JavaScript ES7

Writer: Kim Young-Bo. Date: 2016.09.13 v1

Operator					
Arithmetic operators			Binary bitwise operators		
+	Addition	3	&	Bitwise AND	3
-	Subtraction	3	|	Bitwise OR	3
*	Multiplication	3	^	Bitwise XOR	3
/	Division	3	Bitwise shift operators		
%	Remainder	3	<<	left shift	3
**	Exponentiation	7	>>	right shift	3
Relational operators			>>>	unsigned right shift	3
<	Less than	3	Binary logical operators		
>	Greater than	3	&&	Logical AND	3
<=	Less than or equal	3	||	Logical OR	3
>=	Greater than or equal	3	Assignment operators		
in	In	3	=	Assignment	3
instanceof	Instanceof	3	+=	Addition	3
Increment and decrement operators			-=	Subtraction	3
value++	Postfix increment	3	*=	Multiplication	3
value--	Postfix decrement	3	/=	Division	3
++value	Prefix increment	3	%=	Remainder	3
--value	Prefix decrement	3	&=	Bitwise AND	3
Unary operators			^=	Bitwise XOR	3
+	Unary plus	3	|=	Bitwise OR	3
-	Unary negation	3	<<=	Left shift	3
~	Bitwise NOT	3	>>=	Right shift	3
!	Logical NOT	3	>>>==	Unsigned right shift	3
delete	Delete	3	[x, y] = [1, 2]	Destructuring	6
void	Void	3	{x, y} = {x:1, y:2}	Destructuring	6
typeof	Typeof	3	Comma operator		
Equality operators			,	comma	3
==	Equality	3	Conditional operator		
!=	Inequality	3	(? :)	grouping	3
===	Identity	3			
!==	Nonidentity	3			

Expression					
Primary expressions			Left-hand-side expressions		
[ ]	Array literal	3	new	New Operator	3
{ }	Object literal	3	new.target	Refer to the constructor	6
( )	Grouping operator	3	super	Super keyword	6
` `	Template literal	6	...params	Spread Operator(...rest)	6
/pattern/flags	Regular expression literal	3	...params	Spread Operator	6
this	This keyword	3	object.property	Property accessors	3
			object[property]	Property accessors	3

Statements and Declarations

Statements					
Block	{ code; code;}	3	for	for ([init]; [condition]; [exp]) { code }	3
Empty	; semicolon	3	for-in	for (variable in object) {.....}	3
break	break [Label]	3	for-of	for (variable of object) {.....}	6
if	if (expression) code	3	do-while	statement while expression	3
	if (exp) else code		while	while (expression) { code }	3
throw	throw expression	3	return	return [expression]	3
switch	switch (expression) {	3	debugger	debugger	5
	case expression: [code]		import	import expression	6
	default: [code]		export	export expression	6
try	try {} catch() {}	3	Declarations		
	try {} finally {}		var	var	3
	try {} catch {} finally {}		let	let	6
continue	continue [Label]	3	const	const	6
label	label; statement	3			
with	with (expression) { code }	3			

Keywords

ES3 ~ ES7 Keywords					
break	case	catch	class	const	continue
debugger	default	delete	do	else	export
extends	finally	for	function	if	import
in	instanceof	new	return	super	switch
this	throw	try	typeof	var	void
while	with	yield			
Future Reserved Words					
await	enum	implements	interface	package	private
protected	public				

Function and Class

Function Definitions		Arrow Function Definitions	
function name(param[, param])){ code }	3	(param1, param2) => { code }	6
function [name](param[, param])){ code }	3	(param1, param2) => expression	6
Generator Function Definitions		(param) => { code }	6
function* name([param[, param]]) { code }	6	param => { code }	6
function* [name]([param[, param]]) { code }	6	( ) => { code }	6
[nextParam] = yield [expression]	6	param = ({key: value})	6
[nextParam] = yield* [[expression]]	6	(param1, param2, ...rest) => { code }	6
Class Definitions	6	(param = default) => { code }	6
class name [extends] { code }	6	([x, y] = [1, 2]) => expression	6
static name() { code }	6	({key: value}) = {key: 1} => expression	6

Global Object			
Value Properties of the Global Object		decodeURI (encodedURI)	3
Infinity	3	decodeURIComponent (encodedURI)	3
NaN	3	encodeURI(URI)	3
undefined	3	encodeURIComponent(URI)	3
Function Properties of the Global Objet		escape(string), deprecated	3
eval(string)	3	unescape(string), deprecated	3
isFinite (value)	3	Other Properties of the Global Object	
isNaN(value)	3	JSON	5
parseFloat (string)	3	Math	3
parseInt (string, radix)	3	Reflect	6

Object			
Object Constructor		Object.isFrozen(object)	5
{ [key: value] .....}	3	Object.isSealed(object)	5
Object([value])	3	Object.keys(object)	5
new Object([value])	3	Object.preventExtensions(object)	5
Properties of the Object Constructor		Object.seal(object)	5
Object.prototype	3	Object.setPrototypeOf(object, prototype)	6
Object.assign(target, ...sources)	6	Properties of the Object Prototype Object	
Object.create(prototype[, props])	5	Object.prototype.constructor	3
Object.defineProperties(object, props)	5	Object.prototype.hasOwnProperty(prop)	3
Object.defineProperty(object, prop, descriptor)	5	Object.prototype.isPrototypeOf(object)	3
Object.freeze(object)	5	Object.prototype.propertyIsEnumerable(prop)	3
Object.getOwnPropertyDescriptor(object, prop)	5	Object.prototype.toLocaleString()	3
Object.getOwnPropertyNames(object)	5	Object.prototype.toString()	3
Object.getOwnPropertySymbols(object)	6	Object.prototype.valueOf()	3
Object.getPrototypeOf(object)	5	Additional Properties of the Object.prototype Object	
Object.is(value1, value2);	6	get Object.prototype.__proto__	6
Object.isExtensible(object)	5	set Object.prototype.__proto__	6

Function Object			
Function Constructor		Function.prototype.bind(this[, arg1[, arg2]])	5
Function ([arg1[, arg2],] body)	3	Function.prototype.call(this[, arg1[, arg2]])	3
new Function ([arg1[, arg2],] body)	3	Function.prototype.toString()	3
Properties of the Function Constructor		Function.prototype[@@hasInstance] (object)	6
Function.length	3	Function Instances	
Function.prototype	3	length	3
Properties of the Function Prototype Object		name	6
Function.prototype.constructor	3	prototype	3
Function.prototype.apply(this, [[args]])	3		

Boolean Object			
Boolean Constructor		Properties of the Boolean Prototype Object	
Boolean(value)	3	Boolean.prototype.constructor	3
new Boolean(value)	3	Boolean.prototype.toString()	3
Properties of the Boolean Constructor		Boolean.prototype.valueOf()	3
Boolean.prototype	3		

Symbol Object			
Symbol Constructor		Symbol.toPrimitive	6
Symbol([description])	6	Symbol.toStringTag	6
Properties of the Symbol Constructor		Symbol.unscopables	6
Symbol.hasInstance	6	Symbol.for(key)	6
Symbol.isConcatSpreadable	6	Symbol.keyFor(symbol)	6
Symbol.iterator	6	Properties of the Symbol Prototype Object	
Symbol.match	6	Symbol.prototype.constructor	6
Symbol.prototype	6	Symbol.prototype.toString()	6
Symbol.replace	6	Symbol.prototype.valueOf()	6
Symbol.search	6	Symbol.prototype[@@toPrimitive](hint)	6
Symbol.species	6	Symbol.prototype[@@toStringTag]	6
Symbol.split	6		

Error Object			
Error Constructor		Error.prototype.message	3
Error([message])	3	Error.prototype.name	3
new Error([message])	3	Error.prototype.toString()	3
Properties of the Error Constructor			
Error.prototype	3	Native Error Types	
Properties of the Error Prototype Object		EvalError, RangeError, ReferenceError,	3
Error.prototype.constructor	3	SyntaxError, TypeError, URIError	3

Number Object			
Number Constructor		Number.isInteger(value)	6
Number(value)	3	Number.isNaN(value)	6
new Number(value)	3	Number.isSafeInteger(value)	6
Properties of the Number Constructor		Number.parseFloat (string)	6
Number.EPSILON	6	Number.parseInt(string[, radix])	6
Number.MAX_SAFE_INTEGER	6	Properties of the Number Prototype Object	
Number.MAX_VALUE	3	Number.prototype.constructor	3
Number.MIN_SAFE_INTEGER	6	Number.prototype.toExponential ([digits])	3
Number.MIN_VALUE	3	Number.prototype.toFixed([digits])	3
Number.NaN	3	Number.prototype.toLocaleString([locales[, opts]])	3
Number.NEGATIVE_INFINITY	3	Number.prototype.toPrecision([precision])	3
Number.POSITIVE_INFINITY	3	Number.prototype.toString([radix])	3
Number.prototype	3	Number.prototype.valueOf()	3
Number.isFinite(value)	6		

Math Object			
Value Properties of the Math Object		Math.cosh(x)	6
Math.E	3	Math.exp(x)	3
Math.LN10	3	Math.expm1(x)	6
Math.LN2	3	Math.floor(x)	3
Math.LOG10E	3	Math.fround(x)	6
Math.LOG2E	3	Math.hypot([value1[, value2]])	6
Math.PI	3	Math.imul(x, y)	6
Math.SQRT1_2	3	Math.log(x)	3
Math.SQRT2	3	Math.log1p(x)	6
Math[@@toStringTag]	6	Math.log10(x)	6
Function Properties of the Math Object		Math.log2(x)	6
Math.abs(x)	3	Math.max([value1[, value2]])	3
Math.acos(x)	3	Math.min([value1[, value2]])	3
Math.acosh(x)	6	Math.pow(x, exponent)	3
Math.asin(x)	3	Math.random()	3
Math.asinh(x)	6	Math.round(x)	3
Math.atan(x)	3	Math.sign(x)	6
Math.atanh(x)	6	Math.sin(x)	3
Math.atan2(x, y)	3	Math.sinh(x)	6
Math.cbrt(x)	6	Math.sqrt(x)	3
Math.ceil(x)	3	Math.tan(x)	3
Math.clz32(x)	6	Math.tanh(x)	6
Math.cos(x)	3	Math.trunc(x)	6

Date Object			
Date Constructor		Date.prototype.setDate(day)	3
new Date();	3	Date.prototype.setFullYear(yyyy[, mm[, dd]])	3
new Date(value);	3	Date.prototype.setHours(hh[, mm[, ss[, sss]]])	3
new Date(string);	3	Date.prototype.setMilliseconds(milliseconds)	3
new Date(yyyy, mm[, dd[, hh[, mm[, ss[, sss]]]]]);	3	Date.prototype.setMinutes(mm[, ss[, sss]])	3
Date(yyyy, mm[, dd[, hh[, mm[, ss[, sss]]]]]);	3	Date.prototype.setMonth(month[, day])	3
Properties of the Date Constructor		Date.prototype.setSeconds(seconds[, ms])	3
Date.prototype	3	Date.prototype.setTime(time)	3
Date.now()	5	Date.prototype.setUTCDate(day)	3
Date.parse(string)	3	Date.prototype.setUTCFullYear (yyyy[, mm[, dd]])	3
Date.UTC(yyyy, mm[, dd[, hh[, mm[, ss[, sss]]]]])	3	Date.prototype.setUTCHour (hh[, mm[, ss[, sss]]])	3
Properties of the Date Prototype Object		Date.prototype.setUTCMilliseconds(milliseconds)	3
Date.prototype.constructor	3	Date.prototype.setUTCMinutes(mm[, ss[, sss]])	3
Date.prototype.getDate()	3	Date.prototype.setUTCMonth(month[, day])	3
Date.prototype.getDay()	3	Date.prototype.setUTCSeconds(ss[, sss])	3
Date.prototype.getFullYear()	3	Date.prototype.toDateString()	3
Date.prototype.getHours()	3	Date.prototype.toISOString()	5
Date.prototype.getMilliseconds()	3	Date.prototype.toJSON()	5

Date.prototype.getMinutes()	3	Date.prototype.toLocaleDateString([locales[, opts]])	3
Date.prototype.getMonth()	3	Date.prototype.toLocaleString([locales[, opts]])	3
Date.prototype.getSeconds()	3	Date.prototype.toLocaleTimeString([locales[, opts]])	3
Date.prototype.getTime()	3	Date.prototype.toString()	3
Date.prototype.getTimezoneOffset()	3	Date.prototype.toTimeString()	3
Date.prototype.getUTCDate()	3	Date.prototype.toUTCString()	3
Date.prototype.getUTCDay()	3	Date.prototype.valueOf()	3
Date.prototype.getUTCFullYear()	3	Date.prototype[@@toPrimitive](hint)	6
Date.prototype.getUTCHours()	3	Additional Properties of the Date.prototype Object	
Date.prototype.getUTCMilliseconds()	3	Date.prototype.getYear(), deprecated	3
Date.prototype.getUTCMinutes()	3	Date.prototype.setYear(),deprecated	3
Date.prototype.getUTCMonth()	3	Date.prototype.toGMTString(),deprecated	3
Date.prototype.getUTCSeconds()	3		

String Object			
String Constructor		String.prototype.repeat(count)	6
"string text"	3	String.prototype.replace(regexp, string)	3
String(value)	3	String.prototype.search(regexp)	3
new String(value)	3	String.prototype.slice(start[, end])	3
Properties of the String Constructor		String.prototype.split([separator[, limit]])	3
String.prototype	3	String.prototype.startsWith(search[, position])	6
String.fromCharCode(num1[, num2])	3	String.prototype.substring(start[, snd])	3
String.fromCodePoint(num1[, num2])	6	String.prototype.toLocaleLowerCase()	3
String.raw({raw: 'name', ...params)	6	String.prototype.toLocaleUpperCase()	3
String.raw"template"	6	String.prototype.toLowerCase()	3
Properties of the String Prototype Object		String.prototype.toString()	3
String.prototype.charAt(index)	3	String.prototype.toUpperCase()	3
String.prototype.charCodeAt(index)	3	String.prototype.trim()	3
String.prototype.codePointAt(position)	6	String.prototype.valueOf()	3
String.prototype.concat(string1[, string2])	3	String.prototype[@@iterator]()	6
String.prototype.constructor	3	Properties of String Instances	
String.prototype.endsWith(search[, position])	6	length	3
String.prototype.includes(search[, position])	6	String Iterator Objects	
String.prototype.indexOf(search[, index])	3	next()	6
String.prototype.lastIndexOf(search[, index])	3	[@@toStringTag]	6
String.prototype.localeCompare(str[, locales[, opts]])	3	Additional Properties of the String.prototype Object	
String.prototype.match(regexp)	3	String.prototype.substr(start, length), deprecated	3
String.prototype.normalize([form])	6		

RegExp Object			
flags		RegExp.prototype.global	3
g(global)	3	RegExp.prototype.ignoreCase	3
i(ignoreCase)	3	RegExp.prototype[@@match](string)	6
m(multiline)	3	RegExp.prototype.multiline	3
u(unicode)	6	RegExp.prototype[@@replace](string, string|fn)	6
y(sticky)	6	RegExp.prototype[@@search](string)	6
RegExp Constructor		RegExp.prototype.source	3
RegExp(pattern[, flags])	3	RegExp.prototype[@@split](string[, limit])	6
new RegExp(pattern[, flags])	3	RegExp.prototype.sticky	6
Properties of the RegExp Constructor		RegExp.prototype.test(string)	3
RegExp.prototype	3	RegExp.prototype.toString()	3
RegExp[@@species]	6	RegExp.prototype.unicode	6
Properties of the RegExp Prototype Object		Properties of RegExp Instances	
RegExp.prototype.constructor	3	lastIndex	3
RegExp.prototype.exec(string)	3	Additional Properties of the RegExp.prototype Object	
RegExp.prototype.flags	3	RegExp.prototype.compile(pat, flags), deprecated	3

Array Object			
Array Constructor		Array.prototype.join([separator])	3
[element1, element2]	3	Array.prototype.keys()	3
Array()	3	Array.prototype.lastIndexOf(search[, index])	3
Array(element1[, element2])	3	Array.prototype.map(callback[, this])	3
Array(length)	3	Array.prototype.pop()	3
new Array(element1[, element2])	3	Array.prototype.push(element1, element2)	3
Properties of the Array Constructor		Array.prototype.reduce(callback[, value])	3
Array.prototype	3	Array.prototype.reduceRight(callback [, value])	3
Array.from(iterable[, callback[, this]])	6	Array.prototype.reverse()	3
Array.isArray(object)	3	Array.prototype.shift()	3
Array.of(element1[, element2])	6	Array.prototype.slice([start[, end]])	3
get Array[Symbol,.species]	6	Array.prototype.some(callback[, this])	3
Properties of the Array Prototype Object		Array.prototype.sort([function])	3
Array.prototype.concat(value1[, value2])	3	Array.prototype.splice(start, count[, el1[, el2]])	3
Array.prototype.constructor	3	Array.prototype.toLocaleString()	3
Array.prototype.copyWithin(target[, start[, end]])	3	Array.prototype.toString()	3
Array.prototype.entries()	3	Array.prototype.unshift([element1[, element2]])1	3
Array.prototype.every(callback[, this])	3	Array.prototype.values()	6
Array.prototype.fill(value[, start[, end]])	3	Array.prototype[@@iterator]()	6
Array.prototype.filter(callback[, this])	3	Array.prototype[@@unscopables]	6
Array.prototype.find(callback[, this])	3	Properties of Array Instances	
Array.prototype.findIndex(callback[, this])	3	length	3
Array.prototype.forEach(callback[, this])	3	Array Iterator Objects	
Array.prototype.includes(search[, index])	3	next()	6
Array.prototype.indexOf(search[, index])	3	[@@toStringTag]	6

Map Object			
Map Constructor		Map.prototype.entries()	6
new Map([iterable])	6	Map.prototype.forEach(callback[, this])	6
Properties of the Map Constructor		Map.prototype.get(key)	6
Map.prototype	6	Map.prototype.has(key)	6
get Map[@@species]	6	Map.prototype.keys()	6
Properties of the Map Prototype Object		Map.prototype.set(key, value)	6
Map.prototype.constructor	6	Map.prototype.values()	6
Map.prototype.size	6	Map.prototype[@@iterator]()	6
Map.prototype[@@toStringTag]	6	Map Iterator Objects	
Map.prototype.clear()	6	next()	6
Map.prototype.delete(key)	6	[@@toStringTag]	6

WeakMap Object			
WeakMap Constructor		WeakMap.prototype[@@toStringTag]	6
new WeakMap([iterable])	6	WeakMap.prototype.delete(key)	6
Properties of the WeakMap Constructor		WeakMap.prototype.get(key)	6
WeakMap.prototype	6	WeakMap.prototype.has(key)	6
Properties of the WeakMap Prototype Object		WeakMap.prototype.set(key, value)	6
WeakMap.prototype.constructor	6		

Set Object			
Set Constructor		Set.prototype.delet(value)	6
new Set([iterable])	6	Set.prototype.entries()	6
Properties of the Set Constructor		Set.prototype.forEach(callback[, this])	6
Set.prototype	6	Set.prototype.has(value)	6
get Set[@@species]	6	Set.prototype.keys()	6
Properties of the Set Prototype Object		Set.prototype.values()	6
Set.prototype.constructor	6	Set.prototype[@@iterator]()	6
Set.prototype.size	6		6
Set.prototype[@@toStringTag]	6	Set Iterator Objects	
Set.prototype.add(value)	6	next()	6
Set.prototype.clear()	6	[@@toStringTag]	6

WeakSet Object			
WeakSet Constructor		WeakSet.prototype.constructor	6
new WeakSet([iterable])	6	WeakSet.prototype[@@toStringTag]	6
Properties of the WeakSet Constructor		WeakSet.prototype.add(value)	6
WeakSet.prototype	6	WeakSet.prototype.delete(value)	6
Properties of the WeakSet Prototype Object		WeakSet.prototype.has(value)	6

ArrayBuffer Object			
ArrayBuffer Constructor		Properties of the ArrayBuffer Prototype Object	
new ArrayBuffer(length)	6	get ArrayBuffer.prototype.byteLength	6
Properties of the ArrayBuffer Constructor		ArrayBuffer.prototype.constructor	6
ArrayBuffer.prototype	6	ArrayBuffer.prototype.slice(start[, end])	6
get ArrayBuffer[@@species]	6	ArrayBuffer.prototype[@@toStringTag]	6
ArrayBuffer.isView(object)	6		

TypedArray Object			
TypedArray		TypedArray.prototype.every(callback[, this])	6
Int8Array, Uint8Array, Uint8ClampedArray,	6	TypedArray.prototype.fill(value[, start[, end]])	6
Int16Array, Uint16Array, Int32Array	6	TypedArray.prototype.filter(callback[, this])	6
Uint32Array, Float32Array, Float64Array	6	TypedArray.prototype.find(callback[, this])	6
TypedArray Constructor		TypedArray.prototype.findIndex(callback[, this])	6
new TypedArray()	6	TypedArray.prototype.forEach(callback[, this])	6
new TypedArray(length)	6	TypedArray.prototype.indexOf(search[, index])	6
new TypedArray(typedArray)	6	TypedArray.prototype.includes(search[, index])	6
new TypedArray(object)	6	TypedArray.prototype.join([separator])	6
new TypedArray(buffer[, byteOffset[, length]])	6	TypedArray.prototype.keys()	6
Properties of the TypedArray Constructor		TypedArray.prototype.lastIndexOf(search[, index])	6
TypedArray.BYTES_PER_ELEMENT	6	TypedArray.prototype.map(callback[, this])	6
TypedArray.prototype	6	TypedArray.prototype.reduce(callback[, value])	6
get TypedArray[@@species]	6	TypedArray.prototype.reduceRight(callback[, value])	6
TypedArray.from(source[, callback[, this]])	6	TypedArray.prototype.reverse()	6
TypedArray.of(element1[, element2])	6	TypedArray.prototype.set(array[, offset])	6
Properties of the TypedArray Prototype Objects		TypedArray.prototype.set(typedarray[,offset])	6
TypedArray.prototype.BYTES_PER_ELEMENT	6	TypedArray.prototype.slice([start[, end]])	6
TypedArray.prototype.buffer	6	TypedArray.prototype.some(callback[, this])	6
TypedArray.prototype.byteLength	6	TypedArray.prototype.sort([function])	6
TypedArray.prototype.byteOffset	6	TypedArray.prototype.subarray([begin [,end]])	6
TypedArray.prototype.constructor	6	TypedArray.prototype.toLocaleString()	6
TypedArray.prototype.length	6	TypedArray.prototype.toString()	6
TypedArray.prototype[@@toStringTag]	6	TypedArray.prototype.values()	6
TypedArray.copyWithin(target, start[, end])	6	TypedArray.prototype[@@iterator]()	6
TypedArray.prototype.entries()	6		

DataView Object			
DataView Constructor		DataView.prototype.getInt16(byteOffset[, endian])	6
new DataView(buffer[, offset[, byteLength]])	6	DataView.prototype.getInt32(offset[, endian])	6
Properties of the DataView Constructor		DataView.prototype.getUint8(offset)	6
DataView.prototype	6	DataView.prototype.getUint16(offset[, endian])	6
Properties of the DataView Prototype Object		DataView.prototype.getUint32(offset[, endian])	6
get DataView.prototype.buffer	6	DataView.prototype.setFloat32(offset,value[, endian])	6
get DataView.prototype.byteLength	6	DataView.prototype.setFloat64(offset,value[, endian])	6
get DataView.prototype.byteOffset	6	DataView.prototype.setInt8(offset, value)	6

DataView.prototype.constructor	6	DataView.prototype.setInt16(offset, value[, endian])	6
DataView.prototype[@@toStringTag]	6	DataView.prototype.setInt32(offset, value[, endian])	6
DataView.prototype.getFloat32(offset[, endian])	6	DataView.prototype.setUint8(offset, value)	6
DataView.prototype.getFloat64(offset [, endian])	6	DataView.prototype.setUint16(offset, value[, endian])	6
DataView.prototype.getInt8(offset)	6	DataView.prototype.setUint32(offset, value[, endian])	6

Generator Object			
Properties of the Generator Prototype		Generator.prototype.next(value)	6
Generator.prototype.constructor	6	Generator.prototype.return(value)	6
Generator.prototype[@@toStringTag]	6	Generator.prototype.throw(exception)	6

GeneratorFunction Object			
GeneratorFunction Constructor		GeneratorFunction.prototype.prototype	6
new GeneratorFunction ([arg1[, arg2],] body)	6	GeneratorFunction.prototype[Symbol.toStringTag]	6
Properties of the GeneratorFunction Constructor		GeneratorFunction Instances	
GeneratorFunction.length	6	length	6
GeneratorFunction.prototype	6	name	6
Properties of the GeneratorFunction Prototype Object		prototype	6
GeneratorFunction.prototype.constructor	6		

Promise Object			
Promise Constructor		Promise.resolve(value)	6
new Promise(function(resolve, reject){});	6	Promise.resolve(promise)	6
Properties of the Promise Constructor		Promise.resolve(thenable)	6
Promise.prototype	6	Properties of the Promise Prototype Object	
Promise[@@species]	6	Promise.prototype.constructor	6
Promise.all(iterable)	6	Promise.prototype[@@toStringTag]	6
Promise.race(iterable)	6	Promise.prototype.catch(reject)	6
Promise.reject(reason)	6	Promise.prototype.then(fulfill, rejected)	6

Proxy Object			
Proxy Constructor		handler.get(target, key[, receiver])	6
new Proxy(target, handler)	6	handler.getOwnPropertyDescriptor(target, key)	6
Properties of the Proxy Constructor		handler.getPrototypeOf(target)	6
Proxy.revocable(target, handler)	6	handler.has(target, key)	6
handler		handler.isExtensible(target)	6
handler.apply(target[, this[, [params]]])	6	handler.ownKeys(target)	6
handler.construct(target[, params[, newTarget]])	6	handler.preventExtensions(target)	6
handler.defineProperty(target, key, descriptor)	6	handler.set(target, key, value[, receiver])	6
handler.deleteProperty(target, key)	6	handler.setPrototypeOf(target, prototype)	6

Reflect Object			
Reflect.apply(target[, this[, [params]]])	6	Reflect.has(target, key)	6
Reflect.construct(target[, params[, newTarget]])	6	Reflect.isExtensible(target)	6
Reflect.defineProperty(target, key, descriptor)	6	Reflect.ownKeys(target)	6
Reflect.deleteProperty(target, key)	6	Reflect.preventExtensions(target)	6
Reflect.get(target, key[, receiver])	6	Reflect.set(target, key, value[, receiver])	6
Reflect.getOwnPropertyDescriptor(target, key)	6	Reflect.setPrototypeOf(target, prototype)	6
Reflect.getPrototypeOf(target)	6		