Cheatography

HTML+CSS Cheat Sheet

by raposinha via cheatography.com/197915/cs/42020/

Elements

<section>	
<a>	
<link rel="" href="">	

Positioning elements

position attribute

position: static	The default, their normal position. (inline, block...)
position: relative	Position relative to itself. It only affects selected element so it will not affect others.
	Can be used alongside 'left', 'right','top','bottom' to move in the opposite position
	Can be used alongside 'z-index','x-index' to move alongside the x or y axis, 0 by default, like 3D maps.
position: absolute	Position relative to its container, so its movement will affect other elements because it stops belonging to the same plane as the others, like a layer. The container's position must be relative.
position: fixed	It always stays the same place, because its position is relative to the viewport.

Floating elements

Block elements take up the whole block, line elements take up the line. We can line elements up with properties like **float**.

float: (left/-right)	The element with this property will float on that direction inside the container, and their siblings will float around it.
clear: (left/right/both)	Stops the element from floating around the element who has the float property on.

Floating elements (cont)

.container::after{ content: ''; display: block, clear: both }	Containers don't detect floating elements, so if other content is removed they will **collapse** and the floating element will overflow.
overflow: auto;	
.class-name::before{ content: url()}	To position elements in the page, like stickers or badges
	If you're going to make its position absolute, make the parent's position relative

FlexBox (Flexible box layout)

A layout method for laying out elements in one dimension, row or column

display: flex;	Activates this layout method. Used in container element.
flex-direction: row;	The default, lines elements out horizontally.
flex-direction: column;	Vertically.
flex-direction: column-reverse;;	
flex-direction: row-reverse;	

Aligning items

row	Align across main axis (horizontal)
justify-content (flex-start(default)/flex-end/center/space-evenly/space-around/-space-between)	main axis
column	Align across cross axis (vertical)
align-items (flex-start(default)/flex-end/-center/)	cross axis

Flex containers want to fit the same amount of children in the same line.

FlexBox (Flexible box layout) (cont)

flex-wrap-(no-wrap(-default)/-wrap)	when there's no more container width available, items make themselves smaller to fit in the same line/items jump to the next line.
align-content ()	
align-self(f-lex-start/fl-ex-end)	Use on the child element, overrides other container porperties directed to child

Grid

Grids are useful for distributing elements in rows and columns at the same time.

Under Inspect, if we click the tag 'grid' next to the element we can visualize the grid's size and distribution

Display: grid	Define a grid in the container
grid-template-columns: 100px 100px;	Define 2 columns
grid-template-rows: 100px 100px 100px;	Define 3 rows
grid-template-rows: repeat(3, 100px);	
grid-template: repeat(3, 100px) / repeat(2, 100px)	Define 2 columns and 3 rows
Center grids and its' content (default is stretch)	justify-items: center; align-items: center; justify-content: center; align-content: center;
Stretch to fit size	(DEFAULT)
grid-template: repeat(3, 100px) / 30% 70%	Define two columns, each occupying x% of the available space

Grid (cont)

grid-temp-late: repeat 100px auto 100px/ 30fr 70fr	Define two columns, each occupying the respective fraction of the available space. The row in the middle will decrease and increase with the screen, top and bottom will stay fixed
Defining gaps	
row-gap	
column-gap	
gap	
Placing items	
grid-row	(n)moves item to n row, (n/x) item starts from n and finishes in x row (see grid tag)
grid-colu-mn(n, n/x)	(n) moves item to n column, (n/x) item starts from n and finishes in x line (see grid tag)
grid-area (n/x/y/z)	The first two n represent the start (row,column), the last two the item numbers (start, end)
grid-templat-e-areas("" "")	Set up the grid to host a grid template, the properties are written like classes: two headers: "header header", The second quotes are used for rows:"row1 row2"
grid-area	Reference the properties mentioned in last one

To auto-fill the fraction for the column with as many inputs as given: *grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));*

Animations: 2d transformations

	The transform property is used for these animations. They can be used on pseudo-properties like :hover or alone.
transform: rotate(-15deg);	
transform: scale(1.5);	
transform: skew(15deg);	
transform: translate(10px, 50px);	Moves an element to a specific position. Better than absolute positioning.
To use more than one transformation:	transform: rotate(30deg) scale(2);

The order in which they're called matters, it will rotate first and scale second.

Animations: 3D transformations

The difference with 2d animations is that not just vertical and horizontal axis are included, but X and Y, which are able to position the element 'closer' or 'further away'.. To use 3d in animations transform() is included. The rotation origin is the center of the element in a X or Y axis point of view, like a matrix.

	transform: perspective(-200px) rotateY(50deg);

The position where its transformed from can be changed from the center to others with transform-origin, and it uses X and Y.

0 0 on transform-origin sets the center on the top left corner.	transform-origin: 0 0;
This transformation starts from the left and the middle	transform-origin: 0 50%;

If many elements are to share an animation, they have to use the same class for transform(). This is easily done setting a container class.

Animations: Transitions

```
//For animations to appear smooth between one step and the next we can use transitions. Properties to use: linear, ease-in (starts slow, continues as expected), ease-out (starts as expected, ends slow), cubic- bez ier (.2 9,.1 -3,.29,.8) (you determine the speed of the transform)
.box-2 {
        width: 100px;
        height: 100px;
        bac kgr ound: red;
        margin: 3rem;
    transi tion: transform 0.5s ease-i n-out 0.3, background 1s;
}
.box:hover {
        tra nsform: rotate (-1 5deg);
        tra nsform: scale( 1.5);
        tra nsform: skew(1 5deg);
        tra nsform: transl ate (10px, 50px);
        tra nsform: rotate (30deg) scale(2);
        bac kgr ound: brown;
}
```

cubic-bezier.com lets you manually pick the kind of bezier curve animation to use with its inputs

The next numeric value input is for animation delay, it takes 0.3 seconds to start after hovering it.

You can use more properties than transform to animate, you can also use background which will shift the background color

Animations pt. II

Animations can be more intricate than what we've seen so far.

@keyframes animation-Name{}	We specify what happens in each keyframe. This property is divided in other small properties.
.. 0% { transform: scale(1);} ...	
.. 50% { transform: scale(5);} ...	
To call this animation, in the element we want using it, we declare:	.box3{ animation-name: animation-Name; animation-duration: 5s;}

Other properties include

Animations pt. II (cont)

animation-delay: 1s;

animation-iteration-count: infinite;	The amount of times you want to repeat this animation. Use 0-9 and infinite for a loop.
animation-timing-function: ease-in;	The timing functions to use, like in transform, you can make it start slow and continue at normal speed (ease-in), etc.
animation-direction: alternate;	Determines if you want to start the animation from start to end, from end to start or to alternate from start to end and from end to start.

Reusable animations

<div class="animation-pop"

animate.style lets you use pre-made animations

Best practices

- Follow a naming convention: kebab case(.kebab-case), camel case (camelCase), pascal case (PascalCase), underscore (under_score).

- Create logical sections on your stylesheet: big projects will need different stylesheets for each logical case which are then combined in a main one. Even in one stylesheet try to differentiate different concerns, such as basic styles, typography, forms, navBar, etc.

- Avoid over-specific selectors: avoid direct children, element names, repeating the same name class over the document (specify, such as 'nav-item').

- Avoid !important.

- Sort CSS properties: to automatically sort CSS properties type >sort in the command line up top.

- Take advantage of style inheritance: to get the same font style in a link and a list, for example, give the font to the parent element.

- Extract repetitive patterns.

Best practices (cont)

- Avoid repetitive values in your code.

* Selection> Add cursor below will show you how to edit multiple lines at the same time.
* To address children of a same class use .mom .kid{}
* To automaticam

Variables or custom properties

:root	This is a pseudo-class selector. We can use it to define custom properties or global variables.
{ --color-primary: red; }	This is a variable set on :root.
.mainText { background: var(--color-primary);}	This is a variable applied inside of a selector.

Object Oriented CSS

- Separate the content from the container

Make a style just for the button class so it applies to all buttons, not just the ones inside .container:

.container .btn {}	.btn{}

- Separate structure from skin

Make classes that purely define the structure or logic (a button having rounded edges, a specific font, no border) and clases for 'skin' (such as a button's color, size...)

BEM: Block Element Modifier

This coding practice/convention sees website's grouped elements as 'blocks' of content that form an 'all'. A block can contain elements or other blocks.

In this convention, classes act as separators, and are named likeso: card, card__header, btn (it's used individually of the card and will be adressed as the card's child), body

The 'modifier' bit refers to bits of website's elements that have the same use but look different, like a subscription list where Premium membership is a different color. In this case, the class naming convention will be: card--premium.

BEM: Block Element Modifier (cont)

'__' is used to differentiate a block from an element and '--' to differentiate a block from a modifier.

Setting up a project

Create and open the project folder with VSC.

Create the file index.html with boilerplate code. (! + Shift)

Make a css folder and create a styles.css file and a normalize.css file (Google the last one, it's for compatibility).

Link both stylesheets in <head>

In the Source Control pannel, click 'Initialize Repository'. Add all new files and click the 'Commit' or 'Okay' button.

Color palettes

Typically composed of primary, seocndary and accent colors.

It's recommended to classify colors in variables	:root { --color-primary: #000; --color-accent: #222}

Typography

Use the designer's photoshop mockup to select font types and sizes.

To decrease download price, select the specific styles you need with care. (Regular 300, Bold 500... etc).

Instead of PX its preferrable that you use REM

font-size: 62,5%; means that default font size is not 16px anymore but 10px, so 1rem will be 10px

Measure the distance between p and h, for rule of assiciation, and adjust depoending on perceived space and actual space. Remember that when two margins meet, they collapse, meaning they become one.

Project - Links

Sprites

SVG files can be edited with CSS.

To combine and use many SVG items, we can use a sprite. A sprite can be generated using https://svgsprit.es/

To reference the svgs inside the sprite, use:	<svg class="icon"> <use xlink:href="../images/sprite.svg#wordpress"></use> </svg>

To style them use CSS as usual.

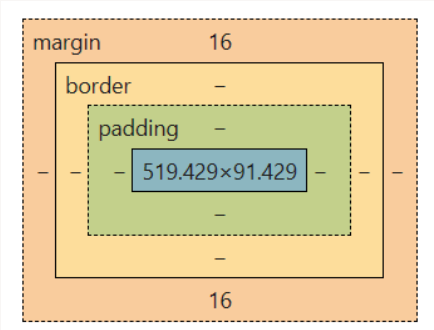
Shadows

box-shadow: x y z w;	x: horizontal, y: vertical; z: blurriness; w: shadow length;

Blocks

Navigation bars

The BOX model



The numbers indicate the amount of space reserved for them. In here, content takes up 519pxx91px and there's a top and bottom margin of 16.

The BOX model

This model refers to an element being put inside an invisible box when the DOM document is rendered.

At the core of the box there's the content area where content is displayed.

Box: <p>	Content area: text blah blah

Outside of the **content area** we have the **padding area** used to add some space outside of the content area.

Next we have the **border area**

On top we have the **margin area** used to create some space between elements, other boxes.

For CSS, the rules are applied with **trouble**: top, right, bottom, left.

p { padding: 10px 20px 10 px 20px;}

10px 20px

10px 20px 10px	10px for top and bottom, 20 for right and left

If two elements are next to each others **their margins collpase**, meaning they're combined and they share the same space.

Sizing elements

width, height properties	they size the content area up or down
padding	it sizes up or down the padding
margin	it sizes up or down the margin
border (style, size color)	can size up the box if size is changed
box-sizing: content-box;	all box pieces add up to the size to the box but margin, which separates elements from others
box-sizing: border-box;	Adds everything up from the border so the total is 100px
	We can use the universal (*) selector to apply the border-box property on all elements. For pseudo-elements: ((*::before, *::after...))

width and height only apply to block level elements, which take up the whole horizontal space. If you add another element after a first one, it will start on the next block of space

Inline elements don't respect width and height

Block elements use display: block; by default, inline display: inline;

display: inline-block	They can use width, height and not start in a new line

Overflow

When an element has a fixed size, content exceedind the designated space might happen. This is overflow.

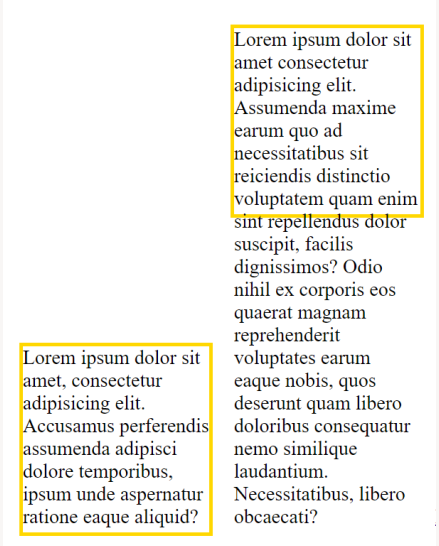
There are CSS properties to control this:

overflow: hidden;	hides the exceeding elements
overflow: scroll;	Gives you the option to scroll down to see the overflow content
overflow: auto;	

The overflow property actually has axis x and y, so you can combine these

overflow: hidden scroll;	Hide the content on x axis and scroll on y

Overflowing content



Measurement units

px	pixel size, absolute: it stays the same size regardless of device or screen size.
%	size relative to the size of the container, it takes up x % of the parent element's size. Browser's default is 100% width and 0% height (extends with content)
vw	size relative to viewport, it takes up the whole horitontal width space, regardless of content
vh	size relative to viewport, it takes up the whole vertical height space, regardless of content
em	size relative to the font size of parent (10em -> 10 times)
rem	size relative to the font size of the root element (16px by default). If we set html's font-size to 65,5% it will be 10px.

Images

There's two kind of images: **raster**, made up by pixels, and **vectors** made up by mathematical vectors. Raster images usually come from cameras or scanners. The more amount of pixels the bigger image file size, if smaller because they have less pixels, the blurrier they look. Vectors are software-made and look sharp at any size.

Images (cont)

To check image compatibility, you can use caniuse.com

Another two types: content images and background images.

background property	It's used to set a background color OR image, to use as image: background: url(/route)
background-repeat:	no repeat / repeat-x (alongside horizontal axis) / repeat-y
background-position	num num (right,down); By default backgrounds start from top left of the conmtainer element, we can move them around with this property
background-size	(right,down) or 100vh - remember that default height of elements is 0 before setting this!
background-attachment	fixed (background will not move even if you scroll and content does move)

To check downloaded images by browser: Inspect, Network, Img. Many images will cause too many requests.

CSS Sprites can help lower the workload of user requests. https://csspritestool.com/ -> We can download, add the new image, copy the css rules and use only one image, likeso: span class="bg-dishes" (class prefix, image name for each). It's useful for small icons, not for all pictures because it will create a huge image file.

Data URIs encode image files. They are protocoled in "data:(...)" form, which goes inside the 'src' tag. It loads faster but is heavier (on desktop).

Clipping creates a path around an image and displays it in different shapes

Images (cont)

clip-path: polygon(x z, 100% 0, etc)	x,z offsets: top-left, top-right, bottom-right, bottom-left
Filters change the look of image elements, can be combined with pseudo-selectors for a clean look	filter: grayscale(70%)/blur(10px)

Supporting high-density screens (high res): Provide two files of the same image, one with x amount of pixels and another with twice the amount. Physical resolution and logical resolution are different, CSS uses logical. To use images with different DPR, we can export the same image into smaller sizes, depending on the sizes we want to use. Generally export at 7. Instead of using 'src' you can use 'scrset' for multiple sources.

srcset="images/meal.jpg 1x, images/meal@2x.jpg 2x"	Support multiple sources of the same image in different DPRs
Resolution switching to fetch one image or the other depending on width	srcset="images/meal.jpg 400w , images/meal@2x.jpg 800w" sizes=" (max-width: 500px) 100vw, (max-width: 500px) 50%"

responsivebreakpoints.com is useful to set different image resolutions for each breakpoint

To use lighter images, we can convert them to webp format. I might have to use 'picture' for better support with type as webp and jpg.

To utilize art direction, for which shows different images on different display sizes. With this, different sources will be picked depending on each query	<picture><source media="-(max-width: 500px)" src=""

Icons

Images (cont)

fontawesome.com gives you free icons to use on your website	<i class="fa-solid fa-leaf fa-rotate-by" style="color: #4b511f; --fa-rotate-angle: 2deg; fa-2x"">

Hiding elements

display: none;	It hides the element as if it was never there.
visibility: hidden;	Allocates space for hidden el

Media queries

Used to create responsive websites because they adapt to the device using it, not the other way around.

On Chrome: View > Developer > Chrome DevTools > Toggle device ToolBar you can check the viewport size and how it looks in different screens

Breakpoint	When after changing sizes the screen looks bad. Use this as a basis, not popular device models.
@media	The type of media it will adress, used in breakpoints
@media screen	For web browsers
@media print	For printers. Useful to set font sizes to pt and cm for sizing.
@media screen and()	Create a condition
@media screen and(min-width: 600px)	If the condition is applied (minimum size of screen is 600px) the rules will apply, otherwise they won't. Classes can be referenced to inside.

Pseudo-class selectors

.box:nth-of-type(x)	Style the x occurrence with this class ('box')

Font types

Serif | Sans-serif | Monospace

Typography

There's 3 fonts, serif, sans-serif and monospace

Styling fonts

font-family		Determines the font used by the element.
Font stacks	font-family: Arial, Helvetica, sans-serif;	Multiple fonts, if the first font is not available, the computer looks for the next in line. The third is a generic font, and will be one of the three: serif, sans-serif or monospace.
font-weight	100-900, bold(700),bolder,lighter,normal(400)	Determines the bolness of the font.
font-style	normal,italic,bold	
font-size	px,em,rem,	Determines the size of the font.
System font stack	Each computer will interpret the font differently depending on OS or version. To make sure the website is readable for the user, this approach is recommended.	font-family: (apple-system)

Sizing fonts

Typography (cont)

Pixels (px) will look different on different OS, browsers ... it's better to use relative units. It's 16px by default. 1 rem is equal to the default font (16px * 1rem = 16px), 62,5% is equal to 10px. Be it px or relative units, always set the font size relative to the html element.

line-height	on body will determine the default height between lines
letter-spacing	will separate letters from each other
word-spacing	

Font formatting

text-align:	(center,end,justify...)
text-indent:	Adds a little space before the paragraph. To avoid adding it to the paragraph subsequent to a heading and add it to the next paragraph, we can use relational selector p + p. (rem,px...)
text-decoration:	(underline,line-through...)
text-transform:	(lowercase,uppercase,capitalize...)
white--space:	(no-wrap)
text-overflow	(ellipsis)
line-clamp:	(n)
column-count:	(n) Separates text into (n) columns
column-gap:	(rem,px...) Makes a gap between prior columns
column-rule:	Creates a visual separator between columns, ex: 3px, dotted, #999
direction:	ltr, ltr (text direction)

Forms: Checkboxes

```
<div>
  <input type="checkbox" name="" id="yes" checked />
  <label for="yes" class="label-inline">Yes</label>
</div>
<div>
  <input type="checkbox" name="" id="no" disabled />
  <label for="no" class="label-inline">No</label>
</div>
```

Checkboxes also have default and disable options

Animations

Imports

@import url();

@import url(animations.css);

Forms

To align fields, wrap label and input into a div

input[type='text'], input[type='email'] {	Adding properties to different input types
input[type='text']:focus, input[type='email']:focus {	Adding properties to different input types at a given point (when the element is being focused on)
	To remove the resize option from the textarea element: resize: none;
	To remove the default ugly outline or border from selected inputs, use outline:none
Attributes	
type	Determine what input type will be used, will have many different input options (text, email, number, password, date)
value	Automatically fills the given input with a set value
placeholder	Automatically fills the given input with a set value and disappears when typing on it

Forms (cont)

readonly value=""	Input can be selected but not modified
disabled	Input can't be selected or modified and won't be sent to the server
maxlength	Input can't exceed this amount
autofocus	

Datalists	These provide input suggestions for autocomplete.
Drop-down lists	They give the user options to choose from in different ways.
Checkboxes	
	input:checkbox -> <input type="checkbox" name="" id="" />
Radio--boxes	Used when we want to select just one choice
	<input type="radio" name="" id="">
Sliders	Allows the user to select from a range of values with JavaScript help
	<input type="range" min="0" max="10" value="5" />
File inputs	
Data validation	We can follow different constraints to make sure users input valid inputs and avoid malware. This can be done through HTML5 alone and JavaScript.

Forms (cont)

	With HTML5 we can use the 'required' attribute to force completion, minlength to force a minimum length for the input, maxlength. Some restriction come with type, like email or date. min and max should be used in numeric fields to avoid corrupt inputs.
Hidden fields	These are used to send data from the form to the server, like IDs.
	Never ever store sensitive values on these.
	<input type="hidden" name="course-id" value="1234" />
Submitting the form	Both buttons and inputs can be created to create a submit option:
	<button type="submit"></button>
	<input type="submit" value="" />
	To actually submit the form to the server, we need aid from server-side technologies like NodeJS, Django, etc.
	To test, we can use the website https://formspree.io/
	Forms need to have an action attribute (where we send data) and a method attribute (how we're sending data)
	<form action="https://formspree.io/f/???" method="POST">
	and inputs need to have a 'name'
	<input type="email" name="email" id="e-mail" />

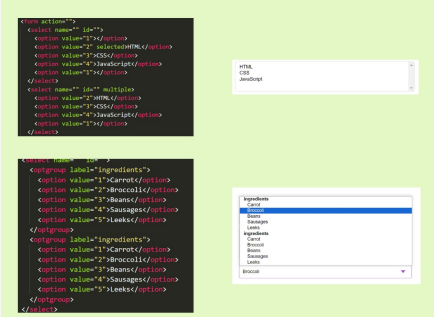
Forms (cont)

with POST, the input value will be included on the body of the HTTP request. With GET, they will be appended to the URL.

Datalists

```
<datalist id="countries">
  <!-- Uniquely identifies country and shows '2' over the country -->
  <option value="2">Australia</option>
  <!-- Uniquely identifies country without ugly output
       and makes it customizable and readable by JavaScript -->
  <option data-value="2">Austria</option>
  <option value="">Spain</option>
  <option value="">Turkey</option>
</datalist>
```

Drop-down lists



Forms: grouping related fields

```
<!-- Grouping related fields -->
<fieldset>
  <legend>Billing Address</legend>
  <input type="text" />
  <input type="text" />
  <input type="text" />
</fieldset>
<fieldset>
  <legend>Payment</legend>
  <input type="text" />
  <input type="text" />
  <input type="text" />
</fieldset>
```

Forms: file inputs

```
<input type="file" />
<input type="file" multiple />
<input type="file" accept=".jpg" />
<input type="file" accept=".jpg, .png" />
<input type="file" accept="image/*" />
<input type="file" accept="video/*" />
<input type="file" accept="audio/*" />
```

Forms: Radio buttons

```
<div>
  <input type="radio" name="membership" id="silver" />
  <label class="label-inline" for="silver">Silver membership</label>
</div>
<div>
  <input type="radio" name="membership" id="silver" />
  <label class="label-inline" for="gold">Gold membership</label>
</div>
```