

Celestra cheatsheet – v7.0.1 – <https://github.com/Serrin/Celestra/>

Core API			DOM API
constant(value); identity(value); noop(); T(); F();	asyncConstant(value); asyncIdentity(value); asyncNoop(); asyncT(); asyncF();	eq(value1, value2); gt(value1, value2); gte(value1, value2); lt(value1, value2); lte(value1, value2);	qsa(selector[,context]); 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(element[,duration[,display]]); domShow(element[,display]); domHide(element); domToggle(element[,display]); domIsHidden(element); domScrollToTop(); domScrollToBottom(); domScrollToElement(element[,top=true]);
VERSION; BASE16 = "0123456789ABCDEF"; BASE32 = "23456789ABCDEFGHIJKLMNPOQRSTUVWXYZ"; BASE36 = "0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZ"; BASE58 = "123456789ABCDEFGHIJKLMNPOQRSTUVWXYZabcdefghijklmnopqrstu vwxyz"; BASE62 = "0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZabcdefghijklmnopq rstuvwxyz"; WORDSAFEALPHABET = "23456789CFGHJMPQRVWXCfghjmqvw";	delay(milisec).then(callback); bind(function,context); unBind(function); curry(function); once(function); tap(function): function(value); compose(function1[,functionN]); pipe(function1[,functionN]); assert(condition[,message error]);	RandomBoolean(); randomUUIDv7(v4=false); timestampID([size=21[,alphabet= "23456789CFGHJMPQRVWXCfghjmqvw x"]]);	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]);
String API			
b64Decode(string); b64Encode(string); strCodePoints(string); strFromCodePoints(iterator); strAt(string,index[,newChar]); strCount(string, substring); strSplice(string,index,count[,add]); strTruncate(string); strReverse(string);	strUpFirst(string); strDownFirst(string); strCapitalize(string); strPropercase(string); strTitlecase(string); strHTMLRemoveTags(string); strHTMLEscape(string); strHTMLUnEscape(string);		

Collections API	Math API	Type API	
castArray(value); 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]]]); arrayCycle(iterator[, n=100]); arrayRepeat(value[, n=100]); iterRange([start=0[, step=1[, end=Infinity]]]); iterCycle(iterator[, n=Infinity]); iterRepeat(value[, n=Infinity]); (iterator, callback); findLast(iterator, callback); zip(iterator1[, iteratorN]); unzip(iterator); zipObj(iterator1, iterator2); sort(iterator[, numbers = false]); concat(iterator1[, iteratorN]); join(iterator[, separator=", "]); slice(iterator[, begin=0[, end=Infinity]]); unique(iterator[, resolver]); includes(collection, val[, comparator]); item(iterator, index); nth(iterator, index); first(iterator); head(iterator); last(iterator); reverse(iterator); initial(iterator); tail(iterator); shuffle(iterator); compact(iterator); size(collection); min(value1[, valueN]); max(value1[, valueN]);	sum(value1[, valueN]); avg(value1[, valueN]); product(value1[, valN]); clamp(value, min, max); minmax(value, min, max); inRange(value, min, max); signbit(value); randomInt([max]); randomInt(min, max); randomFloat([max]); randomFloat(min, max); add(value1, value2); sub(value1, value2); mul(value1, value2); div(value1, value2); divMod(value1, value2); mod(value1, value2); pow(base, power); isEven(value); isOdd(value); isFloat(value); toInteger(value); toIntegerOrInfinity(value);	isPrimitive(value);	toPrimitive(value);
		isObject(value);	toObject(value);
		isPropertyKey(value);	toPropertyKey(value);
		isIndex(value);	toIndex(value);
		isLength(value);	toLength(value);
		typeof(value); constructorOf(value); is(val[, expectedType[, Throw=false]]); isEmpty(value); isTypedCollection(iter, expectedType, Throw=false); isSameType(value1, value2[, type]); isSameInstance(value1, value2, Constructor); isDeepStrictEqual(value1, value2); isCoercedObject(object); isArrayLike(value); toSafeString(value);	
		isNull(value); isUndefined(value); isNullish(value); isPlainObject(value); isProxy(value); isElement(value); isRegexp(value);	
		isFunction(value); isGeneratorFunction(value); isAsyncFunction(value); isAsyncGeneratorFunction(value); isArrowFunction(value);	
		isNonNullable(value);	isNonNullablePrimitive(value);
		isIterator(value);	isAsyncIterator(value);
		isIterable(value);	isAsyncIterable(value);

How to import		
Celestra for Node.js and Deno: <i>celestra.js</i>	Celestra for browser: <i>celestra.js</i>	
<pre>// import the defaultExport object import defaultExport from "./celestra.js"; globalThis.celestra = defaultExport; globalThis.CEL = defaultExport; // import with default with name import { default as celestra } from "./celestra.js"; globalThis.celestra = celestra; globalThis.CEL = celestra; // import all into a new celestra object // includes the DOM API and Cookie API import * as celestra from "./celestra.js"; globalThis.celestra = celestra; globalThis.CEL = celestra; // import some functions import { first, last } from "./celestra.js"; globalThis.first = first; globalThis.last = last;</pre>	<pre><script type="module"> // import all into a new celestra object import * as celestra from "./celestra.js"; globalThis.celestra = celestra; globalThis.CEL = celestra; </script> <script type="module"> // import some functions import { first, last } from "./celestra.js"; globalThis.first = first; globalThis.last = last; </script></pre>	
	Removed APIs in the <i>defaultExport</i>	
	DOM API	Cookie API

Cookie API	Polyfills
<pre>getCookie([name]); hasCookie(name); setCookie(name,value[,hours=8760[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]]); setCookie(Options object: properties are the same as the parameters); removeCookie(name[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]); removeCookie(Options object: properties are the same as the parameters); clearCookies([path="/"[,domain[,sec[,SameSite="Lax"[,HttpOnly]]]]]); clearCookies(Options object: properties are the same as the parameters);</pre>	<pre><u>globalThis.AsyncFunction();</u> <u>globalThis.AsyncGeneratorFunction();</u> <u>globalThis.GeneratorFunction();</u> <u>crypto.randomUUID();</u> <u>Error.isError();</u> <u>globalThis;</u> <u>Math.sumPrecise();</u></pre>

Removed Polyfills - Available in celestra-polyfills.dev.js and celestra-polyfills.min.js				
v3.1.0	v3.8.0	v5.6.0		
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();	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;	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(); <th>v5.9.0</th> BigInt.prototype.toJSON(); <th>v6.5.0</th> Array.fromAsync(); Array.prototype.toReversed(); Array.prototype.toSorted(); Array.prototype.toSpliced(); Array.prototype.with(); Map.groupBy(); and Object.groupBy(); Object.hasOwn(); TypedArray.prototype.toReversed(); TypedArray.prototype.toSorted(); TypedArray.prototype.with();	v5.9.0	v6.5.0