Celestra cheatsheet – v5.3.2 – https://github.com/Serrin/Celestra/

The `celestra` and/or the `CEL` objects contain these functions, except the polyfills. Example: `CEL.qsa("p");`

Core API	DOM	Type checking
signbit(<value>);	qsa(<selector>[,context]).forEach(<cb>);	isMap(<value>); *and* isWeakMap(<v>);
delay(<ms>).then(<callback>);	qs(<selector>[,context]);	isSet(<value>); *and* isWeakSet(<v>);
inherit(<subclass>,<superclass>);	domReady(<callback>);	isNumber(<v>); *and* isNumeric(<v>);
randomInt([max] *or* <min>,<max>);	domCreate(<type>[,properties[,innerHTML]]);	isFloat(<v>); *and* isBigInt(<v>);
randomFloat([max] *or* <min>,<max>);	domCreate(<element descriptive object>);	isString(<v>); *and* isChar(<v>);
randomBoolean();	domToElement(<htmlString>);	isDate(<v>); *and* isError(<v>);
randomID([hyphens=true][,usedate=false]);	domGetCSS(<element>[,property]);	isRegexp(<v>); *and* isSymbol(<v>);
randomString([length[,specChar=false]]);	domSetCSS(<element>,<property>,<value>);	isElement(<v>); *and* isObject(<v>);
inRange(<value>,<min>,<max>);	domSetCSS(<element>,<properties>);	isDataView(<v>);, isBoolean(<v>);
b64Encode(<string>);, b64Decode(<str>);	domFadeIn(<element>[,duration[,display]]);	isNull(<val>);, isUndefined(<val>);
javaHash(<data>[,hexa=false]);	domFadeOut(<element>[,duration]);	isNullOrUndefined(<v>); isNil(<v>);
getUrlVars([str=location.search]);	domFadeToggle(<elem.>[,duration[,display]]);	isPlainObject(<value>);
obj2string(<object>);	domShow(<element>[,display]);	isFunction(<value>);
getType(<variable>[,type][,throw=false]);	domHide(<element>);	isConstructorFn(<value>);
extend([deep,]<target>,<source1>[,srcN]);	domToggle(<element>[,display]);	isGeneratorFn(<value>);
sizeIn(<obj>); *and* forIn(<obj>,<cb>);	domIsHidden(<element>);	isAsyncGeneratorFn(<value>);
filterIn(<obj>,<cb>);, popIn(<obj>,<pr>);	domSiblings(<element>);	isAsyncFn(<value>);
strPropercase(<s>);	domSiblingsPrev(<element>);	isArraylike(<value>);
strCapitalize(<s>);	domSiblingsLeft(<element>);	isTypedArray(<value>);
strUpFirst(<str>);, strDownFirst(<str>);	domSiblingsNext(<element>);	isArrayBuffer(<value>);
strHTMLRemoveTags(<string>);	domSiblingsRight(<element>);	isPrimitive(<value>);
strHTMLEscape(<s>);,strHTMLUnEscape(<s>);	domGetCSSVar(<name>);	isIterator(<v>);, isIterable(<v>);
strReverse(<str>);, strAt(<str>,<index>);	domSetCSSVar(<name>,<value>);	isPromise(<value>);
strCodePoints(<string>);	importScript(<script1>[,scriptN]);	isEmptyObject(<value>);
strFromCodePoints(<collection>);	importStyle(<style1>[,styleN]);	isEmptyArray(<value>);
toFunction(<fn>);, bind(<fn>,<context>);	setFullscreenOn(<selector> *or* <element>);	isEmptyMap(<value>);
constant(<value>); *and* identity(<value>);	setFullscreenOff();	isEmptySet(<value>);
noop(); *and* T(); *and* F();	getFullscreen();	isEmptyIterator(<value>);
assertEq(<msg>,<v1>,<v2>[,strict=true]);	form2array(<form>);	isSameObject(<object1>,<object2>);
assertNotEq(<m>,<v1>,<v2>[,strict=true]);	form2string(<form>);	isSameArray(<array1>,<array2>);
assertTrue(<msg>,<value>);	getDoNotTrack();	isSameMap(<map1>,<map2>);
assertFalse(<msg>,<value>);	getLocation(<success>[,error]);	isSameSet(<set1>,<set2>);
noConflict(); *and* VERSION;	createFile(<filename>,<content>[,dType]);	isSameIterator(<iter1>,<iter2>);

AJAX and CORS

ajax(<Options object>);, getJson(<url>,<success>);, getText(<url>,<success>);
Options object properties (* = default value): url: *string*, data: *string*, queryType: **"ajax"/"cors"*, type: **"get"/"post"*, success: *function*, error: *function*, format: **"text"/"json"/"xml"*, user: *string*, password: *string*

Collections

arrayCreate([length=0]);
arrayDeepClone(<array>);
arrayMerge(<target>,<source1>[,sourceN]);
arrayUnique(<collection>);
arrayAdd(<array>,<value>);
arrayClear(<array>);
arrayRemove(<array>,<value>[,all=false]);
arrayRemoveBy(<array>,<callback>[,all=false]);
arrayRange([start=0[,end=99[,step=1]]]);
arrayCycle(<collection>[,n=100]);
arrayRepeat(<value>[,n=100]);
iterRange([start=0[,step=1[,end=Infinity]]]);
iterCycle(<iter>[,n=Infinity]);
iterRepeat(<value>[,n=Infinity]);
arrayUnion(<collection1>[,collectionN]);
arrayIntersection(<collection1>,<collection2>);
arrayDifference(<collection1>,<collection2>);
arraySymmetricDifference(<collec1>,<collec2>);
setUnion(<collection1>[,collectionN]);
setIntersection(<set1>,<set2>);
setDifference(<set1>,<set2>);
setSymmetricDifference(<set1>,<set2>);
isSuperset(<superCollection>,<subCollection>);
slice(<collection>[,begin=0[,end=Infinity]]);
withOut(<collection>,<filterCollection>);
reduce(<collection>,<callback>[,initialvalue]);
take(<collection>[,n=1]);
takeWhile(<collection>,<callback>);
takeRight(<collection>[,n=1]);
takeRightWhile(<collection>,<callback>);
drop(<collection>[,n=1]);
dropWhile(<collection>,<callback>);
dropRight(<collection>[,n=1]);
dropRightWhile(<collection>,<callback>);

forEach(<collection>,<callback>);
map(<collection>,<callback>);
enumerate(<collection>[,offset=0]);
entries(<collection>[,offset=0]);
size(<collection>);
every(<collection>,<callback>);
some(<collection>,<callback>);
none(<collection>,<callback>);
includes(<collection>,<value>);
contains(<collection>,<value>);
find(<collection>,<callback>);
findLast(<collection>,<callback>);
filter(<collection>,<callback>);
reject(<collection>,<callback>);
partition(<collection>,<callback>);
groupBy(<collection>,<callback>);
shuffle(<collection>);
min(<value1>[,valueN]);
max(<value1>[,valueN]);
sort(<collection>[,numbers=false]);
reverse(<collection>);
zip(<collection1>[,collectionN]);
unzip(<collection>);
zipObj(<collection1>,<collection2>);
item(<collection>,<index>);
nth(<collection>,<index>);
first(<collection>);
head(<collection>);
last(<collection>);
initial(<collection>);
tail(<collection>);
flat(<collection>);
concat(<collection1>[,collectionN]);
join(<collection>[,separator=","]);

Polyfills

Array.prototype.at();
Array.prototype.findLast();
Array.prototype.findLastIndex();
Array.prototype.flat(); + .flatMap();
crypto.randomUUID();, globalThis;
Object.hasOwn(); *and* .fromEntries();
String.prototype.at(); + .matchAll();
String.prototype.padStart();
String.prototype.padEnd();
String.prototype.replaceAll();
String.prototype.trimStart();
String.prototype.trimLeft();
String.prototype.trimEnd();
String.prototype.trimRight();
TypedArray.prototype.at();
TypedArray.prototype.findLast();
TypedArray.prototype.findLastIndex();

Non-standard polyfills

BigInt.prototype.toJSON();
AsyncFunction(); GeneratorFunction();

Abstract functions

hasIn(<obj>,<prop>); getIn(<o>,<pr>);
setIn(<object>,<property>,<value>);
isPropertyKey(<v>); *and* isIndex(<v>);
toPropertyKey(<v>); *and* toIndex(<v>);
toInteger(<value>);
toObject(<value>); *and* type(<value>);
createDataProperty(<obj>,<prop>,<v>);
createMethodProperty(<obj>,<pr>,<v>);
isSameValue(<value1>,<value2>);
isSameValueZero(<value1>,<value2>);
isSameValueNonNumber(<val1>,<val2>);

Cookie

getCookie([name]);, hasCookie(<name>);,
setCookie(<name>,<value>[,hours=8760[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]]);, setCookie(<Optionsobj>);
removeCookie(<name>[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]);, removeCookie(<Options object>);,
clearCookies([path="/"[,domain[,sec[,SameSite="Lax"[,HttpOnly]]]]]);, clearCookies(<Options object>);