Celestra cheatsheet – v7.2.0 – https://github.com/Serrin/Celestra/

Core 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);
```

```
VERSION;

BASE16 = "0123456789ABCDEF";
BASE32 = "234567ABCDEFGHIJKLMNOPQRSTUVWXYZ";
BASE36 =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
BASE58 =
"123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijk
mnopqrstuvwxyz";
BASE62 =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefgh
ijklmnopqrstuvwxyz";
WORDSAFEALPHABET =
"23456789CFGHJMPQRVWXcfghjmpqvwx";
```

```
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]);
```

```
deepAssign(target,source1[,sourceN]);
sizeIn(object);
pick(object,keys);
omit(object,keys);
assoc(object,key,value);
```

```
RandomBoolean();
randomUUIDv7(v4=false);
timestampID([size=21[,alphabet=
"23456789CFGHJMPQRVWXcfghjmpqvw
x"]]);
```

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);
```

DOM API

```
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]);
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]);
```

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);
		isPropertyKey(value); toPropertyKey(value);
		isIndex(value); toIndex(value);
		isIterator(value); isAsyncIterator(value);
		isIterable(value); isAsyncIterable(value);
		typeOf(value);
		constructorOf(value);
		is(value, expectedType);
		isEmpty(value);
		isTypedCollection(iter,expectedType,Throw=false);
		isSameType(value1,value2[,type]);
		isSameInstance(value1,value2,Contructor);
		isDeepStrictEqual(value1,value2);
		toSafeString(value);
		isNull(value);
		isUndefined(value);
		isNullish(value);
		isNonNullable(value);
		isNonNullablePrimitive(value);
		isPrimitive(value);
		isObject(value);
		isPlainObject(value);
		isCoercedObject(object);
		isArraylike(value);
		isTypedArray(value);
		isProxy(value);
		isElement(value);
		isRegexp(value);
		isFunction(value);
		isGeneratorFunction(value);
		isAsyncFunction(value);
		isAsyncGeneratorFunction(value);
		isArrowFunction(value);

How to import

Celestra for Node.js and Deno: *celestra.js*

```
// 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;
```

Celestra for browser: *celestra.js*

```
<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>
```

Removed APIs in the *defaultExport*

DOM API	Cookie API

Cookie API

```
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);
```

Polyfills

```
globalThis.AsyncFunction();
globalthis.AsyncGeneratorFunction();
globalThis.GeneratorFunction();

crypto.randomUUID();

Error.isError();

globalThis;

Math.sumPrecise();
```