

Celestra cheatsheet – v5.5.2 – <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
<code>signbit(<value>);</code> <code>delay(<ms>).then(<callback>);</code> <code>sleep(<ms>).then(<callback>);</code> <code>inherit(<subclass>,<superclass>);</code> <code>randomInt([<max> or <min>,<max>]);</code> <code>randomFloat([<max> or <min>,<max>]);</code> <code>randomBoolean();</code> <code>randomID([hyphens=true],[,usedate=false]);</code> <code>randomString([length[,specChar=false]]);</code> <code>nanoid([size=21[,alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_-"]]);</code> <code>inRange(<value>,<min>,<max>);</code> <code>b64Encode(<string>); b64Decode(<string>);</code> <code>javaHash(<data>[,hexa=false]);</code> <code>getUrlVars([str=location.search]);</code> <code>obj2string(<object>);</code> <code>classof(<variable>[,type[,throw=false]]);</code> <code>extend([deep,]<target>,<source1>[,srcN]);</code> <code>sizeIn(<obj>); and popIn(<obj>,<prop>);</code> <code>forIn(<object>,<callback>);</code> <code>filterIn(<object>,<callback>);</code> <code>strPropercase(<string>)</code> <code>strTitlecase(<string>);</code> <code>strCapitalize(<string>);</code> <code>strUpFirst(<str>); and strDownFirst(<s>);</code> <code>strHTMLRemoveTags(<string>);</code> <code>strHTMLEscape(<s>); strHTMLUnEscape(<s>);</code> <code>strReverse(<str>);, strAt(<str>,<index>);</code> <code>strCodePoints(<string>);</code> <code>strFromCodePoints(<collection>);</code> <code>bind(<fn>,<context>); and unBind(<fn>);</code> <code>constant(<value>); and identity(<value>);</code> <code>noop(); and T(); and F();</code> <code>assertEq(<msg>,<v1>,<v2>[,strict=true]);</code> <code>assertNotEq(<m>,<v1>,<v2>[,strict=true]);</code> <code>assertTrue(<msg>,<value>);</code> <code>assertFalse(<msg>,<value>);</code> <code>noConflict(); and VERSION;</code>	<code>qsa(<selector>[,context]).forEach(<cb>);</code> <code>qs(<selector>[,context]);</code> <code>domReady(<callback>);</code> <code>domCreate(<type>[,properties[,innerHTML]]);</code> <code>domCreate(<element descriptive object>);</code> <code>domToElement(<htmlString>);</code> <code>domGetCSS(<element>[,property]);</code> <code>domSetCSS(<element>,<property>,<value>);</code> <code>domSetCSS(<element>,<properties>);</code> <code>domFadeIn(<element>[,duration[,display]]);</code> <code>domFadeOut(<element>[,duration]);</code> <code>domFadeToggle(<elem.>[,duration[,display]]);</code> <code>domShow(<element>[,display]);</code> <code>domHide(<element>);</code> <code>domToggle(<element>[,display]);</code> <code>domIsHidden(<element>);</code> <code>domSiblings(<element>);</code> <code>domSiblingsPrev(<element>);</code> <code>domSiblingsLeft(<element>);</code> <code>domSiblingsNext(<element>);</code> <code>domSiblingsRight(<element>);</code> <code>domGetCSSVar(<name>);</code> <code>domSetCSSVar(<name>,<value>);</code> <code>importScript(<script1>[,scriptN]);</code> <code>importStyle(<style1>[,styleN]);</code> <code>setFullscreenOn(<selector> or <element>);</code> <code>setFullscreenOff();</code> <code>getFullscreen();</code> <code>form2array(<form>);</code> <code>form2string(<form>);</code> <code>getDoNotTrack();</code> <code>getLocation(<success>[,error]);</code> <code>createFile(<filename>,<content>[,dType]);</code>	<code>isMap(<value>); and isWeakMap(<v>);</code> <code>isSet(<value>); and isWeakSet(<v>);</code> <code>isNumber(<v>); and isNumeric(<v>);</code> <code>isFloat(<val>); and isBigInt(<v>);</code> <code>isString(<v>); and isChar(<val>);</code> <code>isDate(<val>); and isError(<val>);</code> <code>isRegex(<v>); and isSymbol(<v>);</code> <code>isElement(<v>); and isObject(<v>);</code> <code>isDataView(<value>);</code> <code>isBoolean(<value>);</code> <code>isNull(<value>);</code> <code>isUndefined(<value>);</code> <code>isNullOrUndefined(<value>);</code> <code>isNil(<value>);</code> <code>isPlainObject(<value>);</code> <code>isTruthy(<value>);</code> <code>isFalsy(<value>);</code> <code>isFunction(<v>); + isCallable(<v>);</code> <code>isConstructorFn(<value>);</code> <code>isGeneratorFn(<value>);</code> <code>isAsyncGeneratorFn(<value>);</code> <code>isAsyncFn(<value>);</code> <code>isArraylike(<value>);</code> <code>isTypedArray(<value>);</code> <code>isArrayBuffer(<value>);</code> <code>isPrimitive(<value>);</code> <code>isPromise(<value>);</code> <code>isIterator(<value>);</code> <code>isIterable(<value>);</code> <code>isEmptyObject(<value>);</code> <code>isEmptyArray(<value>);</code> <code>isEmptyMap(<value>);</code> <code>isEmptySet(<value>);</code> <code>isEmptyIterator(<value>);</code> <code>isSameObject(<object1>,<object2>);</code> <code>isSameArray(<array1>,<array2>);</code> <code>isSameMap(<map1>,<map2>);</code> <code>isSameSet(<set1>,<set2>);</code> <code>isSameIterator(<iter1>,<iter2>);</code>

Collections		Polyfills
<pre> 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>); </pre>	<pre> 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>); group(<collect.>,<cb>[,map=false]); 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=","]); </pre>	<pre> Array.prototype.at(); Array.prototype.findLast(); Array.prototype.findLastIndex(); Array.prototype.flat(); Array.prototype.flatMap(); Array.prototype.group(); Array.prototype.groupToMap(); crypto.randomUUID(); globalThis; Number.MIN_SAFE_INTEGER; Number.MAX_SAFE_INTEGER; Object.fromEntries(); Object.hasOwn(); Object.is(); String.prototype.at(); String.prototype.matchAll(); String.prototype.replaceAll(); String.prototype.trimStart(); String.prototype.trimLeft(); String.prototype.trimEnd(); String.prototype.trimRight(); TypedArray.prototype.at(); TypedArray.prototype.findLast(); TypedArray.prototype.findLastIndex(); </pre>
		Non-standard polyfills
		<pre> BigInt.prototype.toJSON(); AsyncFunction(); GeneratorFunction(); </pre>

Math functions		Abstract functions
sum(<value1>[,valueN]); avg(<value1>[,valueN]); product(<value1>[,valueN]); clamp(<value>,<min>,<max>); isEven(<value>); isOdd(<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>);
toInt8(<value>); toUInt8(<value>); toInt16(<value>); toUInt16(<value>); toInt32(<value>); toUInt32(<value>); toBigInt64(<value>); toBigUInt64(<value>); toFloat32(<value>);	isInt8(<value>); isUInt8(<value>); isInt16(<value>); isUInt16(<value>); isInt32(<value>); isUInt32(<value>); isBigInt64(<value>); isBigUInt64(<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: <i>"ajax"/"cors"</i> , type: <i>"get"/"post"</i> , success: function, error: function, format: <i>"text"/"json"/"xml"</i> , user: string, password: string		