# Reference

19 февраля 2018 г. 1:02

**About CSS**

**Font**
- Font
- Text

**Color**

**List & Tables**
- Table
- List
- Link
- Form Elements Styles

**Box properties**
- Box
- Box Size, Margin & Padding
  Overflow
- Border
- Outline
- Resize

**Box Decoration**
- Backgrounds
- Gradient
- Border-Image
- Shadows

**Images**
- Image
- object-fit
- Image sprites
- Icons
- Opacity

**Layout & Media Queries**
- Layout
- Float & Clear Properties
- Multiple Columns
- Flexbox
- CSS Grid
- Inline-block

**Responsive Layout**
- CSS Media Queries
- CSS Responsive

**Selectors**
- CSS Selectors
- CSS Combinators
- Specifity

**Transformation & Animation**
- 2D Transform
- 3D transform
- Transitions
- CSS Animation

**Functions**
- CSS Variables
- CSS Counters
- CSS Functions

**Links**
- CSS Properties Reference
  - *Properties Default Values*
  - *CSS Animatable*
- CSS Entity Reference
- Colors

**Samples**
- Samples
- Horizontal & Vertical Align

# CSS Notes

14 февраля 2018 г. 18:12

## About CSS

What is CSS
- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media

How to include in html document:
- <link rel="stylesheet" type="text/css" href="mystyle.css">
- <style></style>
- style="color:blue;margin-left:30px;" > inline style (highest priority)

Back

---

## Font

*The Web Open Font Format (WOFF) > WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata. The goal is to support font distribution from a server to a client over a network with bandwidth constraints.*

| Generic family | Font family | Description |
|---|---|---|
| Serif | Times New Roman<br>Georgia | Serif fonts have small lines at the ends on some characters |
| Sans-serif | Arial<br>Verdana | "Sans" means without - these fonts do not have the lines at the ends of characters |
| Monospace | Courier New<br>Lucida Console | All monospace characters have the same width |

- font-family > to choose from several fonts if some not available Sample
- @font-face > if font is not installed, define own unique font
- font-size > px, %, em. Normally 16px = 100%, 16px=1em, vw > viewport widh

Note

Back

## *Color*

*ways to set > DarkCyan; #ee3e80; rgb(100,100,90); rgba(0,0,0,0.5); hsl(0,0%,78%)*
*The CSS opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).*

Note Color Names Samples of Defining Color

Back

## Text

- pt - used for printers
- :first-letter :first-line :link :visited :hover :active :focus
- color > Sets the color of text
- direction > Specifies the text direction/writing direction
- letter-spacing > Increases or decreases the space between characters in a text
- line-height > Sets the line height
- text-align > Specifies the horizontal alignment of text
  - center
  - left
  - right
  - justify > each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers)
- text-decoration > Specifies the decoration added to text
  - none > Defines a normal text. This is default
  - underline > Defines a line below the text
  - overline > Defines a line above the text
  - line-through > Defines a line through the text
- text-indent > Specifies the indentation of the first line in a text-block
- text-shadow > Specifies the shadow effect added to text
- text-transform > Controls the capitalization of text
- text-align-last > Specifies how to align the last line of a text
- text-overflow > Specifies how overflowed content that is not displayed should be signaled to the user
  - clip
  - ellipsis
- unicode-bidi > Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
- vertical-align Sets the vertical alignment of an element
- white-space Specifies how white-space inside an element is handled
- word-spacing Increases or decreases the space between words in a text
- word-wrap > allows long words to be able to be broken and wrap onto the next line
  - break-word
- word-break > specifies line breaking rules.
  - keep-all > break at hypens (-)
  - break-all > break in any character

Note

Back

---

## Table

border > Sets all the border properties in one declaration
- border-width border-style border-color|initial|inherit > 5px solid red;

border-collapse > Specifies whether or not table borders should be collapsed
- collapse - Borders are collapsed into a single border when possible (border-spacing and empty-cells properties have no effect)
- separate - Default. Borders are separated; each cell will display its own borders.

border-spacing > Specifies the distance between the borders of adjacent cells
- Horizontal Vertical > 5px 10px

caption-side > Specifies the placement of a table caption
- top
- bottom

empty-cells > Specifies whether or not to display borders and background on empty cells in a table
- show
- hide

table-layout > Sets the layout algorithm to be used for a table
- auto - default, the column width is set by the widest unbreakable content in the cells
- fixed - the horizontal layout only depends on the table's width and the width of the columns, not the contents of the cells

Note:
cursor > specifies cursor view
- pointer
- move
- help
- url("") - link to cursor picture

tr:nth-child(even) {...} > other style for even lines

Back

## List

**List** *display items in unordered (the list items are marked with bullet) or ordered list ( the list items are marked with numbers or letters)*

**list-style-type** > property specifies the type of list item marker
- order list
  - decimal
  - lower-alpha (a, b , c)
  - lower-roman (i, ii ,iii)
- unordered list
  - disk
  - circle
  - square

**list-style-image** > property specifies an image as the list item marker ()
- url('sqpurple.gif')

**list-style-position** > property specifies the position of the list-item markers (bullet points).
- inside
- outside

Back

## Link

*a:hover MUST come after a:link and a:visited*
*a:active MUST come after a:hover*

Back

## Form Elements Styles

Input field
- width
- border
- margin
- padding
- focus
- background-image
- transition

Text Area
- resize:none

Back

---

## Box

*The CSS width property specifies the width of the element's **content area**. The content area is the portion inside the padding, border, and margin of an element (the box model)*
*To keep the width, no matter the amount of padding, you can use the **box-sizing: border-box** property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease. Box-sizing property allows us to include the padding and border in an element's total width and height*
- element > padding > border > margin
- top, right, bottom, left > specifying border values
- by default sizes to include all content
- could use % or em, if in another element % in relation to that element
- em size of the box based on size of the text in it
- margin: 10px auto 10px auto; to center content use auto
- visibility: hidden, visible > The visibility property allows you to hide boxes from users but It leaves a space where the element have to be
- display: inline, block, none, inline-block > The display property allows you to turn an inline element into a block-level element or vice versa, and can also be used to hide an element from the page.
- overflow: hidden, scroll
- border-color: - could be 4 different, border-image - could be;

## Box Size, Margin & Padding

**Margin**
*The CSS margin properties are used to create space around elements, outside of any defined borders.*
- auto - the browser calculates the margin
- length - specifies a margin in px, pt, cm, etc.
- % - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element
- Negative values are allowed.
- You can set the margin property to **auto** to horizontally center the element within its container.
- Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

**Padding**
*The CSS padding properties are used to generate space around an element's content, inside of any defined borders.*
*Properties similar to margins*

**Height/Width**
- Max/min-height/width > improve handling of small windows, it resizes without adding scrollbars
- When you set the width and height properties of an element with CSS, you just set the width and height of the

element have to be
- display: inline, block, none, inline-block > The display property allows you to turn an inline element into a block-level element or vice versa, and can also be used to hide an element from the page.
- overflow: hidden, scroll
- border-color: - could be 4 different, border-image - could be;


Note

Back

*The CSS padding properties are used to generate space around an element's content, inside of any defined borders.*
*Properties similar to margins*

**Height/Width**
- Max/min-height/width > improve handling of small windows, it resizes without adding scrollbars
- When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element, you must also add padding, borders and margins.

Back

## Overflow

**Overflow** *specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.*
*The overflow property only works for block elements with a specified height.*
*Exists overflow-x and overflow-y*
- visible - Default. The overflow is not clipped. It renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content
- auto - If overflow is clipped, a scrollbar should be added to see the rest of the content

Back

## Border

- Each border could be styled separately (left top etc)
  - border-style: ***top right bottom left***
- border-style > kind of border to display
- border-width > property specifies the width of the four borders (in px, pt, cm, em, etc)
- border-color > color
- border-radius > used to add rounded borders to an element (could be specified for each corner)
  Possible specify eliptic corners (for all) > border-radius: 50px / 15px;

Samples

Back

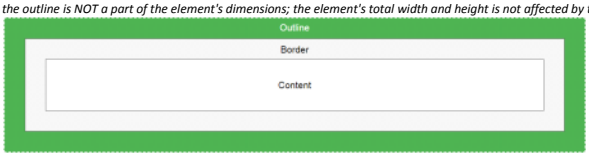
## Resize

*specifies whether or not an element should be resizable by the user. (could be dragged for resizing)*
- resize: horizontal, vertical, both

Back

## Outline

*An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out"*
*Outline differs from borders! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline*




- outline > a shorthand property for setting outline-width, outline-style, and outline-color in one declaration
- outline-color > Sets the color of an outline
- outline-offset > Specifies the space between an outline and the edge or border of an element
- outline-style > Sets the style of an outline
- outline-width > Sets the width of an outline

Back

---

## Backgrounds

*The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer. [few images]*
*Sliding doors pattern > background of 3 images > left, right, repeating middle*
*Full size background image > background: url(img_flower.jpg) no-repeat center fixed; + background-size: cover;*

- background-color
- background-image > Sets the background image for an element
- background-repeat > Sets how a background image will be repeated
- background-attachment > Sets whether a background image is fixed or scrolls with the rest of the page
- background-position > Sets the starting position of a background image
- background-size > allows you to specify the size of background images
  contain - save proportions width/height
  cover - stretch to cover completely content area
- background-origin > specifies where the background image is **positioned** (values the same as for background-clip)
- background-clip > specifies the **painting area** of the background
  border-box - the background image starts from the upper left corner of the border
  padding-box - (default) the background image starts from the upper left corner of the padding edge
  content-box - the background image starts from the upper left corner of the content

Samples

Back

## Gradient

*Linear Gradients (goes down/up/left/right/diagonally)*
*Radial Gradients (defined by their center)*
- background: [repeating-]linear-gradient(direction, color-stop1, color-stop2, ...);
  direction
  - to bottom right
  - to right
  - 40deg

  color stop > color & optional stop position (a percentage between 0% and 100% or a length along the gradient axis)

background: radial-gradient(shape size at position, start-color, ..., last-color);

background: radial-gradient(shape size at position, start-color, ..., last-color);
  shape
  - circle

  size:
  - closest-side
  - farthest-side
  - closest-corner
  - farthest-corner

repeating-radial-gradient(red, yellow 10%, green 15%);

Gradient

Back

## Border-Image

*To use border-mage **border** property have to be set > border: 10px solid transparent;*
- border-image  A shorthand property for setting all the border-image-* properties
- border-image-source  Specifies the path to the image to be used as a border
- border-image-slice  Specifies how to slice the border image (%)
- border-image-width  Specifies the widths of the border image
- border-image-outset  Specifies the amount by which the border image area extends beyond the border box
- border-image-repeat  Specifies whether the border image should be repeated, rounded or stretched

Sample
(stretch|repeat|round|space)

Back

## Shadows

*There is text and box shadows*
*It is possible to add more than one shadow to the text via comma*
*Note: plain border around some text > text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;*

- text-shadow: horizontal shadow, vertical shadow, blur effect, color
- box-shadow > similar to text

Sample

Back

---

## Image

*Normally image sizes (width & height) have to be set > browser reserves space for it*
*Image - inline element*
*:active + :hover + transform:  to simulate button press*
Note
Image Galery
Image Samples > Borders, Responsive images, Center Image, Image Filters, Hover Overlay

Back

## Image sprites

*It is a collection of images put into a single image. Using image sprites will reduce the number of server requests and save bandwidth.*
*Define width + heigth + left & top position*
- background: url('img_navsprites.gif') left_px top_px;

Back

## Icons

**Icons** *scalable vectors that can be customized with CSS (size, color, shadow, etc.)*
*Simplest use icon library, just add class name to any inline HTML element*

Bootstrap icons <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
Google icons  <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
Font Awesome Icons  Sample

Back

## Opacity

*Vary from 0.0 - 1.0 The lower value, the more transparent*
*When using the opacity property to add transparency to the background of an element, all of its child elements inherit the same transparency.*

Back

## object-fit

*used to specify how an <img> or <video> should be resized to fit its container.*

- fill - This is default. The replaced content is sized to fill the element's content box. If necessary, the object will be stretched or squished to fit
- contain - The replaced content is scaled to maintain its aspect ratio while fitting within the element's content box
- cover - The replaced content is sized to maintain its aspect ratio while filling the element's entire content box. The object will be clipped to fit
- none - The replaced content is not resized
- scale-down - The content is sized as if none or contain were specified (would result in a smaller concrete object size)

Sample

Back

---

## Layout

- position:static > default browser behaviour, block elements begin from next line. Static positioned elements are not affected by the top, bottom, left, and right properties.
- position:relative > relative positioning moves an element in relation to where it would have been in normal flow. Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.
- position:absolute > take out of normal flow, The box offset properties specify where the element should appear in relation to its containing element. Containing element have to be ***relative***
- position:fixed > It positions the element in relation to the browser window, stays at the same place even if the page is scrolled
- position:sticky > work with scroll
- z-index > if elements overlap highest value will be shown. If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.
- float left right > take out of normal flow, anything other will flow around element
  - The clear property allows you to say that no element (within the same containing element) should touch the left or righthand sides of a box.
  - E.g. left > The left-hand side of the box should not touch any other elements appearing in the same containing element.
- floats issue, if containing element too small to include all elements - container become invisible
- relative positioning priority e.g. left & right at the same time

.class:after { clear: both; } > Clear floats after the columns

Samples of position  Sample of Layout

Back

## Float & Clear Properties

**Float** *property is used for positioning and layout on web pages*

- left - The element floats to the left of its container
- right- The element floats to the right of its container
- none - The element does not float (will be displayed just where it occurs in the text). This is default
- inherit - The element inherits the float value of its parent

**Clear** *property specifies what elements can float beside the cleared element and on which side.*
*When clearing floats, you should match the clear to the float. If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.*

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side
- right- No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side
- inherit - The element inherits the clear value of its parent

Back

## Multiple Columns

*The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers*

- column-count > Specifies the number of columns an element should be divided into
- column-fill > Specifies how to fill columns
  - balance
  - auto
- column-gap > Specifies the gap between the columns
- column-rule > A shorthand property for setting all the column-rule-* properties
- column-rule-color > Specifies the color of the rule between columns
- column-rule-style > Specifies the style of the rule between columns
  - none Default value. Defines no rule
  - hidden Defines a hidden rule
  - dotted Defines a dotted rule
  - dashed Defines a dashed rule
  - solid Defines a solid rule
  - double Defines a double rule
  - groove Specifies a 3D grooved rule. The effect depends on the width and color values
  - ridge Specifies a 3D ridged rule. The effect depends on the width and color values
  - inset Specifies a 3D inset rule. The effect depends on the width and color values
  - outset Specifies a 3D outset rule. The effect depends on the width and color values
  - initial Sets this property to its default value. Read about initial
  - inherit Inherits this property from its parent element. Read about inherit

- column-rule-width > Specifies the width of the rule between columns
- column-span > Specifies how many columns an element should span across (e.g. header)
- column-width > Specifies a suggested, optimal width for the columns
- columns > A shorthand property for setting column-width and column-count

Back

## Flexbox

*It is possible to achieve similar to float layout result.*
*The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without having to use floats o r positioning*
*A flexbox layout consists of a parent element, with one or more child elements. {.container {display: flex;}}*
*The direct child element(s) of a flexible container automatically becomes flexible items.*

- display > Specifies the type of box used for an HTML element
- flex-direction > Specifies the direction of the flexible items inside a flex container
  - row Default value. The flexible items are displayed horizontally, as a row
  - row-reverse Same as row, but in reverse order
  - column The flexible items are displayed vertically, as a column
  - column-reverse Same as column, but in reverse order
- justify-content > Horizontally aligns the flex items when the items do not use all available space on the main-axis
  - flex-start Default value. Items are positioned at the beginning of the container
  - flex-end Items are positioned at the end of the container
  - center Items are positioned at the center of the container
  - space-between Items are positioned with space between the lines
  - space-around Items are positioned with space before, between, and after the lines
- align-items > Vertically aligns the flex items when the items do not use all available space on the cross-axis
  - stretch Default. Items are stretched to fit the container
  - center Items are positioned at the center of the container
  - flex-start Items are positioned at the beginning of the container
  - flex-end Items are positioned at the end of the container
  - baseline Items are positioned at the baseline of the container
- flex-wrap > Specifies whether the flex items should wrap (change qty of rows etc) or not, if there is not enough room for them on one flex line
  - nowrap Default value. Specifies that the flexible items will not wrap
  - wrap Specifies that the flexible items will wrap if necessary
  - wrap-reverse Specifies that the flexible items will wrap, if necessary, in reverse order
- align-content > Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
  - stretch Default value. Lines stretch to take up the remaining space
  - center Lines are packed toward the center of the flex container
  - flex-start Lines are packed toward the start of the flex container
  - flex-end Lines are packed toward the end of the flex container
  - space-between Lines are evenly distributed in the flex container
  - space-around Lines are evenly distributed in the flex container, with half-size spaces on either end

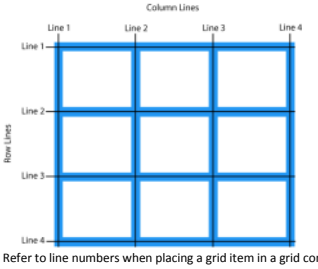
- flex-flow > A shorthand property for flex-direction and flex-wrap
- flex-grow property specifies how much a flex item will grow relative to the rest of the flex items.
  style="flex-grow: 8"
- order > Specifies the order of a flexible item relative to the rest of the flex items inside the same container
  style="order: 3"
- flex-shrink property specifies how much a flex item will shrink relative to the rest of the flex items.
  style="flex-shrink: 0", default is 1
- align-self > Used on flex items. Overrides the container's align-items property
- flex > A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties

Sample  Perfect Centring

Back

## CSS Grid

*The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.*
*An HTML element becomes a grid container by setting the **display** property to **grid** or **inline-grid**.*
*Adjust the gap size by properties > **grid-column-gap**, **grid-row-gap**, **grid-gap***




Refer to line numbers when placing a grid item in a grid container ( grid-column-start: 1;  grid-column-end: 3;)
It is possible to rearrange grid using @media

Try to use properties from flexbox layout Link

- grid-template-columns > property defines the number of columns in your grid layout, and it can define the width of each column. The value is a space-separated-list, where each value defines the length of the respective column.
- grid-template-rows > property defines the height of each row

A grid container contains **grid items**.
By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows.

- grid-column > property defines on which column(s) to place an item (it is a shorthand property for the grid-column-start and the grid-column-end properties).
  grid-column: 1 / 5; > start on line 1 and end on line 5;
  or use span > grid-column: 1 / span 3;
- grid-row > property defines on which row to place an item. (similar to grid-column)
- grid-area > property can be used as a shorthand property for the grid-row-start, grid-column-start, grid-row-end and the grid-column-end properties.
  Can also be used to assign names to grid items.
- grid-template-areas > Used for arranging named grid items > sample

Sample

## Inline-block

*Could be used as well for grid creation. inline-block elements are like inline elements but they can have a width and a height. Blocks just follows one after another like inline elements*
display: inline-block

Back

Back

## CSS Media Queries

*Media queries can be used to check many things, such as:*

- *width and height of the viewport*
- *width and height of the device*
- *orientation (is the tablet/phone in landscape or portrait mode?)*
- *resolution*
- *Complete list of mediafeatures*

*Syntax in css:*

@media not|only **mediatype** and (**mediafeature** and|or|not **mediafeature**) {
  CSS-Code;
}

*Or as separate stylesheet link*

<link rel="stylesheet" media="mediatype and|not|only (expressions)" href="print.css">

**Mediatype**

- all Used for all media type devices
- print Used for printers
- screen Used for computer screens, tablets, smart-phones etc.
- speech Used for screenreaders that "reads" the page out loud

**Meadiafeatures**

- max-width The maximum width of the display area, such as a browser window
- min-width The minimum width of the display area, such as a browser window
- aspect-ratio The ratio between the width and the height of the viewport

Sample Sample2

Back

## CSS Responsive

***Viewport***

*The viewport is the user's visible area of a web page.*
*The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen.*
*<meta name="viewport" content="width=device-width, initial-scale=1.0">*
*A <meta> viewport element gives the browser instructions on how to control the page's dimensions and scaling.*
*width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).*

**Sample of 12 Column Grid**

**Images**

- width: 100%; height: auto; > image will be responsive and scale up and down. The image can be scaled up to be larger than its original size.
- max-width: 100%; height: auto; > the image will scale down if it has to, but never scale up to be larger than its original size
- background-size property is set to "contain", the background image will scale, and try to fit the content area. However, the image will keep its aspect ratio (the proportional relationship between the image's width and height)
- background-size property is set to "100% 100%", the background image will stretch to cover the entire content area
- background-size property is set to "cover", the background image will scale to cover the entire content area. Notice that the "cover" value keeps the aspect ratio, and some part of the background image may be clipped
- The <picture> element works similar to the <video> and <audio> elements. You set up different sources, and the first source that fits the preferences is the one being used Sample

Back

---

## CSS Combinators

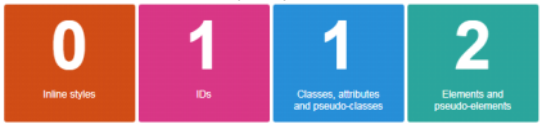
*A combinator is something that explains the relationship between the selectors*
*Read combinated expressions from left to right (as browsers do)*

- descendant selector (**space**) - matches all elements that are descendants of a specified element.
- child selector (**>**) - selects all elements that are the immediate children of a specified element
- adjacent sibling selector (**+**) - selects all elements that are the adjacent (immediately following) siblings (brother/same level) of a specified element
- general sibling selector (**~**) - selects all elements that are siblings (same level below) of a specified element
- Group selector (**,**) - not combinator, any element matches a & b

Samples

Back

## Specify




- The universal selector (*) has no specificity value (0,0,0,0)
- Pseudo-elements (e.g. :first-line) get 0,0,0,1 unlike their pseudo-class brethren 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. It overrides even inline styles from the markup. The only way an !important value can be overridden is with another ! important rule declared later in the CSS and with equal or great specificity value otherwise. You could think of it as adding 1,0,0,0,0 to the specificity value.

Specify Calculator

Back

## CSS Selectors

Selectors:

***** > universal CSS selector is used to select all elements
***element(div)*** > selects all the HTML elements of the same type
***.class*** > selects all HTML elements which have the given CSS class
***#id*** > selects the HTML element which has the given ID
***[attribute]*** > selector is used to select HTML elements by their attributes
***[attr="value"]*** > based on attribute value
***[attr^="begin"]*** > attribute value begins with
***[attr|="begin"]*** > specified attribute starting with the specified value, The value has to be a whole word, either alone, like class="top", or followed by a hyphen( - )
***[attr$="end"]*** > attribute value ends with
***[attr*="value"]*** > attribute value contains
***[attr~="programmer"]*** > attribute value contains word (whitespace separated word)

Pseudo classes:

***a:link, a:visted, a:hover, a:active*** > states of hyperlink. a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective! a:active MUST come after a:hover in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.
***:first-child, :last-child***
***:nth-child()*** > :nth-child(even) / :nth-child(3n)
***:first-of-type, :last-of-type, :nth-of-type()***
***:nth-last-child(), :nth-last-of-type()*** > similar to :nth-child() but counts from the last
***:only-child*** > selects all elements that are the only child of their parent
***:only-of-type*** > selects all elements that are the only child of its kind inside their parent
***:empty*** > selects all HTML elements that are empty, meaning they have no text or child elements inside their body
***:not()*** > selects all of those HTML elements that do not match the CSS selector given as parameter (inside the parentheses)
***:checked, :enabled, :disabled***
***:lang*** > defines different rules for specific languages

Pseudo elements:

***::first-letter, ::first-line >*** can only be applied to block-level elements
***::before, ::after*** > matches a virtual first and last child of the selected HTML element. This is normally used to insert some extra content (text or HTML) before or after that virtual last child using the content CSS property. The content CSS property is used to generate content and insert into the DOM via CSS.
***::selection*** > refers to the selected content when the user selects e.g. a passage of text with the mouse.

Samples Samples 2

Back

---

## CSS Animation

*An animation lets an element gradually change from one style to another.*
*You can change as many CSS properties you want, as many times you want. To use CSS animation, you must first specify some keyframes for the animation. Keyframes hold what styles the element will have at certain times.*

- @keyframes > Specifies the animation code
  - from (0%), to (100%) or list of %
- animation-name > Specifies the name of the @keyframes animation
- animation > A shorthand property for setting all the animation properties
  - name, duration, timing function, delay, iteration count, direction
- animation-delay > Specifies a delay for the start of an animation
- animation-direction > Specifies whether an animation should be played forwards, backwards or in alternate cycles
  - normal - The animation is played as normal (forwards). This is default
  - reverse - The animation is played in reverse direction (backwards)
  - alternate - The animation is played forwards first, then backwards
  - alternate-reverse - The animation is played backwards first, then forwards
- animation-duration > Specifies how long time an animation should take to complete one cycle
- animation-fill-mode > Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
  - none - Default value. Animation will not apply any styles to the element before or after it is executing
  - forwards - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
  - backwards - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
  - both - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions
- animation-iteration-count > Specifies the number of times an animation should be played
  - qty of times
  - infinite
- animation-play-state > Specifies whether the animation is running or paused
- animation-timing-function > Specifies the speed curve of the animation
  - ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
  - linear - Specifies an animation with the same speed from start to end
  - ease-in - Specifies an animation with a slow start
  - ease-out - Specifies an animation with a slow end
  - ease-in-out - Specifies an animation with a slow start and end
  - cubic-bezier(n,n,n,n) - Lets you define your own values in a cubic-bezier function

Sample

## 3D transform

*allows to format your elements using 3D transformations*

- transform > Applies a 2D or 3D transformation to an element
- transform-origin > Allows you to change the position on transformed elements
- transform-style > Specifies how nested elements are rendered in 3D space
  - flat|preserve-3d
- perspective > Specifies the perspective on how 3D elements are viewed. The perspective property defines how many pixels a 3D element is placed from the view. When defining the perspective property for an element, it is the CHILD elements that get the perspective view, NOT the element itself. (length)
- perspective-origin > Specifies the bottom position of 3D elements
- backface-visibility > Defines whether or not an element should be visible when not facing the screen

***Transform Methods***

- translate3d(x,y,z) > Defines a 3D translation
- translateX(x) > Defines a 3D translation, using only the value for the X-axis
- translateY(y) > Defines a 3D translation, using only the value for the Y-axis
- translateZ(z) > Defines a 3D translation, using only the value for the Z-axis
- scale3d(x,y,z) > Defines a 3D scale transformation
- scaleX(x) > Defines a 3D scale transformation by giving a value for the X-axis
- scaleY(y) > Defines a 3D scale transformation by giving a value for the Y-axis
- scaleZ(z) > Defines a 3D scale transformation by giving a value for the Z-axis
- rotate3d(x,y,z,angle) > Defines a 3D rotation
- rotateX(angle) > Defines a 3D rotation along the X-axis
- rotateY(angle) > Defines a 3D rotation along the Y-axis
- rotateZ(angle) > Defines a 3D rotation along the Z-axis
- perspective(n) > Defines a perspective view for a 3D transformed element
- matrix3d(n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n) > Defines a 3D transformation, using a 4x4 matrix of 16 values

Back

Back

## 2D Transform

*CSS transforms allow you to translate, rotate, scale, and skew elements > change shape, size and position*

- **translate()** method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis) > transform: translate(50px, 100px);
- **rotate()** > method rotates an element clockwise or counter-clockwise according to a given degree > transform: rotate(20deg);
- **scale()**, scaleX(), scaleY() method increases or decreases the size of an element (according to the parameters given for the width and height) > transform: scale(2, 3);
- **skew()**, skewX(), skewY() > method skews an element along the X-axis, Y-axis by the given angle. transform: skew(20deg, 10deg);
- **matrix()** > method combines all the 2D transform methods into one > matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

Back

## Transitions

*CSS transitions allows you to change property values smoothly (from one value to another), over a given duration.*
*To create a transition effect, you must specify two things: the CSS property you want to add an effect to, the duration of the effect. The transition effect will start when the specified CSS property (width) changes value.*
*Change several properties > transition: width 2s, height 4s;*

- 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 > property specifies the speed curve of the transition effect
  - ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
  - linear - specifies a transition effect with the same speed from start to end
  - ease-in - specifies a transition effect with a slow start
  - ease-out - specifies a transition effect with a slow end
  - ease-in-out - specifies a transition effect with a slow start and end
  - cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function
- transition-delay > property specifies a delay (in seconds) for the transition effect.
- Transition > a shorthand property for setting the four transition properties into a single property

Back

---

## CSS Variables

- *The var() function can be used to insert the value of a custom property.*
- *Variables in CSS should be declared within a CSS selector that defines its scope. For a global scope you can use either the :root or the body selector.*
- *The variable name must begin with two dashes (--) and is case sensitive!*
- *var(custom-name, value)*

Sample

Back

## CSS Counters

*CSS counters are "variables" maintained by CSS whose values can be incremented by CSS rules (to track how many times they are used). Counters let you adjust the appearance of content based on its placement in the document.*

- counter-reset - Creates or resets a counter
- counter-increment - Increments a counter value
- content - Inserts generated content (used with the :before and :after pseudo-elements).
- counter() or counters() function - Adds the value of a counter to an element

Sample

Back

## CSS Functions

- attr() Returns the value of an attribute of the selected element
  a:after {content: " (" attr(href) ")";}
- calc() Allows you to perform calculations to determine CSS property values. The following operators can be used: + - * /
  width: calc(100% - 100px);
- cubic-bezier() Defines a Cubic Bezier curve
- hsl() Defines colors using the Hue-Saturation-Lightness model (HSL)
- hsla() Defines colors using the Hue-Saturation-Lightness-Alpha model (HSLA)
- linear-gradient() Sets a linear gradient as the background image. Define at least two colors (top to bottom)
- radial-gradient() Sets a radial gradient as the background image. Define at least two colors (center to edges)
- repeating-linear-gradient() Repeats a linear gradient
- repeating-radial-gradient() Repeats a radial gradient
- rgb() Defines colors using the Red-Green-Blue model (RGB)
- rgba() Defines colors using the Red-Green-Blue-Alpha model (RGBA)
- var() Inserts the value of a custom property

Samples

Back

---

## Samples

Navigation Bar Sample

Dropdown Sample

Image Galery

Image Samples > Borders, Responsive images, Center Image, Image Filters, Hover Overlay,

Tooltips

Button Styling

Paggination (<<1 2 3 4>>)

Flexbox Layout

Responsive Image Gallery Using Flexbox

Responsive Website using Flexbox

Back

## Horizontal & Vertical Align

Horizontally center

- text > text-align: center
- element > assign width + margin auto
- Image > display: block + margin left & right: auto

Vertically center

- one line of text > use line-height = height of container
- element > containing element no height + paddings
- element > container position:relative + element position absolute + top & left 50% + transform:translate(-50%, -50%)
- element > display:flex + align-items:center

Back

# Chapter 6 (Missing Manual)

8 февраля 2018 г. 1:23

**Vendor prefixes**
Some features which is under development uses with browser specific prefix > -moz
-moz > Firefox
-webkit > Chrome, Safari, Opera
-ms > IE
-o > old version of Opera

**Transperancy**
Use rgba() or opacity, prefarable opacity can make transparent image, more elements, used in transition

**Background**
multiple background support (few images)
Sliding doors pattern > background of 3 images > left, right, repeating middle

Shadows text and box

**Transition effects**

- Possible to define parameter change over the time (or use all)
- It could be used with transperancy, shadow, gradient, transforms
- Exist timing function of transition effect (start slow end fast etc)

**Transforms**

- It lets move > translate(x,y), scale(x,y), skew(angle), rotate(angle) element
- Transform-origin > shift the starting point around which transform applies
- Transform not rebuild web page content, so it could overlap but it works faster

@font-face

- Use to add not standard font.
- If needed just check google fonts it is enough <link href="http://fonts.googleapis.com/css?family=Metrophobic" rel="stylesheet"> & use it in font-family

column-count > it is possible to put text into multiple columns
Set text size related to ***em*** so wty of worlds will not depend on font

# Samples

8 февраля 2018 г. 1:31

**Multiple background**

```
.decoratedBox {
  margin: 50px;
  padding: 20px;
  background-image: url('top-left.png'), url('bottom-right.png');
  background-position: left top, right bottom;
  background-repeat: no-repeat, no-repeat;
}
```

**Sliding door**

```
.decoratedBox {
  margin: 50px;
  padding: 20px;
  background-image: url('left.png'), url('middle.png'), url('right.png');
  background-position: left top, left top, right bottom;
  background-repeat: no-repeat, repeat-x, no-repeat;
}
```

**Shadows**

```
.shadowedBox {
  border: thin #336699 solid;
  border-radius: 25px;
  box-shadow: 5px 5px 10px gray;
}
```

horizontal & vertical offset, fuzziness, color

**Gradient**

```
.colorBlendBox {
  background: lightblue;
  background: -webkit-linear-gradient(top left, white, lightblue);
  background: linear-gradient(from top left, white, lightblue);
}
```

**Transition**

```
-webkit-transition: background 0.5s, color 0.5s;
transition: background 0.5s, color 0.5s;
```

**Transform**

```
.rotatedElement {
  -ms-transform: scale(1.5) scaleX(10px) skew(10deg);
  -webkit-transform: scale(1.5) scaleX(10px) skew(10deg);
  transform: scale(1.5) scaleX(10px) skew(10deg);
}
```

**New font**

```
1  @font-face {
2    font-family: 'Chantelli1AntiquaRegular';
3    src: url('Chantelli_Antiqua-webfont.eot');
4    src: local('Chantelli Antiqua'),
5      url('Chantelli_Antiqua-webfont.woff') format('woff'),
6      url('Chantelli_Antiqua-webfont.ttf') format('truetype'),
7      url('Chantelli_Antiqua-webfont.svg') format('svg');
8  }
```

1 register font to use it in stylesheets
2 internal name
3 this line is for IE
4 check if it is installed on the users PC, but could security message appears
5-7 other font files with different formats

**Color**

color: #ee3e80; (text color)
background-color: rgb(200,200,200);
opacity: 0.5;

hsl > 0-360 color circle, saturation (amount of grey), lightness

**Text**

Selectors

font-family: Georgia, Times, serif;

@font-face {
font-family: 'ChunkFiveRegular';
src: url('fonts/chunkfive.eot');

font-weight: bold;
font-style: italic
text-transform: uppercase, lowercase, capitalize

text-decoration: underline, overline, line-through, *none (for hyperlinks)*
line-height: 1.4em; letter-spacing; word-spacing
vertical-align
text-shadow
text-indent - move first line
text-align : center, left, right, justify

**Box**

border-style: solid, dotted, dashed, double, groove, ridge etc > like in out etc
box-shadow
Border-radius

**Background Image**

background-image: url("images/pattern.gif")
background-repeat: repeat, repeat-x, repeat-y, no-repeat
background-attachment: fixed, scroll > stay in one position or move during the scroll
background-position: left top, center bottom etc; or % or px

*Set as background picture to whole screen*

```
background: url("../pictures/b1.jpg") no-repeat center center fixed;
background-size: cover;
```

**Borders**

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

border-width: medium; thick

**Icons**

<link href="https://use.fontawesome.com/releases/v5.0.6/css/all.css" rel="stylesheet">

**Animation**

```
/* The animation code */
@keyframes example {
    0%   {background-color: red;}
    25%  {background-color: yellow;}
    50%  {background-color: blue;}
    100% {background-color: green;}
}

/* The element to apply the animation to */
div {
    width: 100px;
    height: 100px;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
}
```

**CSS Variables**

```
:root {
    --main-bg-color: coral;
    --main-txt-color: blue;
    --main-padding: 15px;
}

#div1 {
    background-color: var(--main-bg-color);
    color: var(--main-txt-color);
```

**Perfect Centring**

```
display: flex;
justify-content: center;
align-items: center;
```

**MediaQueries**

```
@media screen and (min-width: 480px) {
    body {
        background-color: lightgreen;
    }
}
```

```
    padding: var(--main-padding);
}

#div2 {
    background-color: var(--main-bg-color);
    color: var(--main-txt-color);
    padding: var(--main-padding);
}
```

**Grid Template Area**

```
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
}
```

**Picture Samples**

```
<picture>
 <source srcset="img_smallflower.jpg" media="(max-width: 400px)">
 <source srcset="img_flowers.jpg">
 <img src="img_flowers.jpg" alt="Flowers">
</picture>
```