Celestra cheatsheet – v5.7.3 – https://github.com/Serrin/Celestra/

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

Core API	Type API	DOM API

noConflict();
VERSION;
	type(value);
	isSameType(value1,value2);
	isSameClass(value1,value2);
	isSameInstance(v1,v2,Contructor);
	isNull(value);
	isUndefined(value);
	isNullOrUndefined(value);
	isNil(value);
	isPrimitive(value);
	isNumeric(value);
	isChar(value);
	isFunction(value);
	isCallable(value);
	isConstructorFn(value);
	isClass(value);
	isGeneratorFn(value);
	isAsyncFn(value);
	isAsyncGeneratorFn(value);
	isObject(value);
	isPlainObject(value)
	isProxy(value);
	isElement(value);
	isRegexp(value);
	isArraylike(value);
	isTypedArray(value);
	isIterator(value);
	isIterable(value);
	isCoercedObject(object);
	isDeepStrictEqual(value1,value2);
	isEmptyValue(value);
	isIndex(value);
	toIndex(value);
	isLength(value);
	toLength(value);
	isPropertyKey(value);
	toPropertyKey(value);
	toPrimitiveValue(value);
		qsa(selector[,context]).forEach(callback);
		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(); *and* 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]);

BASE16; BASE32; BASE36; BASE58; BASE62; WORDSAFEALPHABET;		

javaHash(data[,hexadecimal = false]);
b64Encode(string);
b64Decode(string);
extend([deep,]target,source1[,sourceN]);
sizeIn(object);
popIn(object,property);
forIn(object,callback);
filterIn(object,callback);
delay(milisec).then(callback);
sleep(milisec).then(callback);
createPolyfillMethod(object,prop,value);
createPolyfillProperty(object,prop,value);
deletePropertyOrThrow(object,property);
randomBoolean();
randomUUIDv7();
timestampID([size=21[,alphabet="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789_-"]]);
nanoid([size=21[,alphabet="123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz"]]);
getUrlVars([str=location.search]);
obj2string(object);
classof(value[,class[,throw=false]]);
getType(value[,class[,throw=false]]);
toObject(value);
bind(function,context);
unBind(function);
constant(value);
identity(value);
noop();
T(); *and* F();		

String API	Assertion API	Math API

strPropercase(string);
 	
strTitlecase(string);
 	
strCapitalize(string);
 	
strTruncate(string);
 	
strUpFirst(string);
 	
strDownFirst(string);
 	
strReverse(string);
 	
strCodePoints(string);
 	
strFromCodePoints(iterator);
 	
strAt(string,index[,newChar]);
 	
strSplice(string,index,count[,add]);
 	
strHTMLRemoveTags(string);
 	
strHTMLEscape(string);
 	
strHTMLUnEscape(string);
	assert(condition[,message]);
	assertTrue(condition[,message]);
	
	assertFalse(condition[,message]);
	
	assertEqual(value1,value2[,message]);
	
	assertNotEqual(value1,value2[,message]);
	
	assertStrictEqual(value1,value2[,message]);
	
	assertNotStrictEqual(value1,value2[,message]);
	
	assertDeepEqual(value1,value2[,message]);
	
	assertNotDeepEqual(value1,value2[,message]);
	
	assertDeepStrictEqual(value1,value2[,message]);
	
	assertNotDeepStrictEqual(value1,value2[,message]);
	
	assertType(value,typeString[,message]);
	assertType(value,constructor[,message]);
	
	assertNotType(value,typeString[,message]);
	assertNotType(value,constructor[,message]);
		sum(value1[,valueN]);
		avg(value1[,valueN]);
		product(value1[,valueN]);
		clamp(value,min,max);
		minmax(value,min,max);
		inRange(value,min,max);
		signbit(value);
		
		randomInt([max]);
		randomInt(min,max);
		randomFloat([max]);
		randomFloat(min,max);
		
		isEven(value);
		isOdd(value);
		isInt8(value);
		isInt16(value);
		isUInt32(value);
		isUInt8(value);
		isUInt16(value);
		isInt32(value);
		isBigInt64(value);
		isBigUInt64(value);
		isFloat16(value);
		isFloat(value);
		
		toInteger(value);
		toIntegerOrInfinity(value);
		toInt8(value);
		toInt16(value);
		toInt32(value);
		toUInt8(value);
		toUInt16(value);
		toUInt32(value;
		toBigInt64(value);
		toBigUInt64(value);
		toFloat16(value);
		toFloat32(value);

Collections API		Polyfills

arrayCreate([length = 0]);
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]]]);
iterRange([start=0[,step=1[,end=Infinity]]]);
arrayCycle(iterator[,n = 100]);
iterCycle(iterator[,n = Infinity]);
arrayRepeat(value[,n = 100]);
iterRepeat(value[,n = Infinity]);
unique(iterator[,resolver]);
slice(iterator[,begin=0[,end = Infinity]]);
withOut(iterator,filterIterator);
reduce(iterator,callback[,initialvalue]);
count(iterator,callback);
 	
take(iterator[,n = 1]);
takeWhile(iterator,callback);
takeRight(iterator[,n = 1]);
takeRightWhile(iterator,callback);
drop(iterator[,n = 1]);
dropWhile(iterator,callback);
dropRight(iterator[,n = 1]);
dropRightWhile(iterator,callback);
 	
isSuperset();
arrayDifference();
arrayIntersection();
arraySymmetricDifference();
arrayUnion();
setDifference();
setIntersection();
setSymmetricDifference();
setUnion();
	forEach(iterator,callback);
	map(iterator,callback);
	enumerate(iterator[,offset = 0]);
	entries(iterator[,offset = 0]);
	size(iterator);
	
	every(iterator,callback);
	some(iterator,callback);
	none(iterator,callback);
	includes(iterator,value);
	contains(iterator,value);
	find(iterator,callback);
	findLast(iterator,callback);
	filter(iterator,callback);
	reject(iterator,callback);
	partition(iterator,callback);
	
	zip(iterator1[,iteratorN]);
	unzip(iterator);
	zipObj(iterator1,iterator1);
	shuffle(iterator);
	
	min(value1[,valueN]);
	max(value1[,valueN]);
	sort(iterator[,numbers = false]);
	reverse(iterator);
	
	item(iterator,index);
	nth(iterator,index);
	first(iterator);
	head(iterator);
	last(iterator);
	initial(iterator);
	tail(iterator);
	
	flat(iterator);
	concat(iterator1[,iteratorN]);
	join(iterator[,separator = ","]);
		Array.fromAsync();
		
		Array.prototype.toReversed();
		
		Array.prototype.toSorted();
		
		Array.prototype.toSpliced();
		
		Array.prototype.with();
		
		crypto.randomUUID();
		
		Error.isError();
		
		globalThis;
		
		Map.groupBy();
		
		Math.sumPrecise();
		
		Object.groupBy();
		Object.hasOwn();
		
		TypedArray.prototype.toReversed();
		
		TypedArray.prototype.toSorted();
		
		TypedArray.prototype.with();
		
		Non-standard polyfills
		
		BigInt.prototype.toJSON();
		
		window.AsyncFunction();
		window.GeneratorFunction();

AJAX and CORS API

```
getText(url,success);

getJson(url,success);

ajax(Options object);
```

Options object properties (* = default value):

Property	Value
url	string
data	string
queryType	*"ajax"/"cors"
type	*"get"/"post"
success	function
error	function
format	*"text"/"json"/"xml"
user	string
password	string

Cookie API

```
getCookie([name]);

hasCookie(name);

setCookie(Options object: properties are the same as the parameters);
setCookie(name,value[,hours=8760[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]]);

removeCookie(Options object);
removeCookie(name[,path="/"[,domain[,secure[,SameSite="Lax"[,HttpOnly]]]]]);

clearCookies(Options object: properties are the same as the parameters);
clearCookies([path="/"[,domain[,sec[,SameSite="Lax"[,HttpOnly]]]]]);
```

Removed Polyfills – Available in celestra-polyfills.dev.js and celestra-polyfills.min.js		
V3.1.0	**v3.8.0**	**v5.6.0**
Array.from();		Array.prototype.at();
Array.of();	Array.prototype.values();	
Array.prototype.copyWithin();	Array.prototype.includes();	Array.prototype.findLast();
Array.prototype.fill();		Array.prototype.findLastIndex();
Array.prototype.find();	ChildNode.after();	
Array.prototype.findIndex();	ChildNode.before();	Array.prototype.flat();
Object.create();	ChildNode.remove();	Array.prototype.flatMap();
String.fromCodePoint();	ChildNode.replaceWith();	
String.prototype.codePointAt();		Number.MIN_SAFE_INTEGER;
String.prototype.endsWith();	Element.prototype.closest();	Number.MAX_SAFE_INTEGER;
String.prototype.startsWith();	Element.prototype.getAttributeNames();	
Math.acosh();	Element.prototype.matches();	Object.fromEntries();
Math.asinh();	Element.prototype.toggleAttribute();	
Math.atanh();		Object.is();
Math.cbrt();	ParentNode.append();	
Math.clz32();		String.prototype.at();
Math.cosh();	ParentNode.prepend();	
Math.expm1();		String.prototype.matchAll();
Math.fround();	String.prototype[Symbol.iterator]();	
Math.hypot();	String.prototype.includes();	String.prototype.padStart();
Math.imul();	String.prototype.repeat();	String.prototype.padEnd();
Math.log1p();		
Math.log10();	NodeList.prototype.forEach();	String.prototype.replaceAll();
Math.log2();		
Math.sign();	Object.assign();	String.prototype.trimStart();
Math.sinh();	Object.entries();	String.prototype.trimLeft();
Math.tanh();		
Math.trunc();	Object.getOwnPropertyDescriptors();	String.prototype.trimEnd();
Number.EPSILON;	Object.values();	String.prototype.trimRight();
Number.isNaN(); *and* isNaN();		
Number.isInteger();	RegExp.prototype.flags;	Typedarray.prototype.at();
Number.isFinite();		
Number.isSafeInteger();	window.screenLeft;	TypedArray.prototype.findLast();
Number.parseInt();	window.screenTop;	TypedArray.prototype.findLastIndex();
Number.parseFloat();		