Celestra cheatsheet – v6.0.4 – https://github.com/Serrin/Celestra/

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

Core API		Type API	DOM API
`VERSION;`	`noConflict();`	`type(value);`	`qsa(selector[,context]).forEach(callback);`
`constant(value);`	`asyncConstant(value);`	`is(val[,expectedType[,Throw=false]]);`	`qs(selector[,context]);`
`identity(value);`	`asyncIdentity(value);`	`isSameType(value1,value2);`	`domReady(callback);`
`noop();`	`asyncNoop();`	`isSameInstance(v1,v2,Contructor);`	`domClear(element);`
`T();`	`asyncT();`	`isDeepStrictEqual(value1,value2);`	`domCreate(type[,properties[,innerHTML]]);`
`F();`	`asyncF();`	`isCoercedObject(object);`	`domCreate(element descriptive object);`
		`isEmptyValue(value);`	`domToElement(htmlString);`
`BASE16; BASE32; BASE36; BASE58; BASE62;`		`isNull(value);`	`domGetCSS(element[,property]);`
`WORDSAFEALPHABET;`		`isUndefined(value);`	`domSetCSS(element,property,value);`
`extend([deep,]target,source1[,sourceN]);`		`isNil(value);`	`domSetCSS(element,properties);`
`deleteOwnProperty(obj,prop[,Throw=false]);`		`isNumeric(value);`	`domFadeIn(element[,duration[,display]]);`
`sizeIn(object);`		`isChar(value);`	`domFadeOut(element[,duration]);`
`pick(object,keys);`		`isPlainObject(value);`	`domFadeToggle(elem[,duration[,display]]);`
`omit(object,keys);`		`isFunction(value);`	`domShow(element[,display]);`
`assoc(object,key,value);`		`isCallable(value);`	`domHide(element);`
		`isClass(value);`	`domToggle(element[,display]);`
`delay(milisec).then(callback);`		`isGeneratorFn(value);`	`domIsHidden(element);`
`bind(function,context);`		`isAsyncFn(value);`	`domScrollToTop();` *and* `domScrollToBottom();`
`unBind(function);`		`isAsyncGeneratorFn(value);`	`domScrollToElement(element[,top=true]);`
`curry(function);`		`isProxy(value);`	`domSiblings(element);`
`compose(function1[,functionN]);`		`isElement(value);`	`domSiblingsPrev(element);`
`pipe(function1[,functionN]);`		`isRegexp(value);`	`domSiblingsLeft(element);`
`once(function);`		`isArraylike(value);`	`domSiblingsNext(element);`
`tap(function): function(value);`		`isTypedArray(value);`	`domSiblingsRight(element);`
		`isIterator(value);`	`domGetCSSVar(name);`
`randomBoolean();`		`isIterable(value);`	`domSetCSSVar(name,value);`
`randomUUIDv7();`		`isAsyncIterable(value);`	`importScript(script1[,scriptN]);`
`timestampID([size=21[,alphabet="ABCDEFGHIJ`		`isIndex(value);`	`importStyle(style1[,styleN]);`
`KLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz`		`toIndex(value);`	`setFullscreenOn(selector);`
`0123456789_-"]]);`		`isLength(value);`	`setFullscreenOn(element);`
`nanoid([size=21[,alphabet="123456789ABCDEF`		`toLength(value);`	`setFullscreenOff();`
`GHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxy`		`isPropertyKey(value);`	`getFullscreen();`
`z"]]);`		`toPropertyKey(value);`	`form2array(form);`
`createPolyfillMethod(object,prop,value);`		`isPrimitive(value);`	`form2string(form);`
`createPolyfillProperty(object,prop,value);`		`toPrimitiveValue(value);`	`getDoNotTrack();`
`getUrlVars([str=location.search]);`		`isObject(value);`	`getLocation(success[,error]);`
`obj2string(object);`		`toObject(value);`	`createFile(filename,content[,dType]);`

String API	Assertion API	Math API

b64Decode(string);
 	
b64Encode(string);
 	
strAt(string,index[,newChar]);
 	
strCapitalize(string);
 	
strCodePoints(string);
 	
strDownFirst(string);
 	
strFromCodePoints(iterator);
 	
strHTMLEscape(string);
 	
strHTMLRemoveTags(string);
 	
strHTMLUnEscape(string);
 	
strPropercase(string);
 	
strReverse(string);
 	
strSplice(string,index,count[,add]);
 	
strTitlecase(string);
 	
strTruncate(string);
 	
strUpFirst(string);
	assert(condition[,message|error]);
	assertTrue(condition[,message|error]);
	
	assertFalse(condition[,message|error]);
	
	assertThrows(callback[,message|error]);
	
	assertFail(message|error);
	
	assertEqual(value1,value2[,message|error]);
	assertNotEqual(value1,value2[,message|error]);
	
	assertStrictEqual(value1,value2[,message|error]);
	assertNotStrictEqual(value1,value2[,message|error]);
	
	assertDeepEqual(value1,value2[,message|error]);
	assertNotDeepEqual(value1,value2[,message|error]);
	
	assertDeepStrictEqual(value1,value2[,message|error]);
	assertNotDeepStrictEqual(value1,value2[,message|error]);
	
	assertIs(value,exptectedType[,message|error]);
	assertIsNot(value,exptectedType[,message|error]);
	
	assertIsNil(value[,message|error]);
	assertIsNotNil(value[,message|error]);
	
	assertMatch(string,regexp[,message|error]);
	assertDoesNotMatch(string,regexp[,message|error]);
		sum(value1[,valueN]);
		avg(value1[,valueN]);
		product(value1[,valueN]);
		clamp(value,min,max);
		minmax(value,min,max);
		inRange(value,min,max);
		signbit(value);
		
		randomInt([max]);
		randomInt(min,max);
		randomFloat([max]);
		randomFloat(min,max);
		
		isEven(value);
		isOdd(value);
		isInt8(value);
		isInt16(value);
		isUInt32(value);
		isUInt8(value);
		isUInt16(value);
		isInt32(value);
		isBigInt64(value);
		isBigUInt64(value);
		isFloat16(value);
		isFloat(value);
		
		toInteger(value);
		toIntegerOrInfinity(value);
		toInt8(value);
		toInt16(value);
		toInt32(value);
		toUInt8(value);
		toUInt16(value);
		toUInt32(value;
		toBigInt64(value);
		toBigUInt64(value);
		toFloat16(value);
		toFloat32(value);

Collections API		Polyfills

`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] );`
	`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 = ","]);`
		`Array.fromAsync();`
		
		`Array.prototype.toReversed();`
		
		`Array.prototype.toSorted();`
		
		`Array.prototype.toSpliced();`
		
		`Array.prototype.with();`
		
		`crypto.randomUUID();`
		
		`Error.isError();`
		
		`globalThis;`
		
		`Map.groupBy();`
		
		`Math.sumPrecise();`
		
		`Object.groupBy();`
		
		`Object.hasOwn();`
		
		`TypedArray.prototype.toReversed();`
		
		`TypedArray.prototype.toSorted();`
		
		`TypedArray.prototype.with();`
		
		`globalThis.AsyncFunction();`
		
		`globalthis.AsyncGeneratorFunction();`
		
		`globalThis.GeneratorFunction();`

AJAX and CORS API

```
getText(url,success);

getJson(url,success);

ajax(Options object);
```

Options object properties (* = default value):

Property	Value
url	string
data	string
queryType	*"ajax"/"cors"
type	*"get"/"post"
success	function
error	function
format	*"text"/"json"/"xml"
user	string
password	string

Cookie API

```
getCookie([name]);

hasCookie(name);

setCookie(Options object: properties are the same as the parameters);
setCookie(name,value[,hours=8760[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]]);

removeCookie(Options object: properties are the same as the parameters);
removeCookie(name[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]);

clearCookies(Options object: properties are the same as the parameters);
clearCookies([path="/"[,domain[,sec[,SameSite="Lax"[,HttpOnly]]]]]);
```

Celestra for Node.js and Deno: *celestra.node.mjs*		
How to import	**Removed functions in the celestra.node.mjs**	

`// 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, classof } from "./celestra.node.mjs";`
`globalThis.first = first;`
`globalThis.classof = classof;`
 	
`// dynamic import`
 	
`const celestra = await import("./celestra.node.mjs");`
 	
`globalThis.celestra = celestra;`
`globalThis.CEL = celestra;`
	DOM API
	
	`qsa();`
	`qs();`
	
	`domReady();`
	
	`domClear();`
	
	`domCreate();`
	
	`domToElement();`
	
	`domGetCSS();`
	`domSetCSS();`
	
	`domFadeIn();`
	`domFadeOut();`
	`domFadeToggle();;`
	
	`domShow();`
	`domHide();`
	`domToggle();`
	`domIsHidden();`
	
	`domScrollToTop();`
	`domScrollToBottom();`
	`domScrollToElement();`
	
	`domSiblings();`
	`domSiblingsPrev();`
	`domSiblingsLeft();`
	`domSiblingsNext();`
	`domSiblingsRight();`
		DOM API
		
		`domGetCSSVar();`
		`domSetCSSVar();`
		
		`importScript();`
		`importStyle();`
		
		`setFullscreenOn();`
		`setFullscreenOn();`
		`setFullscreenOff();`
		`getFullscreen();`
		
		`form2array(form);`
		`form2string(form);`
		
		`getDoNotTrack();`
		`getLocation();`
		
		`createFile();`
		
		AJAX and CORS API
		
		`getText();`
		`getJson();`
		`ajax();`
		
		Cookie API
		
		`getCookie();`
		`hasCookie();`
		`setCookie();`
		`removeCookie();`
		`clearCookies();`

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();`

		v5.9.0
		`BigInt.prototype.toJSON();`