Celestra cheatsheet – v5.4.3 – https://github.com/Serrin/Celestra/

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

Suphit(value); qsa(selector)[,context]].forEach(<ch); and="" delay(<ms).then(<callback);="" dom(*callba<="" dom(*callback);="" isste(value);="" iswarkmap(<v);="" qsa(selector)[,context]].foreach(<ch);="" sleep(<ms).then(<callback);="" th="" =""><th></th><th>ects contain these functions, except the polyt</th><th></th></ch);>		ects contain these functions, except the polyt	
<pre>delay(msp).then(<allback>); sleep(msp).then(<allback>); inherit(<subclass>, <superclass>); andomFrie((max) or <min>, <max>); randomFrie((max) or <min>, <max>, <max>); domFrie((max) or <min>, <max>, <m< th=""><th>Core API</th><th>DOM</th><th>Type checking</th></m<></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></max></min></max></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></superclass></subclass></allback></allback></pre>	Core API	DOM	Type checking
<pre>sleep(cms>).then(<calback>); inheric(xsubclass>, cauperclass>); candomInt([max] or <min>, <max>); candomInt([max] or <min>, <max>); candomBolat([max] or <min>, <max>); candomBolat([max] or <min>, <max>); candomBolat([max] or <min>, <max>); candomBolean(); randomBolean(); randomDI([hyphens-true][,usedate=false]); randomDI([hyphens-true][,usedate=false]); randomDI([hyphens-true],secthar=false]); inRange(xvalue>, <min>, <max>; bdAffacode(xstring>); bdAffacode(xstring>); bdAffacode(xstring>); javaHash(<data>[,hexa=false]); gdublavas([str=location.search]); obj2string(<object>); classof(<variable>[,type], throw=false]); rotn(<object>); forIn(<object>, <calback>); domSetUSS(selement>, classlay); gdublavas([deep,] <talback); popin(sobject="">, <calback>); domFadeTogele(<element>[,duration[,display]]); domSiblingsPrev(<element>); domSiblingsPrev(<element>); strPitlecase(string>); strPitlec</element></element></element></calback></talback);></calback></object></object></variable></object></data></max></min></max></min></max></min></max></min></max></min></max></min></calback></pre>	signbit(<value>);</value>	<pre>qsa(<selector>[,context]).forEach(<cb>);</cb></selector></pre>	<pre>isMap(<value>); and isWeakMap(<v>);</v></value></pre>
<pre>inherit(<subclass>,<superclass>); randomfInd([max] or <min>,<max>); randomFlost([max] or <min>,<max>); randomFlost([max] or <min>,<max>); randomFlost([max] or <min>,<max>); randomBlost([max] or <min>,<max>); randomBlost([max] or <min>,<max>); randomBlost([max] or <min>,<max>); randomBlost([max] or <min>,<max>); randomBlost([myhens=true][,usedate=false]); domCreate(<telement descriptive="" object="">); domGreate(<telement descriptive="" descriptive);="" domg<="" domgreate(<telement="" object);="" td=""><td><pre>delay(<ms>).then(<callback>);</callback></ms></pre></td><td><pre>qs(<selector>[,context]);</selector></pre></td><td><pre>isSet(<value>); and isWeakSet(<v>);</v></value></pre></td></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></telement></max></min></max></min></max></min></max></min></max></min></max></min></max></min></max></min></superclass></subclass></pre>	<pre>delay(<ms>).then(<callback>);</callback></ms></pre>	<pre>qs(<selector>[,context]);</selector></pre>	<pre>isSet(<value>); and isWeakSet(<v>);</v></value></pre>
<pre>randomInt([max] or <min>, max>); randomInt([max] or <min>, max>); randomInt([max] or <min>, max>); randomInt([max] or <min>, max>); randomInt([max] or <min>, max>); domCreate(<element descriptive="" object="">); isString(</element></min></min></min></min></min></pre> isString(</td <td><pre>sleep(<ms>).then(<callback>);</callback></ms></pre></td> <td></td> <td><pre>isNumber(<v>); and isNumeric(<v>);</v></v></pre></td>	<pre>sleep(<ms>).then(<callback>);</callback></ms></pre>		<pre>isNumber(<v>); and isNumeric(<v>);</v></v></pre>
<pre>candomFloat([max] or xmin>,<max>); randomBolean(); randomBolean(); randomBolean(); randomBol([hyphens=true][,usedate=false]); randomBol([hyphens=true][,usedate=hol(,usedate)]; randomBol([hyphens=true][,usedate=hol(,usedate=hol(,usedate)]; randomBol([hyphens=true][,usedate=hol(,usedate=hol(,usedate)]; randomBol([hyphens=true][,usedate=hol(,usedate)]; randomBol([hyphens=true][,usedate=hol(,usedate]]; randomBol([hyphens=true][,usedate=hol(,usedate]]; randomBol([hyphens=true][,usedate=hol(,usedate]]; randomBol([hyphens=true][,usedate]]; randomBol([hyphens=true][,usedate]]; randomBol([hyphens=true][,usedate]]; randomBol([hyphens=true][,usedate]]; randomBol([hyph</max></pre>	<pre>inherit(<subclass>,<superclass>);</superclass></subclass></pre>	<pre>domReady(<callback>);</callback></pre>	<pre>isFloat(<val>); and isBigInt(<v>);</v></val></pre>
<pre>randomBoolean(); randomBoolean(); randomDof[hyphens=true][, usedate=false]]; randomDof[hyphens=true][, usedate=false]]; randomString([length[, specchar=false]]); inRange(<value>, dmin>, <max>); b64Encode(<string>); javaBash(<data>[, hexa=false]); gatUrslVars([str=location.search]); obj2string(object>); classof(<variable>[, type(, throw=false]]); sizeIn(<object>); classof(<variable>[, type(, throw=false]]); sizeIn(<object>); sizeIn(<object>); sizeIn(<object>, (sallback>); forn(\cobject>, <callback>); forn(\cobject>, <callback>); forn(\cobject>, <callback>); strPropercase(<string>); strPropercase(<string>); strPropercase(<string>); strPropercase(<string>); strPropercase(<string>); strPrint(RemoveTags(<string>); strHTMLEscape(<string>); strHTMLEscape(<string>); strRevrese(<string>); strRevrese(<string< td=""><td><pre>randomInt([max] or <min>,<max>);</max></min></pre></td><td><pre>domCreate(<type>[,properties[,innerHTML]]);</type></pre></td><td><pre>isString(<v>); and isChar(<val>);</val></v></pre></td></string<></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></callback></callback></callback></object></object></object></variable></object></variable></data></string></max></value></pre>	<pre>randomInt([max] or <min>,<max>);</max></min></pre>	<pre>domCreate(<type>[,properties[,innerHTML]]);</type></pre>	<pre>isString(<v>); and isChar(<val>);</val></v></pre>
<pre>randomID([hyphens=true][,usedate=false]); randomString([length[,specChar=false]); inRange(value>, xain>, xmax); b64Encode(varing>); b64Decode(string>); domSetCSS(velement>, cproperty); isBoalview(value>); isBalviel'salve>); isBalv</pre>	<pre>randomFloat([max] or <min>,<max>);</max></min></pre>	<pre>domCreate(<element descriptive="" object="">);</element></pre>	<pre>isDate(<val>); and isError(<val>);</val></val></pre>
<pre>randomString([length, specchar=false]); inRange(<value>, <min>, <max>); b64Encode(<string>); b64Decode(<string>); javaHash(<data>[, hexa=false]); gomFadeIn(<element>, <pre>, property>, <pre>value>); javaHash(<data>[, hexa=false]); gomFadeIn(<element>, duration[, display]]); domFadeOut(<element>, duration[, display]]); domFadeOut(<element>, duration[); domFadeOut(<element>, duration[); domFadeOut(<element>, display]); domFadeOut(<element>, display]); domFadeOut(<element>, display]); domFadeOut(<element>, display]); domFadeOut(<element>, display]); domFadeOut(<element>); domShadeToggle (<element, (<element="" display]);="" domshadetoggle="">, display]); domShadeToggle (<element>, display]); domShadeToggle (<element>, display]); domShadeToggle (<element>, display]); domShadeOut(<element>); domShadeOut(<element>); domShadeOut(<element>); domShadeToggle (<element>, display]); domShadeToggle (<element>, display]); domShadeToggle (<element>, display]); domShadeToggle (<element>, display]); domShadeOut(<element>, display])</element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element,></element></element></element></element></element></element></element></element></element></element></data></pre></pre></element></data></string></string></max></min></value></pre>	<pre>randomBoolean();</pre>	<pre>domToElement(<htmlstring>);</htmlstring></pre>	<pre>isRegexp(<v>); and isSymbol(<v>);</v></v></pre>
<pre>inRange(<value>,<min>,<max>); b64Encode(<string>); b64Decode(<string>); b64Decode(<string>); javaiash(<data>[,hexa=false]); domFadeIn(<element>,<pre>(domFadeIn(<element)]; isundefined(<value="">); isUndefined(<value>); isUnlefined(<value>); isTrub(<value>); isTrub(<value>); isTrub(<value>); isTrub(<value>); isTrub(<value>); isArray(<xalue>); isPrimitive(<value>); isTrub</value></xalue></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></element)];></pre></element></data></string></string></string></max></min></value></pre>	<pre>randomID([hyphens=true][,usedate=false]);</pre>		<pre>isElement(<v>); and isObject(<v>);</v></v></pre>
b64Encode(<string>); b64Decode(<string>); domPadeIn(<element>[,duration[,display]]); domPadeOut(<element>[,duration[,display]]); domPadeOut(<element>[,duration[,duration[,display]]); domPadeOut(<element>[,duration[,duratio</element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></string></string>	<pre>randomString([length[,specChar=false]]);</pre>	<pre>domSetCSS(<element>,<pre>,<value>);</value></pre></element></pre>	<pre>isDataView(<value>);</value></pre>
<pre>javaHash(</pre> <pre>javaHash(</pre> <pre>javaHash(</pre> <pre>controlly loars([str=location.search]); domFadeToggle(<pre>clem.>[, duration]); domFadeToggle(<pre>clem.>[, duration]); domFadeToggle(<pre>clem.>[, duration]]); domFadeToggle(<pre>clem.>[, duration]]]); domFadeToggle(<pre>cle</pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>	<pre>inRange(<value>,<min>,<max>);</max></min></value></pre>	<pre>domSetCSS(<element>,<pre>,<pre>,</pre>;</pre></element></pre>	<pre>isBoolean(<value>);</value></pre>
<pre>getUrlVars([str=location.search]); obj2string(<object>); obj2string(<object); object="">, starget>, st</object);></object></object></object></object></object></object></object></object></object></object></object></object></pre>	<pre>b64Encode(<string>); b64Decode(<string>);</string></string></pre>	<pre>domFadeIn(<element>[,duration[,display]]);</element></pre>	isNull(<value>);</value>
<pre>obj2string(<object>); classof(<variable>[,type[,throw=false]]); classof(<variable>[,type[,throw=false]]); domShow(<element>[,display]); domHide(<element>); domHide(<element>); domToggle <clement>[,display]); domToggle <clement>[,display]); domToggle <clement>[,display]); domToggle <clement>[,display]); domToggle <clement>[,display]); domToggle <clement>[,display]]; domSide(<element>]; isFlain(object)</element></clement></clement></clement></clement></clement></clement></element></element></element></variable></variable></object></pre> isFuthy(<alue>); isFalsy(<alue>); isGeneratorFn(<alue>); isGeneratorFn(<alue>); isAsyncEn(<alue>); isAsyncEn(<alue>); isAsyncEn(<alue>); isAsyncEn(<alue>); isArraylike(<alue>); is</alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue>	<pre>javaHash(<data>[,hexa=false]);</data></pre>	<pre>domFadeOut(<element>[,duration]);</element></pre>	<pre>isUndefined(<value>);</value></pre>
<pre>classof(<variable>[,type[,throw=false]]); extend([deep,]<target>>,<sourcel>[,srcN]); sizeIn(<object>); forIn(<object>>,<callback>); forIn(<object>>,<callback>); filterIn(<object>,<callback>); forSpoint(object>,<callback>); filterIn(<object>,<callback>); forSpoint(object>,<callback>); forSpoint(object),<callback>); forSpoint(object),<callback>); forSpoint(object),<callback); forspoint(object),<callback="">); forSpoint(object),<callback>); forSpoint(object),<callback>); forSpoint(object),<callback); forspoint(object),<callback);="" forspoint(object),<callbackelement="">);</callback);></callback></callback></callback);></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></callback></object></callback></callback></object></callback></object></callback></object></object></sourcel></target></variable></pre>	<pre>getUrlVars([str=location.search]);</pre>	<pre>domFadeToggle(<elem.>[,duration[,display]]);</elem.></pre>	<pre>isNullOrUndefined(<value>);</value></pre>
<pre>extend([deep,]<target>,<sourcel>[,srcN]); sizeIn(<object>); forIn(<object>); forIn(<object>,<callback>); forIn(<object>,<callback>); forIn(<object>,<callback>); forIn(<object>,<callback>); domSiblings(<element>); domSiblingsPerv(<element>); sisGeneratorFn(<value>); sisGeneratorFn(<value>); sisAsyncGeneratorFn(<value>); strPropercase(<string>); strTitlecase(<string>); strCapitalize(<string>); strUpFirst(<str>); strUpFirst(<str>); strHTMLERcape(<string>); strHTMLUREscape(<string>); strHTMLUREscape(<string>); strHTMLUREscape(<string>); strHTMLUREscape(<string>); strReverse(<string>); strReverse(<string>); strFornocdePoints(<collection>); strStyFornocdePoints(<collection>); strFornocdePoints(<collection>); sosentant(<value>); somof(); and F(); sasertEq(<msg>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); sasertTrue(<msg>, <value>); ssemeMap(<malp>, <malp>, <malp>, <malpolic form="">); ssemeMap(<malp>, <malp>, <malp>, <malpolic form="">); ssemeMap(<malp>, <malp>, <malpolic form="">); ssemeMap(<malp>, <malpolic form="">); ssemeMap(<malp>, <malp>, <malpolic form="">); ssemeMap(<malp>, <malpolic form="">); ssemeMap(<malpolic form="">); ssemeMap(<malpolic form="">); ssemeMap(<malpolic form="">); ssemeMap(<malpolic form="">); ssemeMap(<malpolic form="">); ssameApray(<arrayla, <airayl<="" <airayla,="" <arrayla,="" td=""><td>obj2string(<object>);</object></td><td><pre>domShow(<element>[,display]);</element></pre></td><td>isNil(<value>);</value></td></arrayla,></malpolic></malpolic></malpolic></malpolic></malpolic></malpolic></malp></malpolic></malp></malpolic></malp></malpolic></malp></malpolic></malp></malpolic></malp></malpolic></malp></malpolic></malp></malp></malpolic></malp></malpolic></malp></malp></malpolic></malp></malp></malp></malpolic></malp></malp></malp></value></msg></value></msg></v2></v1></msg></value></collection></collection></collection></string></string></string></string></string></string></string></str></str></string></string></string></value></value></value></element></element></callback></object></callback></object></callback></object></callback></object></object></object></sourcel></target></pre>	obj2string(<object>);</object>	<pre>domShow(<element>[,display]);</element></pre>	isNil(<value>);</value>
<pre>sizeIn(<object>); forIn(<object>,<callback>); forIn(<object>,<callback>); forIn(<object>,<callback>); forIn(<object>,<callback>); popIn(<object>,<pre>comparty>); domSiblings(<element>); domSiblingsPrev(<element>); isConstructorFn(<value>); stepnerase(<string>); strProperase(<string>); strCapitalize(<string>); strUpFirst(<str); and="" strdownfirst(<s="">); strHTMLRemoveTags(<string>); strHTMLEscape(<string>); strHTMLEscape(<string>); strReverse(<string>); strReverse(<string>); strReverse(<string>); strRodePoints(<string>); strCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); constant(<value>); and entity(<value>); sasertEq(<msg>,<value>); assertTrue(<msg>,<value>); getDoNotTrack(); assertTrue(<msg>,<value>); assertTrue(<msg>,<value>); getDoNotTrack(); getDoNotTrack(); getDoLore index(index) is is function(<vx) constructor="" fn(<value="" function(<vx)="" is="">); is domSiblingsReve(<element>); is domSiblingsReve(element>); is Arraylike(<value>); is Primtive(<value>); is Furctio</value></value></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></vx)></value></msg></value></msg></value></msg></value></msg></value></value></fn></context></fn></collection></string></string></string></string></string></string></string></str);></string></string></string></value></element></element></pre></object></callback></object></callback></object></callback></object></callback></object></object></pre>	<pre>classof(<variable>[,type[,throw=false]]);</variable></pre>	<pre>domHide(<element>);</element></pre>	<pre>isPlainObject(<value>);</value></pre>
<pre>forIn(<object>,<callback>); filterIn(<object>,<callback>); filterIn(<object>,<callback>); forIn(<object>,<callback>); domSiblings(<element>); domSiblingsPrev(<element>); strPropercase(<string>); strPropercase(<string>); domSiblingsNext(<element>); domSiblingsNext(element>); do</element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></element></string></string></element></element></callback></object></callback></object></callback></object></callback></object></pre>	<pre>extend([deep,]<target>,<source1>[,srcN]);</source1></target></pre>	<pre>domToggle(<element>[,display]);</element></pre>	<pre>isTruthy(<value>);</value></pre>
<pre>filterIn(\(\)object>,\(\)callback>\); popIn(\(\)object>,\(\)cpoperty>\); strPropercase(\(\)string>\); strPropercase(\(\)string>\); strPropercase(\(\)string>\); strCapitalize(\(\)string>\); strDpFirst(\(\)str); strDpFirst(\(\)str); strHTMLEscape(\(\)string>\); strHTMLEscape(\(\)string>\); strHTMLEscape(\(\)string>\); strReverse(\(\)string>\); strReverse(\(\)string>\); strCopinits(\(\)string>\); strCopinits(\(\)string>\); strBopinits(\(\)string>\); strHTMLInescape(\(\)string>\); strReverse(\(\)string>\); strReverse(\(\)string>\); strReverse(\(\)string>\); strReverse(\(\)string>\); strCodePoints(\(\)string>\); strCodePoints(\(\)string>\); strFomCodePoints(\(\)string>\); sonstant(\(\)string>\); sonstant(\(\)string>\); sasertEq(\(\)string>\); sasertEq(\(\)sg>,\(\)strict=true]); assertTrue(\(\)sg>,\(\)strict=true]); assertTrue(\(\)sg>,\(\)strict=true]); assertTralse(\(\)sg>,\(\)strict=true]); assertTralse(\(\)strict=true]); assertTralse(\(\)strict=true]</pre>	sizeIn(<object>);</object>	<pre>domIsHidden(<element>);</element></pre>	<pre>isFalsy(<value>);</value></pre>
<pre>popIn(<object>,<property>); strPropercase(<string>); strPropercase(<string>); strTitlecase(<string>); strCapitalize(<string>); strUpFirst(<str>); strUpFirst(<str>); strHTMLRemoveTags(<string>); strHTMLEscape(<string>); strHTMLEscape(<string>); strHTMLUnEscape(<string>); strFaverse(<string>); strReverse(<string>); strReverse(<string>); strRopFirst(<str>); strPopCodePoints(<string>); strCapitalize(<string>); strHTMLUnescape(<string>); strHTMLUnescape(<string>); strHTMLUnescape(<string>); strReverse(<string>); strReverse(<string>); strReverse(<string>); strReverse(<string>); strCodePoints(<string>); strCodePoints(<string>); strSupFomCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msy>,<val>,<val>,<val>,<val>,<val>,<iand collection="">,<iand collection="">,</iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></iand></val></val></val></val></val></msy></value></value></fn></context></fn></collection></string></string></string></string></string></string></string></string></string></string></string></str></string></string></string></string></string></string></string></str></str></string></string></string></string></property></object></pre>	<pre>forIn(<object>, <callback>);</callback></object></pre>		<pre>isFunction(<v>); + isCallable(<v>);</v></v></pre>
<pre>strPropercase(<string>); strTitlecase(<string>); strOpitalize(<string>); strOpitalize(<string>); strUpFirst(<str); and="" strdownfirst(<s);="" strhtmlremovetags(<string="">); strHTMLEscape(<string>); strHTMLEscape(<string>); strHTMLEscape(<string>); strHTMLEscape(<string>); strRTMLUnescape(<string>); strRTMLOpitalize(<string>); strRTMLOpitalize(<string>); strHTMLIscape(<string>); strHTMLIscape(<string>); strRTMLOpitalize(<string>); strRTMLOpitalize(<string>); strRTMLOpitalize(<string>); strRTMLOpitalize(<string>); strRt(<string>); strRt(<string>); strRt(<string>); strRt(<string>); strRt(<string>); strRt(<string>); strRt(<string>); strCodePoints(<string>); strCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertTopitalize(<string>); ssertItue(<msg>,<value>); assertTrue(<msg>,<value>); getDoNotTrack(); getDoNotTrack(); getDoNotTrack(); getDoNotTrack(); getDoNotTrack(); getDoNotTrack(); isSameSet(<set1>,<set2>);</set2></set1></value></msg></value></msg></string></value></value></fn></context></fn></collection></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></string></str);></string></string></string></string></pre>	<pre>filterIn(<object>, <callback>);</callback></object></pre>	<pre>domSiblings(<element>);</element></pre>	<pre>isConstructorFn(<value>);</value></pre>
<pre>strTitlecase(<string>); strCapitalize(<string>); strUpFirst(<string>); strUpFirst(<string>); strUpFirst(<str); and="" strdownfirst(<s="">); strHTMLRemoveTags(<string>); strHTMLEscape(<string>); strHTMLEscape(<string>); strHTMLUnEscape(<string>); strReverse(<string>); strReverse(<string>); strReverse(<string>); strPomCodePoints(<collection>); bind(<fn>, <context>); and unBind(<fn>); constant(<value>); and identity(<value>); sasertTue(<msg>, <value>); assertTue(<msg>, <value>); setDoNotTrack(); assertFalse(<msg>, <value>); getDoNotTrack(); getLocation(<success>[,error]); isAsyncFn(<value>); isArraylike(<value>); isArraylike(<value>); isTypedArray(<value>); isArraylike(<value>); isTypedArray(<value>); isArrayBuffer(<value>); isPromise(<value>); isPromise(<value>); isPromise(<value>); isIterator(<value>); isIterator(<value>); isEmptyObject(<value>); isEmptyObject(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptySet(<value>); isEmptyIterator(<value>); isSameArray(<arrayl>, <arrayl>, <arrayly, <ar<="" td=""><td><pre>popIn(<object>,<pre>,<pre>,</pre>;</pre></object></pre></td><td></td><td>isGeneratorFn(<value>);</value></td></arrayly,></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></arrayl></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></value></success></value></msg></value></msg></value></msg></value></value></fn></context></fn></collection></string></string></string></string></string></string></string></str);></string></string></string></string></pre>	<pre>popIn(<object>,<pre>,<pre>,</pre>;</pre></object></pre>		isGeneratorFn(<value>);</value>
<pre>strCapitalize(<string>); strUpFirst(<str>); and strDownFirst(<s>); strHTMLRemoveTags(<string>); strHTMLEscape(<string>); strHTMLEscape(<string>); strHTMLUnEscape(<string>); importScript(<script1)[, ispromise(<value="" scriptn]);="">); strPad(<string>); strReverse(<string>); strReverse(<string>); strAt(<string>, <index>); strPad(<string>); strPad(<string>); importScript(<scriptn]); ispromise(<value="">); isIterator(<value>); isIterator(<value>); isIterator(<value>); strCodePoints(<string>); strFomCodePoints(<collection>); bind(<fn>, <context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertDate(<msg>, <value>); ssertDate(<msg>, <value>); getDoNotTrack(); assertFalse(<msg>, <value>); getLocation(<success>[,error]); isArraylike(<value>); isTpypdArray(<value>); isPrimitive(<value>); isPromise(<value>); isPromise(<value>); isIterator(<value>); isIterator(<value>); isIterator(<value>); isEmptyObject(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isSameObject(<object1>, <object2>); isSameArray(<array1>, <array2>); isSameAp(<array1>, <array2>); isSameAp(<array1>, <array2>); isSameAp(<array1>, <array2>); isSameSet(<set1>, <apre>, <apre>,</apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></apre></set1></array2></array1></array2></array1></array2></array1></array2></array1></object2></object1></value></value></value></value></value></value></value></value></value></value></value></value></success></value></msg></value></msg></value></msg></value></value></fn></context></fn></collection></string></value></value></value></scriptn]);></string></string></index></string></string></string></string></script1)[,></string></string></string></string></s></str></string></pre>	<pre>strPropercase(<string>);</string></pre>	<pre>domSiblingsLeft(<element>);</element></pre>	<pre>isAsyncGeneratorFn(<value>);</value></pre>
<pre>strUpFirst(<str>); and strDownFirst(<s>); strHTMLRemoveTags(<string>); strHTMLRemoveTags(<string>); strHTMLEscape(<string>); strHTMLUnEscape(<string>); strReverse(<string>); strReverse(<string>); strAt(<string>, <index>); strCodePoints(<scollection>); bind(<fn>, <context>); and unBind(<fn>); constant(<value>); and identity(<value>); strSeverse(<string>); strFomCodePoints(<string>); strFromCodePoints(<string>); strFromCodePoints(<string>); strFromCodePoints(<string>); strFromCodePoints(<string>); strFromCodePoints(<collection>); bind(<fn>, <context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msg>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); assertTrue(<msg>, <value>); getDoNotTrack(); getLocation(<success>[, error]); isTypedArray(<value>); isArrayBuffer(<value>); isPrimitive(<value>); isPromise(<value>); isIterator(<value>); isIterator(<value>); isIterator(<value>); isEmptyObject(<value>); isEmptyArray(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptySet(<value>); isSameArray(<array1>, <array2>); isSameArray(<array1>, <array2>); isSameArray(<array1>, <array2>); isSameMap(<map1>, <map2>); isSameSet(<set1>, <set2>);</set2></set1></map2></map1></array2></array1></array2></array1></array2></array1></value></value></value></value></value></value></value></value></value></value></value></value></success></value></msg></value></msg></v2></v1></msg></value></value></fn></context></fn></collection></string></string></string></string></string></string></value></value></fn></context></fn></scollection></index></string></string></string></string></string></string></string></s></str></pre>	<pre>strTitlecase(<string>);</string></pre>	<pre>domSiblingsNext(<element>);</element></pre>	isAsyncFn(<value>);</value>
<pre>strHTMLRemoveTags(<string>); strHTMLEscape(<string>); strHTMLUnEscape(<string>); strReverse(<string>); strAt(<string>,<index>); strCodePoints(<string>); strFromCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); constant(<value>); assertEq(<msg>,<vu1>,<vu2>,(xtrict=true]); assertTrue(<msg>,<vu1ve); assertfalse(<msg="">,<vu1ve); assertfalse(<msg="">,<vu1ve); assertfalse(<msg="">,<vu1ve); assertfalse(<msg="">,<vu1ve); assertfalse(<msg="">,<vu1ve); assertsalse(<msg="">,<vu1ve); assertsalse(<msg="" assertsalse(<msg);="">,<vu1ve); assertsalse(<msg="">,<vu1ve); asserts<="" td=""><td><pre>strCapitalize(<string>);</string></pre></td><td><pre>domSiblingsRight(<element>);</element></pre></td><td><pre>isArraylike(<value>);</value></pre></td></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></vu1ve);></msg></vu2></vu1></msg></value></fn></context></fn></collection></string></index></string></string></string></string></string></pre>	<pre>strCapitalize(<string>);</string></pre>	<pre>domSiblingsRight(<element>);</element></pre>	<pre>isArraylike(<value>);</value></pre>
<pre>strHTMLEscape(<string>); strHTMLUnEscape(<string>); strReverse(<string>); strReverse(<string>); strReverse(<string>); strAt(<string>,<index>); strCodePoints(<string>); strFromCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); constant(<value>); and identity(<value>); assertEq(<msg>,<v1>,<v2>[, strict=true]); assertTrue(<msg>,<value>); assertTrue(<msg>,<value>); assertFalse(<msg>,<value>); assertFalse(<msg>,<value>); assertFalse(<msg>,<value>); assertSalse(<msg>,<value>); getLocation(<success>[,error]);</success></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></v2></v1></msg></value></value></fn></context></fn></collection></string></index></string></string></string></string></string></string></pre> isPrimitive(<value>); isPromise(<value>); isPromise(<value>); isIterator(<value>); isIterator(<value>); isIterator(<value>); isEmptyObject(<value>); isEmptyObject(<value>); isEmptyArray(<value>); isEmptyArray(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptySet(<value>); isSameObject(<object1>,<object2>); isSameArray(<array1>,<array2>); isSameMap(<map1>,<map2>); isSameSet(<set1>,<set2>);</set2></set1></map2></map1></array2></array1></object2></object1></value></value></value></value></value></value></value></value></value></value></value></value></value></value>	<pre>strUpFirst(<str>); and strDownFirst(<s>);</s></str></pre>	<pre>domGetCSSVar(<name>);</name></pre>	<pre>isTypedArray(<value>);</value></pre>
<pre>strHTMLUnEscape(<string>); strReverse(<string>); strReverse(<string>); strAt(<string>,<index>); strCodePoints(<string>); strFromCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msg>,<vul>,<vul>,<vul>,<vul>,<index (some="" and="" asserttrue(<msg="" context);="" formulation="" und="">,<vul>,<vul>,<vul>,<index (some="" and="" asserttrue(<msg="" context);="" formulation="" und="">,<vul>,<vul>,<vul>,<index (some="" asserttrue(<msg="" context);="">,<vulue>); assertTrue(<msg>,<vulue>); assertTrue(<msg>,<vulue>); assertTrue(<msg>,<vulue>); assertFalse(<msg>,<vulue>); assertFalse(<msg>,<vulue>); assertFalse(<msg>,<vulue>); assertTrue(<msg>,<vulue>); assertTrue(<ms< td=""><td><pre>strHTMLRemoveTags(<string>);</string></pre></td><td><pre>domSetCSSVar(<name>, <value>);</value></name></pre></td><td><pre>isArrayBuffer(<value>);</value></pre></td></ms<></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></msg></vulue></index></vul></vul></vul></index></vul></vul></vul></index></vul></vul></vul></vul></msg></value></value></fn></context></fn></collection></string></index></string></string></string></string></pre>	<pre>strHTMLRemoveTags(<string>);</string></pre>	<pre>domSetCSSVar(<name>, <value>);</value></name></pre>	<pre>isArrayBuffer(<value>);</value></pre>
<pre>strReverse(<string>); strAt(<string>,<index>); strCodePoints(<string>); strFromCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msg>,<v1>,<v2>[,strict=true]); assertTrue(<msg>,<value>); assertTrue(<msg>,<value>); assertFalse(<msg>,<value>); assertFalse(<msg>,<value>); assertFalse(<msg>,<value>); assertSalse(<msg>,<value>); assertSalse(<msg>,<value></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></v2></v1></msg></value></value></fn></context></fn></collection></string></index></string></string></pre>	<pre>strHTMLEscape(<string>);</string></pre>		<pre>isPrimitive(<value>);</value></pre>
<pre>strAt(<string>,<index>); strCodePoints(<string>); strCodePoints(<collection>); strFromCodePoints(<collection>); bind(<fn>,<context>); and unBind(<fn>); 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>); getDoNotTrack(); assertFalse(<msg>,<value>); getLocation(<success>[,error]); isIterable(<value>); isEmptyObject(<value>); isEmptyArray(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptyMap(<value>); isEmptySet(<value>); isSameObject(<object1>,<object2>); isSameObject(<object1>,<array2>); isSameArray(<array1>,<array2>); isSameAp(<map1>,<map2>); isSameSet(<set1>,<set2>);</set2></set1></map2></map1></array2></array1></array2></object1></object2></object1></value></value></value></value></value></value></value></value></value></value></success></value></msg></value></msg></value></msg></v2></v1></m></v2></v1></msg></value></value></fn></context></fn></collection></collection></string></index></string></pre>	<pre>strHTMLUnEscape(<string>);</string></pre>	<pre>importScript(<script1>[,scriptN]);</script1></pre>	<pre>isPromise(<value>);</value></pre>
<pre>strCodePoints(<string>); strFromCodePoints(<collection>); strFromCodePoints(<collection>); bind(<fn>, <context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msg>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); assertTrue(<msg>, <value>); assertFalse(<msg>, <value>); assertFalse(<msg>, <value>); assertFalse(<msg>, <value>); assertFalse(<msg>, <value>); assertSalse(<msg>, <value>); a</value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></v2></v1></msg></value></value></fn></context></fn></collection></collection></string></pre>	<pre>strReverse(<string>);</string></pre>	<pre>importStyle(<style1>[,styleN]);</style1></pre>	<pre>isIterator(<value>);</value></pre>
<pre>strFromCodePoints(<collection>); bind(<fn>, <context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msg>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); assertTrue(<msg>, <value>); assertFalse(<msg>, <value>);</value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></value></msg></v2></v1></msg></value></value></fn></context></fn></collection></pre> isEmptyMar(<value>); isEmptyMar(<value>); isEmptyMar(<value>); isEmptyMar(<value>); isEmptyMar(<value>); isEmptyMar(<alue>); isEmpty</alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></alue></value></value></value></value></value>	<pre>strAt(<string>,<index>);</index></string></pre>		<pre>isIterable(<value>);</value></pre>
<pre>bind(<fn>,<context>); and unBind(<fn>); constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msg>,<v1>,<v2>[,strict=true]); assertTrue(<msg>,<value>); assertTrue(<msg>,<value>); assertFalse(<msg>,<value>); assertFalse(<msg>,<value>); assertFalse(<msg>,<value>);</value></msg></value></msg></value></msg></value></msg></value></msg></v2></v1></msg></value></value></fn></context></fn></pre> <pre>isEmptyMap(<value>); isEmptyMap(<value>); isSameObject(<object1>,<object2>); isSameArray(<array1>,<array2>); isSameMap(<map1>,<map2>); isSameSet(<set1>,<set2>);</set2></set1></map2></map1></array2></array1></object2></object1></value></value></value></value></value></value></value></value></value></value></value></pre>	<pre>strCodePoints(<string>);</string></pre>	<pre>setFullscreenOn(<selector> or <element>);</element></selector></pre>	<pre>isEmptyObject(<value>);</value></pre>
<pre>constant(<value>); and identity(<value>); noop(); and T(); and F(); assertEq(<msg>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); assertFalse(<msg>, <value>); assertFalse(<msg>, <value>);</value></msg></value></msg></value></msg></v2></v1></msg></v2></v1></msg></value></value></pre> isEmptySet(<value>); isEmptyIterator(<value>); isSameObject(<object1>, <object2>); isSameArray(<array1>, <array2>); isSameMap(<map1>, <map2>); isSameMap(<map1>, <map2>); isSameSet(<set1>, <set2>);</set2></set1></map2></map1></map2></map1></array2></array1></object2></object1></value></value>	<pre>strFromCodePoints(<collection>);</collection></pre>	<pre>setFullscreenOff();</pre>	<pre>isEmptyArray(<value>);</value></pre>
<pre>noop(); and T(); and F(); assertEq(<msg>, <v1>, <v2>[, strict=true]); assertNotEq(<m>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); assertFalse(<msg>, <value>); assertFalse(<msg>, <value>);</value></msg></value></msg></value></msg></v2></v1></m></v2></v1></msg></pre> form2array(<form>); isEmptyIterator(<value>); isSameObject(<object1>, <object2>); isSameArray(<array1>, <array2>); isSameMap(<map1>, <map2>); isSameSet(<set1>, <set2>);</set2></set1></map2></map1></array2></array1></object2></object1></value></form>	<pre>bind(<fn>,<context>); and unBind(<fn>);</fn></context></fn></pre>	<pre>getFullscreen();</pre>	<pre>isEmptyMap(<value>);</value></pre>
<pre>assertEq(<msg>, <v1>, <v2>[, strict=true]); assertNotEq(<m>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); assertFalse(<msg>, <value>); form2string(<form>); isSameObject(<object1>, <object2>); isSameArray(<array1>, <array2>); isSameMap(<map1>, <map2>); isSameMap(<map1>, <map2>); isSameSet(<set1>, <set2>);</set2></set1></map2></map1></map2></map1></array2></array1></object2></object1></form></value></msg></value></msg></v2></v1></m></v2></v1></msg></pre>	<pre>constant(<value>); and identity(<value>);</value></value></pre>		<pre>isEmptySet(<value>);</value></pre>
<pre>assertNotEq(<m>, <v1>, <v2>[, strict=true]); assertTrue(<msg>, <value>); assertFalse(<msg>, <value>); getDoNotTrack(); getLocation(<success>[,error]); isSameArray(<array1>, <array2>); isSameMap(<map1>, <map2>); isSameSet(<set1>, <set2>);</set2></set1></map2></map1></array2></array1></success></value></msg></value></msg></v2></v1></m></pre>	noop(); and T(); and F();	<pre>form2array(<form>);</form></pre>	<pre>isEmptyIterator(<value>);</value></pre>
<pre>assertTrue(<msg>, <value>); assertFalse(<msg>, <value>); getDoNotTrack(); getLocation(<success>[,error]); isSameMap(<map1>, <map2>); isSameSet(<set1>, <set2>);</set2></set1></map2></map1></success></value></msg></value></msg></pre>	<pre>assertEq(<msg>, <v1>, <v2>[, strict=true]);</v2></v1></msg></pre>	<pre>form2string(<form>);</form></pre>	<pre>isSameObject(<object1>,<object2>);</object2></object1></pre>
<pre>assertTrue(<msg>, <value>); assertFalse(<msg>, <value>); getDoNotTrack(); getLocation(<success>[,error]); isSameMap(<map1>, <map2>); isSameSet(<set1>, <set2>);</set2></set1></map2></map1></success></value></msg></value></msg></pre>	<pre>assertNotEq(<m>, <v1>, <v2>[, strict=true]);</v2></v1></m></pre>		<pre>isSameArray(<array1>,<array2>);</array2></array1></pre>
		<pre>getDoNotTrack();</pre>	
	<pre>assertFalse(<msg>, <value>);</value></msg></pre>	<pre>getLocation(<success>[,error]);</success></pre>	<pre>isSameSet(<set1>, <set2>);</set2></set1></pre>
nocontrice(), and vincint, (tell), (tell),	noConflict(); and VERSION;	<pre>createFile(<filename>, <content>[,dType]);</content></filename></pre>	<pre>isSameIterator(<iter1>,<iter2>);</iter2></iter1></pre>

```
Collections
                                                                                                      Polyfills
arrayCreate([length=0]);
                                                 forEach (<collection>, <callback>);
arrayDeepClone(<array>);
                                                 map(<collection>, <callback>);
                                                                                       Array.prototype.at();
arrayMerge(<target>, <source1>[, sourceN]);
                                                 enumerate(<collection>[,offset=0]);
                                                                                       Array.prototype.findLast();
arrayUnique(<collection>);
                                                 entries(<collection>[,offset=0]);
                                                                                       Array.prototype.findLastIndex();
                                                 size(<collection>);
arrayAdd(<array>,<value>);
                                                                                       Array.prototype.flat();
arrayClear(<array>);
                                                                                       Array.prototype.flatMap();
arrayRemove(<array>, <value>[,all=false]);
                                                 every(<collection>, <callback>);
                                                                                       Array.prototype.groupBy();
arrayRemoveBy(<array>, <callback>[,all=false]);
                                                 some(<collection>,<callback>);
                                                                                       Array.prototype.groupByToMap();
arrayRange([start=0[,end=99[,step=1]]]);
                                                 none(<collection>, <callback>);
arrayCycle(<collection>[,n=100]);
                                                                                       crypto.randomUUID();
arrayRepeat(<value>[,n=100]);
                                                 includes(<collection>, <value>);
                                                 contains(<collection>,<value>);
                                                                                       globalThis;
iterRange([start=0[,step=1[,end=Infinity]]]);
iterCycle(<iter>[,n=Infinity]);
                                                 find(<collection>,<callback>);
                                                                                       Number.MIN SAFE INTEGER;
iterRepeat(<value>[,n=Infinity]);
                                                 findLast(<collection>, <callback>);
                                                                                       Number.MAX SAFE INTEGER;
arrayUnion(<collection1>[,collectionN]);
                                                                                       Object.fromEntries();
                                                 filter(<collection>, <callback>);
arrayIntersection(<collection1>,<collection2>); reject(<collection>,<callback>);
                                                                                       Object.hasOwn();
arrayDifference(<collection1>, <collection2>);
                                                 partition(<collection>,<callback>);
                                                                                       Object.is();
arraySymmetricDifference(<collec1>, <collec2>);
                                                 groupBy(<collec.>,<cb>[,map=false]);
setUnion(<collection1>[,collectionN]);
                                                                                       String.prototype.at();
                                                 shuffle(<collection>);
                                                                                       String.prototype.matchAll();
setIntersection(<set1>,<set2>);
                                                 min(<value1>[,valueN]);
                                                                                       String.prototype.replaceAll();
setDifference(<set1>,<set2>);
                                                 max(<value1>[,valueN]);
                                                                                       String.prototype.trimStart();
setSymmetricDifference(<set1>, <set2>);
                                                 sort(<collection>[,numbers=false]);
                                                                                       String.prototype.trimLeft();
                                                 reverse (<collection>):
                                                                                       String.prototype.trimEnd();
isSuperset(<superCollection>, <subCollection>);
                                                                                       String.prototype.trimRight();
                                                 zip(<collection1>[,collectionN]);
slice(<collection>[,begin=0[,end=Infinity]]);
                                                 unzip(<collection>);
                                                                                       TypedArray.prototype.at();
withOut(<collection>, <filterCollection>);
                                                 zipObj(<collection1>, <collection2>); | TypedArray.prototype.findLast();
                                                 item(<collection>,<index>);
reduce(<collection>, <callback>[,initialvalue]);
                                                                                        TypedArray.prototype.findLastIndex();
                                                 nth(<collection>,<index>);
take(<collection>[,n=1]);
                                                 first(<collection>);
                                                                                               Non-standard polyfills
takeWhile (<collection>, <callback>);
                                                 head(<collection>):
takeRight(<collection>[,n=1]);
                                                 last(<collection>);
takeRightWhile(<collection>, <callback>);
                                                 initial(<collection>);
                                                                                        BigInt.prototype.toJSON();
drop(<collection>[,n=1]);
                                                 tail(<collection>);
dropWhile(<collection>, <callback>);
                                                 flat(<collection>);
                                                                                       AsyncFunction();
dropRight(<collection>[,n=1]);
                                                 concat(<collection1>[,collectionN]); | GeneratorFunction();
dropRightWhile(<collection>, <callback>);
                                                 join(<collection>[,separator=","]);
```

```
Math plugin (with celestra-math.min.js)
                                                                                       Abstract functions
                                   toFloat32(<value>);
sum(<value1>[,valueN]);
                                                                      getIn(<object>,,,;
avg(<value1>[,valueN]);
                                                                      getInV(<object>,,,;
product(<value1>[,valueN]);
                                   toInt8(<value>);
                                                                      hasIn(<object>,,,;
                                   toUInt8(<value>);
                                                                      setIn(<object>,,<value>);
clamp(<value>, <min>, <max>);
                                   toInt16(<value>);
                                   toUInt16(<value>);
                                                                      toIndex(<value>);
isEven(<value>);
                                   toInt32(<value>);
                                                                      toPropertyKey(<value>);
                                   toUInt32(<value>);
isOdd(<value>);
                                                                      toInteger(<value>);
                                   toBigInt64(<value>);
                                                                      toArray(value);
                                   toBigUInt64(<value>);
                                                                      toObject(<value>);
                                   isInt8(<value>);
                                                                      isIndex(<value>);
                                   isUInt8(<value>);
                                                                      isPropertyKey(<value>);
                                   isInt16(<value>);
                                                                      isSameValue(<value1>, <value2>);
                                   isUInt16(<value>);
                                                                      isSameValueZero(<value1>, <value2>);
                                   isInt32(<value>);
                                                                      isSameValueNonNumber(<value1>, <value2>);
                                   isUInt32(<value>);
                                   isBigInt64(<value>);
                                                                      type(<value>);
                                   isBigUInt64(<value>);
                                                                      createDataProperty(<object>,,,<value>);
                                                                      createMethodProperty(<object>,,,<value>);
                                                          Cookie
getCookie([name]);
hasCookie(<name>);
setCookie(<Options object>);
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>);
clearCookies([path="/"[,domain[,sec[,SameSite="Lax"[,HttpOnly]]]]]);
                                                      AJAX and CORS
getText(<url>, <success>);
getJson(<url>, <success>);
ajax(<Options object>);
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
```