Celestra cheatsheet – v4.5.2 – 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

signbit(<value>);
delay(<ms>).then(<callback>);
inherit(<subclass>,<superclass>);
randomInt([max] *or* <min>,<max>);
randomFloat([max] *or* <min>,<max>);
randomBoolean();
randomID([hyphens=false]);
randomString([length[,specChar=false]]);
inRange(<value>,<min>,<max>);
b64Encode(<string>);, b64Decode(<str>);
javaHash(<data>[,hexa=false]);
getUrlVars([str=location.search]);
obj2string(<object>);
getType(<variable>[,type]);
extend([deep,]<target>,<source1>[,srcN]);
deepAssign(<target>,<source1>[,srcN]);
sizeIn(<obj>); *and* forIn(<obj>,<cb>);
filterIn(<obj>,<cb>);, popIn(<obj>,<pr>);
strPropercase(<s>);, strCapitalize(<s>);
strUpFirst(<str>);, strDownFirst(<str>);
strHTMLRemoveTags(<string>);
strHTMLEscape(<string>);
strHTMLUnEscape(<string>);
strReverse(<str>);, strAt(<str>,<index>);
strCodePoints(<string>);
strFromCodePoints(<collection>);
toFunction(<fn>);, bind(<fn>,<context>);
constant(<value>); *and* identity(<value>);
noop(); *and* T(); *and* F();
assertEq(<msg>,<v1>,<v2>[,strict=true]);
assertNotEq(<m>,<v1>,<v2>[,strict=true]);
assertTrue(<msg>,<value>);
assertFalse(<msg>,<value>);
noConflict(); *and* VERSION;
	qsa(<selector>[,context]).forEach(<cb>);
	qs(<selector>[,context]);
	domReady(<callback>);
	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>);
	domSiblings(<element>);
	domSiblingsPrev(<element>);
	domSiblingsLeft(<element>);
	domSiblingsNext(<element>);
	domSiblingsRight(<element>);
	domGetCSSVar(<name>);
	domSetCSSVar(<name>,<value>);
	
	importScript(<script1>[,scriptN]);
	importStyle(<style1>[,styleN]);
	setFullscreenOn(<selector> *or* <element>);
	setFullscreenOff();
	getFullscreen();
	form2array(<form>);
	form2string(<form>);
	getDoNotTrack();
	getLocation(<success>[,error]);
	createFile(<filename>,<content>[,dType]);
		isMap(<value>); *and* isWeakMap(<v>);
		isSet(<value>); *and* isWeakSet(<v>);
		isNumber(<v>); *and* isNumeric(<v>);
		isFloat(<v>); *and* isBigInt(<v>);
		isString(<v>); *and* isChar(<v>);
		isDate(<v>); *and* isError(<v>);
		isRegexp(<v>); *and* isSymbol(<v>);
		isElement(<v>); *and* isObject(<v>);
		isNull(<value>);
		isUndefined(<value>);
		isNullOrUndefined(<value>);
		isNil(<value>);
		isPlainObject(<value>);
		isFunction(<value>);
		isGeneratorFn(<value>);
		isAsyncFn(<value>);
		isDataView(<value>);
		isBoolean(<value>);
		isArraylike(<value>);
		isTypedArray(<value>);
		isArrayBuffer(<value>);
		isPrimitive(<value>);
		isIterator(<v>);, isIterable(<v>);
		isPromise(<value>);
		isEmptyObject(<value>);
		isEmptyArray(<value>);
		isEmptyMap(<value>);
		isEmptySet(<value>);
		isEmptyIterator(<value>);
		isSameObject(<object1>,<object2>);
		isSameArray(<array1>,<array2>);
		isSameMap(<map1>,<map2>);
		isSameSet(<set>,<set2>);
		isSameIterator(<iter1>,<iter2>);

AJAX and CORS

ajax(<Options object>);, getJson(<url>,<success>);, getText(<url>,<success>);
Options object properties (* = default value): url: *string*, data: *string*, queryType: **"ajax"/"cors"*, type: **"get"/"post"*, success: *function*, error: *function*, format: **"text"/"json"/"xml"*, user: *string*, password: *string*

Collections

```
arrayMerge([flat=false,]<target>,<src1>[,srN]);
arrayUnique(<collection>);
arrayAdd(<array>,<value>);
arrayClear(<array>);
arrayRemove(<array>,<value>[,all=false]);
arrayRemoveBy(<array>,<callback>[,all=false]);
arrayRange([start=0[,end=100[,step=1]]]);
arrayCycle(<collection>[,n=100]);
arrayRepeat(<value>[,n=100]);
iterRange([start=0[,step=1[,end=Infinity]]]);
iterCycle(<iter>[,n=Infinity]);
iterRepeat(<value>[,n=Infinity]);
arrayUnion(<collection1>[,collectionN]);
arrayIntersection(<collection1>,<collection2>);
arrayDifference(<collection1>,<collection2>);
arraySymmetricDifference(<collec1>,<collec2>);
setUnion(<collection1>[,collectionN]);
setIntersection(<set1>,<set2>);
setDifference(<set1>,<set2>);
setSymmetricDifference(<set1>,<set2>);
isSuperset(<superCollection>,<subCollection>);
slice(<collection>[,begin=0[,end=Infinity]]);
withOut(<collection>,<filterCollection>);
reduce(<collection>,<callback>[,initialvalue]);
shuffle(<collection>);
take(<collection>[,n=1]);
takeWhile(<collection>,<callback>);
takeRight(<collection>[,n=1]);
takeRightWhile(<collection>,<callback>);
drop(<collection>[,n=1]);
dropWhile(<collection>,<callback>);
dropRight(<collection>[,n=1]);
dropRightWhile(<collection>,<callback>);
forEach(<collection>,<callback>);
map(<collection>,<callback>);
enumerate(<collection>[,offset=0]);
entries(<collection>[,offset=0]);
size(<collection>);
every(<collection>,<callback>);
some(<collection>,<callback>);
none(<collection>,<callback>);
includes(<collection>,<value>);
contains(<collection>,<value>);
find(<collection>,<callback>);
findLast(<collection>,<callback>);
filter(<collection>,<callback>);
reject(<collection>,<callback>);
partition(<collection>,<callback>);
groupBy(<collection>,<callback>);
min(<collection>);
max(<collection>);
sort(<collection>[,numbers=false]);
reverse(<collection>);
zip(<collection1>[,collectionN]);
unzip(<collection>);
zipObj(<collection1>,<collection2>);
item(<collection>,<index>);
nth(<collection>,<index>);
first(<collection>);
head(<collection>);
last(<collection>);
initial(<collection>);
tail(<collection>);
flat(<collection>);
concat(<collection1>[,collectionN]);
join(<collection>[,separator=","]);
```

Polyfills

```
Array.prototype.at();
Array.prototype.findLast();
Array.prototype.findLastIndex();
Array.prototype.flat();
Array.prototype.flatMap();

globalThis;

Object.fromEntries();
Object.hasOwn();

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();
```

Non-standard polyfills

```
BigInt.prototype.toJSON();
window.AsyncFunction();
window.GeneratorFunction();
```

Cookie

```
getCookie([name]);, hasCookie(<name>);,
setCookie(<name>,<value>[,hours=8760[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]]);, setCookie(<Optionsobj>);
removeCookie(<name>[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]);, removeCookie(<Options object>);,
clearCookies([path="/"[,domain[,sec[,SameSite="Lax"[,HttpOnly]]]]]);, clearCookies(<Options object>);
```