

Celestra cheatsheet – v3.8.1 – <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>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>randomString([<length>,<specChar>]);</code> <code>b64Encode(<string>);</code> <code>b64Decode(<str>);</code> <code>javaHash(<data>[,<hexa>]);</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>forIn(<object>,<callback>);</code> <code>strRemoveTags(<string>);</code> <code>strReverse(<string>);</code> <code>strReplaceAll(<str>,<search>,<replace>);</code> <code>strCodePoints(<string>);</code> <code>strFromCodePoints(<collection>);</code> <code>strAt(<string>,<pos>);</code> <code>toFunction(<function>);</code> <code>bind(<function>,<context>);</code> <code>constant(<value>); and identity(<value>);</code> <code>noop(); and T(); and F();</code> <code>assert(<condition>[,<message>]);</code> <code>assertLog(<condition>[,<message>]);</code> <code>assertAlert(<condition>[,<message>]);</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>]);, domHide(<el>);</code> <code>domToggle(<element>[,<display>]);</code> <code>domIsHidden(<element>);</code> <code>domSiblings(<element>);</code> <code>domGetCSSVar(<name>);</code> <code>domSetCSSVar(<name>,<value>);</code> <code>importScript(<url>[,<success>]);</code> <code>importScripts(<scripts> or <script1>[,<scN>]);</code> <code>importStyle(<href>[,<success>]);</code> <code>importStyles(<styles> or <style1>[,<styleN>]);</code> <code>setFullscreenOn(<selector> or <element>);</code> <code>setFullscreenOff(); and getFullscreen();</code> <code>form2array(<form>); and form2string(<form>);</code> <code>getDoNotTrack();</code> <code>getLocation(<success>[,<error>]);</code> <code>createFile(<filename>,<content>[,<dType>]);</code>	<code>isMap(<v>); and isWeakMap(<v>);</code> <code>isSet(<v>); and isWeakSet(<v>);</code> <code>isString(<v>); and isChar(<v>);</code> <code>isNumber(<v>); and isNumeric(<v>);</code> <code>isFloat(<v>); and isBigInt(<v>);</code> <code>isDate(<v>); and isError(<v>);</code> <code>isBoolean(<value>);</code> <code>isElement(<value>);</code> <code>isObject(<value>);</code> <code>isEmptyObject(<value>);</code> <code>isArraylike(<value>);</code> <code>isSameArray(<array1>,<array2>);</code> <code>isEmptyArray(<value>);</code> <code>isTypedArray(<value>);</code> <code>isArrayBuffer(<value>);</code> <code>isNull(<value>);</code> <code>isUndefined(<value>);</code> <code>isNullOrUndefined(<value>);</code> <code>isNil(<value>);</code> <code>isPrimitive(<value>);</code> <code>isRegexp(<value>);</code> <code>isSymbol(<value>);</code> <code>isIterator(<value>);</code> <code>isIterable(<value>);</code> <code>isFunction(<value>);</code> <code>isGeneratorFn(<value>);</code> <code>isAsyncFn(<value>);</code> <code>isPromise(<value>);</code>
AJAX and CORS		
<code>ajax(<Options object>);, getJson(<url>,<success>);, getText(<url>,<success>);</code>		
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		
Cookie		
<code>getCookie([<name>]);, hasCookie(<name>);,</code> <code>setCookie(<name>,<value>[,<hours>=8760[,<path>="/"[,<domain>[,<secure>[,<SameSite>="Lax"[,<HttpOnly>]]]]]]];, setCookie(<Options obj>);</code> <code>removeCookie(<name>[,<path>="/"[,<domain>[,<secure>[,<SameSite>="Lax"[,<HttpOnly>]]]]]]];, removeCookie(<Options object>);,</code> <code>clearCookies([<path>="/"[,<domain>[,<sec>[,<SameSite>="Lax"[,<HttpOnly>]]]]]]];, clearCookies(<Options object>);</code>		

Collections	Polyfills
arrayMerge([deep,]<target>,<source1>[,srcN]); arrayUnique(<collection>); and arrayAdd(<array>,<value>); arrayClear(<array>); and arrayRemove(<array>,<value>[,all]); arrayRange([start=0[,end=100[,step=1]]]); and iterRange([start=0[,step=1[,end]]]); arrayCycle(<collection>[,n]); and iterCycle(<iter>[,n]); arrayRepeat(<value>[,n]); and iterRepeat(<value>[,n]); arrayUnion(<collection1>[,collectionN]); arrayIntersection(<collection1>,<collection2>); arrayDifference(<collection1>,<collection2>); arraySymmetricDifference(<collection1>,<collection2>); setUnion(<collection1>[,collectionN]); setIntersection(<set1>,<set2>); setDifference(<set1>,<set2>); setSymmetricDifference(<set1>,<set2>); isSuperset(<superset>,<subset>); partition(<collection>,<callback>); and groupBy(<collection>,<callback>); zip(<collection1>[,collectionN]); and unzip(<collection>); min(<collection>); and max(<collection>); size(<collection>); item(<collection>,<index>); and itemOf(<collection>,<index>); forEach(<collection>,<callback>); map(<collection>,<callback>); filter(<collection>,<callback>); includes(<collection>,<value>); find(<collection>,<callback>); every(<collection>,<callback>); and some(<collection>,<callback>); none(<collection>,<callback>); first(<collection>); and last(<collection>); slice(<collection>[,begin[,end]]); reverse(<collection>); sort(<collection>); reduce(<collection>,<callback>[,initialvalue]); concat(<collection1>[,collectionN]); and flat(<collection>); enumerate(<collection>); join(<collection>[,separator=","]); take(<collection>[,n]); and takeWhile(<collection>,<callback>); takeRight(<collection>[,n]); and takeRightWhile(<collection>,<callback>); drop(<collection>[,n]); and dropWhile(<collection>,<callback>); dropRight(<collection>[,n]); and dropRightWhile(<collection>,<callback>);	Array.prototype.flat(); Array.prototype.flatMap(); globalThis; Object.fromEntries(); Object.hasOwn(); String.prototype.matchAll(); String.prototype.padStart(); String.prototype.padEnd(); String.prototype.replaceAll(); String.prototype.trimStart(); String.prototype.trimLeft(); String.prototype.trimEnd(); String.prototype.trimRight();
	Non-standard polyfills BigInt.prototype.toJSON(); window.AsyncFunction(); window.GeneratorFunction();