Celestra cheatsheet – v5.4.3 – 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>);
sleep(<ms>).then(<callback>);		isNumber(<v>); *and* isNumeric(<v>);
inherit(<subclass>,<superclass>);	domReady(<callback>);	isFloat(<val>); *and* isBigInt(<v>);
randomInt([max] *or* <min>,<max>);	domCreate(<type>[,properties[,innerHTML]]);	isString(<v>); *and* isChar(<val>);
randomFloat([max] *or* <min>,<max>);	domCreate(<element descriptive object>);	isDate(<val>); *and* isError(<val>);
randomBoolean();	domToElement(<htmlString>);	isRegexp(<v>); *and* isSymbol(<v>);
randomID([hyphens=true][,usedate=false]);	domGetCSS(<element>[,property]);	isElement(<v>); *and* isObject(<v>);
randomString([length[,specChar=false]]);	domSetCSS(<element>,<property>,<value>);	isDataView(<value>);
inRange(<value>,<min>,<max>);	domSetCSS(<element>,<properties>);	isBoolean(<value>);
b64Encode(<string>); b64Decode(<string>);	domFadeIn(<element>[,duration[,display]]);	isNull(<value>);
javaHash(<data>[,hexa=false]);	domFadeOut(<element>[,duration]);	isUndefined(<value>);
getUrlVars([str=location.search]);	domFadeToggle(<elem.>[,duration[,display]]);	isNullOrUndefined(<value>);
obj2string(<object>);	domShow(<element>[,display]);	isNil(<value>);
classof(<variable>[,type[,throw=false]]);	domHide(<element>);	isPlainObject(<value>);
extend([deep,]<target>,<source1>[,srcN]);	domToggle(<element>[,display]);	isTruthy(<value>);
sizeIn(<object>);	domIsHidden(<element>);	isFalsy(<value>);
forIn(<object>,<callback>);		isFunction(<v>); + isCallable(<v>);
filterIn(<object>,<callback>);	domSiblings(<element>);	isConstructorFn(<value>);
popIn(<object>,<property>);	domSiblingsPrev(<element>);	isGeneratorFn(<value>);
strPropercase(<string>);	domSiblingsLeft(<element>);	isAsyncGeneratorFn(<value>);
strTitlecase(<string>);	domSiblingsNext(<element>);	isAsyncFn(<value>);
strCapitalize(<string>);	domSiblingsRight(<element>);	isArraylike(<value>);
strUpFirst(<str>); *and* strDownFirst(<s>);	domGetCSSVar(<name>);	isTypedArray(<value>);
strHTMLRemoveTags(<string>);	domSetCSSVar(<name>,<value>);	isArrayBuffer(<value>);
strHTMLEscape(<string>);		isPrimitive(<value>);
strHTMLUnEscape(<string>);	importScript(<script1>[,scriptN]);	isPromise(<value>);
strReverse(<string>);	importStyle(<style1>[,styleN]);	isIterator(<value>);
strAt(<string>,<index>);		isIterable(<value>);
strCodePoints(<string>);	setFullscreenOn(<selector> *or* <element>);	isEmptyObject(<value>);
strFromCodePoints(<collection>);	setFullscreenOff();	isEmptyArray(<value>);
bind(<fn>,<context>); *and* unBind(<fn>);	getFullscreen();	isEmptyMap(<value>);
constant(<value>); *and* identity(<value>);		isEmptySet(<value>);
noop(); *and* T(); *and* F();	form2array(<form>);	isEmptyIterator(<value>);
assertEq(<msg>,<v1>,<v2>[,strict=true]);	form2string(<form>);	isSameObject(<object1>,<object2>);
assertNotEq(<m>,<v1>,<v2>[,strict=true]);		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>);

Collections		Polyfills
arrayCreate([length=0]);	forEach(<collection>,<callback>);	
arrayDeepClone(<array>);	map(<collection>,<callback>);	Array.prototype.at();
arrayMerge(<target>,<source1>[,sourceN]);	enumerate(<collection>[,offset=0]);	Array.prototype.findLast();
arrayUnique(<collection>);	entries(<collection>[,offset=0]);	Array.prototype.findLastIndex();
arrayAdd(<array>,<value>);	size(<collection>);	Array.prototype.flat();
arrayClear(<array>);		Array.prototype.flatMap();
arrayRemove(<array>,<value>[,all=false]);	every(<collection>,<callback>);	Array.prototype.groupBy();
arrayRemoveBy(<array>,<callback>[,all=false]);	some(<collection>,<callback>);	Array.prototype.groupByToMap();
arrayRange([start=0[,end=99[,step=1]]]);	none(<collection>,<callback>);	
arrayCycle(<collection>[,n=100]);		crypto.randomUUID();
arrayRepeat(<value>[,n=100]);	includes(<collection>,<value>);	
	contains(<collection>,<value>);	globalThis;
iterRange([start=0[,step=1[,end=Infinity]]]);		
iterCycle(<iter>[,n=Infinity]);	find(<collection>,<callback>);	Number.MIN_SAFE_INTEGER;
iterRepeat(<value>[,n=Infinity]);	findLast(<collection>,<callback>);	Number.MAX_SAFE_INTEGER;
arrayUnion(<collection1>[,collectionN]);	filter(<collection>,<callback>);	Object.fromEntries();
arrayIntersection(<collection1>,<collection2>);	reject(<collection>,<callback>);	Object.hasOwn();
arrayDifference(<collection1>,<collection2>);	partition(<collection>,<callback>);	Object.is();
arraySymmetricDifference(<collec1>,<collec2>);	groupBy(<collec.>,<cb>[,map=false]);	
setUnion(<collection1>[,collectionN]);		String.prototype.at();
	shuffle(<collection>);	String.prototype.matchAll();
setIntersection(<set1>,<set2>);	min(<value1>[,valueN]);	String.prototype.replaceAll();
setDifference(<set1>,<set2>);	max(<value1>[,valueN]);	String.prototype.trimStart();
setSymmetricDifference(<set1>,<set2>);	sort(<collection>[,numbers=false]);	String.prototype.trimLeft();
	reverse(<collection>);	String.prototype.trimEnd();
isSuperset(<superCollection>,<subCollection>);		String.prototype.trimRight();
	zip(<collection1>[,collectionN]);	
slice(<collection>[,begin=0[,end=Infinity]]);	unzip(<collection>);	TypedArray.prototype.at();
withOut(<collection>,<filterCollection>);	zipObj(<collection1>,<collection2>);	TypedArray.prototype.findLast();
reduce(<collection>,<callback>[,initialvalue]);	item(<collection>,<index>);	TypedArray.prototype.findLastIndex();
	nth(<collection>,<index>);	
take(<collection>[,n=1]);	first(<collection>);	**Non-standard polyfills**
takeWhile(<collection>,<callback>);	head(<collection>);	
takeRight(<collection>[,n=1]);	last(<collection>);	
takeRightWhile(<collection>,<callback>);	initial(<collection>);	BigInt.prototype.toJSON();
drop(<collection>[,n=1]);	tail(<collection>);	
dropWhile(<collection>,<callback>);	flat(<collection>);	AsyncFunction();
dropRight(<collection>[,n=1]);	concat(<collection1>[,collectionN]);	GeneratorFunction();
dropRightWhile(<collection>,<callback>);	join(<collection>[,separator=","]);	

Math plugin (with celestra-math.min.js)		Abstract functions

sum(<value1>[,valueN]);
avg(<value1>[,valueN]);
product(<value1>[,valueN]);
 	
clamp(<value>,<min>,<max>);
 	
isEven(<value>);
isOdd(<value>);
	toFloat32(<value>);
	
	toInt8(<value>);
	toUInt8(<value>);
	toInt16(<value>);
	toUInt16(<value>);
	toInt32(<value>);
	toUInt32(<value>);
	toBigInt64(<value>);
	toBigUInt64(<value>);
	
	isInt8(<value>);
	isUInt8(<value>);
	isInt16(<value>);
	isUInt16(<value>);
	isInt32(<value>);
	isUInt32(<value>);
	isBigInt64(<value>);
	isBigUInt64(<value>);
		getIn(<object>,<property>);
		getInV(<object>,<property>);
		hasIn(<object>,<property>);
		setIn(<object>,<property>,<value>);
		
		toIndex(<value>);
		toPropertyKey(<value>);
		toInteger(<value>);
		toArray(value);
		toObject(<value>);
		
		isIndex(<value>);
		isPropertyKey(<value>);
		isSameValue(<value1>,<value2>);
		isSameValueZero(<value1>,<value2>);
		isSameValueNonNumber(<value1>,<value2>);
		
		type(<value>);
		createDataProperty(<object>,<property>,<value>);
		createMethodProperty(<object>,<property>,<value>);

Cookie

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

AJAX and CORS

getText(<url>,<success>);
getJson(<url>,<success>);
ajax(<Options object>);
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*