CSS

[CSS INTRODUCTION 1](#_Toc85318963)

[ADD CSS TO AN HTML PAGE 2](#_Toc85318964)

[CSS PROPERTIES 2](#_Toc85318965)

[CSS SELECTORS 3](#_Toc85318966)

[PRIMARY SELECTORS 3](#_Toc85318967)

[NESTED SELECTORS 4](#_Toc85318968)

[PSEUDO-CLASSES AND PSEUDO-ELEMENTS 4](#_Toc85318969)

[CSS & TYPOGRAPHY 5](#_Toc85318970)

[CSS WEBSITE LAYOUT 6](#_Toc85318971)

[BOX MODEL 6](#_Toc85318972)

[POSITION & FLOAT 8](#_Toc85318973)

[FLEFBOX 8](#_Toc85318974)

[ABOUT FLEXBOX 8](#_Toc85318975)

[PROPERTIES FOR THE PARENT 8](#_Toc85318976)

[PROPERTIES FOR THE CHILDREN 9](#_Toc85318977)

[CSS LAYOUT EXAMPLES 10](#_Toc85318978)

[HORIZONTAL ALIGNMENT 10](#_Toc85318979)

[VERTICAL ALIGNMENT 12](#_Toc85318980)

[HORIZONTAL AND VERTICAL ALIGNMENT 12](#_Toc85318981)

Academind: JS Data Structures, algorithms

scrum agile

Google: css tricks, css grid tricks; css modules, styled components, etc.

Frontend Masters

Google and Bootstrap Icons

CSS transitions animate search input field

# CSS INTRODUCTION

|  |  |
| --- | --- |
| About CSS | CSS stands for Cascading Style Sheets and defines the visual presentation of HTML elements. Every CSS document is a collection of CSS rules. Every rule contains a selector (an identifier of the HTML element/s to be styled) and one or more declarations separated by semicolon. "Cascading" means that every rule for an HTML element will also be applied to all its child elements. |
| CSS Syntax | selector {  property: value; } |

# USEFUL TOOLS AND PLATFORMS

|  |  |
| --- | --- |
| View CSS in the Browser | CSS Viewer extension for Chrome |
| Online Code Editor for HTML, CSS and JS | codepen.io |
| Make a Screenshot of an Entire Web Page | GoFullPage extension for Chrome |
| Create Design | Figma |
| Pick Exact Color | Just Color Picker |
| Check Browser Support for a Specific Selector/Element | caniuse.com |
| CSS Selector Game | https://flukeout.github.io/ |
| Design Guidelines | Material Design (material.io) |
| Get a Random Image | Lorem Picsum (picsum.photos) |
| Book on Writing Good Code | *Code Complete (2)* by Steve McConnell |
|  |  |
| SoftUni Guidelines | https://zeroheight.com/646c17950/p/608bfb-button/b/32e1a2 |
| Workshop ITCSS & BEM | Design: https://www.figma.com/file/D1hWjDJ2oDVSAXWz3TnmgM/CSS-  Advanced?node-id=1%3A26  Google Material Design Shadows:  https://gist.github.com/serglo/f9f0be9a66fd6755a0bda85f9c64e85f  Sample ITCSS Architecture: https://github.com/aarongarciah/itcss-  sample/tree/master/src/scss |
| SASS | Design: https://www.figma.com/file/D1hWjDJ2oDVSAXWz3TnmgM/CSS-  Advanced?node-id=1%3A26  Easy Compile VS Code Extension:  https://marketplace.visualstudio.com/items?itemName=refgd.easy-compile  When to Use Extend/When to Use Mixin: https://csswizardry.com/2014/11/when-  to-use-extend-when-to-use-a-mixin/ |
| Building Layouts 1 | Design: https://www.figma.com/file/D1hWjDJ2oDVSAXWz3TnmgM/CSS-  Advanced?node-id=2556%3A623 |
| Building Layouts 2 | Design: https://www.figma.com/file/D1hWjDJ2oDVSAXWz3TnmgM/CSS-Advanced |

# ADD CSS TO AN HTML PAGE

|  |  |
| --- | --- |
| External Style Sheet (in <head>) | <link rel="stylesheet" type="text/css" href="site.css"> <!-- recommended --> |
| Internal Style Sheet  (in <head>) | <style>...</style> <!-- when a single document has a unique style --> |
| Inline Style | <p *style*="color:red">Some red text</p> |
| Style Inheritance | When the same element is styled differently, the last style overwrites the previous ones. In increasing order of priority: external styles, internal styles (if declared after the external ones), inline styles, !important. |

# CSS PROPERTIES

|  |  |
| --- | --- |
| Background | background-color: #fff | transparent | navy | rgb(125, 125, 125);  background-image: url('../image.jpg');  background-image: linear-gradient(rgba(255, 255, 255, 0.5), rgba(255, 255,  255, 0.5)), url('../image.jpg'); /\* semi-transparent background image \*/  background-repeat: repeat | repeat-x | repeat-y | no-repeat; /\* for images \*/  background-position: top left | center | 50px 50px | 25% 75%;  background-size: cover | 100% 100%; /\* to cover the whole element or stretch \*/  background-attachment: fixed | scroll; /\* whether the image will remain fixed \*/  opacity: 1 | 0.0; /\* opaque ‒ transparent; inherited by children!, better rgba() \*/  background: #fff url('image.png') no-repeat fixed -47px 0; /\* shorthand \*/ |
| Text | (text-)color: #000 | red | #05ffb0 | rgb(125, 125, 255) | rgba(255, 0, 0, 0.5);  text-align: left | right | center | justify; /\* the horizontal alignment \*/  text-decoration: underline | overline | line-through | none;  text-transform: uppercase | lowercase | capitalize;  text-indent: 50px; /\* the indentation of the first line \*/  line-height: 1.5 | 16px; /\* relative to the font size or in px \*/  letter-spacing: 2px; /\* the space between the characters \*/  word-spacing: 2px; /\* the space between the words \*/  white-space: nowrap; /\* text wrapping disabled \*/  overflow: scroll | hidden; /\* behaviour in case of overflowing \*/  text-overflow: ellipsis; /\* + width, white-space: nowrap, overflow: hidden \*/  overflow-x: auto | scroll; /\* adds a horizontal scroller \*/  overflow-y: scroll; /\* adds a vertical scroller \*/  word-break: normal | break-all; /\* the word break at the end of the line \*/  text-shadow: none | 2px 4px 10px red; /\* horizontal, vertical, blur, color \*/  text-shadow: 0 0 3px #f00, 0 0 5px #00f; /\* red and blue neon glow \*/  direction: rtl; unicode-bidi: bidi-override; /\* text will be displayed right to left \*/  vertical-align: baseline | text-top | text-bottom | sub | super;  /\* vertical alignment of an inline element in a text or a text inside a table \*/ |
| Font | font-family: Verdana; /\* family name \*/  font-family: Verdana, Arial, sans-serif; /\* browser doesn't have Verdana > Arial;  doesn't have Arial either > default non-serif font \*/  font-family: serif | sans-serif | monospace | cursive | fantasy /\* generic name \*/  font-style: italic | normal | oblique; /\* oblique: more slanted than italic \*/  font-size: 16px; /\* default: 18px \*/  font-size: 1rem; /\* relative to the root's font size (body element) \*/  font-size: 1em; /\* relative to the parent element's font size \*/  font-weight: bold | bolder | normal | 100; /\* 100: thin, 900: bolder \*/  font-variant: normal | small-caps; /\* Normal vs. SMALL CAPS \*/  letter-spacing: normal | 2px; /\* the spacing between the characters \*/ |
| Border | border-width: 2px | thick; /\* all 4 sides \*/  border-top-width: 2px; /\* only the top border \*/  border-width: 25px 10px 5px 35px; /\* top, right, bottom, left \*/  border-width: 25px 10px 5px; /\* top, right and left, bottom \*/  border-width: 25px 10px; /\* top and bottom, right and left \*/  border-style: solid | dotted | dashed | double | groove | ridge | inset | outset |  hidden | none;  border-style: dotted dashed solid double; /\* top right bottom left \*/  border-color: #C00;  border-color: red green blue yellow; /\* top right bottom left \*/  border-radius: 10px; /\* rounded corners \*/  border-collapse: collapse; /\* single table border instead of double \*/  border-spacing: 10px; /\* 10px between table border and cell border \*/  border: 1px solid black; /\* shorthand for width, style and color \*/  border-left: 3px solid red; /\* width, style and color for the left border \*/ |
| Margin (outside Border) | margin-top: 10px;  margin-right: 10px;  margin-bottom: 10px;  margin-left: 10px; |
| Padding (between Content and Border) | padding-top: 10px;  padding-right: 10px;  padding-bottom: 10px;  padding-left: 10px; |
| Outline (outside Border, NOT Calculated in Dimensions) | outline-width: 4px | thin | medium | thick; /\* the width of the element's outline \*/  outline-style: dotted| dashed | solid | ...; /\* the style of the element's outline \*/  outline-color: 4px; /\* the color of the element's outline \*/  outline: 4px dotted red; /\* width, style, color \*/  outline-offset: 15px; /\* the space between the outline and the border \*/ |
| Shadow | box-shadow: 1px 1px 6px -2px black;  box-shadow: 0px 10px 6px -6px gray; /\* blurred shadow only on the bottom \*/ |
| Overflow (Only for Block Elements with Height) | overflow: visible | scroll | auto | hidden; /\* behaviour in case of overflowing;  visible: default; auto: adds scrollbars only when needed \*/  overflow-x: auto | scroll; /\* adds a horizontal scroller \*/  overflow-y: scroll; /\* adds a vertical scroller \*/ |
| List Styles | list-style-type: disc | none | circle | square | upper-roman | lower-alpha;  list-style-position: inside | outside; /\* bullet part of the list or not \*/  list-style-image: url('../image');  list-style: square inside url('../image'); /\* backup type, position, image \*/ |
| Images | width: 100%; height: 100%; /\* always specify width and height \*/  object-fit: contain;  object-fit: cover;  object-fit: fill;  object-fit: none;  object-fit: scale-down; |
| Textarea | resize: none; /\* disables the "grabber" in the bottom right corner \*/ |
| Opacity | opacity: 1 | 0.0; /\* 1: opaque, 0: transparent;  all the element's children will have the same opacity, better use rgba() \*/  background-color: rgba(0, 0, 255, 0.5); /\* semi-transparent blue background \*/  background-image: linear-gradient(rgba(255, 255, 255, 0.5), rgba(255, 255,  255, 0.5)), url('../image.jpg'); /\* semi-transparent background image \*/ |
| Cursor | cursor: pointer | move | auto | crosshair | default | e-resize | n-resize | ne-resize  | nw-resize | s-resize | se-resize | sw-resize | w-resize | help | progress | text |  wait; /\* the mouse cursor when covering the element \*/  pointer-events: none; /\* link disabled \*/ |
| Display | display: inline | block | inline-block | flex | none; /\* as per box model \*/  visibility: hidden; /\* like display: none, but the element still takes the space \*/ |

# CSS SELECTORS

## PRIMARY SELECTORS

|  |  |
| --- | --- |
| Select All HTML Elements | \* { /\* applies to all elements \*/  background-color: blue;  } |
| Select by Tag | h1 { /\* applies to all <h1> elements \*/  background-color: blue;  text-align: center;  } |
| Select by ID | #top { /\* applies to the element with id "top" \*/  background-color: blue; } |
| Select by Tag and ID | nav#top { /\* applies to nav element with id "top" \*/  background-color: blue; } |
| Select by Class | .code { /\* applies to all elements with class "code" \*/  font-family: Helvetica, sans-serif; } |
| Select by Multiple Classes | .top.bottom { ... } |
| Select by Tag and Class | nav.top { /\* applies to all nav elements with class "top" \*/  background-color: blue; } |
| Multiple Selectors | h1, h2, p {  background: yellow; } |
| Select by Pseudo-Class | a:hover { /\* all <a> elements when the user's mouse is over them \*/  text-decoration: underline;  } |
| Select by Attribute | [title="menu"] { ... } /\* all elements with title "menu" \*/  [title~="menu"] { ... } /\* "pizza menu" and "egg menu", but not "pizza-menu" \*/  [class|="pizza"] { ... } /\* "pizza" and "pizza-menu", but not "pizza( )menu" \*/  [class\*="pi"] { ... } /\* "pizza", "menu-pizza", "pie", "hippie" \*/ |

## COMBINATORS (NESTED SELECTORS)

|  |  |
| --- | --- |
| Descendant | article p { ... } |
| Direct Child | div > p { ... } |
| Adjacent Sibling | div + p { ... } /\* the first <p> element that comes directly after a <div> \*/ |
| General Next Sibling | div ~ p { ... } /\* all <p> elements that come after a <div> \*/ |

## PSEUDO-CLASSES AND PSEUDO-ELEMENTS

|  |  |
| --- | --- |
| General Pseudo-Classes | p:first-child { ... } /\* only the first <p> \*/  p i:last-child { ... } /\* the last <i> in all <p> elements \*/  p:first-of-type { ... } /\* all <p> elements that are the first <p> of their parent \*/  p:nth-of-type(3) { ... } /\* all <p> that are the third <p> of their parent \*/  p:last-of-type { ... } /\* all <p> elements that are the last <p> of their parent \*/  p:only-of-type { ... } /\* all <p> elements that are the only <p> of their parent \*/  tr:nth-child(2) { ... } /\* the second <tr> element \*/  tr:nth-child(2n) { ... } /\* every second <tr> element starting from the 2nd \*/  tr:nth-child(2n + 1) { ... } /\* every second <tr> starting from the 1st \*/  tr:nth-last-child(2) { ... } /\* the second <tr> element counting from the end \*/  tr:nth-child(odd) { ... } /\* all odd <tr> elements (1st, 3rd, 5th, etc.) \*/  tr:only-child { ... } /\* all <tr> elements that are the only child of their parent \*/  q:lang(no) { quotes: "~" "~" } /\* quotation marks for NO language: ~~ \*/  p:empty { ... } /\* all <p> elements that have no children \*/  :not(p) { ... } /\* all elements that are not <p> \*/  :root { ... } /\* the document's root element \*/  :target { ... } /\* the current active element (when the user clicked on a URL containing this element's name as anchor) \*/ |
| Pseudo-Classes Used for Hyperlinks | a:link { ... } /\* unvisited <a> elements \*/  a:visited { ... } /\* visited <a> elements \*/  a:hover { ... } /\* <a> on mouse over; must come after :link and :visited \*/  a:active { ... } /\* selected <a> elements; after :hover \*/ |
| Pseudo-Classes Used for Input Elements | input:focus { ... } /\* the <input> that has focus \*/  input:checked { ... } /\* all checked <input> elements \*/  input:disabled { ... } /\* all <input> elements that are disabled \*/  input:enabled { ... } /\* all <input> elements that are enabled \*/  input:in-range { ... } /\* all <input> elements with value within range \*/  input:out-of-range { ... } /\* all <input> elements with value out of range \*/  input:valid { ... } /\* all <input> elements with a valid value \*/  input:invalid { ... } /\* all <input> elements with an invalid value \*/  input:required { ... } /\* all <input> elements with "required" attribute \*/  input:optional { ... } /\* all <input> elements with no "required" attribute \*/  input:read-only { ... } /\* all <input> elements with "readonly" attribute \*/  input:read-write { ... } /\* all <input> elements with no "readonly" attribute \*/ |
| Use Several Pseudo-Classes at Once | tbody tr:nth-child(odd):not(:hover) td { ... } /\* all <td>'s in a <tr> when the user's mouse is NOT over the <tr> \*/ |
| Pseudo-Elements Used to Insert Something | ::after /\* insert content after an element; always with property "content":  content: url("image.png"); or content: ''; \*/  ::before /\* insert content before an element; always with property "content" \*/ |
| Pseudo-Elements Used to Select Part of an Element | ::first-letter /\* only applied to block-level elements \*/  ::first-line /\* only applied to block-level elements \*/  ::marker /\* selects the markers of list items \*/  ::selection /\* matches the portion of an element that is selected by a user \*/ |

## CSS ARCHITECTURE

### ITCSS LAYERS

|  |  |
| --- | --- |
| About ITCSS | Inverted Triangle CSS is a methodology that helps developers organize their project CSS files in layers (subfolders) in order to better deal with CSS specifics like global namespace, cascade and selectors specifity. |
| Layer Architecture in the Project Folder | Create a folder "styles" in the project folder. It should contain the following subfolders: 01-settings, 02-tools, 03-generic, 04-elements, 05-objects, 06-components, 07-utilities. |
| Import Stylesheets into HTML | When using pure CSS, import all CSS files (in the given order) into the HTML page, each in its own <link> tag, or import all CSS files into a styles/style.css file using @import url('./...') and then import only this file in the HTML. |
| Layers Settings & Tools | Used with preprocessors, no actual CSS code generated. |
| Generic Layer | 03-generic/generic-normalize.css: /\* tag selectors \*/  github.com/sindresorhus/modern-normalize/modern-normalize.css  /\* resets browser settings for HTML elements (styles, box-sizing definition, etc.) \*/  03-generic/generic-fonts.css:  @import url('https://fonts.googleapis.com/css2?family=Lato&display=swap');  03-generic/generic-scrollbar.css:  ::-webkit-scrollbar { width: 8px; }  ::-webkit-scrollbar-track { box-shadow: inset 0 0 5px red; border-radius: 8px; }  ::-webkit-scrollbar-thumb { background-color: red; border-radius: 8px; }  03-generic/generic-selection.css:  ::selection { color: white; background-color: tomato; } |
| Elements Layer | 04-elements/elements-body.css: /\* tag selectors \*/  body { min-height: 100vh; display: flex; flex-flow: column wrap; }  04-elements/elements-heading.css:  h1, h2, h3, h4, h5 { font-family: Georgia, serif; }  h1 { font-size: 36px; line-height: 43px; }  04-elements/elements-links.css:  a { color: black; text-decoration: none; }  a:hover { text-decoration: underline; } |
| Objects Layer | 05-objects/objects-containers.css: /\* class selectors for abstract objects \*/  .o-centered-container { display: flex; flex-flow: column wrap; }  .o-row-container { display: flex; flex-flow: row wrap; } |
| Components Layer | 06-components/components-site-header.css: /\* class (id) selectors \*/  .c-site-header { border: 2px solid gray; border-radius: 8px; padding: 10px; }  06-components/components-card.css:  .c-card { border: 2px solid gray; border-radius: 8px; padding: 10px; }  .c-card p { margin: 0; }  .c-card h1 { margin-top: 0; }  06-components/components-thumbnail.css:  .c-thumbnail { width: 40px; height: 40px; }  .c-thumbnail--circle { border-radius: 50%; }  .c-thumbnail--big { width: 100px; height: 100px; } |
| Utilities Layer | 07-utilities/utilities-hidden.css: /\* class (id) selectors \*/  .u-visually-hidden { /\* hidden, but read by reading software \*/  position: absolute;  padding: 0 **!important**;  border: 0 **!important**;  width: 1px **!important**;  height: 1px **!important**;  overflow: hidden;  } |

### BEM NAMING CONVENTION

|  |  |
| --- | --- |
| About BEM | Block, Element, Modifier is a popular naming convention for classes in HTML and CSS. It helps developers experience less problems from cascading and also be able to transfer blocks (components) from finished projects to new ones.  It can be perfectly combined with the ITCSS architecture, playing a key role in the sixth layer, components. |
| Parent Element | .btn |
| Child Element | .btn\_\_price |
| Modifier | .btn--big |
| Example | <a *class*="btn btn--big btn--orange" *href*="#"></a> |

### OBJECT ORIENTED CSS

|  |  |  |
| --- | --- | --- |
| Purpose and Principles | The purpose of OOCSS is to encourage code reuse and, ultimately, faster and more efficient stylesheets that are easier to add and to maintain. It is based on two main principles: separation of structure from skin and separation of containers and content. | |
| Separation of Structure from Skin | It is the abstraction of repeated visual features ("skins") into class-based modules, so that they become reusable. | |
| CSS | OOCSS |
| #button {  width: 200px;  height: 50px;  border: solid 1px #ccc;  background-color: lightgray;  }  #box {  width: 400px;  overflow: hidden;  border: solid 1px #ccc;  background-color: lightgray;  } | .button {  width: 200px;  height: 50px;  }  .box {  width: 400px;  overflow: hidden;  }  .skin {  border: solid 1px #ccc;  background-color: lightgray;  } |
| Separation of Containers and Content |  | |

# CSS COLORS

|  |  |
| --- | --- |
| Parts of an Element We Can Set the Color for | background-color: blue; /\* background color will be blue \*/  color: red; /\* text color will be red \*/  border-color: yellow; /\* border color will be yellow \*/ |
| Color Names | red, tomato, lightblue, mediumseagreen /\* 140 standard color names \*/ |
| RGB Values: Red, Green, Blue | rgb(255, 255, 255) /\* white \*/  rgb(0, 0, 0) /\* black \*/  rgb(238, 238, 238) /\* light gray \*/  rgb(90, 90, 90) /\* dark gray \*/  rgb(255, 0, 0) /\* red \*/  rgb(0, 255, 0) /\* green \*/  rgb(0, 0, 255) /\* blue \*/  rgb(255, 0, 255) /\* magenta \*/  rgb(255, 99, 71) /\* tomato \*/ |
| HEX Values: Hexadecimal #RRGGBB | #ffffff /\* white; shorthand: #fff \*/  #000000 /\* black; shorthand: #000 \*/  #616161 /\* dark gray; no shorthand, different values for each component \*/  #ff0000 /\* red; shorthand: #f00 \*/ |
| HSL Values: Hue, Saturation, Lightness | hsl(0, 0%, 100%) /\* white; lightness 100% is always white \*/  hsl(0, 0%, 0%) /\* black; lightness 100% is always black \*/  hsl(0, 0%, 50%) /\* gray; saturation 0% is gray, more lightness ‒ lighter \*/  hsl(0, 100%, 50%) /\* red; 100% pure color (no gray) \*/  hsl(120, 100%, 50%) /\* green; 100% pure color (no gray) \*/  hsl(240, 100%, 50%) /\* blue; 100% pure color (no gray) \*/ |
| RGBA Values: Red, Green, Blue, Alpha (Opacity) | rgba(255, 99, 71, 0.5) /\* tomato, but 50% transparent \*/ |
| HSLA Values: Hue, Saturation, Lightness, Alpha (Opacity) | hsla(0, 100%, 50%, 0.5) /\* red, but 50% transparent \*/ |

# CSS & TYPOGRAPHY

## GENERAL TYPOGRAPHY

|  |  |
| --- | --- |
| Specify Fonts | font-family: Verdana, sans-serif;  /\* if the user doesn't have Verdana: web safe font from the sans-serif family \*/  font-family: serif | sans-serif | monospace | cursive | fantasy /\* generic name \*/  font-style: italic | normal | oblique; /\* oblique: more slanted than italic \*/  font-size: 16px; /\* default: 18px \*/  font-size: 1rem; /\* relative to the root's font size (body element) \*/  font-size: 1em; /\* relative to the parent element's font size \*/  font-weight: bold | bolder | normal | 100; /\* 100: thin, 900: bolder \*/  font-variant: normal | small-caps; /\* Normal vs. SMALL CAPS \*/  letter-spacing: normal | 2px; /\* the spacing between the characters \*/  letter-spacing: normal | 2px; /\* the spacing between the characters \*/  font: 2px Arial sans-serif; /\* style, variant, weight, size/line height, family\*/ |
| Web Safe Fonts | sans-serif: Arial, Verdana, Helvetica, Tahoma, 'Trebuchet MS'  serif: 'Times New Roman', Georgia, Garamond  monospace: 'Courier New'  cursive: 'Brush Script MT' |
| Popular Font Pairings | Georgia for headings, Verdana for text  Merriweather for headings, 'Open Sans' for text  'Abril Fatface' for headings, Poppins for text  Spectral for headings, Rubik for text |
| Relative Font Size | body { font-size: 100%; }  h1 { font-size: 2.5em; }  h2 { font-size: 1.875em; }  p { font-size: 0.875em; } |
| Responsive Font Size | font-size: 10vw; |
| Text Format | (text-)color: #000 | red | #05ffb0 | rgb(125, 125, 255) | rgba(255, 0, 0, 0.5);  text-align: left | right | center | justify;  text-decoration: underline | overline | line-through | none;  text-transform: uppercase | lowercase | capitalize;  text-indent: 50px; /\* the indentation of the first line \*/  line-height: 1.5 | 16px; /\* relative to the font size or in px \*/  overflow: scroll | hidden; /\* behaviour in case of overflowing \*/  text-overflow: ellipsis; /\* + width, whitespace: nowrap, overflow: hidden \*/  overflow-x: scroll; /\* adds a horizontal scroller \*/  overflow-y: scroll; /\* adds a vertical scroller \*/  word-break: normal | break-all; /\* the word break at the end of the line \*/  text-shadow: none | 2px 4px 10px red; /\* horizontal, vertical, blur, color \*/ |

## GOOGLE FONTS

|  |  |
| --- | --- |
| About Google Fonts | Google Fonts is a place where we can check and download a font (type something in cyrillic!). |
| Import Google Font in the HTML File | <link *rel*="stylesheet" *href*="https://fonts.googleapis.com/css?family=Sofia"> |
| Import Multiple Fonts in the HTML File | <link *rel*="stylesheet"  *href*="https://fonts.googleapis.com/css?family=Sofia|Trirong|Audiowide"> |
| Import Font Effects in HTML File | <link *rel*="stylesheet"  *href*="https://fonts.googleapis.com/css?family=Sofia&effect=fire">  <p *class*="font-effect-fire">Sofia on fire</p> |
| Import Google Font in the CSS File with @import (for JudgeW) | @import url('https://fonts.googleapis.com/css2?family=Roboto&display=swap');  /\* Google Fonts > Search "Roboto" > Select this style > @import \*/ |
| Import Google Font in the CSS File with @font-face | @font-face {  font-family: "Gret";  src: url("./Gret/Gret-bold.woff") format("woff"); /\* .ttf > format("truetype") \*/  }  /\* Search "Gret" > Select this style > download font into project folder \*/ |

## FONT AWSOME ICONS

|  |  |
| --- | --- |
| Import Font Awsome in the HTML File | <head>  *Font Awsome > Sign In > Profile Pic > Font Awsome CDN > copy <link>*  </head> |
| Import Font Awsome in the CSS File | @import url("*Font Awsome > Sign In > Profile Pic > Font Awsome CDN > get href value from <link>*"); |
| Use Font Awsome Icons | <i *class*="fas fa-star"></i> |

# CSS IMAGES

|  |  |
| --- | --- |
| Image Sprites |  |
|  |  |

# CSS WEBSITE LAYOUT

## BOX MODEL

|  |  |  |
| --- | --- | --- |
| About CSS Box Model | It is a box that wraps around every HTML element and consists of (from outside to inside) margins, borders, padding and the actual content. |  |
| Block Elements | Start on a new line and fill up the full width available: main, header, article, section, fieldset, nav, ul, ol, li, form, h1-h6, p, div, etc. | |
| Inline Elements | Don't start on a new line, margin can be added just on the right and left sides, the top and bottom padding is ignored when positioning the element, width ant height are not applied: a, label, map, span, strong, em, i, img, textarea, input, button, select, etc. Inline elements cannot contain block elements. | |
| Inline-Block Elements | Don't start on a new line, but can have padding and margin added on all four sides. Declaration in CSS code: display: inline-block;. | |
| Width and Height Dimensions (NOT Applied to Inline Elements) | width: auto; /\* adjusts to display the content correctly \*/  width: 240px | 1(r)em;  width: 50%; /\* 50% of the container's width \*/  width: calc(100% - 20px); /\* 100% of the container's width minus 20px \*/  width: 50vw; /\* 50% of the current screen size \*/  width: fit-content;  max-width: none | 200px | 50%; /\* the maximum width of the element \*/  min-width: none | 200px | 50%; /\* the minimum width of the element \*/  height: auto | 240px | 1(r)em | 50% | 50vh;  max-height: none | 200px | 50%; /\* the maximum width of the element \*/  min-height: none | 200px | 50%; /\* the minimum width of the element \*/ | |
| Overflow | overflow: scroll | hidden; /\* container behaviour in case the content doesn't fit \*/  text-overflow: ellipsis; /\* + width, whitespace: nowrap, overflow: hidden \*/  overflow-x: scroll; /\* adds a horizontal scroller \*/  overflow-y: scroll; /\* adds a vertical scroller \*/ | |
| Margins (the Space Wrapping the Border) | margin-top: 10px | 20% | auto; /\* auto centers the element horizontally \*/  margin-right: 20px;  margin-bottom: 30px; /\* collapses with the top margin of the element below \*/  margin-left: 40px;  margin: 10px 20px 30px 40px; /\* shorthand property \*/  margin: 10px 20px 30px; /\* top: 10px, left & right: 20px, bottom: 30px \*/  margin: 10px 20px; /\* top & bottom: 10px, left & right: 20px \*/ | |
| Paddings (the Space between the Content and the Border) | padding-top: 10px;  padding-right: 20px;  padding-bottom: 30px;  padding-left: 40px;  padding: 10px 20px 30px 40px; /\* shorthand property \*/  padding: 10px 20px; /\* top & bottom: 10px, left & right: 20px \*/ | |
| Border | border-width: 2px;  border-style: solid | dotted | dashed | double | groove | ridge | inset | outset |   none;  border-color: #C00;  border-width: 2px solid black; /\* shorthand property \*/  border-radius: 5px; /\* rounded corners; for table + overflow: hidden; \*/  border-top-left-radius: 10px; /\* rounded top left corner \*/ | |
| Outline |  | |
| Set the Width/Height Calculation | box-sizing: content-box; /\* default value, width and height only include content \*/  box-sizing: border-box; /\* size includes content, border, paddings and margins \*/ | |
| Set the Sizing for the Whole Document | html { box-sizing: border-box; }  \*, \*::before, \*::after { box-sizing: inherit; } | |

## POSITION & FLOAT

|  |  |
| --- | --- |
| Make the Rest of a Container Wrap around a Given Child | float: right; /\* floats the element to the right of its container \*/  float: left; /\* floats the element to the left of its container \*/  float: none; /\* default; restricts the element to float \*/  float: initial; /\* the element remains in its default position \*/  float: inherit; /\* the element inherits the property from its parent \*/ |
| Specify on Which Side of an Element Floating Elements are NOT Allowed to Float | clear: none; /\* other elements are allowed to float to the left and to the right \*/  clear: left; /\* other elements are NOT allowed to the left \*/  clear: right; /\* other elements are NOT allowed to the right \*/  clear: both; /\* other elements are NOT allowed to float around \*/ |
| The Clearfix Hack | .parent { overflow: auto; } /\* when a floated element is taller than its parent \*/  .parent::after { content: ''; clear: both; display: table; } /\* same as above \*/ |
| Specify the Type of Positioning Method Used for an Element | position: static; /\* default; element is in its normal position in the layout flow \*/  position: relative; /\* looks like static (in natural flow) but is relative to parent \*/  position: absolute; /\* relative to closest positioned ancestor/root (NOT in flow) \*/  position: fixed; /\* positioned according to the viewport \*/  position: sticky; /\* relative until a given position is met, then fixed \*/ |
| Positioning Properties (Don't Work with Static Position!) | top: 20px; /\* relative: 20 px down from original position;  absolute: 20px down from the parent's (closest positioned element's) top \*/  left: -100px; /\* relative: 100 px left from original position;  absolute: 100px left from the parent's left border \*/  top: 50%; left: 50%; transform: translate(-50%, -50%); /\* center (absolute) \*/  z-index: 1; /\* behind the elements with z-index 2, 3, etc.; position: absolute! \*/ |

## FLEFBOX

### ABOUT FLEXBOX

|  |  |
| --- | --- |
| About Flexbox | It is a method for laying out rows and columns that offers space distribution between items in an interface and powerful alignment capabilities. Flexbox expands items to fill available free space or shrinks them to prevent overflow. |
| Advantages of Flexbox | With flexbox, we can vertically center an element inside its parent, make all the children of a container take up an equal amount of the available width/height, make all the columns in a multiple-column layout adopt the same height even if they contain a different amout of content, etc. |

### PROPERTIES FOR THE PARENT

|  |  |
| --- | --- |
| Create a Flexbox Container | display: flex; /\* the child elements are turned into flexbox items \*/  display: inline-flex; /\* container is displayed inline, its children ‒ flexbox items \*/ |
| Specify Flex Direction | flex-direction: row; /\* default; flexbox items are ordered in the text direction \*/  flex-direction: row-reverse; /\* the opposite way of the text direction \*/  flex-direction: row; /\* the flexbox items are ordered like the text direction \*/  flex-direction: column; /\* the same way as the text direction (cross axis) \*/  flex-direction: column-reverse; /\* the text direction (cross axis) reversed \*/ |
| Specify the Number of Lines the Fexbox Items Will Appear on | flex-wrap: nowrap; /\* default, the flexbox items will remain on a single line \*/  flex-wrap: wrap; /\* will be distributed among multiple lines if needed \*/  flex-wrap: wrap-reverse; /\* multiple lines if needed, next line before previous \*/ |
| Specify Flex Direction and Number of Lines | flex-flow: row wrap; /\* default; shorthand for flex-direction and flex-wrap \*/ |
| Horizontal Alignment of Items within Flexbox Container | justify-content: flex-start; /\* pushed towards the start of the container \*/  justify-content: flex-end; /\* pushed towards the end of the container \*/  justify-content: center; /\* centered along the container's main axis \*/  justify-content: space-between; /\* remaining space distributed between items \*/  justify-content: space-around; /\* remaining space distributed around items \*/ |
| Vertical Alignment of Items within Flexbox Container | align-items: stretch; /\* default; will stretch across the whole cross axis \*/  align-items: flex-start; /\* aligned at the start of the cross axis \*/  align-items: flex-end; /\* aligned at the end of the cross axis \*/  align-items: center; /\* aligned at the center of the cross axis \*/  align-items: baseline; /\* according to the baseline of the content \*/ |
| Vertical Alignment of Lines (when Multiple Lines) | align-content: stretch; /\* each line will stretch to fill the remaining space \*/  align-content: flex-start; /\* only needed space from the start of container \*/  align-content: flex-end; /\* only needed space to the end of container (down) \*/  align-content: center; /\* only needed space at the center of container \*/  align-content: space-between; /\* remaining space appears between the lines \*/  align-content: space-around; /\* remaining space distributed around the lines \*/ |

### PROPERTIES FOR THE CHILDREN

|  |  |
| --- | --- |
| Specify the Order of a Flexbox Item | order: 0; /\* will appear after item with order: -7 and before item with order: 9 \*/ |
| Specify Item Growth When Space Available | flex-grow: 0; /\* item will NOT grow, it will only fill the space it needs \*/  flex-grow: 1; /\* will fill up remaining space if the only item with flex-grow \*/ |
| Specify How Much an Item Should Shrink if not Enough Space | flex-shrink: 0; /\* will NOT shink and will NOT wrap its content \*/  flex-shrink: 1; /\* default; will shink by a factor of 1 and will wrap its content \*/ |
| Speify the Initial Size of a Flexbox Item (Main Axis) | flex-basis: auto; /\* automatically sized based on its content or width/height \*/  flex-basis: 80px; /\* will wrap its content to avoid overflow \*/  flex-basis: 100%; /\* = width: 100% when flex-direction: row \*/ |
| Specify Initial Size and Size Changes | flex: 0 1 auto; /\* default; flex-grow: 0, flex-shrink: 1, flex-basis: auto \*/ |
| Align a Single Item | align-self: auto; /\* will use the value of align-items \*/  align-self: flex-start; /\* aligns the item at the start of the container's cross axis \*/  align-self: flex-end; /\* aligns the item at the end of the container's cross axis \*/  align-self: center; /\* aligns the item at the center of the container's cross axis \*/  align-self: stretch; /\* will stretch across the container's whole cross axis \*/ |

## CSS LAYOUT EXAMPLES

### HORIZONTAL ALIGNMENT

|  |  |  |
| --- | --- | --- |
| Center One or More Inline or Inline-Block Element Horizontally inside Parent | .parent {  text-align: center;  } |  |
| Center One Block Element Horizontally inside Parent | margin: 0 auto; |  |
| Center Several Block Elements Horizontally inside Parent | .parent {  display: flex;  justify-content: center;  } |  |
| Stretch an Inline-Block or Block Element to Take the Whole Width of the Parent | width: 100%; |  |
| Stretch a Flex Item to Take the Whole Width of the Parent | width | flex-basis: 100%;  or  flex-grow: 1;  /\* + align-items: flex-start; to prevent stretching \*/ |  |
| Distribute Space between Inline-Block or Flex Items 50/50 | <div>1</div><div>2</div>  <!-- inline-block: no whitespace/new line between the HTML elements! -->  .first, .second { width: 50%; } |  |
| Distribute Space between Flex Items 50/50 | .parent {  display: flex; }  .first, .second {  flex-grow: 1; } |  |
| Distribute Space Evenly between 3 Inline-Block or Flex Items | <div>1</div><div>2</div><div>...  <!-- inline-block: no whitespace/new line between the HTML elements! -->  .first, .second, .third {  width: 33.333%; } |  |
| Distribute Space Evenly between 3 Block or Flex Items | <div>1</div><div>2</div><div>...  <!-- block: no whitespace/new line between the HTML elements! -->  .first, .second, .third {  width: 33.333%; float: left; } |  |
| Distribute Space Evenly between 3 or More Flex Items | .parent {  display: flex; }  .first, .second, .third {  flex-grow: 1; } |  |
| Distribute Space between Inline-Block or Flex Items 70/30 | <div>1</div><div>2</div>  <!-- inline-block: no whitespace/new line between the HTML elements! -->  .first { width: 70%; }  .second { width: 30%; } |  |
| Distribute Space between Flex Items 70/30 when Enough Space | .parent { display: flex; }  .first { flex-grow: 3; }  .second { flex-grow: 1; } |  |
| Distribute Space between Flex Items 70/30 when NOT Enough Space | .parent { display: flex; }  .first { flex-shrink: 0; }  .second { flex-shrink: 3; } |  |
| Position Inline or Inline-Block Elements with All Available Empty Space in between | .parent { position: relative; }  .first { position: absolute; left: 0; }  .first { position: absolute; right: 0; } |  |
| Position Flex Items with All Available Empty Space in between | justify-content: space-between;  /\* + align-items: flex-start; to prevent stretching \*/ |  |
| Position Inline or Inline-Block Elements with All Available Empty Space Equally Distributed around Them | .parent { position: relative; }  .first { position: absolute; left: 25%;  transform: translate(-50%, 0) }  .second { position: absolute;  right: 25%;  transform: translate(50%, 0) } |  |
| Position Flex Items with All Available Empty Space Distributed around Them | justify-content: space-around;  /\* + align-items: flex-start; to prevent stretching \*/ |  |
|  |  |  |
|  |  |  |

### VERTICAL ALIGNMENT

|  |  |  |
| --- | --- | --- |
| Position Inline-Block Elements Vertically next to Each Other | .first { vertical-align: bottom; }  .second, .third {  vertical-align: middle; }  .fourth { vertical-align: top; } |  |
| Position Flex Items Vertically next to Each Other | .parent { display: flex; }  .first { align-self: flex-end; }  .second { align-self: stretch; }  .third { align-self: center; }  .fourth { align-self: flex-start; } |  |
| Center Any Element Vertically inside Parent | .parent { position: relative; }  .child { position: absolute; top: 50%;  transform: translate(0%, -50%);  margin-top: 0; } |  |
| Center a Flex Item Vertically inside Parent with a Parent Property | .parent {  display: flex;  align-items: center;  } /\* parent must have height! \*/ |  |
| Center a Flex Item Vertically inside Parent with a Child Property | .parent {  display: flex; }  .child {  align-self: center; } |  |
| Stretch Inline-Block and Block Elements Vertically | .first, .second {  height: 100%;  } |  |
| Stretch Flex Elements Vertically | .parent {  display: flex;  /\* align-items: stretch; (default) \*/  } |  |
| Display Rows of Flex Items at the Top of the Parent | .parent {  align-content: flex-start;  } |  |
| Display Rows of Flex Items at the Bottom of the Parent | .parent {  align-content: flex-end;  } |  |
| Display Rows of Flex Items in the Middle of the Parent | .parent {  align-content: center;  } |  |
| Display All Available Vertical Space between the Rows of Flex Items | .parent {  align-content: space-between;  } |  |
| Distribute Evenly the Available Vertical Space between the Rows of Flex Items | .parent {  align-content: space-around;  } |  |
| Stretch the Rows of Flex Items to Fill the Whole Vertical Space | .parent {  align-content: stretch;  } |  |
| columns | column-count: 3;  .item {  break-inside: avoid-column;  } |  |

### HORIZONTAL AND VERTICAL ALIGNMENT

|  |  |  |
| --- | --- | --- |
| Center Any Element Horizontally and Vertically inside Parent | .parent { position: relative; }  .child { position: absolute;  top: 50%; left: 50%;  transform: translate(-50%, -50%);  margin: 0; } |  |
| Center a Flex Item Horizontally and Vertically inside Parent | .parent {  display: flex;  justify-content: center;  align-items: center;  } /\* parent must have height! \*/ |  |
|  |  |  |

### DISPLAYING IMAGES

|  |  |  |
| --- | --- | --- |
| Make a Large Image Scalable | <article><img></article>  img { width: 100%; height: 100%; } |  |
|  |  |  |

### DROPDOWN MENUES

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

# RESPONSIVE WEB DESIGN

|  |  |
| --- | --- |
| About Responsive Web Design | Responsive web design is using HTML and CSS to automatically resize, hide, shrink or enlarge a website to make it look good on all devices. |
| Set the Viewport in HTML | <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">  <!-- gives the browser instructions on how to control the page's dimensions and scaling --> |

## RESPONSIVE WEB DESIGN

|  |  |
| --- | --- |
| About Media Queries | A CSS technique which uses the @media rule to include a block of CSS properties only if a certain condition is true. Good practice is to initially style the page for mobile phones and then add media queries for larger screens (AFTER the original CSS declaration). |
| Use Media Queries | @media only screen and (min-width: 600px) {  p { color: red;} /\* only applies when screen width > 600 px \*/  }  @media print { p { color: red;} } /\* only when printed \*/ |
| Media Types | all | print | screen | speech |
| Media Feature Rules | (min/max-)width | height | orientation | (min/max-)aspect-ratio |
| Logical Operators | and /\* combines multiple media features \*/  not /\* negates a media query \*/  only /\* applies a style only if an entire query matches \*/  , (comma) /\* combines multiple media queries into a single rule \*/ |
| Typical Device Breakpoints | @media only screen and (max-width: 600px) { ... } /\* extra small devices \*/  @media only screen and (min-width: 600px) { ... } /\* small devices \*/  @media only screen and (min-width: 768px) { ... } /\* medium devices \*/  @media only screen and (min-width: 992px) { ... } /\* large devices \*/  @media only screen and (min-width: 1200px) { ... } /\* extra large devices \*/ |
| Media-Dependent @import Rules | @import url("finerprint.css") print;  @import url("landscape.css") screen and (orientation: landscape); |