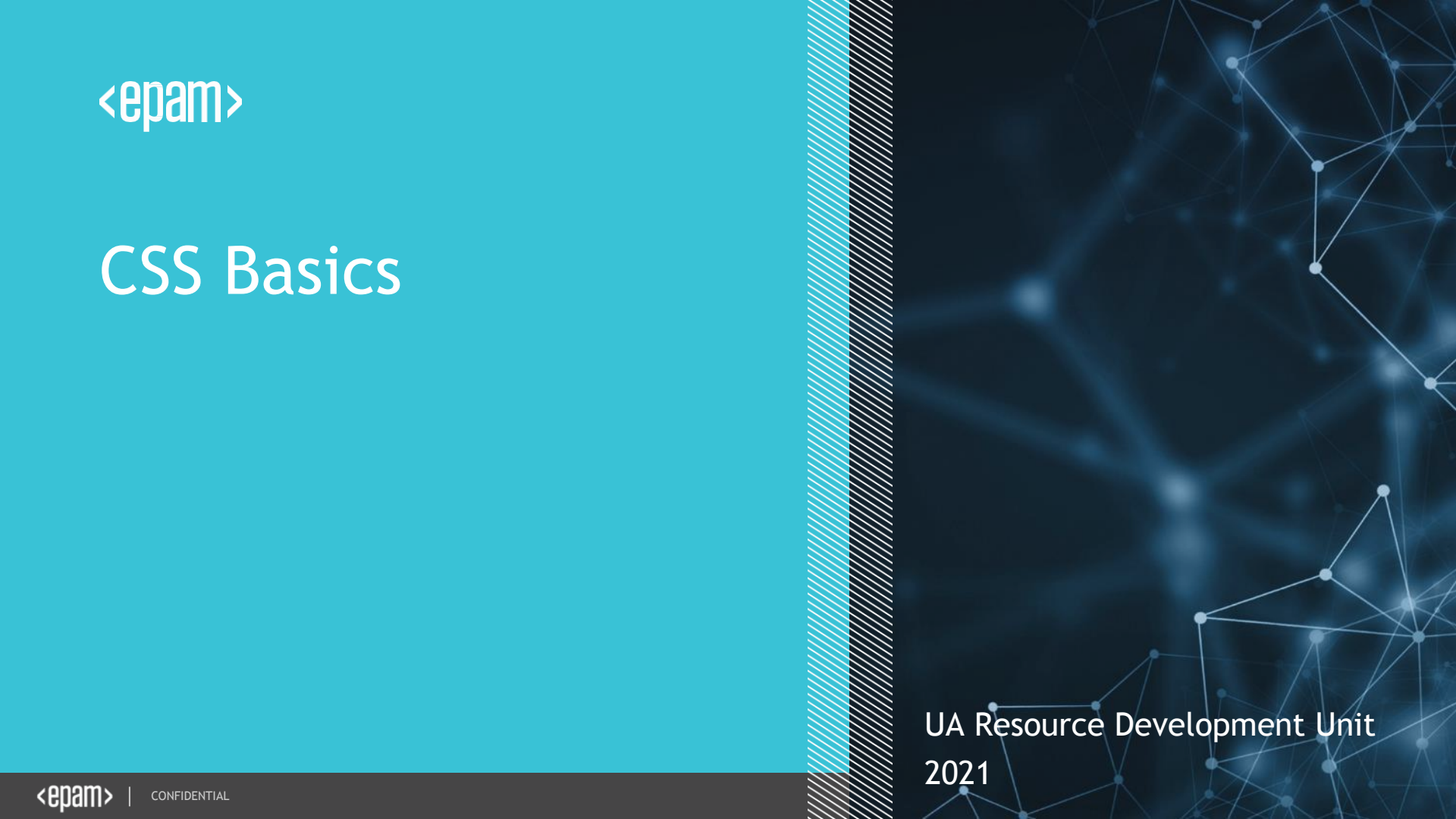

epam
CSS Basics
UA Resource Development Unit
2021
epam | CONFIDENTIAL

AGENDA

1. CSS definitions
2. The basic syntax of CSS
3. How to add styles to the page
4. Basic selectors
5. CSS Style Guide

CSS definitions

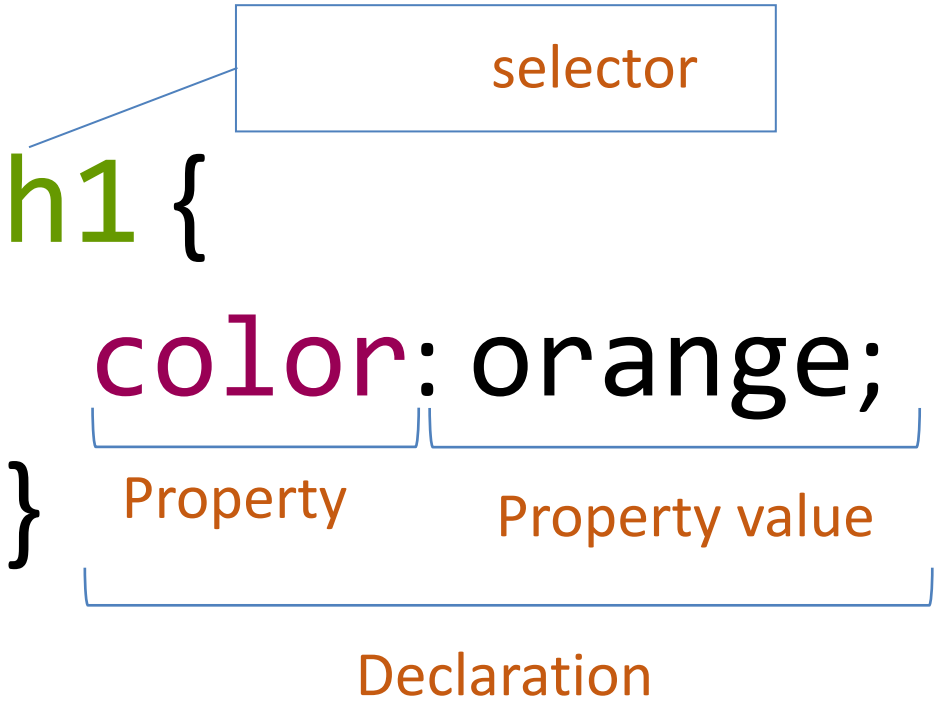
Cascading Style Sheets (CSS) are a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects like SVG or XHTML).

CSS describes how elements should be displayed on screen, on paper, in speech, or on other media. CSS is the only document styling language that browsers understand.

CSS has a standardized W3C specification.
CSS1 is now obsolete,
CSS2.1 is a recommendation,
CSS3 is splitted into smaller modules, progressing on the standardization track.

The types of styles:
a browser's style
an author's style
a user's styles

The basic syntax of CSS



Priority matters:

```
p { color: green; }
p { color: RED; }
```

Comments in CSS-file:

```
/*
Style is designed for introductory purposes
*/
div {
width: 200px; /* a width of the block */
}
```

CSS Style Guide

- Put a space before the opening brace { in rule declarations
- In properties, put a space after, but not before, the : character.
- Put closing braces } of rule declarations on a new line
- Trailing semi-colon (;) on our last declaration
- Put blank lines between rule declarations
- 80 character wide columns

CSS Style Guide

- Use soft tabs (2 spaces) for indentation
- Prefer dashes over camelCasing in class names
- Underscores and PascalCasing are okay if you are using BEM
- Do not use ID selectors
- When using multiple selectors in a rule declaration, give each selector its own line
- Related selectors on the same line
- Unrelated selectors on new lines

CSS Style Guide

```
.snapshot-box h2 { padding: 0 0 6px 0; font-weight: bold; position:
absolute; left: 0; top: 0; }
```

```
.snapshot-box h2
{
  position: absolute;
  left: 0;
  top: 0;
  padding: 0 0 6px 0;
  font-weight: bold;
}
```

```
.snapshot-box h2
{
 padding: 0 0 6px 0;
}
```

CSS Style Guide

```
.snapshot-box h2, .profile-box h2, .order-box h2 {
  padding: 0 0 6px 0;
  font-weight: bold;
}
```

```
.snapshot-box h2,
.profile-box h2,
.order-box h2 {
  padding: 0 0 6px 0;
  font-weight: bold;
}
```

CSS Style Guide

```
.avatar{
    border-radius:50%;
    border:2px solid white; }
.no, .nope, .not_good {
    // ...
}
#lol-no {
  // ...
}
```

```
.avatar {
  border-radius: 50%;
  border: 2px solid white;
}

.one,
.selector,
.per-line {
  // ...
}
```

CSS Style Guide

```
.news {
  background: #eee;
  border-radius: 5px;
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, .25);
}
.social {
  background: #eee;
  border-radius: 5px;
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, .25);
}
```

```
.news,
.social {
  background: #eee;
  border-radius: 5px;
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, .25);
}
```

```
.modal {
  background: #eee;
  border-radius: 5px;
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, .25);
}
```

How to add styles to the page. External style

example-1.html:

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
    <title>Styles</title>
    <link rel="stylesheet" href=
"css/style.css">
</head>
<body>
    <h1>Heading</h1>
    <p>Content</p>
</body>
</html>
```

style.css:

```
h1 {
color: #000080;
font-size: 200%;
text-align: center;
}
p {
padding: 20px;
background: yellow;
}
```

How to add styles to the page. Global page styles

example-2.html:

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
    <title>Global styles</title>
    <style>
        h1 {
            font-size: 120%;
            font-family: Verdana, Arial, Helvetica, sans-serif;
            color: #333366;
        }
    </style>
</head>
<body>
    <h1>Hello, world!</h1>
</body>
</html>
```

How to add styles to the page. Tags inner styles

example-3.html:

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Inner styles</title>
</head>
<body>
<h1 style="font-size: 120%; color: #cd66cc">Hello, world!</h1>
</body>
</html>
```

How to add styles to the page. Import of styles

example-4.html:

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Styles</title>
<style>
@import "style-1.css";
@import url("style-2.css");
</style>
</head>
<body>
<h1>Heading</h1>
<p>Content</p>
</body>
</html>
```

style-1.css:

```
h1 {
color: #000080;
font-size: 200%;
text-align: center;
}
p {
padding: 20px;
background: yellow;
}
```

style-2.css:

```
body { background: #fc0; }
p { font-weight: bold; }
```

How to add styles to the page

All described methods of using CSS can be used either alone or in combination with each other.
In the second case, is necessary to remember their hierarchy.

- tag inner style - highest priority
- global style, external style - lower priority

Specify the type of media. @import

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Style import</title>
<style>
@import "/style/main.css" screen; /* Style for output to monitor */
@import "/style/print-and-speech.css" print, speech; /* Style Print and
Screenreaders */
</style>
</head>
<body>
<p>...</p>
</body>
</html>
```

Device types

Value	Description	Status
all	Used for all media type devices	Active
aural	Used for speech and sound synthesizers	Deprecated
braille	Used for braille tactile feedback devices	Deprecated
embossed	Used for paged braille printers	Deprecated
handheld	Used for small or handheld devices	Deprecated
print	Used for printers	Active
projection	Used for projected presentations, like slides	Deprecated
screen	Used for computer screens, tablets, smart-phones etc.	Active
speech	Used for screenreaders that "read" the page out loud	Active
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals	Deprecated
tv	Used for television-type devices	Deprecated

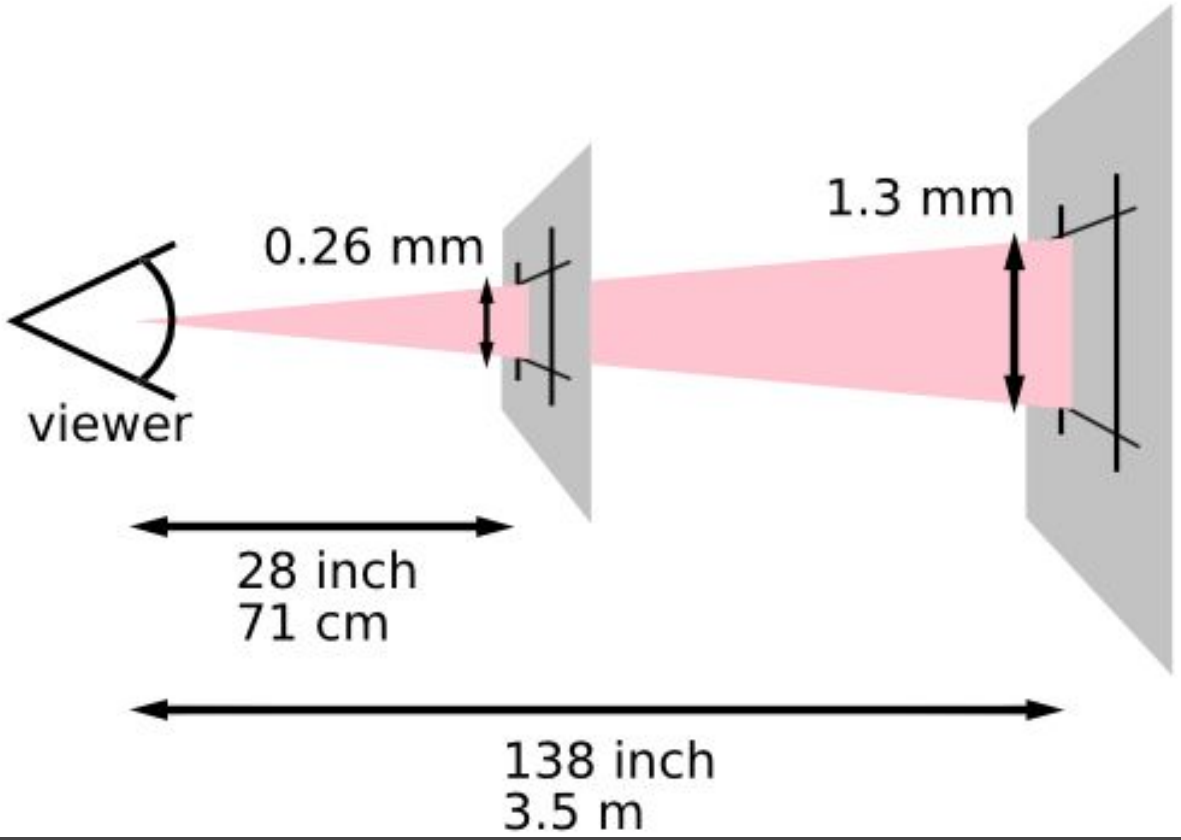
Specify the type of media. @media

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>Device types</title>
<style>
@media screen {
/* Style for display in a browser */
body {
font-family: Arial, Verdana, sans-serif;
}
}
@media print {
/* Style for printing */
body {
font-family: Times, 'Times New Roman', serif;
}
}
</style>
</head>
<body>
<p>...</p>
</body>
</html>
```

Specify the type of media. Attribute “media”

```
<html>
<head>
<meta charset="utf-8">
<title>Devices</title>
<link media="print, handheld" rel="stylesheet"
href="print.css">
<link media="screen" rel="stylesheet"
href="main.css">
</head>
<body>
<p>...</p>
</body>
</html>
```

Values and Units



Absolute Lengths

cm	centimeters
mm	millimeters
in	inches (1in = 2.54cm)
px	pixels
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

Relative Lengths

em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport
vh	Relative to 1% of the height of the viewport
vmin	Relative to 1% of viewport's smaller dimension
vmax	Relative to 1% of viewport's larger dimension

Using CSS custom properties

CSS Variables

CSS Variables are entities defined by CSS authors which contain specific values to be reused throughout a document

They are set using custom property notation (e.g. --main-color: black;) and are accessed using the var() function (e.g. color: var(--main-color);)

[Read more in spec]
[Read more in MDN]

```
// Global option
:root {
--component-shadow: 0 .5rem 1rem rgba(0,0,0,.1);
}
// Put it to use
.component {
 box-shadow: var(--component-shadow);
}
.remove-shadow {
// Because the custom property is within this ruleset,
// it will only remove the shadow for this class
 --component-shadow: none;
 }
```

Declaring a variable

```
element {
--main-bg-color: brown;
}
```

```
:root {
--main-bg-color: brown;
}
```

```
.two {
--test: 10px;
}
.three {
--test: 2em;
}
```

Main types of style properties

Strings:

```
li:before {content: 'Hello'}
```

Numbers:

```
p {
font-weight: 600; line-height: 1.2;
}
```

URLs:

```
a { background: url(warn.png)
no-repeat }
```

Keywords

```
p { text-align: right; }
```

Color:

- By hexadecimal values: #6609CF, #fc0
- By name: white, silver, gray, black, red, orange, ...
- RGB: rgb(255, 0, 0)
- RGBA: rgba(0,255,0,0.3)
- HSL: hsl(120,100%, 25%)
- HSLA: hsla(120,100%, 50%, 0.3)

Using the variable

```
element {
  background-color: var(--main-bg-color);
}
.two {
  color: var(--my-var, red);
/* Red if --my-var is not defined */
}
.three {
  background-color: var(--my-var, var(--my-background, pink));
/* pink if my-var and --my-background are not defined */
}
.three {
  background-color: var(--my-var, --my-background, pink);
/* Invalid: "--my-background, pink" */
}
```

Basic selectors

.class

.intro

Selects all elements with class="intro"

```
.intro {
font-size: 11px;
}
```

#id

#firstname

Selects the element with id="firstname"

```
#firstname {
width: 520px;
padding: 100px;
background: #fc0;
}
```

*

*****

Selects all elements

```
* {
font-size: 11px;
}
```

element

p

Selects all <p> elements

```
p {
text-align:  right;
color: green;
}
```

Combinators

element,element

div, p

Selects all elements and all <p> elements

element element

div p

Selects all <p> elements inside elements

element>element

div > p

Selects all <p> elements where the parent is a element

element+element

div + p

Selects all <p> elements that are placed immediately after elements

element1~element2

div ~ p

Selects every <p> element that are preceded by a element

Descendant Selectors

```
* { margin: 0; }
a {
color: red;
}
div b {
font-family: Times, serif;
}
.main-nav {
margin: 0;
padding: 0;
list-style: none;
}
.main-nav li {
margin: 0 0 10px 0;
padding: 3px;
background: #fc0;
}
.main-nav a {
color: #000;
}
```

```
<p><b>Bold text</b>, normal text</p>
<div>Hello <b>another bold text</b></div>
<p><a href="#">Some link</a></p>
<ul class="main-nav">
<li><a href="#">Home page</a></li>
<li><a href="#">About me</a></li>
<li><a href="#">Contacts</a></li>
</ul>
```

Adjacent Sibling Selector (one next)

```
<style>
b + i {
color: red;
}
</style>
<p>Lorem <b>ipsum </b> dolor sit amet,
<i>"first i"</i> adipiscing <i>"second i"</i>
elit.</p>
```

Demo:

Lorem **ipsum** dolor sit amet, *"first i"* adipiscing *"second i"* elit.

General Sibling Selector (all next)

```
<style>
b ~ i {
color: red;
}
</style>
<p>Lorem <b>ipsum </b> dolor sit amet,
<i>"first i"</i> adipiscing <i>"second i"</i>
elit.</p>
```

Lorem **ipsum** dolor sit amet, *"first i"* adipiscing *"second i"* elit.

Child Selector

```
<style>
h2 {
color: green;
margin-top: 0;
}
h1 + h2 {
margin-top: 30px;
color: blue;
}
p .name {
outline: dotted #333 1px;
}
p > .name {
color: red;
}
</style>
```

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h2>Heading 2</h2>
<h2>Heading 2</h2>
<p>
Lorem <i class="name">ipsum</i>
dolor <span>sit
<i class="name">amet</i></span>
</p>
```

Demo:

Heading 1

Heading 2

Heading 2

Heading 2

Lorem *ipsum* dolor sit *amet*

Attribute selectors

[target]

Selects all elements with a target attribute

[title~=flower]

Selects all elements with a title attribute containing the word "flower"

a[href^="https"]

Selects every <a> element whose href attribute value begins with "https"

a[href*="w3c"]

Selects every <a> element whose href attribute value contains the substring "w3c"

[target=_blank]

Selects all elements with target="_blank"

[lang|=en]

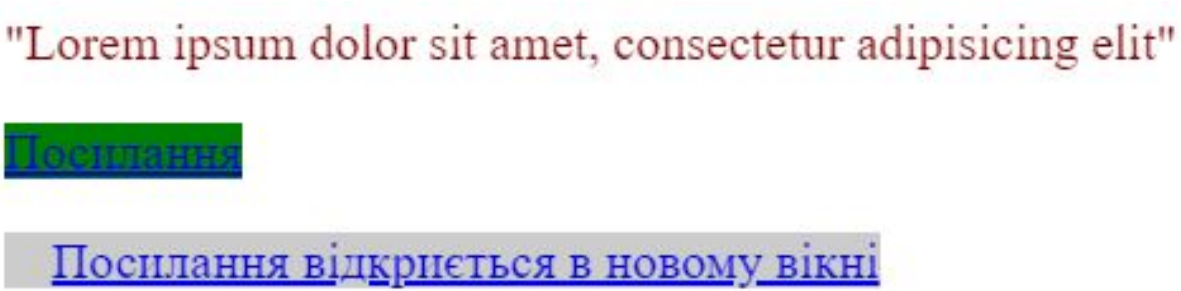
Selects all elements with a lang attribute value starting with "en"

a[href$=".pdf"]

Selects every <a> element whose href attribute value ends with ".pdf"

Attributes selectors

```
<style>
[title] {
color: maroon;
}
a[href] {
background: green;
}
a[target="_blank"] {
background: #ccc;
padding-left: 15px;
}
</style>
<q title="Some title text">Lorem ipsum dolor sit
amet, consectetur adipisicing elit</q>
<p><a href="#link">Посилання</a></p>
<p><a href="#link" target="_blank">Посилання
відкриється в новому вікні</a></p>
```



Advanced attributes selectors

```
/* Attribute value starts with some text */
a[href^="http://" ] {
color: red;
}
/* If link ends with ".com" */
a[href$=".com"] {
background: #fc0;
}
/* If link contains "google" */
[href*="google"] {
background: yellow;
}
/* One of the several attribute values */
[title~="block"] {
color: green;
}
```

```
<p><a href="http://gmail.com"
target="_blank">External link to
gmail.com</a></p>
<p><a
href="http://www.yahoo.com">Yahoo.com</a></p>
<p><a href="http://google.com.ua">Search the
web</a></p>
<h3 title="block tag">Heading</h3>
```

External link to gmail.com

Yahoo.com

Search the web

Heading

Pseudo-classes

Read more

:checked

```
:checked {
  margin-left: 25px;
  border: 1px solid blue;
}
```

:disabled

```
input:disabled {
  background: #ccc;
}
```

:focus

```
a:focus {
  border-bottom: 1px solid;
  background: #BAE498;
}
```

:link

```
a:link { color: #265301; }
```

:active

```
a:active {
  background: #265301;
  color: #CDFEAA;
}
```

:empty

```
div:empty { background: lime; }
```

:hover

```
a:hover {
  border-bottom: 1px solid;
  background: #CDFEAA;
}
```

:not (selector)

```
input:not([type="submit"]){}
```

Structural pseudo-classes

Selector	Example	Example description
:first-child	p:first-child	Selects every <p> element that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:last-child	p:last-child	Selects every <p> element that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
: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
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent

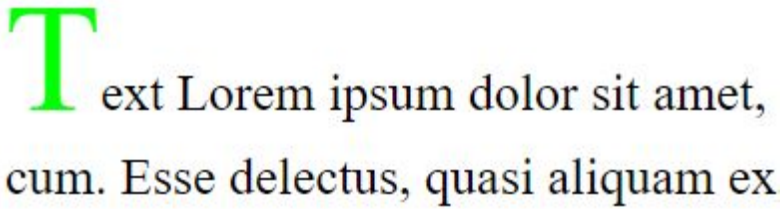
Pseudoclasses

```
a:link {
color: #036; /* The color of not visited links */
}
a:hover {
color: #f00; /* The color of links on mouse pointer hovering */
}
a:visited {
color: #606; /* The color of visited links */
}
a:visited:hover {
color: #303; /* The color of not visited links on hover */
}
a:active {
color: #ff0; /* The color of active links */
}
b:first-child {
color: red; /* The color of the first tag */
}
b:last-child {
color: green; /* The color of the last tag */
}
```

Pseudo-elements

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element

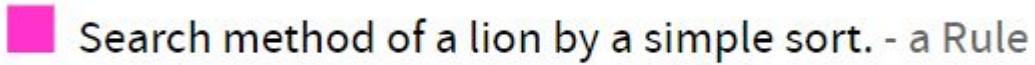
```
p::first-letter {
  color: lime;
  font-size: 300%
}
```



Pseudoelements

```
<style>
p:before {
content: "";
display: inline-block;
width: 20px;
height: 1em;
margin-right: 10px;
background: #f3c;
}
p:after {
content: " - a Rule";
color: #666;
}
</style>
```

```
<p>Search method of a lion by a simple sort.</p>
```



Avoid qualifying ID and class names with type selectors.

Unless necessary (for example with helper classes), do not use element names in conjunction with IDs or classes.

Avoiding unnecessary ancestor selectors is useful for performance reasons.

```
ul#example {}
div.error {}
```

```
#example {}
.error {}
```

```
<main class='mainly'>
  <p>Lorem ipsum</p> <!-- I'd like to style this paragraph-->
</main>
```

```
main.mainly p {

/*This style*/

}
```

```
/* Instead, assign a class name to p : <p
                                class='paragraphly' /> */
                                .paragraphly { }
```

Grouping

```
h1,
h2,
h3 {
font-family: Arial, Helvetica, sans-serif;
}
h1 {
font-size: 160%;
color: #003;
}
```

!important

```
selector
{
    property: property value !important;
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Document</title>
    <style>
    #one { color: red; }
    #two { color: blue !important; }
    </style>
  </head>
  <body>

    <p id="one" style="color: yellow;">First paragraph</p>
    <p id="two" style="color: yellow;">Second paragraph.</p>

  </body>
</html>
```

cascading

Origin and Importance

Scope

Specificity

Order of Appearance

Cascading

Cascading refers simultaneous use of different style rules to document elements by connecting multiple style files, inheritance of properties and other methods.

The higher style rule is placed in this list, the lower its priority and vice versa:

1. Browser’s style
2. Author’s style
3. User’s style
4. The author's style adding !Important
5. The user’s style adding !Important

Cascading

1. All the declared values applied to an element are collected, for each property on each element.
2. There may be zero or many declared values applied to the element.
3. There is at most one cascaded value per property per element
4. Every element has exactly one specified value per property.
5. Every element has exactly one computed value per property.
6. The used value is transformed to the actual value based on constraints of the display environment.

Example

	Property	Winning declaration	Cascaded value	Specified value	Computed value	Used value	Actual value
(a)	'text-align'	`text-align: left`	'left'	'left'	'left'	'left'	'left'
(b)	'border-top-width', 'border-right-width', 'border-bottom-width', 'border-left-width'	`border-width: inherit`	'inherit'	'4.2px'	'4.2px'	'4.2px'	'4px'
(c)	'width'	(none)	(none)	'auto' (initial value)	'auto'	'120px'	'120px'
(d)	'list-style-position'	`list-style-position: inherit`	'inherit'	'inside'	'inside'	'inside'	'inside'
(e)	'list-style-position'	`list-style-position: initial`	'initial'	'outside' (initial value)	'outside'	'outside'	'outside'
(f)	'font-size'	`font-size: 1.2em`	'1.2em'	'1.2em'	'14.1px'	'14.1px'	'14px'
(g)	'width'	`width: 80%`	'80%'	'80%'	'80%'	'354.2px'	'354px'
(h)	'width'	`width: auto`	'auto'	'auto'	'auto'	'134px'	'134px'
(i)	'height'	`height: auto`	'auto'	'auto'	'auto'	'176px'	'176px'

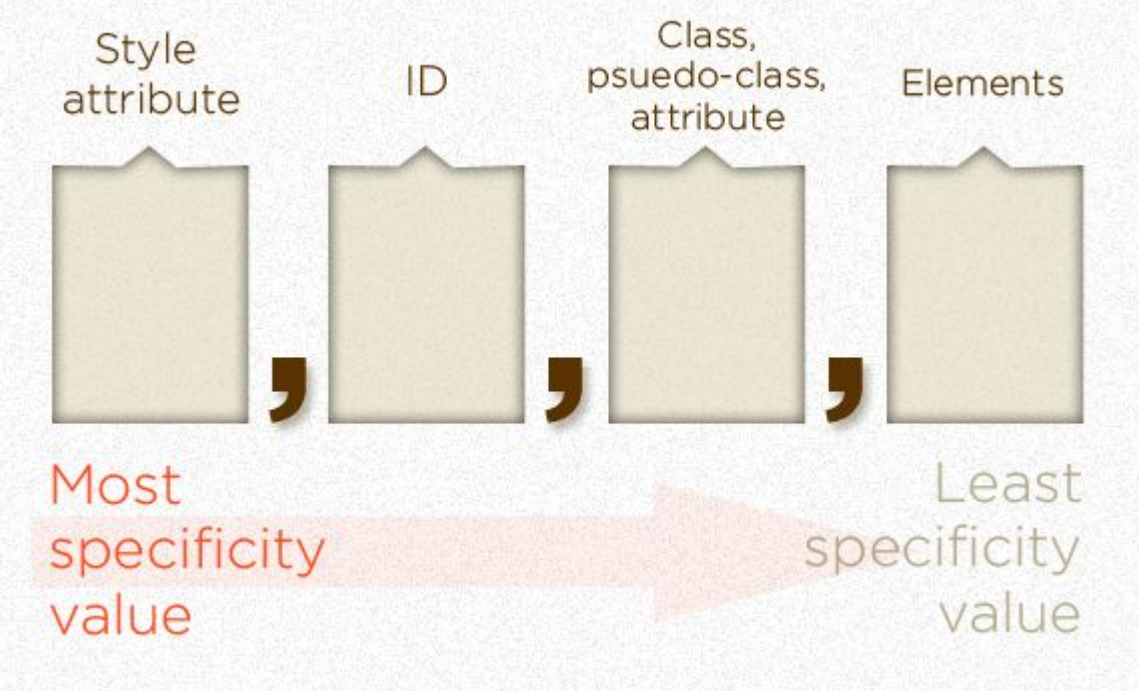
Selector’s weight

For each identifier (hereinafter denote the number through a) there are 100 point, for each class and pseudoclass (b) there are 10 point for each tag selectors and pseudoselector (c) there is 1 point.

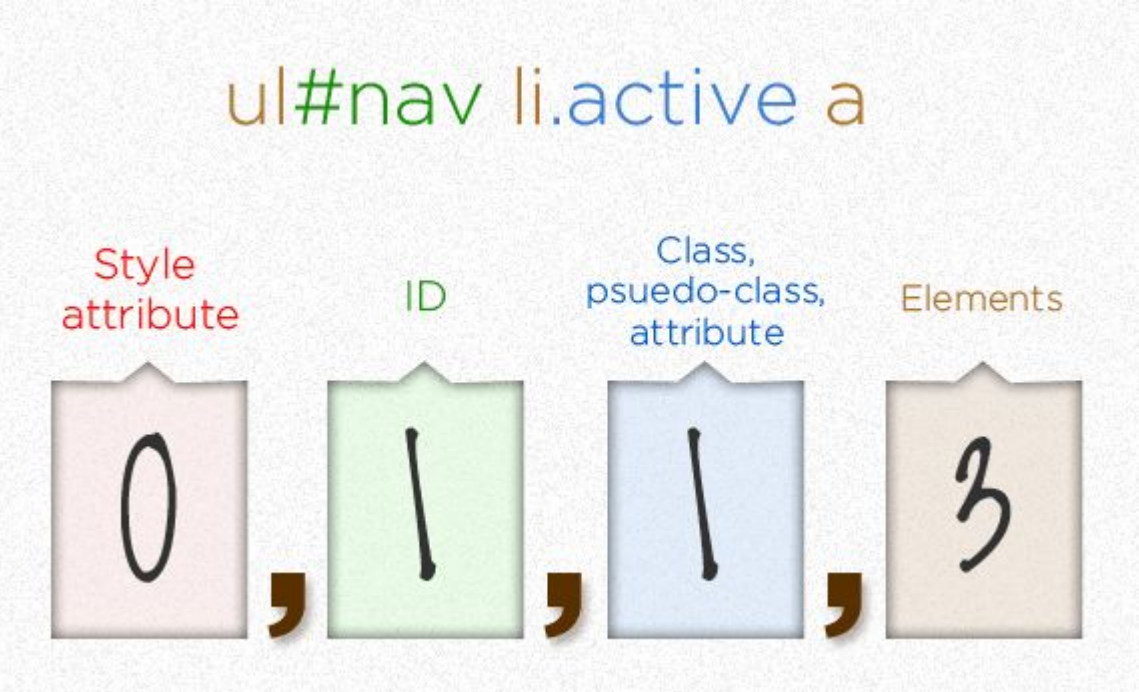
Composing listed values in any particular order, we obtain the weight for the selector:

Selector	Points	Result
* {}	a=0 b=0 c=0	points = 0
li {}	a=0 b=0 c=1	points = 1
li:first-line {}	a=0 b=0 c=2	points = 2
ul li {}	a=0 b=0 c=2	points = 2
ul ol+li {}	a=0 b=0 c=3	points = 3
ul li.red {}	a=0 b=1 c=2	points = 12
li.red.level {}	a=0 b=2 c=1	points = 21
#t34 {}	a=1 b=0 c=0	points = 100
#content #wrap {}	a=2 b=0 c=0	points = 200

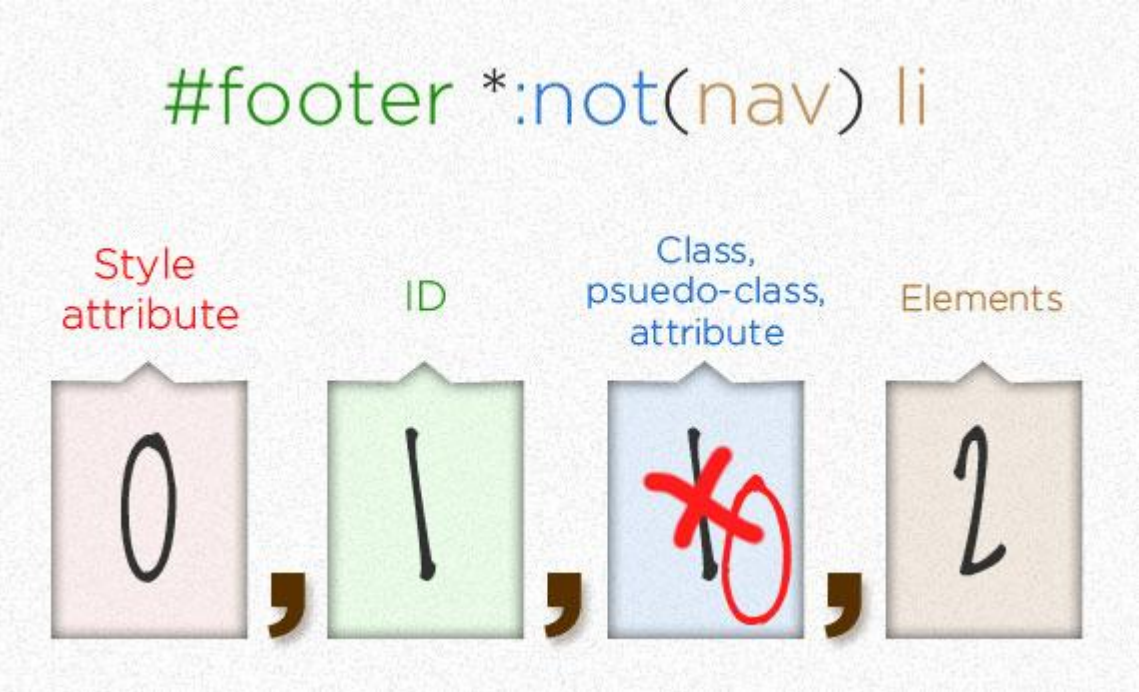
Calculating CSS Specificity Value



Calculating CSS Specificity Value



Calculating CSS Specificity Value



CSS formatting rules

Declaration order Following properties should be grouped together

Positioning (position, top, right)

Box model (display, float, width, height)

Typographic (font, line-height, color)

Visual (background-color, border)

Misc (opacity)

Reset | Normalize

The goal of a reset stylesheet is to reduce browser inconsistencies in things like default line heights, margins and font sizes of headings, and so on

```
* {margin: 0; padding: 0;}
```

- normalize stylesheet - Николас Галлахер и Джонатан Нил.
- reset stylesheet - Эрик А. Мейер.

Reset

```
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
main, menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
     padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
```

http://meyerweb.com/eric/tools/css/reset/

Reset

- HTML5 Reset :: style.css

- https://github.com/murtaugh/HTML5-Reset/blob/master/assets/css/reset.css

Normalize.css

http://necolas.github.io/normalize.css/

Normalize.css makes browsers render all elements more consistently and in line with modern standards. It precisely targets only the styles that need normalizing.

color

hsla

rgba

rgb

hsl

Color units

Color	Color name	Hex rgb	Decimal
	black	#000000	0,0,0
	silver	#C0C0C0	192,192,192
	gray	#808080	128,128,128
	white	#FFFFFF	255,255,255
	maroon	#800000	128,0,0
	red	#FF0000	255,0,0
	purple	#800080	128,0,128
	fuchsia	#FF00FF	255,0,255
	green	#008000	0,128,0
	lime	#00FF00	0,255,0
	olive	#808000	128,128,0
	yellow	#FFFF00	255,255,0
	navy	#000080	0,0,128
	blue	#0000FF	0,0,255
	teal	#008080	0,128,128
	aqua	#00FFFF	0,255,255

RGB color values

```
em { color: #f00 } /* #rgb
*/

em { color: #ff0000 } /*
#rrggbb */

em { color: rgb(255,0,0) }

em { color: rgb(100%, 0%,
0%)    em { color: rgb(255,0,0) } /* integer range 0 -
       255 */
       em { color: rgb(300,0,0) } /* clipped to
       rgb(255,0,0) */
       em { color: rgb(255,-10,0) } /* clipped to
       rgb(255,0,0) */
       em { color: rgb(110%, 0%, 0%) } /* clipped to
       rgb(100%,0%,0%) */
```

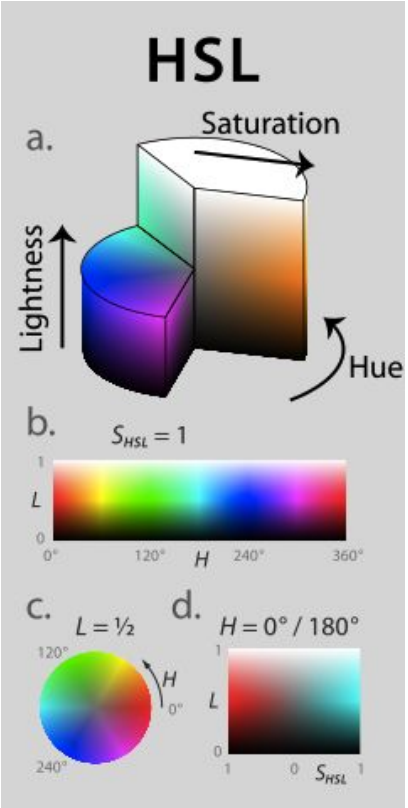
The rgb() function accepts the RGB value as a parameter. The RGB value is provided as a comma-separated list of three values — providing the red, green, and blue hues respectively

rgba

```
rgba(255,0,0,0.
5)
rgba(100%,0%,0%
,0.5)
body {
background:
url('/pix/samples/bg2.png') beige;
color: rgba(0, 0, 0, 1);
font-size: 2em;
}
article {
background-color: rgba(30, 255, 50,
0.5);
border: 5px solid olive;
margin: 20px;
text-align: center;
}
```

The CSS rgba() function can be used to provide a color value with alpha transparency when using CSS. It allows you to specify an RGB color value, as well as an alpha value to determine the color's transparency

HSL



Hue a value ranging from 0 to 360, defines which color you want.

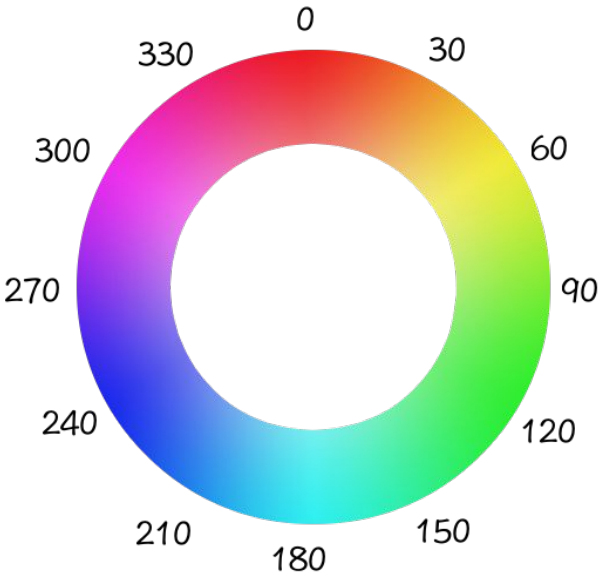
Saturation percentage, ranging from 0% to 100%, defines how much of that color you want.

Lightness percentage, ranging from 0% to 100%, defines how bright you want that color to be

```
body {
  background: hsl(30, 100%, 50%);
  color: hsl(30, 100%, 75%);
  font-size: 1.3em;
}
```

HSLA

CSS hsla() function can be used to add transparency to a color when using the HSL model.



hsla(30, 100%, 50%, 0.5);

Image Sprites

Sprites are two-dimensional images which are made up of combining small images into one larger image at defined X and Y coordinates.

Reducing the number of HTTP requests has the major impact on reducing response time that makes the web page more responsive for the user.

EXAMPLE - A

Number of HTTP request:
10

Total size of the images:
70.7 KB

This example uses seperate images to create navigation list.

EXAMPLE - B

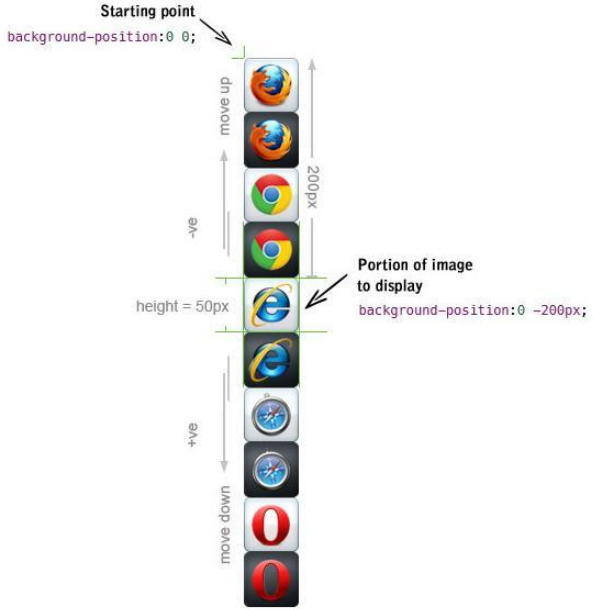
Number of HTTP request:
1

Total size of the images:
32.5 KB

This example uses image sprite to create navigation list.

Making the Image Sprite

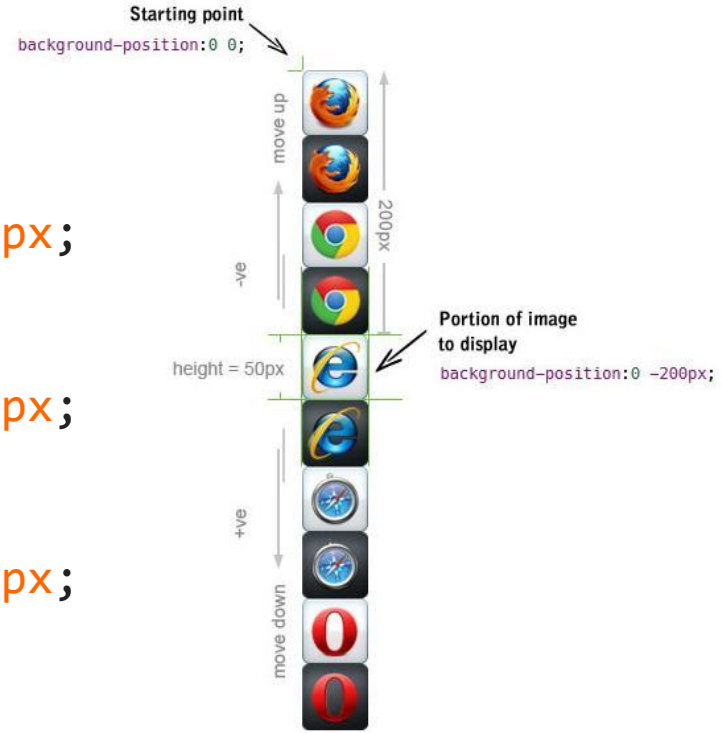
```
<ul class="menu">
  <li class="firefox"><a
href="#">Firefox</a></li>
  <li class="chrome"><a
href="#">Chrome</a></li>
  <li class="ie"><a
href="#">Explorer</a></li>
  <li class="opera"><a
href="#">Opera</a></li>
  <li class="safari"><a
href="#">Safari</a></li>
</ul>
```



https://www.tutorialrepublic.com/codelab.php?topic=css&file=complete-navigation-menu-based-on-image-sprite
https://www.tutorialrepublic.com/css-tutorial/css-sprites.php

Setting Default State of Each Links

```
ul.menu li.firefox a {
background-position: 0 0;
}
ul.menu li.chrome a {
background-position: 0 -100px;
}
ul.menu li.ie a {
background-position: 0 -200px;
}
ul.menu li.safari a {
background-position: 0 -300px;
}
ul.menu li.opera a {
background-position: 0 -400px;
}
```



The reader should be able to read the message of a text easily and comfortably. This depends to a not inconsiderable extent on the size of the type, the length of the lines and the leading (line-height).
—Josef Mueller-Brockmann

Font

font-style: normal | italic

font-variant: normal | small-caps

font-weight: normal | bold

font-size: <length> | <percentage>

line-height: <number> | <length> | <percentage>

font-family: [<family-name> | <generic-family>]#

fonts

WEB SAFE BROWSER FONTS

Sans Serif

- Verdana
- Arial
- Helvetica
- Tahoma
- Trebuchet Ms

Serif

- Times
- Georgia
- Palatino
- Cambria

Monospace

- Courier New
- Lucida Console

Cursive

- Comic Sans Ms

```
body {
        font-family: Arial,
Helvetica, sans-serif;
    }
```

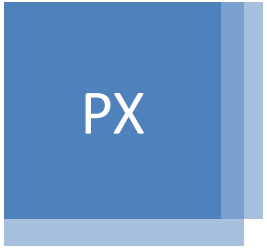
CUSTOM FONTS

FOIT (Flash of Invisible Text)

FOUT (Flash of Unstyled Text)

FOFT (Flash of Faux Text)

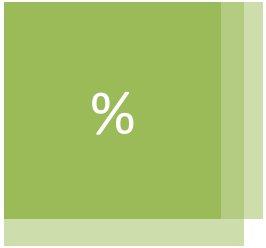
FONT-SIZE UNITS



If you need fine-grained control, renders the letters exactly that number of pixels in height

1em is equal to the current font-size of the element in question.

By default **1em = 16px**.
If you were to go and set a font-size of 20px on your body, then 1em = 20px.



Just like em's the very nature of percentage sizing is that it is relative. It also cascades in the same way.

If a parent has the font-size of 20px and the child has a font-size of 50%, it will be 10px.

Inherited from the root element (html) and do not cascade.

Font-size

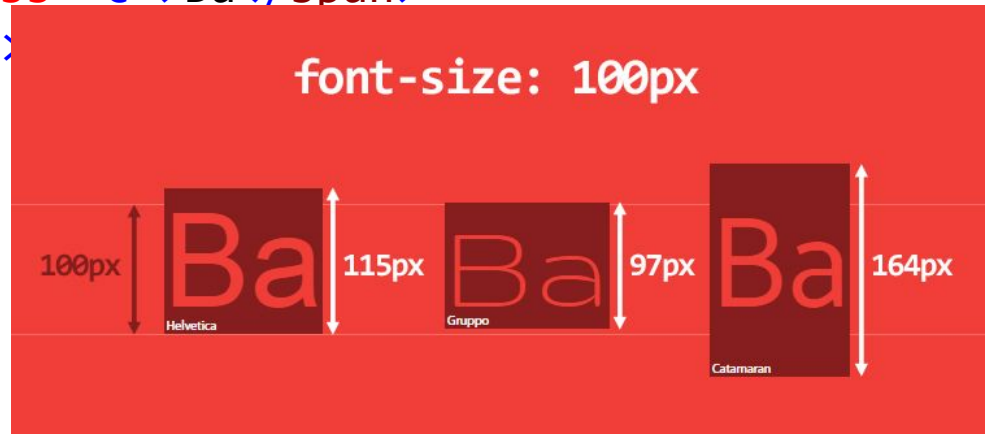
```
div {
  font-size: 14px;
  font-size: 2em; /* ==200% */
}
div p {
  font-size: 2em; /* ==200% */
}
```

```
 div {
   font-size: 20px;
 }
 div p {
   font-size: 50%;
 }
```

```
html{
  font-size: 20px;
}
div {
  font-size: 40px;
}
div p {
  font-size: 1.5rem;
}
```

font-size

```
<p>
  <span
class="a">Ba</span>
  <span
class="b">Ba</span>
  <span
class="c">Ba</span>
</p>
```




```
p {
  font-size: 100px
}
.a {
  font-family:
Helvetica
}
.b {
  font-family: Gruppo
}
.c {
  font-family:
Catamaran
}
```

LINE-HEIGHT

number

The used value is this unitless <number> multiplied by the element's own font size. In most cases, this is the preferred way to set line-height and avoid unexpected results due to inheritance

length

The specified <length> is used in the calculation of the line box height

percentage

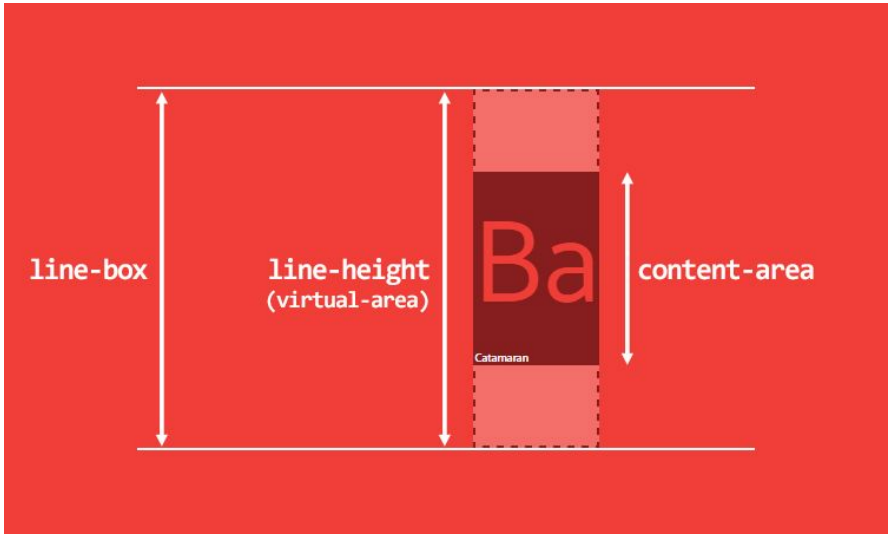
Relative to the font size of the element itself. The computed value is this <percentage> multiplied by the element's computed font size.

normal

Depends on the user agent

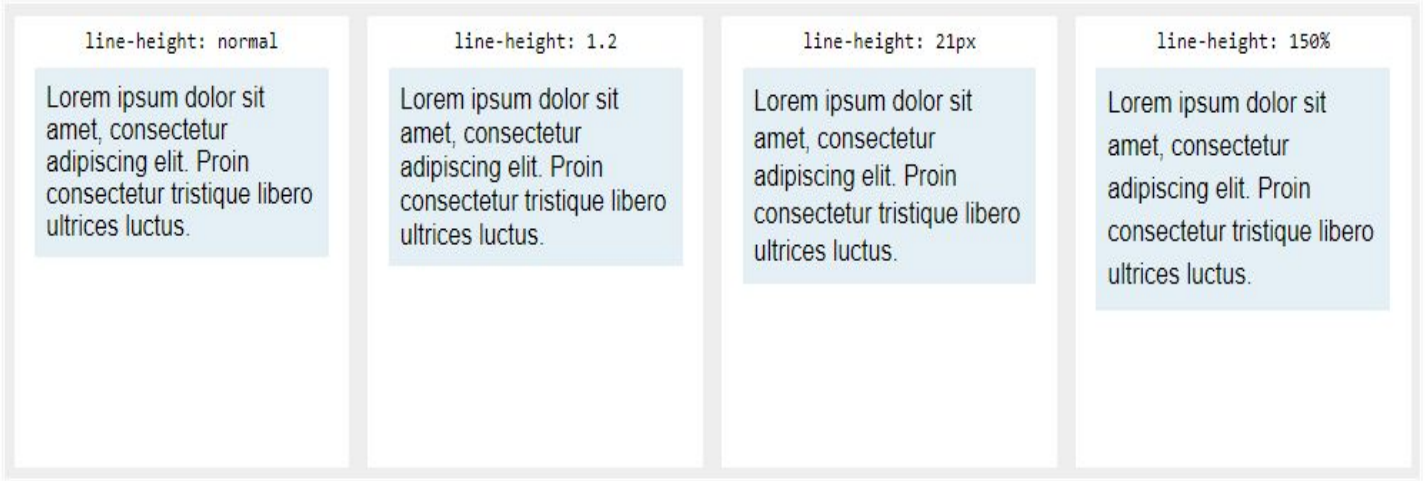
line-height

the content-area height is defined by the font metrics
the virtual-area height is the line-height, and it is the height used to compute the line-box's height

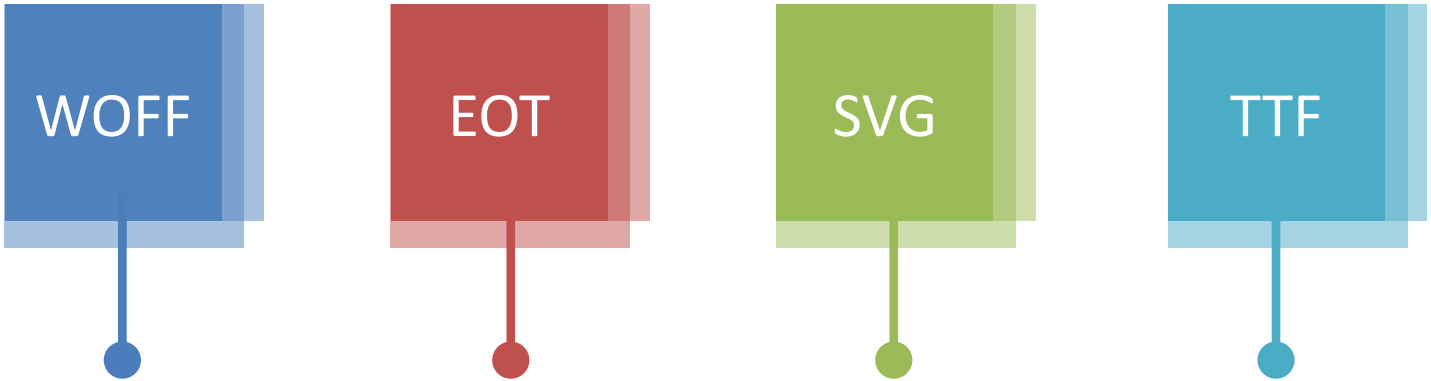


line-height

```
div {
  line-height: 1.2;
  font-size: 10px
} /* number */
div {
  line-height: 1.2em;
  font-size: 18px
} /* length */
div {
  line-height: 150%;
  font-size: 10px
 } /* percentage */
div {
  font: 10px/1.2 Georgia,"Bitstream Charter",serif
} /* font shorthand */
```



FONT FORMATS



Web Open Font Format

.woff files are supported by all modern browsers

Embedded OpenType

.eot files for older Internet Explorer versions (< 8)

Scalable Vector Graphics

.svg files are supported by Safari

TrueType Font

.ttf .otf files partial support in IE and supported by all modern browsers

WOFF – WEB OPEN FONT FORMAT

WOFF

Compressed TrueType/OpenType font that contains additional meta information about the font's source.

Created for use on the web (2009), and developed by Mozilla in conjunction with other organizations, WOFF fonts often load faster than other formats because they use a compressed.

This format can also include metadata and license info within the font file.

This format seems to be the winner and where all browsers are headed.

Supported by the W3C, aims to make it the standard. All browsers will support this in the future .

EOT – EMBEDDED OPENTYPE FONTS

EOT

Type of font that can be derived from a regular font, allowing small files and legal use of high-quality fonts.

This format was created by Microsoft (the original innovators of @font-face) over 15 years ago.

It’s the only format that IE8 and below will recognize when using @font-face.

SVG – SCALABLE VECTOR GRAPHICS

Method of using fonts defined as SVG shapes.

SVG is a vector re-creation of the font, which makes it much lighter in file size, and also makes it ideal for mobile use.

SVG fonts allow SVG to be used as glyphs when displaying text.

SVGZ is a zipped version of SVG.

TTF – TRUETYPE / OTF - OPENTYPE

TTF (True Type Font)
Font file format created by Apple, but used on both Macintosh and Windows platforms; can be resized to any size without losing quality; also looks the same when printed as it does on the screen.

OTF (Open Type)
Font format developed by Adobe and Microsoft; combines aspects of PostScript and TrueType font formats; fully scalable, meaning the font can be resized without losing quality.

BROWSER SUPPORT FOR FONT FORMATS

Font format					
TTF/OTF	9.0*	4.0	3.5	3.1	10.0
WOFF	9.0	5.0	3.6	5.1	11.1
WOFF2	14.0	36.0	39.0	10.0	26.0
SVG	Not supported	Not supported	Not supported	3.2	Not supported
EOT	6.0	Not supported	Not supported	Not supported	Not supported

@font-face DECLARATION

```
@font-face {
      font-family: 'MyWebFont';
      src: url('webfont.eot');                                    /*
IE9 Compat Modes */
      src: url('webfont.eot?#iefix')
format('embedded-opentype'),              /*  IE6-8 */
         url('webfont.woff2') format('woff2'),                 /* Supe
Modern Browsers */
  body {  url('webfont.woff') format('woff'),                 /* Pretty
Modern Browsers */
  font-family:
 'MyWebFont', Fallback,
Android, iOS */
     }    url('webfont.svg#svgFontName') format('svg');
/* Legacy iOS */
```

Useful links

- Mozilla Developer Network (MDN)
- CSS Validation Service
- CSS CURRENT STATUS by W3C
- Can I use - Browser's support checker
- CSS Weekly Newsletter
- Specificity Calculator

Games

https://flukeout.github.io/

https://flexboxfroggy.com/

https://cssgridgarden.com/

https://rupl.github.io/unfold/

FE Online UA Training Course Feedback

I hope that you will find this material useful.

If you find errors or inaccuracies in this material or know how to improve it, please report on to the electronic address:

serhii_shcherbak@epam.com

With the note [FE Online UA Training Course Feedback]

Thank you.

Q&A

UA Frontend Online LAB