Number() = 42

PROPERTIES

- .POSITIVE_INFINITY +∞ equivalent
- .NEGATIVE_INFINITY -∞ equivalent
- .MAX_VALUE largest positive value
- .MIN_VALUE smallest positive value
- .EPSILON diff between 1 & smallest >1
- .NaN not-a-number value

METHODS

- .toExponential(dec) exp. notation
- .toFixed(dec) fixed-point notation
- .toPrecision(p) change precision
- .isFinite(n) check if number is finite
- .isInteger(n) check if number is int.
- .isNaN(n) check if number is NaN
- .parseInt(s, radix) string to integer
- .parseFloat(s, radix) string to float

Regexp() = /.+/ig

PROPERTIES

- .lastIndex index to start global regexp
- .flags active flags of current regexp
- .global flag g (search all matches)
- .ignoreCase flag i (match lower/upper)
- .multiline flag m (match multiple lines)
- .sticky flag y (search from lastIndex)
- .unicode flag u (enable unicode feat.)
- .source current regexp (w/o slashs)

METHODS

- .exec(str) exec search for a match
- .test(str) check if regexp match w/str

CLASSES

. any character	\t tabulator
\d digit [0-9]	\r carriage return
\D no digit [^0-9]	\n line feed

\w any alphanumeric char [A-Za-z0-9_]
\W no alphanumeric char [^A-Za-z0-9_]
\s any space char (space, tab, enter...)
\S no space char (space, tab, enter...)

\xN char with code N	[\b] backspace
\uN char with unicode N	\0 NUL char

CHARACTER SETS OR ALTERNATION

[abc] match any character set
[^abc] match any char. set not enclosed
a|b match a or b

BOUNDARIES

^ begin of input	$ end of input

\b zero-width word boundary
\B zero-width non-word boundary

GROUPING

(x) capture group (?:x) no capture group
\n reference to group n captured

QUANTIFIERS

x* preceding x 0 or more times {0,}
x+ preceding x 1 or more times {1,}
x? preceding x 0 or 1 times {0,1}
x{n} n ocurrences of x
x{n,} at least n ocurrences of x
x{n,m} between n & m ocurrences of x

ASSERTIONS

x(?=y) x (only if x is followed by y)
x(?!y) x (only if x is not followed by y)

String() = 'text'

PROPERTIES

- .length string size

METHODS

- .charAt(index) char at position [i]
- .charCodeAt(index) unicode at pos.
- .codePointAt(index) cp at position
- .fromCharCode(n1, n2...) code to char
- .fromCodePoint(n1, n2...) cp to char
- .concat(str1, str2...) combine text +
- .startsWith(str, size) check beginning
- .endsWith(str, size) check ending
- .includes(str, from) include substring?
- .indexOf(str, from) find substr index
- .lastIndexOf(str, from) find from end
- .search(regex) search & return index
- .localeCompare(str, locale, options)
- .match(regex) matches against string
- .matchAll(regex) return iterator w/all
- .normalize(form) unicode normalize
- .padEnd(len, pad) add end padding
- .padStart(len, pad) add start padding
- .repeat(n) repeat string n times
- .replace(str|regex, newstr|func)
- .slice(ini, end) str between ini/end
- .substr(ini, len) substr of len length
- .substring(ini, end) substr fragment
- .split(sep|regex, limit) divide string
- .toLowerCase() string to lowercase
- .toUpperCase() string to uppercase
- .trim() remove space from begin/end
- .trimEnd() remove space from end
- .trimStart() remove space from begin
- .raw`` template strings with ${vars}

Date()

METHODS

- .UTC(y, m, d, h, i, s, ms) timestamp
- .now() timestamp of current time
- .parse(str) convert str to timestamp
- .setTime(ts) set UNIX timestamp
- .getTime() return UNIX timestamp

UNIT GETTERS / SETTERS (ALSO .getUTC*() / .setUTC*())

.get / .setFullYear(y, m, d)	(yyyy)
.get / .setMonth(m, d)	(0-11)
.get / .setDate(d)	(1-31)
.get / .setHours(h, m, s, ms)	(0-23)
.get / .setMinutes(m, s, ms)	(0-59)
.get / .setSeconds(s, ms)	(0-59)
.get / .setMilliseconds(ms)	(0-999)
.getDay() return day of week	(0-6)

LOCALE & TIMEZONE METHODS

- .getTimezoneOffset() offset in mins
- .toLocaleDateString(locale, options)
- .toLocaleTimeString(locale, options)
- .toLocaleString(locale, options)
- .toUTCString() return UTC date
- .toDateString() return American date
- .toTimeString() return American time
- .toISOString() return ISO8601 date
- .toJSON() return date ready for JSON

Array() = [1, 2, 3]

PROPERTIES

- .length number of elements

METHODS

- .isArray(obj) check if obj is array
- .includes(obj, from) include element?
- .indexOf(obj, from) find elem. index
- .lastIndexOf(obj, from) find from end
- .join(sep) join elements w/separator
- .slice(ini, end) return array portion
- .concat(obj1, obj2...) return joined array
- .flat(depth) return flat array at n depth

MODIFY SOURCE ARRAY METHODS

- .copyWithin(pos, ini, end) copy elems
- .fill(obj, ini, end) fill array with obj
- .reverse() reverse array & return it
- .sort(cf(a,b)) sort array (unicode sort)
- .splice(ini, del, o1, o2...) del&add elem

ITERATION METHODS

- .entries() iterate key/value pair array
- .keys() iterate only keys array
- .values() iterate only values array

CALLBACK FOR EACH METHODS

- .every(cb(e,i,a), arg) test until false
- .some(cb(e,i,a), arg) test until true
- .map(cb(e,i,a), arg) make array
- .filter(cb(e,i,a), arg) make array w/true
- .find(cb(e,i,a), arg) return elem w/true
- .findIndex(cb(e,i,a), arg) return index
- .flatMap(cb(e,i,a), arg) map + flat(1)
- .forEach(cb(e,i,a), arg) exec for each
- .reduce(cb(p,e,i,a), arg) accumulative
- .reduceRight(cb(p,e,i,a), arg) from end

ADD/REMOVE METHODS

- .pop() remove & return last element
- .push(o1, o2...) add elem & return length
- .shift() remove & return first element
- .unshift(o1, o2...) add elem & return len

Function() = function(a, b) { ... }

PROPERTIES

- .length return number of arguments
- .name return name of function
- .prototype prototype object

METHODS

- .call(newthis, arg1, arg2...) change this
- .apply(newthis, arg1) with args array
- .bind(newthis, arg1, arg2...) bound func

number	date
NaN (not-a-number)	regular expresion
string	function
boolean (true/false)	object
array	undefined

available on ECMAScript 2015 or higher

static (ex: Math.random())
non-static (ex: new Date().getDate())

argument required
argument optional