

Celestra cheatsheet – v4.5.0 – <https://github.com/Serrin/Celestra/>

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

Core API	DOM	Type checking
<code>signbit(<value>);</code> <code>delay(<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>randomString([<length>,<specChar=false>]);</code> <code>b64Encode(<string>);, b64Decode(<str>);</code> <code>javaHash(<data>[,<hexa=false>];</code> <code>getUrlVars([<str=location.search>];</code> <code>obj2string(<object>);</code> <code>getType(<variable>[,<type>];</code> <code>extend([<deep>,<target>,<source1>[,<srcN>];</code> <code>deepAssign(<target>,<source1>[,<srcN>];</code> <code>sizeIn(<obj>); and forIn(<obj>,<cb>);</code> <code>filterIn(<object>,<callback>);</code> <code>popIn(<object>,<property>);</code> <code>strPropercase(<s>);, strCapitalize(<s>);</code> <code>strUpFirst(<str>);, strDownFirst(<str>);</code> <code>strHTMLRemoveTags(<string>);</code> <code>strHTMLEscape(<string>);</code> <code>strHTMLUnEscape(<string>);</code> <code>strReverse(<string>);</code> <code>strCodePoints(<string>);</code> <code>strFromCodePoints(<collection>);</code> <code>strAt(<string>,<index>);</code> <code>toFunction(<fn>);, bind(<fn>,<context>);</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(<v>); and isBigInt(<v>);</code> <code>isString(<v>); and isChar(<v>);</code> <code>isDate(<v>); and isError(<v>);</code> <code>isRegex(<v>); and isSymbol(<v>);</code> <code>isElement(<v>); and isObject(<v>);</code> <code>isNull(<value>);</code> <code>isUndefined(<value>);</code> <code>isNullOrUndefined(<value>);</code> <code>isNil(<value>);</code> <code>isFunction(<value>);</code> <code>isGeneratorFn(<value>);</code> <code>isAsyncFn(<value>);</code> <code>isDataView(<value>);</code> <code>isBoolean(<value>);</code> <code>isArraylike(<value>);</code> <code>isTypedArray(<value>);</code> <code>isArrayBuffer(<value>);</code> <code>isPrimitive(<value>);</code> <code>isIterator(<value>);</code> <code>isIterable(<value>);</code> <code>isPromise(<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(<set>,<set2>);</code> <code>isSameIterator(<iter1>,<iter2>);</code>
AJAX and CORS		
<code>ajax(<Options object>);, getJson(<url>,<success>);, getText(<url>,<success>);</code> Options object properties (* = default value): <code>url: string, data: string, queryType: *"ajax"/"cors", type: *"get"/"post", success: function, error: function, format: *"text"/"json"/"xml", user: string, password: string</code>		

Collections		Polyfills
arrayMerge([flat=false,]<target>,<src1>[,srN]); arrayUnique(<collection>); arrayAdd(<array>,<value>); arrayClear(<array>); arrayRemove(<array>,<value>[,all=false]); arrayRange([start=0[,end=100[,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]); shuffle(<collection>); 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(<collect.>,<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>); min(<collection>); max(<collection>); sort(<collection>[,numbers=false]); reverse(<collection>); zip(<collection1>[,collectionN]); unzip(<collection>); item(<collection>,<index>); nth(<collection>,<index>); first(<collection>); head(<collection>); last(<collection>); initial(<collection>); tail(<collection>); flat(<collection>); concat(<collection1>[,collectionN]); join(<collection>[,separator=","]);	Array.prototype.at(<index>); Array.prototype.findLast(); Array.prototype.findLastIndex(); Array.prototype.flat(); Array.prototype.flatMap(); globalThis; Object.fromEntries(); Object.hasOwn(); String.prototype.at(<index>); String.prototype.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(<index>); TypedArray.prototype.findLast(); TypedArray.prototype.findLastIndex();
		Non-standard polyfills
		BigInt.prototype.toJSON(); window.AsyncFunction(); window.GeneratorFunction();
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>);		