Celestra cheatsheet – v3.4.1 – 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

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>);
strRemoveTags(<string>);
strReverse(<string>);
strReplaceAll(<str>,<search>,<replace>);
toFunction(<function>);
bind(<function>,<context>);
hasOwn(<object>,<property>);
constant(<value>);
identity(<value>);
noop();
T();
F();
noConflict();
VERSION;
	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>);
	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();
	getFullscreen();
	form2array(<form>); *and* form2string(<form>);
	getDoNotTrack();
	getLocation(<success>[,error]);
	createFile(<filename>,<content>[,dType]);
		isEqual(<value1>,<value2>); **ES5**
		isString(<v>); *and* isChar(<v>);
		isNumber(<value>);
		isNumeric(<value>);
		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(<value>);
		isUndefined(<value>);
		isNullOrUndefined(<value>);
		isNil(<value>);
		isPrimitive(<value>);
		isRegexp(<value>);
		isSymbol(<value>);
		isMap(<value>);
		isWeakMap(<value>);
		isSet(<value>);
		isWeakSet(<value>);
		isIterator(<value>);
		isIterable(<value>);
		isGenerator(<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

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

Collections	Polyfills

isSuperset(<superset>,<subset>);
arrayMerge([deep,]<target>,<source1>[,srcN]);
zip(<collection1>[,collectionN]); *and* unzip(<collection>);
uniqueArray(<value>); *and* uniquePush(<array>,<value>);
arrayClear(<array>); *and* 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>);
forOf(<collection>,<callback>); *and* forEach(<collection>,<callback>);
mapOf(<collection>,<callback>); *and* map(<collection>,<callback>);
filterOf(<collection>,<callback>);
hasOf(<collection>,<value>);
findOf(<collection>,<callback>);
everyOf(<collection>,<callback>); *and* 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]); *and* flatOf(<collection>);
enumerateOf(<collection>);
joinOf(<collection>[,separator]);
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();
	String.prototype[Symbol.iterator]();
	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;
	BigInt.prototype.toJSON();
	GeneratorFunction();