Celestra cheatsheet – v3.0.2 – https://github.com/Serrin/Celestra/

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

DOM

```
qsa(<selector>[,context]).forEach(<fn>);
qs(<selector>[,context]).argument;
domReady(<function>);
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>);
domOn(<eventTarget>,<eventType>,<callback>);
domOff(<eventTarget>,<eventType>,<callback>);
domTrigger(<eventTarget>,<eventType>);
domSiblings(<element>);
importScript(<url>[,success]);
importScripts(<scripts> or <script1>[,scN]);
importStyle(<href>[,success]);
importStyles(<styles> or <style1>[,styleN]);
getFullscreen();
setFullscreenOn(<selector> or <element>);
setFullscreenOff();
form2array(<form>);
form2string(<form>);
getDoNotTrack();
getLocation(<success>[,error]);
createFile(<filename>,<content>[,dType]);
```

Core API

```
inherit(<subclass>,<superclass>);
random(<min>,<max> or [max]);
randomString([length[,specChar]]);
b64Encode(<string>);
b64Decode(<string>);
javaHash(<data>[,hexa]);
getUrlVar([name]);
getUrlVarFromString(<querystr>[,name]);
obj2string(<object>);
getType(<variable>[,type]);
extend([deep,]<target>,<source1>[,srcN]);
deepAssign(<target>,<source1>[,srcN]);
forIn(<object>,<callback>);
mapIn(<object>,<callback>);
strRemoveTags(<string>);
strReverse(<string>);
toFunction(<function>);
bind(<function>,<context>);
hasOwn(<object>,<property>);
constant(<value>);
identity(<value>);
noop();
T();
F();
noConflict();
VERSION;
```

Type checking

```
isEqual(<value1>,<value2>); ES5
isString(<v>); and isChar(<v>);
isNumber(<v>); and isNumeric(<v>);
isFloat(<value>);
isBigInt(<value>);
isDate(<value>);
isBoolean(<value>);
isElement(<value>);
isObject(<value>);
isEmptyObject(<value>);
isFunction(<value>);
isArraylike(<value>);
isEmptyArray(<value>);
isTypedArray(<value>);
isArrayBuffer(<value>);
isNull(<v>); and isUndefined(<v>);
isNullOrUndefined(<value>);
isNil(<value>);
isPrimitive(<value>);
isRegexp(<value>);
isSymbol(<value>);
isMap(<v>); and isWeakMap(<v>);
isSet(<v>); and isWeakSet(<v>);
isIterator(<value>);
isIterable(<value>);
isGenerator(<value>);
```

AJAX and CORS

```
ajax(<Options obj.>);, 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*

Cookie

```
setCookie(<name>,<value>[,hours[,path[,domain[,secure[,HttpOnly]]]]]);
getCookie([name]); and hasCookie(<name>);
removeCookie(<name>[,path[,domain[,secure[,HttpOnly]]]]);
clearCookies([path[,domain[,secure[,HttpOnly]]]]);
```

Collections	Polyfills

isSuperset(<superset>,<subset>);
arrayMerge([deep,]<target>,<source1>[,srcN]);
zip(<collection1>[,collectionN]); *and* unzip(<collection>);
uniqueArray(<value>);
uniquePush(<array>,<value>);
arrayClear(<array>);
arrayRemove(<array>,<value>[,all]);
min(<collection>); *and* minIndex(<collection>);
max(<collection>); *and* maxIndex(<collection>);
setUnion(<collection1>[,collectionN]);
setIntersection(<set1>,<set2>);
setDifference(<set1>,<set2>);
setSymmetricDifference(<set1>,<set2>);
arrayUnion(<collection1>[,collectionN]);
arrayIntersection(<collection1>,<collection2>);
arrayDifference(<collection1>,<collection2>);
arraySymmetricDifference(<collection1>,<collection2>);
arrayRange(<start>,<end>[,step]); *and* iterRange([start[,step[,end]]]);
arrayCycle(<collection>[,n]); *and* iterCycle(<iter>[,n]);
arrayRepeat(<value>[,n]);, *and* iterRepeat(<value>[,n]);
sizeOf(<collection>);
item(<collection>,<index>); *and* itemOf(<collection>,<index>);
forEach(<collection>,<callback>); *and* forOf(<collection>,<callback>);
map(<collection>,<callback>); *and* mapOf(<collection>,<callback>);
filterOf(<collection>,<callback>);
hasOf(<collection>,<value>);
findOf(<collection>,<callback>);
everyOf(<collection>,<callback>);
someOf(<collection>,<callback>);
noneOf(<collection>,<callback>);
firstOf(<collection>); *and* lastOf(<collection>);
sliceOf(<collection>[,begin[,end]]);
reverseOf(<collection>);
sortOf(<collection>);
reduceOf(<collection>,<callback>[,initialvalue]);,
concatOf(<collection1>[,collectionN]);
takeOf(<collection>[,n]); *and* takeWhile(<collection>,<callback>);
takeRight(<collection>[,n]); *and* takeRightWhile(<collection>,<callback>);
dropOf(<collection>[,n]); *and* dropWhile(<collection>,<callback>);
dropRight(<collection>[,n]); *and* dropRightWhile(<collection>,<callback>);
	Array.prototype.values();
	Array.prototype.includes();
	Array.prototype.flat();
	Array.prototype.flatMap();
	String.prototype.includes();
	String.prototype.trimStart();
	String.prototype.trimLeft();
	String.prototype.trimEnd();
	String.prototype.trimRight();
	String.prototype.padStart();
	String.prototype.padEnd();
	String.prototype.repeat();
	String.prototype.matchAll();
	Object.assign();
	Object.fromEntries();
	Object.entries();
	Object.values();
	Object.getOwnPropertyDescriptors();
	RegExp.prototype.flags;
	NodeList.prototype.forEach();
	ChildNode.after();
	ChildNode.before();
	ChildNode.remove();
	ChildNode.replaceWith();
	ParentNode.append();
	ParentNode.prepend();
	Element.prototype.matches();
	Element.prototype.closest();
	Element.prototype.toggleAttribute();
	Element.prototype.getAttributeNames();
	window.screenLeft;
	window.screenTop;
	globalThis;
	GeneratorFunction();

DEPRECATED FUNCTIONS

```
getAjax(<url>,<format>,<success>[,error[,user,<password>]]);
postAjax(<url>,<data>,<format>,<success>[,error[,user,<password>]]);
getCors(<url>,<format>,<success>[,error[,user,<password>]]);
postCors(<url>,<data>,<format>,<success>[,error[,user,<password>]]);

isInteger(<value>);
isArray(<value>);

arrayKeys(<collection>);
arrayValues(<collection>);
arrayEntries(<collection>);
```

DEPRECATED POLYFILLS

```
Object.create();

String.prototype.startsWith();, String.prototype.endsWith();, Object.is();, String.fromCodePoint();,
String.prototype.codePointAt();,

Array.from();, Array.of();, Array.prototype.fill();, Array.prototype.find();, Array.prototype.findIndex();,
Array.prototype.copyWithin();

Number.MIN_SAFE_INTEGER;, Number.MAX_SAFE_INTEGER();, Number.EPSILON;, Number.isNaN();, isNaN();, Number.isInteger();,
Number.isFinite();, Number.isSafeInteger();, Number.parseInt();, Number.parseFloat();

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();
```