

## Celestra cheatsheet – v6.2.0 – <https://github.com/Serrin/Celestra/>

| Core API   | Type API  | DOM API  |
|--|---|--|
| <pre>constant(value); identity(value); noop(); T(); F(); VERSION; eq(value1, value2); gt(value1, value2); gte(value1, value2); lt(value1, value2); lte(value1, value2); extend([deep,]target,source1[,sourceN]); deleteOwnProperty(obj,prop[,Throw=false]); sizeIn(object); pick(object,keys); omit(object,keys); assoc(object,key,value); delay(ms).then(callback); bind(function,context); unBind(function); curry(function); compose(function1[,functionN]); pipe(function1[,functionN]); once(function); tap(function): function(value); randomBoolean(); randomUUIDv7(v4=false); timestampID([size=21[,alphabet="123456789ABCDEFHJKLMNPQRSTUWXYZabcdefghijklmnopqrstuvwxyz"]]); nanoid([size=21[,alphabet="A-Za-z0-9_-"]]); createPolyfillMethod(object,prop,value); createPolyfillProperty(object,prop,value); getUrlVars([str=location.search]); obj2string(object); assert(condition[,message error]);</pre> | <pre>asyncConstant(value); asyncIdentity(value); asyncNoop(); asyncT(); asyncF();  BASE16; BASE32; BASE36; BASE58; BASE62; WORDSAFEALPHABET;  isTypeOf(value); is(val[,expectedType[,Throw=false]]); isTypedCollection(iter,expectedType, Throw=false); isSameType(value1,value2); isSameInstance(v1,v2,Constructor); isDeepStrictEqual(value1,value2); isCoercedObject(object); isEmptyValue(value); isNull(value); isUndefined(value); isNullish(value); isNonNullable(value); isNonNullablePrimitive(value); isNumeric(value); isChar(value); isPlainObject(value); isFunction(value); isCallable(value); isGeneratorFn(value); isAsyncFn(value); isAsyncGeneratorFn(value); isProxy(value); isElement(value); isRegexp(value); isArraylike(value); isTypedArray(value); isIterator(value); isIterable(value); isAsyncIterable(value); isPropertyKey(value); toPropertyKey(value); isPrimitive(value); toPrimitiveValue(value); isObject(value); and toObject(value); isIndex(value); and toIndex(value); isLength(value); and toLength(value); toSafeString(value);</pre> | <pre>qsa(selector[,context]).forEach(callback); qs(selector[,context]); domReady(callback); domClear(element); domCreate(type[,properties[,innerHTML]]); domCreate(element descriptive object); domToElement(htmlString); domGetCSS(element[,property]); domSetCSS(element,property,value); domSetCSS(element,properties); domFadeIn(element[,duration[,display]]); domFadeOut(element[,duration]); domFadeToggle(elem[,duration[,display]]); domShow(element[,display]); domHide(element); domToggle(element[,display]); domIsHidden(element); domScrollToTop(); domScrollToBottom(); domScrollToElement(element[,top=true]); domSiblings(element); domSiblingsPrev(element); domSiblingsLeft(element); domSiblingsNext(element); domSiblingsRight(element); domGetCSSVar(name); domSetCSSVar(name,value); importScript(script1[,scriptN]); importStyle(style1[,styleN]); setFullscreenOn(selector); setFullscreenOn(element); setFullscreenOff(); getFullscreen(); form2array(form); form2string(form); getDoNotTrack(); getLocation(success[,error]); createFile(filename,content[,dType]);</pre> |

| <b>String API</b>                    | <b>Cookie API</b>  | <b>Math API</b>   |
|--------------------------------------|--|---|
| b64Decode(string);                   | getCookie([name]);   | sum(value1[,valueN]);<br>avg(value1[,valueN]);<br>product(value1[,valueN]);<br>mod(value1,value2);<br>rem(value1,value2);<br>clamp(value,min,max);<br>minmax(value,min,max);<br>inRange(value,min,max);<br>signbit(value);<br>randomInt([max]);<br>randomInt(min,max);<br>randomFloat([max]);<br>randomFloat(min,max);<br>isEven(value);<br>isOdd(value);<br>isInt8(value);<br>isInt16(value);<br>isUInt32(value);<br>isUInt8(value);<br>isUInt16(value);<br>isInt32(value);<br>isBigInt64(value);<br>isBigUInt64(value);<br>isFloat16(value);<br>isFloat(value);<br>toInteger(value);<br>toIntegerOrInfinity(value);<br>toInt8(value);<br>toInt16(value);<br>toInt32(value);<br>toUInt8(value);<br>toUInt16(value);<br>toUInt32(value);<br>toBigInt64(value);<br>toBigUInt64(value);<br>toFloat16(value);<br>toFloat32(value); |
| b64Encode(string);                   | hasCookie(name);   |   |
| strAt(string,index[,newChar]);       | setCookie(name,value[,hours=8760[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]]); |   |
| strCapitalization(string);           | setCookie(Options object: properties are the same as the parameters);                        |   |
| strCodePoints(string);               | removeCookie(name[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]);                 |   |
| strDownFirst(string);                | removeCookie(Options object: properties are the same as the parameters);                     |   |
| strFromCodePoints(iterator);         | clearCookies([path="/"[,domain[,sec[,SameSite="Lax"[,HttpOnly]]]]]);                         |   |
| strHTMLEscape(string);               | clearCookies(Options object: properties are the same as the parameters);                     |   |
| strHTMLRemoveTags(string);           |  |   |
| strHTMLUnEscape(string);             |  |   |
| strPropercase(string);               |  |   |
| strReverse(string);                  |  |   |
| strSplice(string,index,count[,add]); |  |   |
| strTitlecase(string);                |  |   |
| strTruncate(string);                 |  |   |
| strUpFirst(string);                  |  |   |

| Collections API   | Polyfills   |
|---|---|
| <pre> castArray(value); compact(iterator); arrayDeepClone(array); arrayMerge(target,source1[,sourceN]); arrayAdd(array,value); arrayClear(array); arrayRemove(array,value[,all = false]); arrayRemoveBy(array,callback[,all=false]);  arrayRange([start=0[,end = 99[,step = 1]]]); iterRange([start=0[,step=1[,end=Infinity]]]);  arrayCycle(iterator[,n = 100]); iterCycle(iterator[,n = Infinity]);  arrayRepeat(value[,n = 100]); iterRepeat(value[,n = Infinity]);  unique(iterator[,resolver]); slice(iterator[,begin=0[,end = Infinity]]); withOut(iterator,filterIterator); reduce(iterator,callback[,initialvalue]); count(iterator,callback);  take(iterator[,n = 1]); takeWhile(iterator,callback); takeRight(iterator[,n = 1]); takeRightWhile(iterator,callback); drop(iterator[,n = 1]); dropWhile(iterator,callback); dropRight(iterator[,n = 1]); dropRightWhile(iterator,callback);  isSuperset(superCollection,subCollection); setDifference(set1,set2); setIntersection(set1,set2); setSymmetricDifference(set1,set2); setUnion(iterator1[,iteratorN] ); </pre> | <pre> forEach(iterator,callback); map(iterator,callback); enumerate(iterator[,offset = 0]); size(iterator);  every(iterator,callback); some(iterator,callback); none(iterator,callback);  includes(collection,value[,comparator]); find(iterator,callback); findLast(iterator,callback); filter(iterator,callback); reject(iterator,callback); partition(iterator,callback);  zip(iterator1[,iteratorN]); unzip(iterator); zipObj(iterator1,iterator1); shuffle(iterator);  min(value1[,valueN]); max(value1[,valueN]); sort(iterator[,numbers = false]); reverse(iterator);  item(iterator,index); nth(iterator,index); first(iterator); head(iterator); last(iterator); initial(iterator); tail(iterator);  flat(iterator); concat(iterator1[,iteratorN]); join(iterator[,separator = ","]); </pre> |

| How to import  |   |            |
|--|---|------------|
| Celestra for browser: celestra.browser.js  | Celestra for Node.js and Deno: celestra.node.mjs  |            |
| <pre> &lt;script type="module"&gt; // import the defaultExport object import defaultExport from "./celestra.browser.js"; globalThis.celestra = defaultExport; globalThis.CEL = defaultExport; &lt;/script&gt;  &lt;script type="module"&gt; // import with default with name import { default as celestra } from "./celestra.browser.js"; globalThis.celestra = celestra; globalThis.CEL = celestra; &lt;/script&gt;  &lt;script type="module"&gt; // import all into a new celestra object import * as celestra from "./celestra.browser.js"; globalThis.celestra = celestra; globalThis.CEL = celestra; &lt;/script&gt;  &lt;script type="module"&gt; // import some functions import { first, map } from "./celestra.browser.js"; globalThis.first = first; globalThis.map = map; &lt;/script&gt;  &lt;script type="module"&gt; // dynamic import const celestra = await import("./celestra.browser.js"); globalThis.celestra = celestra; globalThis.CEL = celestra; &lt;/script&gt; </pre> | <pre> // import the defaultExport object import defaultExport from "./celestra.node.mjs"; globalThis.celestra = defaultExport; globalThis.CEL = defaultExport;  // import with default with name import { default as celestra } from "./celestra.node.mjs"; globalThis.celestra = celestra; globalThis.CEL = celestra;  // import all into a new celestra object import * as celestra from "./celestra.node.mjs"; globalThis.celestra = celestra; globalThis.CEL = celestra;  // import some functions import { first, map } from "./celestra.node.mjs"; globalThis.first = first; globalThis.map = map;  // dynamic import const celestra = await import("./celestra.node.mjs"); globalThis.celestra = celestra; globalThis.CEL = celestra; </pre> |            |
| Removed APIs in the celestra.node.mjs  |   |            |
| DOM API  | Legacy AJAX API   | Cookie API |

| Removed Polyfills - Available in celestra-polyfills.dev.js and celestra-polyfills.min.js   |   |   |
|--|---|---|
| v3.1.0   | v3.8.0  | v5.6.0  |
| <pre>Array.from(); Array.of(); Array.prototype.copyWithin(); Array.prototype.fill(); Array.prototype.find(); Array.prototype.findIndex(); Object.create(); String.fromCodePoint(); String.prototype.codePointAt(); String.prototype.endsWith(); String.prototype.startsWith(); Math.acosh(); Math.asinh(); Math.atanh(); Math.cbrt(); Math.clz32(); Math.cosh(); Math.expm1(); Math.fround(); Math.hypot(); Math.imul(); Math.log1p(); Math.log10(); Math.log2(); Math.sign(); Math.sinh(); Math.tanh(); Math.trunc(); Number.EPSILON; Number.isNaN(); isNaN(); Number.isInteger(); Number.isFinite(); Number.isSafeInteger(); Number.parseInt(); Number.parseFloat();</pre> | <pre>Array.prototype.values(); Array.prototype.includes(); ChildNode.after(); ChildNode.before(); ChildNode.remove(); ChildNode.replaceWith();  Element.prototype.closest(); Element.prototype.getAttributeNames(); Element.prototype.matches(); Element.prototype.toggleAttribute();  ParentNode.append(); ParentNode.prepend();  String.prototype[Symbol.iterator](); String.prototype.includes(); String.prototype.repeat();  NodeList.prototype.forEach();  Object.assign(); Object.entries();  Object.getOwnPropertyDescriptors(); Object.values();  RegExp.prototype.flags;  window.screenLeft; window.screenTop;</pre> | <pre>Array.prototype.at();  Array.prototype.findLast(); Array.prototype.findLastIndex();  Array.prototype.flat(); Array.prototype.flatMap();  Number.MIN_SAFE_INTEGER; Number.MAX_SAFE_INTEGER;  Object.fromEntries(); Object.is();  String.prototype.at();  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();  TypedArray.prototype.findLast(); TypedArray.prototype.findLastIndex();</pre> |
| v5.9.0   |   |   |
| BigInt.prototype.toJSON();   |   |   |