HTML/CSS/Java

Reference Guide
version 0.1.0

HTML

Audio and Video

<audio controls>
<source src="filename**" type="audio/mp3">**
Your browser does not support the audio.
</audio>
<video width="pixels**" height="**pixels**" controls>**
<source src="filename**" type="video/mp4">**
Your browser does not support the video.
</video>

Canvas

<canvas id="id**" width="**pixels**" height="**pixels**">**
</canvas>: define a canvas on the document

Document structure

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8**">**: character set
<meta name="viewport" content="width=device-width, initial-scale=1.0">: standard settings
<meta name="description" content="Description of the page.**>**
<meta name="keywords" content="SEO tags**">**
<title> browsers title bar **</title>**
</head>: metadata
<header> </header>: introductory content
<nav> </nav>: navigation links
<main>
<article> </article>: independent content
<section> </section>: a section in a document
</main>: main content
<body> </body>: body part
<footer> </footer>: footer part
</html>

Drag and drop

** **text** **: make a division draggable
** **text** **: reserve a droppable, see Drag and Drop within JavaScript section

Forms

<form>
<label for="id**">** label:**</label>**: define a label text
<input list="id**" id="**id**" name="**field**" autocomplete="**bool**">**: make a dropdown list with id, using a datalist with id
<datalist id="id**" <option value="**text**"> … </datalist>**: set the values for the dropdown list
<input type="text" id="id**" name="**field**">**
<input type="number" id="id**" name="**field**">**
<input type="range" id="id**" name="**field**" min="**int**" max="**int**">**
<input type="color" id="id**" name="**field**" value="#**rrggbb**">**
<input type="file" id="id**" name="**field**">**
<input type="password" id="id**" name="**field**" required>**
<input type="email" id="id**" name="**field**">**
<input type="url" id="id**" name="**field**">**
<input type="date" id="id**" name="**field**">**
<input type="radio" id="id**" name="**field**" value="**value**">**
<input type="checkbox" id="id**" name="**field**" checked>**
<textarea id="id**" name="**field**" rows="**#rows**" </textarea>**
<input type="submit" value="button text**">**
<output name="field**" for="**expression**">**text **</output>**: display calculation in forms
</form>

IFrame

<iframe width="int**" height="**int**" src="**url**" title="**title**" frameborder="**int**" allowfullscreen>**
</iframe>: makes a new inset frame

Image maps

<img src="filename**" usemap="#**id**" alt="**text**">**
<map name="id**">**: define an image map
<area shape="rect" coords="x1,y1,x2,y2**" alt="**text**" href="**link**">**: set a rect shape link
<area shape="circle" coords="x,y,r**" alt="**text**" href="**link**">**: set a circular shape link
<area shape="poly" coords="x1,y1,xn,yn**" alt="**text**" href="**link**">**: set a polygon shape link
</map>

Microdata

<div itemscope itemtype="schema**">**: creates a new microdata item of specific type
****value****
** **
<div data-info="value**" data-id="**property**"** Text ** **: custom data attributes
** **
****value****
** **: RDFa microdata

Scalable Vector Graphics

<svg xmlns="http://www.w3.org/2000/svg" viewBox="x1 y1 x2 y2**" fill="**color**" stroke="**color**" stroke-width="**int**" stroke-linecap="round" stroke-linejoin="round">**
<circle cx="int**" cy="**int**" r="**int**"> </circle>**
<line x1="int**" y1="**int**" x2="**int**" y2="**int**"> </line>**
</svg>

Tables

<table border="border width**">**
<thead>: table header
<tr>: table row
<th>header text**</th>**: header text
</tr>
</thead>
<tbody>: table contents
<tr>
<td>row data**</td>**: row data text
</tr>
</tbody>
</table>

Tags

** ** division ** **: creates a new division
<h1> heading text **</h1>**: you can use h1 to h6
** ** new paragraph ** **
**** bold textstyle ****
**** italic textstyle ****
<ul>: unordered list
**
** unordered list item **
**
**
**
<ol>: ordered list, with numbers
**
** ordered list item **
**
**
**
<a href="website**"**link text****: anchor text
<a href="#sectionid**"**link****: link within site
<details>: make summary element
<summary>summary_text**</summary>**
** … **: the hidden details
</details>: close the summary element
<img src="filename**" alt="**text**">**: image, if not found display alternate text
<link rel="import" href="link**">**: imports link in the current document
** **items** **: ARIA role attribute
<template id="id**"> </template>**: define template

CSS

Backgrounds & reflection

background-color: color
background-image: url(url**)**
background-repeat: repeat | **repeat-x** | **repeat-y** | **no-repeat**
background-position: top | **bottom** | **left** | **right**
background-attachment: scroll | **fixed**
background-size: cover | **contain** | xsize ysize
background-origin: padding-box | **border-box** | **content-box**: origin position of a background
background-clip: padding-box | **border-box** | **content-box**: how far should background extend
box-reflect: below | **above** | **left** | **right** int**px** **linear-gradient(**direction**,** color**,** color**)**

Borders

border-width: int
border-style: solid | **dashed** | **dotted** | **double**
border-color: color
border-radius: int int: horizontal and vertical radius
border-top-left-radius: int: specific corner radius
border-image: url(url**)**: url or gradient
border-image-source: url(url**)**: url or gradient
border-image-slice: int**%**: how to slide the source
border-image-width: value: defines the width
border-image-repeat: repeat | **stretch** | **round** | **space**
border-collapse: bool

Colors

keyword: eg. red, blue, green
#RRGGBB: rgb in hexadecimal notation
RGB(Red**,** Green**,** Blue**)**: rgb in csv notation
RGBA(Red**,** Green**,** Blue**,** Alpha**)**: with alpha value
HSL(Hue**,** Saturation**,** Lightness**)**: hue=angle, sl=%
opacity: %: value between 0=invisible and 1=visible
linear-gradient (direction**,** color**,** color**)**: direction eg. to right, to top, to bottom-right from color to color. You can define more than two colors

HTML/CSS/Java

Reference Guide
version 0.1.0

radial-gradient (color**,** color**)**: radial from color to color gradient

Image gallery

** **
<img src=“file**” alt=“**alternate text**”>**
…: add as many images as needed
** <style>**
.gallery {display: flex; flex-wrap: wrap; justify-content: space-around; margin: x**px;}**
.gallery img {width: 100%; height: auto; transition: transform time**s ease-in-out;}**
.gallery img:hover {transform: scale(value**);}**
</style>

Lists

list-style-type: disc | **circle** | **square** | **decimal** | **lower-alpha** | **upper-alpha** | **lower-roman** | **upper-roman** | **none**
list-style-image: url(url**)**
list-style-position: outside | **inside**
list-style: allows to set all styles in one declaration

Margins & Padding

margin-bottom | **margin-top:** value
margin-left | **margin-right:** value
margin: value
padding*: value

Math

calc(expression**)**: make a calculation
min(values**)**: gets the minimum of a series
max(values**)**: gets the maximum of a series
clamp(minvalue**,** preferedvalue**,** maxvalue**)**
abs(value**)**: returns the absolute value
sqrt(value**)**: returns the square root
sin(value**)** | **cos(**value**)** | **tan(**value**)**: trigonometric
rad(value**)** | **deg(**value**)** | **grad(**value**)** | **turn(**value**)**

Navigation & Dropdowns

<nav>
****Nav item****: insert html nav item
…: insert as many navigation items you need
** **: make a dropdown class
<button class=“dropbtn”>drop item**</button>**
** **
****drop item****: insert drop item
…: insert as many dropdown items you need
** </nav>**
<style>
nav {background-color: color**; overflow: hidden;}**
nav a {float: left; display: block; color: color**; text-align:center; padding:** x**px** y**px; }**
nav a:hover {background-color: color**; color:** color**;**}: choose action when mouse hovering
.dropdown {float: left; overflow: hidden;}
.dropdown .dropbtn {font-size: int**px; border: none; outline: none; color:** color**; padding:** x**px** y**px; background-color:inherit; margin:0;}**
.dropcontent {display: none; position: absolute; background-color: color**; min-width:** x**px; box-shadow** int int**px** int**px** int color**); z-index 1;}**
.dropcontent a {float: none; color: value**; padding** x**px** y**px; text-decoration:none; display: block; text-align: left; }**: dropdown content style
.dropcontent a:hover {background-color: color**; }**
.dropdown: hover .dropcontent {display: block;}: show the dropdown menu on mouse hover
</style>

Outline

outline: int**px dashed** | **solid** | **dotted #**color

Position and Float

position: static: elements positioned normally
position: relative: move elements relative to its normal position
position: absolute: element is removed from normal flow and relative to its ancestor
position: fixed: element is removed from normal flow and relative to the viewport
position: sticky: element is treated as relative within its container until it crosses a specified scroll threshold after which it becomes fixed
z-index: int | **auto**
float: left | **right**: position an element horizontally within its containing parent
clear: left | **right** | **both**: prevent elements from wrapping around a floated element
display: inline-block: an inline level element with block level properties

Rules

@media (max-width: int**px) {}**: media query, executes if query is true

Selectors

tag **{**…**}**: select a tag eg. body
.class **{**…**}**: selects a class
.class**:** event **{**…**}**: pseudo class event eg. button: hover, button: active, button: focus, button: visited, checkbox: checked, button: disabled, text: empty, link:target
.class**::** element **{**…**}**: select a pseudo element eg. ::before (generates a virtual element before the content), ::after, ::first-line, ::first-letter, ::selection (selected by the user), ::placeholder, ::marker (bullet or number in a list), ::backdrop (background behind a modal dialog), ::last-line, ::placeholder-shown (input area when placeholder text is shown)
#id **{**…**}**: selects an id
tag **[**attribute**=“**value**”] {**commands**}**: select elements based on attribute eg. text. **$=** is a suffix search, **~=** is a space separated search, **|=** is a prefixed search, **=“”** absence search. attribute* selects substring attributes, attribute^ selects prefix attributes
tr:nth-child(odd) {commands**}**: select pseudo class element eg. odd rows
div > p:first-child {commands**}**: select parent > child, eg. first child
tag **+** tag **{**…**}**: adjacent sibling selector
tag **~** tag **{**…**}**: general sibling selector
tag*** {**…**}**: selects all elements from tag
tag**,** tag **{**…**}**: grouping selector

Size & Overflow

height: value | **min-height** | **max-height** | **auto**
width: value | **min-width** | **max-width** | **auto**
max-width: value
overflow: visible (overflow is rendered) | **hidden** (overflow is hidden) | **scroll** (a scrollbar is added) | **auto** (scrollbar is added when needed)
overflow-x|**y:** value: horizontal|vertical overflow

Styles

/* comments ***/**: insert comments
<style>
.div {
display:flex;: make a flexbox
justify-content:space-between;: equal space
}
.div { flex:int**; }**: flexbox of int rows
</style>
<style>
.div {
display: grid;: make a grid
grid-template-columns: repeat(int**,** int**fr);**: make int columns with int flex rate width
gap: int**px;**: leave a gap of int pixels
stylerule**:** value **!important**: give rule priority
} </style>

Tables

border-collapse: collapse | **separate**
border-spacing: value value: x and y spacing
width: value
text-align: left | **right** | **center** | **justify**
vertical-align: baseline | **sub** | **super** | **top** | **text-top** | **middle** | **bottom** | **text-bottom** | **initial** | **inherit**
background-colors: color

Text and Fonts

font-family: font
font-size: value
font-weight: bold | **normal** | value
font-style: normal | **italic** | **oblique**
font-variant: normal | **small-caps**
text-decoration: underline | **overline** | **line-through** | **none**
text-align: left | **right** | **center** | **justify**
line-height: value
letter-spacing: value
text-transform: uppercase | **lowercase** | **capitalizing**
white-space: normal | **no-wrap** | **pre**
text-overflow: clip | **string** | **ellipsis** | **initial** | **inherit**
word-wrap: normal | **break-word** | **initial** | **inherit**
vertical-align: baseline | **sub** | **super** | **top** | **text-top** | **middle** | **bottom** | **text-bottom** | **initial** | **inherit**
text-shadow: value**px** value**px**: x and y direction
direction: ltr | **rtl**
opacity: %: sets opacity between 0 and 1

HTML/CSS/Java

Reference Guide
version 0.1.0

Transforms

transform: translate(xpx, ypx): move element
transform: rotate(intdeg): rotates int degrees
transform: scale(float**)**: scales float percentage
transform: skew(intdeg): skew int degrees
transform-origin: value value: sets the origin for the transformations in absolute or relative value
translate3D(x, y, z**)**: 3d translation
rotate3D(x, y, z, int**deg)**: 3d rotation
scale3D(x, y, z**)**: 3d scaling
perspective: int**px**: sets a perspective offset
perspective-origin: left | **right** | **center** | **length** | **% , top** | **center** | **bottom** | **length** | **%**
backface-visibility: visible | **hidden**: the backside is visible or invisible when x rotation > 90 deg

Transitions and animations

animation-name: string
animation-duration: int**s**
animation-delay: int**s**
animation-timing-function: ease | **linear** | **ease-in** | **ease-out** | **ease-in-out** | **step-start** | **step-end** | **steps(**int, **start** | **end)** | **cubic-bezier(**%, %, %**)**
animation-iteration-count: int
animation-direction: forward | **backward** | **alternate**
animation-fill-mode: none | **forwards** | **backwards** | **both**: how styles are applied on the elements
animation-play-state: running | **paused**
transition: width int**s, height** int**s, background-color** ints: animate width, height or background changes with int seconds
@keyframes name **{ from { transform: translateX (-**int**%); } to { transform: translateX (**int**%); } }**: define keyframes with a name that does a horizontal move from position int to int
.slide-in-box { width: int**px; height:** int**px; background-color:** color**; animation:** name int**s ease-in-out;}**: a slide in box animation using the name keyframes you defined before
transition-duration: int**s**: duration of the transition
transition-timing-function: ease | **linear** | **ease-in** | **ease-out** | **ease-in-out** | **step-start** | **step-end** | **steps(**int, **start** | **end)** | **cubic-bezier(**%, %, %**)**
transition-delay: int**s**: delay for the transition

Variables

--var: define a custom css property variable
var(--var**)**: gets the value of the variable
counter-reset: variable: set the counter to 0
counter-increment: variable: increments counter
content: counter(variable**)**: gets counter value

JavaScript

Arrays

array**.concat(**array**)**: concatenates arrays
array **=** array**.filter(**function**)**: filters an array using a function that returns Boolean value true
array **=** array**.map(**function**)**: make a new array by using a function on an existing array
array**.find(**function**)**: finds the first value that true
array**.indexOf(**value, start**)**: returns first occurrence of value from a start position
array**.includes(**value**)**: returns true if value present
array**.push(**value**)**: adds an element to the array
array**.pop()**: removes the last element
array**.shift()**: removes the first element
array**.unshift(**value**)**: adds element at beginning
array**.splice(**index**,** number**,** value…**)**: at position index, add (if values provided) or remove (no value given) number items
array **=** array**.slice(**start**,** end**)**: make a new array from an existing array from start until end
array**.reverse()**: reverses order of the array
string **=** array**.join(**separator**)**: join elements string
array**.sort()**: sorts the array
array**.reduce(**function**(**total**,** current**))**: executes a reducer function for all elements in the array
array**.forEach(**function**)**: execute function for all
array**.some(**function**)**: checks function (bool) on all elements. Returns true if valid for one element
array**.every(**function**)**: see some, but only returns true if all elements pass the function test (true)

Canvas

Const canvar **= document.getElementById(“**id**”)**
Const var = canvar**.getContext(“2d”)**
Var**.fillStyle = “**color**”**: define a fill color
Var**.fillRect(**x1,y1,x2,y2**)**: draws a filled rectangle
Var**.strokeStyle = “**color**”**: define a brush color
Var**.strokeRect(**x1,y1,x2,y2**)**: draws a rectangle
Var**.beginPath()**: begins or resets current path
Var**.arc(**xc,yc,radius,start,stop**)**: draws an arc with a center point and a radius with start and stop radians (circle = 2*Math.Pi)
Var**.fill()**: fills the closed path
Var**.lineWidth =** int: defines the stroke width
Var**.stroke()**: changes the stroke of the path
Var**.closePath()**: closes the current path

Date

let var **= new Date()**: gets current date and time
let var **= new Date(**yyyy**,** mm**,** dd**,** hh**,** mm**,** ss**)**: set a specific date and time
datevar**.getFullYear** | **getMonth** | **getDate** | **getHours** | **getMinutes** | **getSeconds**
datevar**.toLocaleDateString()**: return locale string
datevar**.toUTCString()**: return UTC date notation
datevar**.toISOString()**: return ISO8601 notation

Drag and drop

const var **= document.getElementById(‘**id**’)**
const var2 **= document.getElementById(‘**id**’)**
var**.addEventListener(‘dragstart’, (event) => { event.dataTransfer.setData(‘text/plain’, ‘text’); })**: define start of dragging, copy text
var2**.addEventListener(‘dragover’, (event) => { event.preventdefault(); })**: no dragover action
var2**.addEventListener(‘drop’, (event) => { const** var **= event.dataTransfer.getData(‘text/plain’) })**: get the dragged data

Events

element**.addEventListener(“click** | **mouseover** | **mouseout** | **keydown** | **keypress** | **keyup** | **change** | **submit** | **load** | **unload”,** function | **function() {**…**})**: adds an event handler to an element assigned by eg. getElementById

Interaction HTML/CSS

var var **= document.getElementById(“**id**”)**
var**.innerHTML=“**html code**”**: replace html code

Logging

//: comments
/* … ***/**: multiline comments
console.log(variable | **“**text**”)**: log to console

Math

Math.abs(value**)** | **ceil(**value**)** | **floor(**value**)** | **round (**value**)** | **max(**value, …**)** | **min(**value,…**)** | **pow (**value, exp**)** | **sqrt(**value**)** | **exp(**value**)** | **log (**value**)** | **log10(**value**)** | **sin(**value**)** | **cos(**value**)** | **tan(**value**)** | **asin(**value**)** | **acos(**value**)** | **atan (**value**)** | **atan2(**y,x**)** | **random()**: random value between 0 and 1 | **degrees(**value**)** | **radians (**value**)**

Numbers

BigInt(“number**”)**: >64-bit floating-point numbers
isNaN(value**)**: returns true if not a number value
isFinite(value**)**: true if value is finite
parseFloat(string**)**: returns the first float from string
parseInt(string**,** radix**)**: parses a string and returns an integer based on the provided radix (base)
number**.toExponential(**fractiondigits**)**: return a string in exponential notation
number**.toFixed(**digits**)**: return a string with a fixed number of digits after the decimal point
number**.toPrecision(**int**)**: returns a string from the numer with the specified precision
number**.toString(**radix**)**: returns a string from the given number with base radix
number**.valueOf()**: returns the primitive value

Operators

+ | **-** | ***** | **/** | **%**: arithmetic operators
= | **+=** | **-=** | ***=** | **/=**: assignment operators
&& | **||** | **!**: logical operators
== | **===** | **!=** | **>** | **<** | **>=** | **<=**: comparison operators. === means same data type
+ | **-** | **++** | **—** | **!**: unary operators
? expression command **:** command: ternary operat
& | **|** | **^** | **~** | **<<** | **>>**: bitwise operators. ^ = xor, ~ = not, << = shift left, >> = shift right

Service workers

if (‘ServiceWorker’ in navigator) { navigator.ServiceWorker.register(“script.js**”) .then(registration => {** commands**, registration. scope)**: registers the service worker
}) .catch(error => {commands**, error) }) }**: if error

Statements

var var **=** function scoped expression
const var **=** constant expression
let var **=** block scoped expression: eg. let sum=x+y
let var **= [“**item1**”, “**item…**”]**: make a list
let var **= new Set([**value**, …])**: unordered collection of unique values

let var **= new Map([**[key, value]…**])**: stores key-value pairs. Both the key and value can be of any data type
let var **= {** key **:**"value"**,** key **:**"value"**}**: make a map
let string **=** `text…**${**var**}**`: embedded expressions
if (expression**) {**…**} else {**…**}**
switch (var**) {**: control flow statement
case "value"**: {**…**}**: checks if value
default: {…**} }**: the default value
for (init**;** condition**;** iterationexpression**) {**…**}**
for (variable **in** object**) {**…**}**: iterates over enumerable properties of an object
for (variable **of** iterable**) {**…**}**: iterates over the values of iterable objects
while (conditiontrue**) {**…**}**: loop condition
break: used to exit a loop eg. for, while, switch
function name**(**var**,**…**) {**…; **return}**
function name**(**var**=**value**) {**…**}**: default value
let name **= (**par**,** …**) =>** expression: arrow function

Storage

localStorage.setItem("field"**,**"value"**)**: locally stores a field value
const var **= localStorage.getItem(**"field"**)**: gets field

String

string**.length()**: returns the number of chars
string**.charAt(**index**)**: returns char at position
string**.charCodeAt(**index**)**: returns Unicode value
string**.toUpperCase()**: returns upper case string
string**.toLowerCase()**: returns lower case string
string**.substring(**startindex**,** endindex**)**: sub string
string**.slice(**startindex**,** endindex**)**: see substring
string**.indexOf(**searchstring**,** startindex**)**: returns the index of the first occurrence
string**.lastIndexOf(**searchstring**,** startindex**)**: last
string**.startsWith(**searchstring**,** position**)**: checks if the string starts with searchstring from position
string**.endsWith(**searchstring**,** position**)**: checks end
string**.includes(**searchstring**,** position**)**: contains
string**.replace(**searchstring**,**replace**)**: replaces text
string**.trim()**: remove leading and trailing spaces
string**.split(**separator**,**limit**)**: split a string in an array of substrings with a separator and limited parts
string**.concat(**string1**,** …**)**: concatenates strings
string**.repeat(**count**)**: repeat the string count times
string**.match(**regex**)**: search against a regular expression and returns the matches as array
string**.search(**regex**)**: search against a regular expression and returns the index of first match

Template

const var **= document.getElementById(**"id"**)**
const var2 **= document.importNode(**var**.content, true)**: import the template content
document. body.appendChild(var2**)**: add to doc

Validation

<form onsubmit="return function**()">**: set function
function function**() {**
var variable **= document.getElementById ("**field"**).value**: get the value of a specific field
if (variable expression**) {**commands**}** **return** boolean: validity check with commands, return false when the validation failed

Web sockets

const var **= new WebSocket('wss://**url'**)**
var**.addEventListener('open',(event) => {socket. Send(**'text'**) })**: connection opened
var**.addEventListener('message',(event) =>** {commands**, event.data) })**: listen for message

Web workers

const var **= new Worker(**"script.js"**)**: start thread
var**.postMessage(**"msg"**)**: sends a message
var**.onmessage=function(event) {**commands, **event.data)**: listen for message event.data

---