Celestra cheatsheet – v4.0.0 – 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

delay(<ms>).then(<callback>);
inherit(<subclass>,<superclass>);
randomInt([max] *or* <min>,<max>);
randomFloat([max] *or* <min>,<max>);
randomString([length[,specChar]]);
b64Encode(<string>);
b64Decode(<str>);
javaHash(<data>[,hexa]);
getUrlVars([str=location.search]);
obj2string(<object>);
getType(<variable>[,type]);
extend([deep,]<target>,<source1>[,srcN]);
deepAssign(<target>,<source1>[,srcN]);
forIn(<object>,<callback>);
strRemoveTags(<string>);
strReverse(<string>);
strReplaceAll(<str>,<search>,<replace>);
strCodePoints(<string>);
strFromCodePoints(<collection>);
strAt(<string>,<pos>);
toFunction(<function>);
bind(<function>,<context>);
constant(<value>); *and* identity(<value>);
noop(); *and* T(); *and* F();
assert(<condition>[,message]);
assertLog(<condition>[,message]);
assertAlert(<condition>[,message]);
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(<el>);
	domToggle(<element>[,display]);
	domIsHidden(<element>);
	domSiblings(<element>);
	domGetCSSVar(<name>);
	domSetCSSVar(<name>,<value>);
	importScript(<url>[,success]);
	importScripts(<scripts> *or* <script1>[,scN]);
	importStyle(<href>[,success]);
	importStyles(<styles> *or* <style1>[,styleN]);
	setFullscreenOn(<selector> *or* <element>);
	setFullscreenOff(); *and* getFullscreen();
	form2array(<form>); *and* form2string(<form>);
	getDoNotTrack();
	getLocation(<success>[,error]);
	createFile(<filename>,<content>[,dType]);
		isMap(<v>); *and* isWeakMap(<v>);
		isSet(<v>); *and* isWeakSet(<v>);
		isString(<v>); *and* isChar(<v>);
		isNumber(<v>); *and* isNumeric(<v>);
		isFloat(<v>); and isBigInt(<v>);
		isDate(<v>); *and* isError(<v>);
		isBoolean(<value>);
		isElement(<value>);
		isObject(<value>);
		isEmptyObject(<value>);
		isArraylike(<value>);
		isSameArray(<array1>,<array2>);
		isEmptyArray(<value>);
		isTypedArray(<value>);
		isArrayBuffer(<value>);
		isNull(<value>);
		isUndefined(<value>);
		isNullOrUndefined(<value>);
		isNil(<value>);
		isPrimitive(<value>);
		isRegexp(<value>);
		isSymbol(<value>);
		isIterator(<value>);
		isIterable(<value>);
		isFunction(<value>);
		isGeneratorFn(<value>);
		isAsyncFn(<value>);
		isPromise(<value>);

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*

Cookie

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

Collections	Polyfills

arrayMerge([deep,]<target>,<source1>[,srcN]);
arrayUnique(<collection>); *and* arrayAdd(<array>,<value>);
arrayClear(<array>); *and* arrayRemove(<array>,<value>[,all]);
arrayRange([start=0[,end=100[,step=1]]]); *and* iterRange([start=0[,step=1[,end]]]);
arrayCycle(<collection>[,n]); *and* iterCycle(<iter>[,n]);
arrayRepeat(<value>[,n]); *and* iterRepeat(<value>[,n]);
arrayUnion(<collection1>[,collectionN]);
arrayIntersection(<collection1>,<collection2>);
arrayDifference(<collection1>,<collection2>);
arraySymmetricDifference(<collection1>,<collection2>);
setUnion(<collection1>[,collectionN]);
setIntersection(<set1>,<set2>);
setDifference(<set1>,<set2>);
setSymmetricDifference(<set1>,<set2>);
isSuperset(<superset>,<subset>);
partition(<collection>,<callback>); *and* groupBy(<collection>,<callback>);
zip(<collection1>[,collectionN]); *and* unzip(<collection>);
min(<collection>); *and* max(<collection>);
size(<collection>);
item(<collection>,<index>);
forEach(<collection>,<callback>);
map(<collection>,<callback>);
filter(<collection>,<callback>);
includes(<collection>,<value>);
find(<collection>,<callback>);
every(<collection>,<callback>); *and* some(<collection>,<callback>);
none(<collection>,<callback>);
first(<collection>); *and* last(<collection>);
slice(<collection>[,begin[,end]]);
reverse(<collection>);
sort(<collection>);
reduce(<collection>,<callback>[,initialvalue]);
concat(<collection1>[,collectionN]); *and* flat(<collection>);
enumerate(<collection>);
join(<collection>[,separator=","]);
take(<collection>[,n]); *and* takeWhile(<collection>,<callback>);
takeRight(<collection>[,n]); *and* takeRightWhile(<collection>,<callback>);
drop(<collection>[,n]); *and* dropWhile(<collection>,<callback>);
dropRight(<collection>[,n]); *and* dropRightWhile(<collection>,<callback>);
	Array.prototype.flat();
	Array.prototype.flatMap();
	globalThis;
	Object.fromEntries();
	Object.hasOwn();
	String.prototype.matchAll();
	String.prototype.padStart();
	String.prototype.padEnd();
	String.prototype.replaceAll();
	String.prototype.trimStart();
	String.prototype.trimLeft();
	String.prototype.trimEnd();
	String.prototype.trimRight();
	Non-standard polyfills
	BigInt.prototype.toJSON();
	window.AsyncFunction();
	window.GeneratorFunction();