

## Celestra cheatsheet – v5.5.0 – <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(&lt;value&gt;);</code> <code>delay(&lt;ms&gt;).then(&lt;callback&gt;);</code> <code>sleep(&lt;ms&gt;).then(&lt;callback&gt;);</code> <code>inherit(&lt;subclass&gt;,&lt;superclass&gt;);</code> <code>randomInt([&lt;max&gt; or &lt;min&gt;,&lt;max&gt;]);</code> <code>randomFloat([&lt;max&gt; or &lt;min&gt;,&lt;max&gt;]);</code> <code>randomBoolean();</code> <code>randomID([hyphens=true],[,usedate=false]);</code> <code>randomString([length[,specChar=false]]);</code> <code>inRange(&lt;value&gt;,&lt;min&gt;,&lt;max&gt;);</code> <code>b64Encode(&lt;string&gt;); b64Decode(&lt;string&gt;);</code> <code>javaHash(&lt;data&gt;[,hexa=false]);</code> <code>getUrlVars([str=location.search]);</code> <code>obj2string(&lt;object&gt;);</code> <code>classof(&lt;variable&gt;[,type[,throw=false]]);</code> <code>extend([deep,]&lt;target&gt;,&lt;source1&gt;[,srcN]);</code> <code>sizeIn(&lt;object&gt;);</code> <code>forIn(&lt;object&gt;,&lt;callback&gt;);</code> <code>filterIn(&lt;object&gt;,&lt;callback&gt;);</code> <code>popIn(&lt;object&gt;,&lt;property&gt;);</code> <code>strPropercase(&lt;string&gt;);</code> <code>strTitlecase(&lt;string&gt;);</code> <code>strCapitalize(&lt;string&gt;);</code> <code>strUpFirst(&lt;str&gt;); and strDownFirst(&lt;s&gt;);</code> <code>strHTMLRemoveTags(&lt;string&gt;);</code> <code>strHTMLEscape(&lt;string&gt;);</code> <code>strHTMLUnEscape(&lt;string&gt;);</code> <code>strReverse(&lt;string&gt;);</code> <code>strAt(&lt;string&gt;,&lt;index&gt;);</code> <code>strCodePoints(&lt;string&gt;);</code> <code>strFromCodePoints(&lt;collection&gt;);</code> <code>bind(&lt;fn&gt;,&lt;context&gt;); and unBind(&lt;fn&gt;);</code> <code>constant(&lt;value&gt;); and identity(&lt;value&gt;);</code> <code>noop(); and T(); and F();</code> <code>assertEq(&lt;msg&gt;,&lt;v1&gt;,&lt;v2&gt;[,strict=true]);</code> <code>assertNotEq(&lt;m&gt;,&lt;v1&gt;,&lt;v2&gt;[,strict=true]);</code> <code>assertTrue(&lt;msg&gt;,&lt;value&gt;);</code> <code>assertFalse(&lt;msg&gt;,&lt;value&gt;);</code> <code>noConflict(); and VERSION;</code>	<code>qsa(&lt;selector&gt;[,context]).forEach(&lt;cb&gt;);</code> <code>qs(&lt;selector&gt;[,context]);</code>  <code>domReady(&lt;callback&gt;);</code> <code>domCreate(&lt;type&gt;[,properties[,innerHTML]]);</code> <code>domCreate(&lt;element descriptive object&gt;);</code> <code>domToElement(&lt;htmlString&gt;);</code> <code>domGetCSS(&lt;element&gt;[,property]);</code> <code>domSetCSS(&lt;element&gt;,&lt;property&gt;,&lt;value&gt;);</code> <code>domSetCSS(&lt;element&gt;,&lt;properties&gt;);</code> <code>domFadeIn(&lt;element&gt;[,duration[,display]]);</code> <code>domFadeOut(&lt;element&gt;[,duration]);</code> <code>domFadeToggle(&lt;elem.&gt;[,duration[,display]]);</code> <code>domShow(&lt;element&gt;[,display]);</code> <code>domHide(&lt;element&gt;);</code> <code>domToggle(&lt;element&gt;[,display]);</code> <code>domIsHidden(&lt;element&gt;);</code>  <code>domSiblings(&lt;element&gt;);</code> <code>domSiblingsPrev(&lt;element&gt;);</code> <code>domSiblingsLeft(&lt;element&gt;);</code> <code>domSiblingsNext(&lt;element&gt;);</code> <code>domSiblingsRight(&lt;element&gt;);</code> <code>domGetCSSVar(&lt;name&gt;);</code> <code>domSetCSSVar(&lt;name&gt;,&lt;value&gt;);</code>  <code>importScript(&lt;script1&gt;[,scriptN]);</code> <code>importStyle(&lt;style1&gt;[,styleN]);</code>  <code>setFullscreenOn(&lt;selector&gt; or &lt;element&gt;);</code> <code>setFullscreenOff();</code> <code>getFullscreen();</code>  <code>form2array(&lt;form&gt;);</code> <code>form2string(&lt;form&gt;);</code>  <code>getDoNotTrack();</code> <code>getLocation(&lt;success&gt;[,error]);</code> <code>createFile(&lt;filename&gt;,&lt;content&gt;[,dType]);</code>	<code>isMap(&lt;value&gt;); and isWeakMap(&lt;v&gt;);</code> <code>isSet(&lt;value&gt;); and isWeakSet(&lt;v&gt;);</code> <code>isNumber(&lt;v&gt;); and isNumeric(&lt;v&gt;);</code> <code>isFloat(&lt;val&gt;); and isBigInt(&lt;v&gt;);</code> <code>isString(&lt;v&gt;); and isChar(&lt;val&gt;);</code> <code>isDate(&lt;val&gt;); and isError(&lt;val&gt;);</code> <code>isRegex(&lt;v&gt;); and isSymbol(&lt;v&gt;);</code> <code>isElement(&lt;v&gt;); and isObject(&lt;v&gt;);</code> <code>isDataView(&lt;value&gt;);</code> <code>isBoolean(&lt;value&gt;);</code> <code>isNull(&lt;value&gt;);</code> <code>isUndefined(&lt;value&gt;);</code> <code>isNullOrUndefined(&lt;value&gt;);</code> <code>isNil(&lt;value&gt;);</code> <code>isPlainObject(&lt;value&gt;);</code> <code>isTruthy(&lt;value&gt;);</code> <code>isFalsy(&lt;value&gt;);</code> <code>isFunction(&lt;v&gt;); + isCallable(&lt;v&gt;);</code> <code>isConstructorFn(&lt;value&gt;);</code> <code>isGeneratorFn(&lt;value&gt;);</code> <code>isAsyncGeneratorFn(&lt;value&gt;);</code> <code>isAsyncFn(&lt;value&gt;);</code> <code>isArraylike(&lt;value&gt;);</code> <code>isTypedArray(&lt;value&gt;);</code> <code>isArrayBuffer(&lt;value&gt;);</code> <code>isPrimitive(&lt;value&gt;);</code> <code>isPromise(&lt;value&gt;);</code> <code>isIterator(&lt;value&gt;);</code> <code>isIterable(&lt;value&gt;);</code> <code>isEmptyObject(&lt;value&gt;);</code> <code>isEmptyArray(&lt;value&gt;);</code> <code>isEmptyMap(&lt;value&gt;);</code> <code>isEmptySet(&lt;value&gt;);</code> <code>isEmptyIterator(&lt;value&gt;);</code> <code>isSameObject(&lt;object1&gt;,&lt;object2&gt;);</code> <code>isSameArray(&lt;array1&gt;,&lt;array2&gt;);</code> <code>isSameMap(&lt;map1&gt;,&lt;map2&gt;);</code> <code>isSameSet(&lt;set1&gt;,&lt;set2&gt;);</code> <code>isSameIterator(&lt;iter1&gt;,&lt;iter2&gt;);</code>

Collections		Polyfills
<pre> arrayCreate([length=0]); arrayDeepClone(&lt;array&gt;); arrayMerge(&lt;target&gt;,&lt;source1&gt;[,&lt;sourceN&gt;]); arrayUnique(&lt;collection&gt;); arrayAdd(&lt;array&gt;,&lt;value&gt;); arrayClear(&lt;array&gt;); arrayRemove(&lt;array&gt;,&lt;value&gt;[,all=false]); arrayRemoveBy(&lt;array&gt;,&lt;callback&gt;[,all=false]); arrayRange([start=0[,end=99[,step=1]]]); arrayCycle(&lt;collection&gt;[,n=100]); arrayRepeat(&lt;value&gt;[,n=100]);  iterRange([start=0[,step=1[,end=Infinity]]]); iterCycle(&lt;iter&gt;[,n=Infinity]); iterRepeat(&lt;value&gt;[,n=Infinity]);  arrayUnion(&lt;collection1&gt;[,&lt;collectionN&gt;]); arrayIntersection(&lt;collection1&gt;,&lt;collection2&gt;); arrayDifference(&lt;collection1&gt;,&lt;collection2&gt;); arraySymmetricDifference(&lt;collec1&gt;,&lt;collec2&gt;); setUnion(&lt;collection1&gt;[,&lt;collectionN&gt;]);  setIntersection(&lt;set1&gt;,&lt;set2&gt;); setDifference(&lt;set1&gt;,&lt;set2&gt;); setSymmetricDifference(&lt;set1&gt;,&lt;set2&gt;);  isSuperset(&lt;superCollection&gt;,&lt;subCollection&gt;);  slice(&lt;collection&gt;[,begin=0[,end=Infinity]]); without(&lt;collection&gt;,&lt;filterCollection&gt;); reduce(&lt;collection&gt;,&lt;callback&gt;[,initialvalue]);  take(&lt;collection&gt;[,n=1]); takeWhile(&lt;collection&gt;,&lt;callback&gt;); takeRight(&lt;collection&gt;[,n=1]); takeRightWhile(&lt;collection&gt;,&lt;callback&gt;); drop(&lt;collection&gt;[,n=1]); dropWhile(&lt;collection&gt;,&lt;callback&gt;); dropRight(&lt;collection&gt;[,n=1]); dropRightWhile(&lt;collection&gt;,&lt;callback&gt;); </pre>	<pre> forEach(&lt;collection&gt;,&lt;callback&gt;); map(&lt;collection&gt;,&lt;callback&gt;); enumerate(&lt;collection&gt;[,offset=0]); entries(&lt;collection&gt;[,offset=0]); size(&lt;collection&gt;);  every(&lt;collection&gt;,&lt;callback&gt;); some(&lt;collection&gt;,&lt;callback&gt;); none(&lt;collection&gt;,&lt;callback&gt;);  includes(&lt;collection&gt;,&lt;value&gt;); contains(&lt;collection&gt;,&lt;value&gt;);  find(&lt;collection&gt;,&lt;callback&gt;); findLast(&lt;collection&gt;,&lt;callback&gt;);  filter(&lt;collection&gt;,&lt;callback&gt;); reject(&lt;collection&gt;,&lt;callback&gt;); partition(&lt;collection&gt;,&lt;callback&gt;); group(&lt;collect.&gt;,&lt;cb&gt;[,map=false]);  shuffle(&lt;collection&gt;); min(&lt;value1&gt;[,&lt;valueN&gt;]); max(&lt;value1&gt;[,&lt;valueN&gt;]); sort(&lt;collection&gt;[,numbers=false]); reverse(&lt;collection&gt;);  zip(&lt;collection1&gt;[,&lt;collectionN&gt;]); unzip(&lt;collection&gt;); zipObj(&lt;collection1&gt;,&lt;collection2&gt;); item(&lt;collection&gt;,&lt;index&gt;); nth(&lt;collection&gt;,&lt;index&gt;); first(&lt;collection&gt;); head(&lt;collection&gt;); last(&lt;collection&gt;); initial(&lt;collection&gt;); tail(&lt;collection&gt;); flat(&lt;collection&gt;); concat(&lt;collection1&gt;[,&lt;collectionN&gt;]); join(&lt;collection&gt;[,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>);
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>);	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>); <b>Options object properties (* = default value):</b> 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		