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)

figma

Academind: JS Data Structures, algorithms

scrum agile

css selector game

flex

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

Frontend Masters

# 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. "Cascading" means that every rule for an HTML element will also be applied to all its child elements. |
| CSS Syntax | selector {  property: value; } |
| View CSS in the Browser | The CSS Viewer extension for Chrome shows the CSS parameters of any element in a web page. |

# 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, inline styles, !important. |

# CSS PROPERTIES

|  |  |
| --- | --- |
| Background | background-color: #fff | transparent | navy | rgb(125, 125, 125);  background-image: url('../image.jpg');  background-repeat: repeat | repeat-x | repeat-y | no-repeat;  background-position: top left; |
| Text | (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 \*/ |
| Font | font-family: Verdana, Arial, sans-serif; /\* family name \*/  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 \*/  letter-spacing: normal | 2px; /\* the spacing between the characters \*/ |
| Border | border-width: 2px;  border-style: solid | dotted | dashed | double;  border-color: #C00; |
| Margin | margin-top: 10px;  margin-right: 10px;  margin-bottom: 10px;  margin-left: 10px; |
| Padding | padding-top: 10px;  padding-right: 10px;  padding-bottom: 10px;  padding-left: 10px; |
| Outline | outline-width: 4px; /\* the width of the element's outline \*/  outline-style: dotted; /\* the style of the element's outline \*/  outline-color: 4px; /\* the color of the element's outline \*/  outline: 4px dotted red; /\* width, style, color \*/ |
| List Styles | list-style-type: none | circle | disc;  list-style-position: inside | outside;  list-style-image: url(../image); |
| Images | object-fit: contain;  object-fit: cover;  object-fit: fill;  object-fit: none;  object-fit: scale-down; |
| Cursor | cursor: pointer | move; /\* the mouse cursor when covering the element \*/ |

# 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 | nav#top { /\* applies to nav element with id "top" \*/  background-color: blue; } |
| Select by Class | nav.top { /\* applies to all nav elements with class "top" \*/  background-color: blue; }  .code { /\* applies to all elements with class "code" \*/  font-family: Helvetica, sans-serif; } |
| Multiple Selectors | h1, h2, p {  background: yellow; } |
| Select by Pseudo-Class | a:hover { /\* applies to all <a> elements when the user's mouse is over them \*/  text-decoration: underline;  } |

## NESTED SELECTORS

|  |  |
| --- | --- |
| Descendant | article p {  padding: 10px; } |
| Adjacent Sibling | p + p {  font-style: italic; } |
| Direct Child | div > p {  font-style: italic; } |
| Multiple Classes | .top.bottom {  font-style: italic; } |
| Attribute Selectors | a[title="menu"] {  text-decoration: none; } |

# 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-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 { ... } /\* all <a> elements when the user's mouse is over them \*/  a:visited { ... } /\* all <a> elements when the user's mouse is over them \*/  a:hover { ... } /\* all <a> elements when the user's mouse is over them \*/  a:active { ... } /\* all <a> elements when the user's mouse is over them \*/ |
| 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" \*/  ::before /\* insert content before an element; always with property "content" \*/ |
| Pseudo-Elements Used to Select Part of an Element | ::first-letter  ::first-line  ::marker  ::selection |

# CSS & 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 \*/  letter-spacing: normal | 2px; /\* the spacing between the characters \*/ |
| 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 | Google Fonts is a place where we can check a font (type somethyng in cyrillic!). Click on "Select this style" to see how to use the current font (use @import for Judge). Download the selected font, place its folder into the project folder and type in the CSS file:  @font-face {  font-family: "Gret";  src: url("./Gret/Gret-bold.woff") format("woff"); /\* .ttf > format("truetype") \*/  } |
| Font Awsome Icons | index.html: <i *class*="fas fa-star"></i>  style.css: @import url("*Font Awsome > Sign In > Profile Pic > Font Awsome CDN > get href value from <link>*"); |

# 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 horizontal space left and right: 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-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 | width: auto; /\* adjusts to display the content correctly \*/  width: 240px | 1(r)em;  width: 50%; /\* 50% of the container's width \*/  width: 50vw; /\* 50% of the current screen size \*/  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;  margin-right: 20px;  margin-bottom: 30px;  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 \*/ | |
| 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; /\* 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 \*/ |
| 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; /\* 20 px down from the parent's (closest positioned element's) top \*/  left: -100px; /\* 100 px left from the parent's (when relative) left border \*/  top: 50%; left: 50%; transform: translate(-50%, -50%); /\* center (absolute) \*/  z-index: 1; /\* behind the elements with z-index 2, 3, etc. \*/ |

## 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 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! \*/ |  |
|  |  |  |

### WORKING WITH IMAGES

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