Daftar Metode Bawaan JavaScript

String Methods

charAt	charCodeAt	codePointAt	concat
includes	endsWith	indexOf	lastIndexOf
localeCompare	match	matchAll	normalize
padEnd	padStart	repeat	replace
replaceAll	search	slice	split
startsWith	substring	toLocaleLowerCase	toLocaleUpperCase
toLowerCase	toString	toUpperCase	trim
trimStart	trimEnd	valueOf	

Array Methods

at	concat	copyWithin	entries
every	fill	filter	find
findIndex	findLast	findLastIndex	flat
flatMap	forEach	includes	indexOf
join	keys	lastIndexOf	map
pop	push	reduce	reduceRight
reverse	shift	slice	some
sort	splice	toLocaleString	toReversed
toSorted	toSpliced	toString	unshift
values	with		

Object Methods

assign	create	defineProperties	defineProperty
entries	freeze	fromEntries	getOwnPropertyDescriptor
getOwnPropertyDescriptors	getOwnPropertyNames	getOwnPropertySymbols	getPrototypeOf
hasOwn	is	isExtensible	isFrozen
isSealed	keys	preventExtensions	seal
setPrototypeOf	values		

Function Methods

apply	bind	call	toString

Number Methods

toExponential	toFixed	toLocaleString	toPrecision
toString	valueOf		

Math Methods

abs	acos	acosh	asin
asinh	atan	atan2	atanh
cbrt	ceil	clz32	cos
cosh	exp	expm1	floor
fround	hypot	imul	log
log1p	log2	log10	max
min	pow	random	round
sign	sin	sinh	sqrt
tan	tanh	trunc	

Daftar Metode Bawaan JavaScript

Date Methods

getDate	getDay	getFullYear	getHours
getMilliseconds	getMinutes	getMonth	getSeconds
getTime	getTimezoneOffset	getUTCDate	getUTCDay
getUTCFullYear	getUTCHours	getUTCMilliseconds	getUTCMinutes
getUTCMonth	getUTCSeconds	now	parse
setDate	setFullYear	setHours	setMilliseconds
setMinutes	setMonth	setSeconds	setTime
setUTCDate	setUTCFullYear	setUTCHours	setUTCMilliseconds
setUTCMinutes	setUTCMonth	setUTCSeconds	toDateString
toISOString	toJSON	toLocaleDateString	toLocaleString
toLocaleTimeString	toString	toTimeString	toUTCString
UTC	valueOf		

RegExp Methods

exec	test	toString	compile

JSON Methods

parse	stringify

Promise Methods

all	allSettled	any	race
reject	resolve	then	catch
finally			

Symbol Methods

for	keyFor

Global Functions

eval	isFinite	isNaN	parseFloat
parseInt	decodeURI	decodeURIComponent	encodeURI
encodeURIComponent	escape	unescape	

String Methods

charAt - String.prototype.charAt()

Penjelasan: Mengembalikan karakter pada indeks tertentu dalam string.
Syntax: string.charAt(index)
Contoh:
    "Hello".charAt(1)
Output:
    "e"

charCodeAt - String.prototype.charCodeAt()

Penjelasan: Mengembalikan nilai Unicode dari karakter pada posisi tertentu.
Syntax: string.charCodeAt(index)
Contoh:
    "ABC".charCodeAt(0)
Output:
    65

codePointAt - String.prototype.codePointAt()

Penjelasan: Mengembalikan nilai code point dari karakter Unicode pada posisi tertentu.
Syntax: string.codePointAt(pos)
Contoh:
    "ABC".codePointAt(0)
Output:
    65

concat - String.prototype.concat()

Penjelasan: Menggabungkan dua atau lebih string dan mengembalikan string baru.
Syntax: string.concat(string2, ..., stringN)
Contoh:
    "Hello".concat(" ", "World")
Output:
    "Hello World"

includes - String.prototype.includes()

Penjelasan: Memeriksa apakah string mengandung substring tertentu.
Syntax: string.includes(searchString, position)
Contoh:
    "Hello World".includes("World")
Output:
    true

endsWith - String.prototype.endsWith()

Penjelasan: Memeriksa apakah string berakhir dengan substring tertentu.
Syntax: string.endsWith(searchString, length)
Contoh:
    "Hello".endsWith("lo")
Output:
    true

String Methods

indexOf - String.prototype.indexOf()

Penjelasan: Mengembalikan indeks pertama dari nilai yang ditentukan.
Syntax: string.indexOf(searchValue, fromIndex)
Contoh:

```
"banana".indexOf("a")
```

Output:

```
1
```

lastIndexOf - String.prototype.lastIndexOf()

Penjelasan: Mengembalikan indeks terakhir dari nilai yang ditentukan.
Syntax: string.lastIndexOf(searchValue, fromIndex)
Contoh:

```
"banana".lastIndexOf("a")
```

Output:

```
5
```

localeCompare - String.prototype.localeCompare()

Penjelasan: Membandingkan dua string berdasarkan lokal pengaturan.
Syntax: string.localeCompare(compareString)
Contoh:

```
"a".localeCompare("b")
```

Output:

```
-1
```

match - String.prototype.match()

Penjelasan: Mencocokkan string dengan ekspresi reguler.
Syntax: string.match(regexp)
Contoh:

```
"abc123".match(/\d+/)
```

Output:

```
["123"]
```

matchAll - String.prototype.matchAll()

Penjelasan: Mengembalikan iterator semua hasil cocok dengan ekspresi reguler.
Syntax: string.matchAll(regexp)
Contoh:

```
[... "test1test2".matchAll(/test\d/g)]
```

Output:

```
[["test1"], ["test2"]]
```

normalize - String.prototype.normalize()

Penjelasan: Mengembalikan bentuk Unicode normalisasi string.
Syntax: string.normalize([form])
Contoh:

```
"café".normalize("NFD")
```

Output:

```
"cafe■"
```

String Methods

padEnd - String.prototype.padEnd()

Penjelasan: Mengisi string dari akhir sampai panjang tertentu.
Syntax: string.padEnd(targetLength [, padString])
Contoh:

```
"5".padEnd(3, "0")
```

Output:

```
"500"
```

padStart - String.prototype.padStart()

Penjelasan: Mengisi string dari awal sampai panjang tertentu.
Syntax: string.padStart(targetLength [, padString])
Contoh:

```
"5".padStart(3, "0")
```

Output:

```
"005"
```

repeat - String.prototype.repeat()

Penjelasan: Mengembalikan string baru dengan menyalin string asli sebanyak count.
Syntax: string.repeat(count)
Contoh:

```
"ha".repeat(3)
```

Output:

```
"hahaha"
```

replace - String.prototype.replace()

Penjelasan: Mengganti bagian dari string dengan string lain atau fungsi.
Syntax: string.replace(regexp|substr, newSubStr|function)
Contoh:

```
"Hello world".replace("world", "there")
```

Output:

```
"Hello there"
```

replaceAll - String.prototype.replaceAll()

Penjelasan: Mengganti semua bagian dari string yang cocok.
Syntax: string.replaceAll(regexp|substr, newSubStr|function)
Contoh:

```
"aabbcc".replaceAll("b", "z")
```

Output:

```
"aazzcc"
```

search - String.prototype.search()

Penjelasan: Mencari kecocokan menggunakan ekspresi reguler dan mengembalikan indeks pertama.
Syntax: string.search(regexp)
Contoh:

```
"hello".search(/e/)
```

Output:

```
1
```

String Methods

slice - String.prototype.slice()

Penjelasan: Menyalin bagian dari string dan mengembalikan substring baru.
Syntax: string.slice(beginIndex, endIndex)
Contoh:

```
"hello".slice(1, 4)
```

Output:

```
"ell"
```

split - String.prototype.split()

Penjelasan: Membagi string menjadi array substring.
Syntax: string.split(separator, limit)
Contoh:

```
"a,b,c".split(",")
```

Output:

```
["a", "b", "c"]
```

startsWith - String.prototype.startsWith()

Penjelasan: Memeriksa apakah string dimulai dengan substring tertentu.
Syntax: string.startsWith(searchString, position)
Contoh:

```
"Hello".startsWith("He")
```

Output:

```
true
```

substring - String.prototype.substring()

Penjelasan: Mengembalikan subset karakter dari string.
Syntax: string.substring(indexStart, indexEnd)
Contoh:

```
"hello".substring(1, 4)
```

Output:

```
"ell"
```

toLocaleLowerCase - String.prototype.toLocaleLowerCase()

Penjelasan: Mengubah string menjadi huruf kecil, berdasarkan lokal.
Syntax: string.toLocaleLowerCase()
Contoh:

```
"HELLO".toLocaleLowerCase()
```

Output:

```
"hello"
```

toLocaleUpperCase - String.prototype.toLocaleUpperCase()

Penjelasan: Mengubah string menjadi huruf besar, berdasarkan lokal.
Syntax: string.toLocaleUpperCase()
Contoh:

```
"hello".toLocaleUpperCase()
```

Output:

```
"HELLO"
```

String Methods

toLowerCase - String.prototype.toLowerCase()

Penjelasan: Mengembalikan string dengan semua huruf kecil.
Syntax: string.toLowerCase()
Contoh:

```
"HELLO".toLowerCase()
```

Output:

```
"hello"
```

toUpperCase - String.prototype.toUpperCase()

Penjelasan: Mengembalikan string dengan semua huruf besar.
Syntax: string.toUpperCase()
Contoh:

```
"hello".toUpperCase()
```

Output:

```
"HELLO"
```

toString - String.prototype.toString()

Penjelasan: Mengembalikan representasi string dari objek.
Syntax: string.toString()
Contoh:

```
"abc".toString()
```

Output:

```
"abc"
```

trim - String.prototype.trim()

Penjelasan: Menghapus spasi dari awal dan akhir string.
Syntax: string.trim()
Contoh:

```
"  hello  ".trim()
```

Output:

```
"hello"
```

trimStart - String.prototype.trimStart()

Penjelasan: Menghapus spasi dari awal string.
Syntax: string.trimStart()
Contoh:

```
"  hello".trimStart()
```

Output:

```
"hello"
```

trimEnd - String.prototype.trimEnd()

Penjelasan: Menghapus spasi dari akhir string.
Syntax: string.trimEnd()
Contoh:

```
"hello  ".trimEnd()
```

Output:

```
"hello"
```

String Methods

valueOf - String.prototype.valueOf()

Penjelasan: Mengembalikan nilai primitif dari objek string.
Syntax: string.valueOf()
Contoh:

```
new String("hello").valueOf()
```

Output:

```
"hello"
```

Array Methods (Revisi Lengkap)

at - Array.prototype.at()

Penjelasan: Mengambil elemen pada posisi tertentu dalam array, mendukung indeks negatif.
Syntax: array.at(index)
Contoh:

```
[10, 20, 30, 40].at(-1);
```

Output:

```
40
```

concat - Array.prototype.concat()

Penjelasan: Menggabungkan dua atau lebih array dan mengembalikan array baru.
Syntax: array.concat(value1, value2, ..., valueN)
Contoh:

```
[1, 2].concat([3, 4]);
```

Output:

```
[1, 2, 3, 4]
```

copyWithin - Array.prototype.copyWithin()

Penjelasan: Menyalin sebagian array ke posisi lain dalam array yang sama.
Syntax: array.copyWithin(target, start, end)
Contoh:

```
[1, 2, 3, 4, 5].copyWithin(0, 3);
```

Output:

```
[4, 5, 3, 4, 5]
```

entries - Array.prototype.entries()

Penjelasan: Mengembalikan objek iterator berisi pasangan kunci/nilai untuk setiap indeks array.
Syntax: array.entries()
Contoh:

```
Array.from(['a', 'b'].entries());
```

Output:

```
[[0, 'a'], [1, 'b']]
```

every - Array.prototype.every()

Penjelasan: Mengembalikan true jika semua elemen dalam array lulus uji fungsi.
Syntax: array.every(callback(element, index, array), thisArg)
Contoh:

```
[1, 2, 3].every(x => x > 0);
```

Output:

```
true
```

fill - Array.prototype.fill()

Penjelasan: Mengisi elemen dalam array dengan nilai statis.
Syntax: array.fill(value, start, end)
Contoh:

```
[1, 2, 3].fill(0);
```

Output:

```
[0, 0, 0]
```

Array Methods (Revisi Lengkap)

filter - Array.prototype.filter()

Penjelasan: Membuat array baru dengan semua elemen yang lulus tes dari fungsi yang diberikan.
Syntax: array.filter(callback(element, index, array), thisArg)
Contoh:

```
[1, 2, 3, 4].filter(x => x % 2 === 0);
```

Output:

```
[2, 4]
```

find - Array.prototype.find()

Penjelasan: Mengembalikan nilai elemen pertama yang lulus pengujian fungsi.
Syntax: array.find(callback(element, index, array), thisArg)
Contoh:

```
[5, 12, 8, 130, 44].find(x => x > 10);
```

Output:

```
12
```

findIndex - Array.prototype.findIndex()

Penjelasan: Mengembalikan indeks elemen pertama yang memenuhi fungsi pengujian.
Syntax: array.findIndex(callback(element, index, array), thisArg)
Contoh:

```
[5, 12, 8, 130, 44].findIndex(x => x > 13);
```

Output:

```
3
```

flat - Array.prototype.flat()

Penjelasan: Menggabungkan sub-array ke dalam array utama hingga kedalaman tertentu.
Syntax: array.flat(depth)
Contoh:

```
[1, [2, [3]]].flat(2);
```

Output:

```
[1, 2, 3]
```

flatMap - Array.prototype.flatMap()

Penjelasan: Pertama memetakan setiap elemen menggunakan fungsi lalu diratakan menjadi array baru.
Syntax: array.flatMap(callback(element, index, array), thisArg)
Contoh:

```
[1, 2, 3].flatMap(x => [x, x * 2]);
```

Output:

```
[1, 2, 2, 4, 3, 6]
```

forEach - Array.prototype.forEach()

Penjelasan: Menjalankan fungsi tertentu sekali untuk setiap elemen array.
Syntax: array.forEach(callback(currentValue, index, array), thisArg)
Contoh:

```
let result = [];
[1, 2, 3].forEach(x => result.push(x * 2));
result;
```

Output:

```
[2, 4, 6]
```

Array Methods (Revisi Lengkap)

includes - Array.prototype.includes()

Penjelasan: Memeriksa apakah array mencakup nilai tertentu.
Syntax: array.includes(valueToFind, fromIndex)
Contoh:

```
[1, 2, 3].includes(2);
```

Output:

```
true
```

indexOf - Array.prototype.indexOf()

Penjelasan: Mengembalikan indeks pertama dari elemen yang ditemukan, atau -1 jika tidak ditemukan.
Syntax: array.indexOf(searchElement, fromIndex)
Contoh:

```
[1, 2, 3].indexOf(2);
```

Output:

```
1
```

join - Array.prototype.join()

Penjelasan: Menggabungkan semua elemen array menjadi satu string.
Syntax: array.join(separator)
Contoh:

```
[1, 2, 3].join('-');
```

Output:

```
'1-2-3'
```

keys - Array.prototype.keys()

Penjelasan: Mengembalikan iterator yang berisi kunci untuk setiap indeks dalam array.
Syntax: array.keys()
Contoh:

```
Array.from(['a', 'b'].keys());
```

Output:

```
[0, 1]
```

lastIndexOf - Array.prototype.lastIndexOf()

Penjelasan: Mengembalikan indeks terakhir dari elemen yang ditemukan.
Syntax: array.lastIndexOf(searchElement, fromIndex)
Contoh:

```
[1, 2, 1, 2].lastIndexOf(2);
```

Output:

```
3
```

map - Array.prototype.map()

Penjelasan: Membuat array baru dengan hasil pemanggilan fungsi untuk setiap elemen.
Syntax: array.map(callback(currentValue, index, array), thisArg)
Contoh:

```
[1, 2, 3].map(x => x * 2);
```

Output:

```
[2, 4, 6]
```

Array Methods (Revisi Lengkap)

pop - Array.prototype.pop()

Penjelasan: Menghapus elemen terakhir dari array dan mengembalikannya.
Syntax: array.pop()
Contoh:

```
let arr = [1, 2, 3];
arr.pop();
```

Output:

```
3
```

push - Array.prototype.push()

Penjelasan: Menambahkan satu atau lebih elemen ke akhir array dan mengembalikan panjang baru.
Syntax: array.push(element1, ..., elementN)
Contoh:

```
let arr = [1, 2];
arr.push(3);
```

Output:

```
3
```

reduce - Array.prototype.reduce()

Penjelasan: Mengurangi array menjadi satu nilai dengan menjalankan fungsi pada setiap elemen.
Syntax: array.reduce(callback(accumulator, currentValue, index, array), initialValue)
Contoh:

```
[1, 2, 3, 4].reduce((acc, val) => acc + val, 0);
```

Output:

```
10
```

reduceRight - Array.prototype.reduceRight()

Penjelasan: Sama seperti reduce, tetapi dimulai dari elemen terakhir ke pertama.
Syntax: array.reduceRight(callback, initialValue)
Contoh:

```
[1, 2, 3, 4].reduceRight((acc, val) => acc - val);
```

Output:

```
-2
```

reverse - Array.prototype.reverse()

Penjelasan: Membalik urutan elemen dalam array.
Syntax: array.reverse()
Contoh:

```
[1, 2, 3].reverse();
```

Output:

```
[3, 2, 1]
```

shift - Array.prototype.shift()

Penjelasan: Menghapus elemen pertama dari array dan mengembalikannya.
Syntax: array.shift()
Contoh:

```
let arr = [1, 2, 3];
arr.shift();
```

Output:

```
1
```

Array Methods (Revisi Lengkap)

slice - Array.prototype.slice()

Penjelasan: Mengembalikan salinan sebagian dari array.
Syntax: array.slice(begin, end)
Contoh:

```
[1, 2, 3, 4].slice(1, 3);
```

Output:

```
[2, 3]
```

some - Array.prototype.some()

Penjelasan: Mengembalikan true jika setidaknya satu elemen lulus pengujian fungsi.
Syntax: array.some(callback(element, index, array), thisArg)
Contoh:

```
[1, 2, 3].some(x => x > 2);
```

Output:

```
true
```

sort - Array.prototype.sort()

Penjelasan: Mengurutkan elemen array secara in-place dan mengembalikannya.
Syntax: array.sort([compareFunction])
Contoh:

```
[3, 1, 2].sort();
```

Output:

```
[1, 2, 3]
```

splice - Array.prototype.splice()

Penjelasan: Mengubah isi array dengan menghapus atau menambahkan elemen.
Syntax: array.splice(start, deleteCount, item1, ..., itemN)
Contoh:

```
let arr = [1, 2, 3];
arr.splice(1, 1, 4);
```

Output:

```
[2]
```

toLocaleString - Array.prototype.toLocaleString()

Penjelasan: Mengembalikan string lokal dari elemen array.
Syntax: array.toLocaleString()
Contoh:

```
[1, 'a'].toLocaleString();
```

Output:

```
'1,a'
```

toReversed - Array.prototype.toReversed()

Penjelasan: Mengembalikan salinan array yang dibalik (immutable).
Syntax: array.toReversed()
Contoh:

```
[1, 2, 3].toReversed();
```

Output:

```
[3, 2, 1]
```

Array Methods (Revisi Lengkap)

toSorted - Array.prototype.toSorted()

Penjelasan: Mengembalikan salinan array yang terurut (immutable).
Syntax: array.toSorted()
Contoh:

```
[3, 1, 2].toSorted();
```

Output:

```
[1, 2, 3]
```

toSpliced - Array.prototype.toSpliced()

Penjelasan: Versi immutable dari splice, mengembalikan array baru.
Syntax: array.toSpliced(start, deleteCount, ...items)
Contoh:

```
[1, 2, 3].toSpliced(1, 1, 4);
```

Output:

```
[1, 4, 3]
```

toString - Array.prototype.toString()

Penjelasan: Mengembalikan representasi string dari array.
Syntax: array.toString()
Contoh:

```
[1, 2, 3].toString();
```

Output:

```
'1,2,3'
```

unshift - Array.prototype.unshift()

Penjelasan: Menambahkan elemen ke awal array dan mengembalikan panjang baru array.
Syntax: array.unshift(element1, ..., elementN)
Contoh:

```
let arr = [2, 3];
arr.unshift(1);
```

Output:

```
3
```

values - Array.prototype.values()

Penjelasan: Mengembalikan iterator yang berisi nilai-nilai untuk setiap indeks dalam array.
Syntax: array.values()
Contoh:

```
Array.from([1, 2].values());
```

Output:

```
[1, 2]
```

with - Array.prototype.with()

Penjelasan: Mengembalikan salinan array dengan satu elemen diganti (immutable).
Syntax: array.with(index, value)
Contoh:

```
[1, 2, 3].with(1, 4);
```

Output:

```
[1, 4, 3]
```

Array Methods (Revisi Lengkap)

findLast - Array.prototype.findLast()

Penjelasan: Mengembalikan elemen terakhir yang memenuhi kondisi dari fungsi callback.
Syntax: array.findLast(callback(element, index, array), thisArg)
Contoh:

```
[1, 2, 3, 4, 5].findLast(x => x % 2 === 0);
```

Output:

```
4
```

findLastIndex - Array.prototype.findLastIndex()

Penjelasan: Mengembalikan indeks dari elemen terakhir yang memenuhi kondisi.
Syntax: array.findLastIndex(callback(element, index, array), thisArg)
Contoh:

```
[1, 2, 3, 4, 5].findLastIndex(x => x % 2 === 0);
```

Output:

```
3
```

from - Array.prototype.from()

Penjelasan: Membuat array baru dari array-like atau iterable object.
Syntax: Array.from(arrayLike, mapFn, thisArg)
Contoh:

```
Array.from('abc');
```

Output:

```
['a', 'b', 'c']
```

Object Methods (Lengkap)

assign - Object.assign()

Penjelasan: Menyalin properti enumerable dari satu atau lebih objek sumber ke objek target.
Syntax: Object.assign(target, ...sources)
Contoh:

```
Object.assign({a: 1}, {b: 2});
```

Output:

```
{a: 1, b: 2}
```

create - Object.create()

Penjelasan: Membuat objek baru dengan prototipe dan properti tertentu.
Syntax: Object.create(prototype, propertiesObject)
Contoh:

```
const obj = Object.create({greet() { return 'hi'; }});
obj.greet();
```

Output:

```
'hi'
```

defineProperty - Object.defineProperty()

Penjelasan: Menambahkan atau memodifikasi properti langsung pada objek.
Syntax: Object.defineProperty(obj, prop, descriptor)
Contoh:

```
const obj = {};
Object.defineProperty(obj, 'x', { value: 42 });
obj.x;
```

Output:

```
42
```

entries - Object.entries()

Penjelasan: Mengembalikan array pasangan key-value dari properti enumerable objek.
Syntax: Object.entries(obj)
Contoh:

```
Object.entries({a: 1, b: 2});
```

Output:

```
[['a', 1], ['b', 2]]
```

freeze - Object.freeze()

Penjelasan: Membekukan objek, mencegah modifikasi terhadap properti yang ada.
Syntax: Object.freeze(obj)
Contoh:

```
const obj = {a: 1};
Object.freeze(obj);
obj.a = 2;
obj.a;
```

Output:

```
1
```

Object Methods (Lengkap)

defineProperties - Object.defineProperties()

Penjelasan: Menambahkan beberapa properti ke objek sekaligus.
Syntax: Object.defineProperties(obj, props)
Contoh:

```
const obj = {};
Object.defineProperties(obj, {
  x: { value: 1 },
  y: { value: 2 }
});
[obj.x, obj.y];
```

Output:

```
[1, 2]
```

fromEntries - Object.fromEntries()

Penjelasan: Mengubah array pasangan [key, value] menjadi objek.
Syntax: Object.fromEntries(iterable)
Contoh:

```
Object.fromEntries([['a', 1], ['b', 2]]);
```

Output:

```
{a: 1, b: 2}
```

getOwnPropertyDescriptor - Object.getOwnPropertyDescriptor()

Penjelasan: Mengembalikan deskripsi properti langsung dari objek.
Syntax: Object.getOwnPropertyDescriptor(obj, prop)
Contoh:

```
Object.getOwnPropertyDescriptor({x: 42}, 'x');
```

Output:

```
{ value: 42, writable: true, enumerable: true, configurable: true }
```

getOwnPropertyDescriptors - Object.getOwnPropertyDescriptors()

Penjelasan: Mengembalikan semua deskriptor properti dari objek.
Syntax: Object.getOwnPropertyDescriptors(obj)
Contoh:

```
Object.getOwnPropertyDescriptors({x: 1, y: 2});
```

Output:

```
{ x: {...}, y: {...} }
```

getOwnPropertyNames - Object.getOwnPropertyNames()

Penjelasan: Mengembalikan array nama properti milik langsung objek (termasuk non-enumerable).
Syntax: Object.getOwnPropertyNames(obj)
Contoh:

```
Object.getOwnPropertyNames({a: 1, b: 2});
```

Output:

```
['a', 'b']
```

Object Methods (Lengkap)

getOwnPropertySymbols - Object.getOwnPropertySymbols()

Penjelasan: Mengembalikan array simbol milik langsung objek.
Syntax: Object.getOwnPropertySymbols(obj)
Contoh:

```
const sym = Symbol('s');
const obj = {[sym]: 123};
Object.getOwnPropertySymbols(obj);
```

Output:

```
[Symbol(s)]
```

getPrototypeOf - Object.getPrototypeOf()

Penjelasan: Mengembalikan prototipe dari objek.
Syntax: Object.getPrototypeOf(obj)
Contoh:

```
Object.getPrototypeOf({});
```

Output:

```
Object.prototype
```

hasOwn - Object.hasOwn()

Penjelasan: Memeriksa apakah objek memiliki properti sendiri (bukan dari prototype).
Syntax: Object.hasOwn(obj, prop)
Contoh:

```
Object.hasOwn({a: 1}, 'a');
```

Output:

```
true
```

is - Object.is()

Penjelasan: Membandingkan dua nilai untuk kesamaan (seperti === tapi lebih tepat).
Syntax: Object.is(value1, value2)
Contoh:

```
Object.is(NaN, NaN);
```

Output:

```
true
```

isExtensible - Object.isExtensible()

Penjelasan: Memeriksa apakah properti baru bisa ditambahkan ke objek.
Syntax: Object.isExtensible(obj)
Contoh:

```
Object.isExtensible({});
```

Output:

```
true
```

isFrozen - Object.isFrozen()

Penjelasan: Memeriksa apakah objek dibekukan.
Syntax: Object.isFrozen(obj)
Contoh:

```
Object.isFrozen(Object.freeze({}));
```

Output:

```
true
```

Object Methods (Lengkap)

isSealed - Object.isSealed()

Penjelasan: Memeriksa apakah objek disegel.
Syntax: Object.isSealed(obj)
Contoh:

```
Object.isSealed(Object.seal({}));
```

Output:

```
true
```

keys - Object.keys()

Penjelasan: Mengembalikan array nama properti enumerable.
Syntax: Object.keys(obj)
Contoh:

```
Object.keys({a: 1, b: 2});
```

Output:

```
['a', 'b']
```

preventExtensions - Object.preventExtensions()

Penjelasan: Mencegah penambahan properti baru ke objek.
Syntax: Object.preventExtensions(obj)
Contoh:

```
const obj = {};
Object.preventExtensions(obj);
Object.isExtensible(obj);
```

Output:

```
false
```

seal - Object.seal()

Penjelasan: Menyegel objek agar tidak bisa ditambah atau hapus propertinya.
Syntax: Object.seal(obj)
Contoh:

```
const obj = {a: 1};
Object.seal(obj);
delete obj.a;
obj.a;
```

Output:

```
1
```

setPrototypeOf - Object.setPrototypeOf()

Penjelasan: Mengatur prototipe dari objek.
Syntax: Object.setPrototypeOf(obj, prototype)
Contoh:

```
const proto = {greet() { return 'hi'; }};
const obj = {};
Object.setPrototypeOf(obj, proto);
obj.greet();
```

Output:

```
'hi'
```

Object Methods (Lengkap)

values - Object.values()

Penjelasan: Mengembalikan array dari nilai properti enumerable.
Syntax: Object.values(obj)
Contoh:

```
Object.values({a: 1, b: 2});
```

Output:

```
[1, 2]
```

Function Methods

apply - Function.prototype.apply()

Penjelasan: Memanggil fungsi dengan this tertentu dan argumen dalam bentuk array.
Syntax: func.apply(thisArg, [argsArray])
Contoh:

```
function sum(a, b) { return a + b; }
sum.apply(null, [1, 2]);
```

Output:

```
3
```

bind - Function.prototype.bind()

Penjelasan: Membuat fungsi baru dengan this yang ditetapkan dan argumen awal tetap.
Syntax: func.bind(thisArg, arg1, arg2, ...)
Contoh:

```
function greet(name) { return 'Hi ' + name; }
const hiJohn = greet.bind(null, 'John');
hiJohn();
```

Output:

```
'Hi John'
```

call - Function.prototype.call()

Penjelasan: Memanggil fungsi dengan this tertentu dan argumen terpisah.
Syntax: func.call(thisArg, arg1, arg2, ...)
Contoh:

```
function sayHello(greet) { return greet + ', world'; }
sayHello.call(null, 'Hello');
```

Output:

```
'Hello, world'
```

toString - Function.prototype.toString()

Penjelasan: Mengembalikan representasi string dari fungsi.
Syntax: func.toString()
Contoh:

```
function f() {}
f.toString();
```

Output:

```
'function f() {}'
```

Number Methods

toExponential - Number.prototype.toExponential()

Penjelasan: Mengembalikan string dalam notasi eksponensial.
Syntax: num.toExponential(fractionDigits)
Contoh:

```
let num = 123456;
num.toExponential(2);
```

Output:

```
'1.23e+5'
```

toFixed - Number.prototype.toFixed()

Penjelasan: Mengembalikan string dengan jumlah angka di belakang desimal tetap.
Syntax: num.toFixed(digits)
Contoh:

```
let num = 3.14159;
num.toFixed(2);
```

Output:

```
'3.14'
```

toLocaleString - Number.prototype.toLocaleString()

Penjelasan: Mengembalikan string dengan representasi angka lokal.
Syntax: num.toLocaleString([locales], [options])
Contoh:

```
let num = 1234567.89;
num.toLocaleString('de-DE');
```

Output:

```
'1.234.567,89'
```

toPrecision - Number.prototype.toPrecision()

Penjelasan: Mengembalikan string angka dengan panjang presisi tertentu.
Syntax: num.toPrecision(precision)
Contoh:

```
let num = 123.456;
num.toPrecision(5);
```

Output:

```
'123.46'
```

toString - Number.prototype.toString()

Penjelasan: Mengembalikan representasi string dari angka.
Syntax: num.toString([radix])
Contoh:

```
let num = 255;
num.toString(16);
```

Output:

```
'ff'
```

Number Methods

valueOf - Number.prototype.valueOf()

Penjelasan: Mengembalikan nilai primitif dari objek Number.
Syntax: num.valueOf()
Contoh:

```
let num = new Number(10);
num.valueOf();
```

Output:

```
10
```

Math Methods

abs - Math.abs()

Penjelasan: Mengembalikan nilai absolut dari angka.
Syntax: Math.abs(x)
Contoh:

```
Math.abs(-5);
```

Output:

```
5
```

ceil - Math.ceil()

Penjelasan: Mengembalikan nilai terkecil yang lebih besar atau sama dengan x (dibulatkan ke atas).
Syntax: Math.ceil(x)
Contoh:

```
Math.ceil(4.2);
```

Output:

```
5
```

floor - Math.floor()

Penjelasan: Mengembalikan nilai terbesar yang lebih kecil atau sama dengan x (dibulatkan ke bawah).
Syntax: Math.floor(x)
Contoh:

```
Math.floor(4.8);
```

Output:

```
4
```

round - Math.round()

Penjelasan: Membulatkan angka ke bilangan bulat terdekat.
Syntax: Math.round(x)
Contoh:

```
Math.round(4.5);
```

Output:

```
5
```

sqrt - Math.sqrt()

Penjelasan: Mengembalikan akar kuadrat dari angka.
Syntax: Math.sqrt(x)
Contoh:

```
Math.sqrt(9);
```

Output:

```
3
```

pow - Math.pow()

Penjelasan: Mengembalikan nilai basis dipangkatkan dengan eksponen.
Syntax: Math.pow(base, exponent)
Contoh:

```
Math.pow(2, 3);
```

Output:

```
8
```

Math Methods

max - Math.max()

Penjelasan: Mengembalikan nilai terbesar dari sejumlah argumen.
Syntax: Math.max(x1, x2, ..., xn)
Contoh:

```
Math.max(1, 3, 2);
```

Output:

```
3
```

min - Math.min()

Penjelasan: Mengembalikan nilai terkecil dari sejumlah argumen.
Syntax: Math.min(x1, x2, ..., xn)
Contoh:

```
Math.min(1, 3, 2);
```

Output:

```
1
```

random - Math.random()

Penjelasan: Mengembalikan angka acak antara 0 (inklusif) dan 1 (eksklusif).
Syntax: Math.random()
Contoh:

```
Math.random();
```

Output:

```
0.123... (angka acak)
```

trunc - Math.trunc()

Penjelasan: Menghapus bagian desimal dari angka.
Syntax: Math.trunc(x)
Contoh:

```
Math.trunc(4.9);
```

Output:

```
4
```

Math Methods (Lanjutan)

acos - Math.acos()

Penjelasan: Mengembalikan arc cosine dari x dalam radian.
Syntax: Math.acos(x)
Contoh:

```
Math.acos(1);
```

Output:

```
0
```

asin - Math.asin()

Penjelasan: Mengembalikan arc sine dari x dalam radian.
Syntax: Math.asin(x)
Contoh:

```
Math.asin(0);
```

Output:

```
0
```

atan - Math.atan()

Penjelasan: Mengembalikan arc tangent dari x dalam radian.
Syntax: Math.atan(x)
Contoh:

```
Math.atan(1);
```

Output:

```
0.7853981633974483
```

atan2 - Math.atan2()

Penjelasan: Mengembalikan arc tangent dari rasio y/x dalam radian.
Syntax: Math.atan2(y, x)
Contoh:

```
Math.atan2(1, 1);
```

Output:

```
0.7853981633974483
```

cos - Math.cos()

Penjelasan: Mengembalikan nilai cosine dari x (dalam radian).
Syntax: Math.cos(x)
Contoh:

```
Math.cos(0);
```

Output:

```
1
```

sin - Math.sin()

Penjelasan: Mengembalikan nilai sine dari x (dalam radian).
Syntax: Math.sin(x)
Contoh:

```
Math.sin(Math.PI / 2);
```

Output:

```
1
```

Math Methods (Lanjutan)

tan - Math.tan()

Penjelasan: Mengembalikan nilai tangent dari x (dalam radian).
Syntax: Math.tan(x)
Contoh:

```
Math.tan(0);
```

Output:

```
0
```

log - Math.log()

Penjelasan: Mengembalikan logaritma natural (basis e) dari x.
Syntax: Math.log(x)
Contoh:

```
Math.log(Math.E);
```

Output:

```
1
```

log10 - Math.log10()

Penjelasan: Mengembalikan logaritma basis 10 dari x.
Syntax: Math.log10(x)
Contoh:

```
Math.log10(100);
```

Output:

```
2
```

log2 - Math.log2()

Penjelasan: Mengembalikan logaritma basis 2 dari x.
Syntax: Math.log2(x)
Contoh:

```
Math.log2(8);
```

Output:

```
3
```

Math Methods (Final Batch)

exp - Math.exp()

Penjelasan: Mengembalikan nilai eksponensial dari x (e^x).
Syntax: Math.exp(x)
Contoh:

```
Math.exp(1);
```

Output:

```
2.718281828459045
```

expm1 - Math.expm1()

Penjelasan: Mengembalikan e^x - 1.
Syntax: Math.expm1(x)
Contoh:

```
Math.expm1(1);
```

Output:

```
1.718281828459045
```

sign - Math.sign()

Penjelasan: Mengembalikan tanda dari angka (1, -1, 0, -0, NaN).
Syntax: Math.sign(x)
Contoh:

```
Math.sign(-5);
```

Output:

```
-1
```

clz32 - Math.clz32()

Penjelasan: Mengembalikan jumlah leading zero dalam representasi 32-bit integer.
Syntax: Math.clz32(x)
Contoh:

```
Math.clz32(1);
```

Output:

```
31
```

cbrt - Math.cbrt()

Penjelasan: Mengembalikan akar pangkat tiga dari x.
Syntax: Math.cbrt(x)
Contoh:

```
Math.cbrt(27);
```

Output:

```
3
```

fround - Math.fround()

Penjelasan: Mengembalikan representasi 32-bit float dari angka.
Syntax: Math.fround(x)
Contoh:

```
Math.fround(1.337);
```

Output:

```
1.3370000123977661
```

Math Methods (Final Batch)

hypot - Math.hypot()

Penjelasan: Mengembalikan akar kuadrat dari jumlah kuadrat argumen (teorema Pythagoras).
Syntax: Math.hypot(...values)
Contoh:

```
Math.hypot(3, 4);
```

Output:

```
5
```

imul - Math.imul()

Penjelasan: Melakukan perkalian 32-bit integer.
Syntax: Math.imul(a, b)
Contoh:

```
Math.imul(2, 4);
```

Output:

```
8
```

sinh - Math.sinh()

Penjelasan: Mengembalikan hyperbolic sine dari x.
Syntax: Math.sinh(x)
Contoh:

```
Math.sinh(0);
```

Output:

```
0
```

cosh - Math.cosh()

Penjelasan: Mengembalikan hyperbolic cosine dari x.
Syntax: Math.cosh(x)
Contoh:

```
Math.cosh(0);
```

Output:

```
1
```

tanh - Math.tanh()

Penjelasan: Mengembalikan hyperbolic tangent dari x.
Syntax: Math.tanh(x)
Contoh:

```
Math.tanh(0);
```

Output:

```
0
```

asinh - Math.asinh()

Penjelasan: Mengembalikan arc hyperbolic sine dari x.
Syntax: Math.asinh(x)
Contoh:

```
Math.asinh(1);
```

Output:

```
0.881373587019543
```

Math Methods (Final Batch)

acosh - Math.acosh()

Penjelasan: Mengembalikan arc hyperbolic cosine dari x.
Syntax: Math.acosh(x)
Contoh:

```
Math.acosh(2);
```

Output:

```
1.3169578969248166
```

atanh - Math.atanh()

Penjelasan: Mengembalikan arc hyperbolic tangent dari x.
Syntax: Math.atanh(x)
Contoh:

```
Math.atanh(0.5);
```

Output:

```
0.5493061443340548
```

Date Methods (Lengkap)

UTC - Date.UTC()

Penjelasan: Mengembalikan waktu dalam milidetik berdasarkan UTC.
Syntax: Date.UTC(year, month, day, hours, minutes, seconds, ms)
Contoh:

```
Date.UTC(2023, 0, 15);
```

Output:

```
1673740800000
```

getDate - Date.prototype.getDate()

Penjelasan: Mengembalikan tanggal (1-31) dari objek Date.
Syntax: date.getDate()
Contoh:

```
new Date('2023-01-15').getDate();
```

Output:

```
15
```

getDay - Date.prototype.getDay()

Penjelasan: Mengembalikan hari dalam minggu (0-6), di mana 0 adalah Minggu.
Syntax: date.getDay()
Contoh:

```
new Date('2023-01-15').getDay();
```

Output:

```
0
```

getFullYear - Date.prototype.getFullYear()

Penjelasan: Mengembalikan tahun dari objek Date.
Syntax: date.getFullYear()
Contoh:

```
new Date('2023-01-15').getFullYear();
```

Output:

```
2023
```

getHours - Date.prototype.getHours()

Penjelasan: Mengembalikan jam (0-23) dari objek Date.
Syntax: date.getHours()
Contoh:

```
new Date('2023-01-15T13:45:00').getHours();
```

Output:

```
13
```

getMilliseconds - Date.prototype.getMilliseconds()

Penjelasan: Mengembalikan milidetik (0-999) dari objek Date.
Syntax: date.getMilliseconds()
Contoh:

```
new Date('2023-01-15T13:45:00.123').getMilliseconds();
```

Output:

```
123
```

Date Methods (Lengkap)

getMinutes - Date.prototype.getMinutes()

Penjelasan: Mengembalikan menit (0-59) dari objek Date.
Syntax: date.getMinutes()
Contoh:

```
new Date('2023-01-15T13:45:00').getMinutes();
```

Output:

```
45
```

getMonth - Date.prototype.getMonth()

Penjelasan: Mengembalikan bulan (0-11), di mana 0 adalah Januari.
Syntax: date.getMonth()
Contoh:

```
new Date('2023-01-15').getMonth();
```

Output:

```
0
```

getSeconds - Date.prototype.getSeconds()

Penjelasan: Mengembalikan detik (0-59) dari objek Date.
Syntax: date.getSeconds()
Contoh:

```
new Date('2023-01-15T13:45:30').getSeconds();
```

Output:

```
30
```

getTime - Date.prototype.getTime()

Penjelasan: Mengembalikan waktu dalam milidetik sejak 1 Januari 1970.
Syntax: date.getTime()
Contoh:

```
new Date('1970-01-02').getTime();
```

Output:

```
86400000
```

getTimezoneOffset - Date.prototype.getTimezoneOffset()

Penjelasan: Mengembalikan perbedaan waktu lokal terhadap UTC dalam menit.
Syntax: date.getTimezoneOffset()
Contoh:

```
new Date('2023-01-15').getTimezoneOffset();
```

Output:

```
-420 (misalnya untuk UTC+7)
```

getUTCDate - Date.prototype.getUTCDate()

Penjelasan: Mengembalikan tanggal (1-31) dari waktu UTC.
Syntax: date.getUTCDate()
Contoh:

```
new Date('2023-01-15T00:00:00Z').getUTCDate();
```

Output:

```
15
```

Date Methods (Lengkap)

getUTCDay - Date.prototype.getUTCDay()

Penjelasan: Mengembalikan hari dalam minggu (0-6) dari waktu UTC.
Syntax: date.getUTCDay()
Contoh:

```
new Date('2023-01-15T00:00:00Z').getUTCDay();
```

Output:

```
0
```

getUTCFullYear - Date.prototype.getUTCFullYear()

Penjelasan: Mengembalikan tahun dari waktu UTC.
Syntax: date.getUTCFullYear()
Contoh:

```
new Date('2023-01-15T00:00:00Z').getUTCFullYear();
```

Output:

```
2023
```

getUTCHours - Date.prototype.getUTCHours()

Penjelasan: Mengembalikan jam UTC (0-23).
Syntax: date.getUTCHours()
Contoh:

```
new Date('2023-01-15T13:00:00Z').getUTCHours();
```

Output:

```
13
```

getUTCMilliseconds - Date.prototype.getUTCMilliseconds()

Penjelasan: Mengembalikan milidetik UTC (0-999).
Syntax: date.getUTCMilliseconds()
Contoh:

```
new Date('2023-01-15T13:00:00.456Z').getUTCMilliseconds();
```

Output:

```
456
```

getUTCMinutes - Date.prototype.getUTCMinutes()

Penjelasan: Mengembalikan menit UTC (0-59).
Syntax: date.getUTCMinutes()
Contoh:

```
new Date('2023-01-15T13:45:00Z').getUTCMinutes();
```

Output:

```
45
```

getUTCMonth - Date.prototype.getUTCMonth()

Penjelasan: Mengembalikan bulan UTC (0-11).
Syntax: date.getUTCMonth()
Contoh:

```
new Date('2023-01-15T00:00:00Z').getUTCMonth();
```

Output:

```
0
```

Date Methods (Lengkap)

getUTCSeconds - Date.prototype.getUTCSeconds()

Penjelasan: Mengembalikan detik UTC (0-59).
Syntax: date.getUTCSeconds()
Contoh:

```
new Date('2023-01-15T13:45:30Z').getUTCSeconds();
```

Output:

```
30
```

now - Date.now()

Penjelasan: Mengembalikan waktu saat ini dalam milidetik sejak 1 Januari 1970.
Syntax: Date.now()
Contoh:

```
Date.now();
```

Output:

```
1715184000000 (contoh, nilai bervariasi)
```

parse - Date.parse()

Penjelasan: Mengurai string tanggal dan mengembalikan waktu dalam milidetik.
Syntax: Date.parse(dateString)
Contoh:

```
Date.parse('2023-01-15');
```

Output:

```
1673740800000
```

setDate - Date.prototype.setDate()

Penjelasan: Mengatur tanggal bulan (1-31).
Syntax: date.setDate(day)
Contoh:

```
let d = new Date('2023-01-01');
d.setDate(15);
d.getDate();
```

Output:

```
15
```

setFullYear - Date.prototype.setFullYear()

Penjelasan: Mengatur tahun dari objek Date.
Syntax: date.setFullYear(year)
Contoh:

```
let d = new Date();
d.setFullYear(2020);
d.getFullYear();
```

Output:

```
2020
```

Date Methods (Lengkap)

setHours - Date.prototype.setHours()

Penjelasan: Mengatur jam dari objek Date.
Syntax: date.setHours(hours)
Contoh:

```
let d = new Date('2023-01-15');
d.setHours(10);
d.getHours();
```

Output:

```
10
```

setMilliseconds - Date.prototype.setMilliseconds()

Penjelasan: Mengatur milidetik dari objek Date.
Syntax: date.setMilliseconds(ms)
Contoh:

```
let d = new Date('2023-01-15T13:00:00');
d.setMilliseconds(123);
d.getMilliseconds();
```

Output:

```
123
```

setMinutes - Date.prototype.setMinutes()

Penjelasan: Mengatur menit dari objek Date.
Syntax: date.setMinutes(minutes)
Contoh:

```
let d = new Date('2023-01-15T13:00:00');
d.setMinutes(45);
d.getMinutes();
```

Output:

```
45
```

setMonth - Date.prototype.setMonth()

Penjelasan: Mengatur bulan (0-11) dari objek Date.
Syntax: date.setMonth(month)
Contoh:

```
let d = new Date('2023-01-15');
d.setMonth(5);
d.getMonth();
```

Output:

```
5
```

setSeconds - Date.prototype.setSeconds()

Penjelasan: Mengatur detik dari objek Date.
Syntax: date.setSeconds(seconds)
Contoh:

```
let d = new Date('2023-01-15T13:00:00');
d.setSeconds(30);
d.getSeconds();
```

Output:

```
30
```

Date Methods (Lengkap)

setTime - Date.prototype.setTime()

Penjelasan: Mengatur waktu berdasarkan milidetik sejak 1 Jan 1970.
Syntax: date.setTime(milliseconds)
Contoh:

```
let d = new Date();
d.setTime(86400000);
d.getDate();
```

Output:

```
2
```

setUTCDate - Date.prototype.setUTCDate()

Penjelasan: Mengatur tanggal UTC (1-31).
Syntax: date.setUTCDate(day)
Contoh:

```
let d = new Date('2023-01-01T00:00:00Z');
d.setUTCDate(15);
d.getUTCDate();
```

Output:

```
15
```

setUTCFullYear - Date.prototype.setUTCFullYear()

Penjelasan: Mengatur tahun UTC.
Syntax: date.setUTCFullYear(year)
Contoh:

```
let d = new Date();
d.setUTCFullYear(2022);
d.getUTCFullYear();
```

Output:

```
2022
```

setUTCHours - Date.prototype.setUTCHours()

Penjelasan: Mengatur jam UTC.
Syntax: date.setUTCHours(hours)
Contoh:

```
let d = new Date();
d.setUTCHours(10);
d.getUTCHours();
```

Output:

```
10
```

setUTCMilliseconds - Date.prototype.setUTCMilliseconds()

Penjelasan: Mengatur milidetik UTC.
Syntax: date.setUTCMilliseconds(ms)
Contoh:

```
let d = new Date();
d.setUTCMilliseconds(200);
d.getUTCMilliseconds();
```

Output:

```
200
```

Date Methods (Lengkap)

setUTCMinutes - Date.prototype.setUTCMinutes()

Penjelasan: Mengatur menit UTC.
Syntax: date.setUTCMinutes(minutes)
Contoh:

```
let d = new Date();
d.setUTCMinutes(40);
d.getUTCMinutes();
```

Output:

```
40
```

setUTCMonth - Date.prototype.setUTCMonth()

Penjelasan: Mengatur bulan UTC (0-11).
Syntax: date.setUTCMonth(month)
Contoh:

```
let d = new Date();
d.setUTCMonth(6);
d.getUTCMonth();
```

Output:

```
6
```

setUTCSeconds - Date.prototype.setUTCSeconds()

Penjelasan: Mengatur detik UTC.
Syntax: date.setUTCSeconds(seconds)
Contoh:

```
let d = new Date();
d.setUTCSeconds(30);
d.getUTCSeconds();
```

Output:

```
30
```

toDateString - Date.prototype.toDateString()

Penjelasan: Mengembalikan bagian tanggal sebagai string.
Syntax: date.toDateString()
Contoh:

```
new Date('2023-01-15').toDateString();
```

Output:

```
'Sun Jan 15 2023'
```

toISOString - Date.prototype.toISOString()

Penjelasan: Mengembalikan string ISO (format standar internasional).
Syntax: date.toISOString()
Contoh:

```
new Date('2023-01-15T00:00:00Z').toISOString();
```

Output:

```
'2023-01-15T00:00:00.000Z'
```

Date Methods (Lengkap)

toJSON - Date.prototype.toJSON()

Penjelasan: Mengembalikan string ISO sebagai representasi JSON dari Date.
Syntax: date.toJSON()
Contoh:

```
new Date('2023-01-15T00:00:00Z').toJSON();
```

Output:

```
'2023-01-15T00:00:00.000Z'
```

toLocaleDateString - Date.prototype.toLocaleDateString()

Penjelasan: Mengembalikan tanggal dalam format lokal.
Syntax: date.toLocaleDateString([locales], [options])
Contoh:

```
new Date('2023-01-15').toLocaleDateString('id-ID');
```

Output:

```
'15/1/2023'
```

toLocaleString - Date.prototype.toLocaleString()

Penjelasan: Mengembalikan representasi lokal dari tanggal dan waktu.
Syntax: date.toLocaleString()
Contoh:

```
new Date('2023-01-15T13:30:00').toLocaleString('id-ID');
```

Output:

```
'15/1/2023 13.30.00'
```

toLocaleTimeString - Date.prototype.toLocaleTimeString()

Penjelasan: Mengembalikan string waktu dalam format lokal.
Syntax: date.toLocaleTimeString([locales], [options])
Contoh:

```
new Date('2023-01-15T13:45:00').toLocaleTimeString('id-ID');
```

Output:

```
'13.45.00'
```

toString - Date.prototype.toString()

Penjelasan: Mengembalikan string representasi dari objek Date.
Syntax: date.toString()
Contoh:

```
new Date('2023-01-15').toString();
```

Output:

```
'Sun Jan 15 2023 ...'
```

toTimeString - Date.prototype.toTimeString()

Penjelasan: Mengembalikan bagian waktu dari Date sebagai string.
Syntax: date.toTimeString()
Contoh:

```
new Date('2023-01-15T13:45:00').toTimeString();
```

Output:

```
'13:45:00 GMT+0000 (Coordinated Universal Time)'
```

Date Methods (Lengkap)

toUTCString - Date.prototype.toUTCString()

Penjelasan: Mengembalikan representasi string waktu dalam UTC.
Syntax: date.toUTCString()
Contoh:

```
new Date('2023-01-15T00:00:00Z').toUTCString();
```

Output:

```
'Sun, 15 Jan 2023 00:00:00 GMT'
```

valueOf - Date.prototype.valueOf()

Penjelasan: Mengembalikan nilai primitif dari objek Date dalam milidetik.
Syntax: date.valueOf()
Contoh:

```
new Date('1970-01-02').valueOf();
```

Output:

```
86400000
```

RegExp Methods

exec - RegExp.prototype.exec()

Penjelasan: Menjalankan pencocokan RegExp terhadap string dan mengembalikan hasil sebagai array.
Syntax: regexp.exec(str)
Contoh:

```
const regex = /a(b+)/;
regex.exec('abbc');
```

Output:

```
['abb', 'bb']
```

test - RegExp.prototype.test()

Penjelasan: Mengembalikan true jika ada kecocokan pola dalam string, false jika tidak.
Syntax: regexp.test(str)
Contoh:

```
const regex = /hello/;
regex.test('hello world');
```

Output:

```
true
```

toString - RegExp.prototype.toString()

Penjelasan: Mengembalikan representasi string dari ekspresi reguler.
Syntax: regexp.toString()
Contoh:

```
const regex = /abc/i;
regex.toString();
```

Output:

```
'/abc/i'
```

compile - RegExp.prototype.compile()

Penjelasan: (Deprecated) Menyusun ulang ekspresi reguler dengan pola dan flag baru.
Syntax: regexp.compile(pattern, flags)
Contoh:

```
const regex = /abc/;
regex.compile('xyz');
regex.toString();
```

Output:

```
'/xyz/'
```

JSON Methods

parse - JSON.parse()

Penjelasan: Menguraikan string JSON dan mengubahnya menjadi objek JavaScript.
Syntax: JSON.parse(text[, reviver])
Contoh:

```
JSON.parse('{"a":1, "b":2}');
```

Output:

```
{ a: 1, b: 2 }
```

stringify - JSON.stringify()

Penjelasan: Mengubah objek JavaScript menjadi string JSON.
Syntax: JSON.stringify(value[, replacer[, space]])
Contoh:

```
JSON.stringify({x: 10, y: 20});
```

Output:

```
'{"x":10,"y":20}'
```

Promise Methods

all - Promise.all()

Penjelasan: Mengembalikan satu Promise yang resolved ketika semua promise dalam iterable resolved, atau r
Syntax: Promise.all(iterable)
Contoh:

```
Promise.all([Promise.resolve(1), Promise.resolve(2)]).then(console.log);
```

Output:

```
[1, 2]
```

allSettled - Promise.allSettled()

Penjelasan: Mengembalikan promise yang selesai ketika semua promise dalam iterable selesai (fulfilled atau re
Syntax: Promise.allSettled(iterable)
Contoh:

```
Promise.allSettled([Promise.resolve(1), Promise.reject('err')])
  .then(console.log);
```

Output:

```
[{status: 'fulfilled', value: 1}, {status: 'rejected', reason: 'err'}]
```

any - Promise.any()

Penjelasan: Mengembalikan promise yang fulfilled pertama, atau rejected jika semua promise rejected.
Syntax: Promise.any(iterable)
Contoh:

```
Promise.any([Promise.reject('x'), Promise.resolve('y')]).then(console.log);
```

Output:

```
'y'
```

race - Promise.race()

Penjelasan: Mengembalikan promise pertama yang selesai (fulfilled atau rejected).
Syntax: Promise.race(iterable)
Contoh:

```
Promise.race([Promise.resolve(1), Promise.reject('err')]);
```

Output:

```
1
```

reject - Promise.reject()

Penjelasan: Mengembalikan promise yang langsung ditolak dengan alasan yang diberikan.
Syntax: Promise.reject(reason)
Contoh:

```
Promise.reject('error');
```

Output:

```
Promise {<rejected>: 'error'}
```

resolve - Promise.resolve()

Penjelasan: Mengembalikan promise yang langsung diselesaikan dengan nilai yang diberikan.
Syntax: Promise.resolve(value)
Contoh:

```
Promise.resolve(42);
```

Output:

```
Promise {<fulfilled>: 42}
```

Promise Methods

then - Promise.prototype.then()

Penjelasan: Menentukan callback untuk saat promise fulfilled atau rejected.
Syntax: promise.then(onFulfilled, onRejected)
Contoh:

```
Promise.resolve(1).then(val => val + 1);
```

Output:

```
2
```

catch - Promise.prototype.catch()

Penjelasan: Menangani penolakan (rejection) dari promise.
Syntax: promise.catch(onRejected)
Contoh:

```
Promise.reject('err').catch(e => 'caught: ' + e);
```

Output:

```
'caught: err'
```

finally - Promise.prototype.finally()

Penjelasan: Menentukan callback untuk dipanggil saat promise selesai (fulfilled atau rejected).
Syntax: promise.finally(onFinally)
Contoh:

```
Promise.resolve('done').finally(() => console.log('selesai'));
```

Output:

```
'selesai' (dari finally), 'done' (dari resolve)
```

Symbol Methods

Symbol()

Penjelasan: Membuat symbol baru yang unik dengan deskripsi opsional.
Syntax: Symbol(description)
Contoh:

```
const sym = Symbol('id');
sym.toString();
```

Output:

```
'Symbol(id)'
```

for - Symbol.for()

Penjelasan: Mengakses atau membuat symbol global dengan key tertentu.
Syntax: Symbol.for(key)
Contoh:

```
Symbol.for('shared') === Symbol.for('shared');
```

Output:

```
true
```

keyFor - Symbol.keyFor()

Penjelasan: Mengembalikan key dari symbol global.
Syntax: Symbol.keyFor(symbol)
Contoh:

```
const sym = Symbol.for('globalKey');
Symbol.keyFor(sym);
```

Output:

```
'globalKey'
```

Global Functions (Lengkap)

decodeURI

Penjelasan: Mendekode URI yang telah dikodekan oleh encodeURI.
Syntax: decodeURI(encodedURI)
Contoh:

```
decodeURI('https%3A%2F%2Fexample.com');
```

Output:

```
'https://example.com'
```

decodeURIComponent

Penjelasan: Mendekode komponen URI yang dikodekan oleh encodeURIComponent.
Syntax: decodeURIComponent(encodedURIComponent)
Contoh:

```
decodeURIComponent('%E4%BD%A0%E5%A5%BD');
```

Output:

```
'■■'
```

encodeURI

Penjelasan: Mengkodekan URI dengan mengecualikan karakter tertentu.
Syntax: encodeURI(uri)
Contoh:

```
encodeURI('https://example.com/a file');
```

Output:

```
'https://example.com/a%20file'
```

encodeURIComponent

Penjelasan: Mengkodekan komponen URI.
Syntax: encodeURIComponent(uriComponent)
Contoh:

```
encodeURIComponent('■■');
```

Output:

```
'%E4%BD%A0%E5%A5%BD'
```

eval

Penjelasan: Mengeksekusi kode JavaScript dari string.
Syntax: eval(string)
Contoh:

```
eval('2 + 2');
```

Output:

```
4
```

isFinite

Penjelasan: Mengembalikan true jika nilai adalah bilangan hingga.
Syntax: isFinite(value)
Contoh:

```
isFinite(10);
```

Output:

```
true
```

Global Functions (Lengkap)

isNaN

Penjelasan: Mengembalikan true jika nilai adalah NaN.
Syntax: isNaN(value)
Contoh:

```
isNaN(NaN);
```

Output:

```
true
```

parseFloat

Penjelasan: Menguraikan argumen string dan mengembalikan angka floating point.
Syntax: parseFloat(string)
Contoh:

```
parseFloat('3.14');
```

Output:

```
3.14
```

parseInt

Penjelasan: Menguraikan argumen string dan mengembalikan bilangan bulat.
Syntax: parseInt(string, radix)
Contoh:

```
parseInt('10', 2);
```

Output:

```
2
```

escape

Penjelasan: (Deprecated) Mengembalikan string dalam bentuk escaped (karakter non-ASCII diubah menjadi %
Syntax: escape(string)
Contoh:

```
escape('Hello World!');
```

Output:

```
'Hello%20World%21'
```

unescape

Penjelasan: (Deprecated) Mendekode string escaped kembali ke bentuk aslinya.
Syntax: unescape(string)
Contoh:

```
unescape('Hello%20World%21');
```

Output:

```
'Hello World!'
```

Daftar Metode DOM JavaScript

1. Document Methods
Digunakan untuk mengambil elemen dari DOM.

getElementById	querySelectorAll	adoptNode
getElementsByClassName	createElement	open
getElementsByTagName	createTextNode	close
getElementsByName	createDocumentFragment	write
querySelector	importNode	writeln

2. Element Methods
Manipulasi elemen HTML secara langsung.

setAttribute	toggleAttribute	closest
getAttribute	insertAdjacentHTML	matches
removeAttribute	insertAdjacentElement	
hasAttribute	insertAdjacentText	

3. Node Methods
Digunakan untuk manipulasi struktur DOM.

appendChild	hasChildNodes	compareDocumentPosition
removeChild	normalize	contains
replaceChild	isEqualNode	
cloneNode	isSameNode	

4. Event Methods
Berhubungan dengan penanganan event.

addEventListener	removeEventListener	dispatchEvent

5. Traversal & Navigation
Navigasi antar node di dalam DOM.

parentNode	nextSibling	lastElementChild
childNodes	previousSibling	nextElementSibling
firstChild	children	previousElementSibling
lastChild	firstElementChild	

6. Attribute Methods

getAttribute	removeAttribute
setAttribute	hasAttribute

7. Style & Class Methods

Mengatur gaya dan class elemen.

style	classList.toggle	getComputedStyle
classList.add	classList.contains	
classList.remove	classList.replace	

8. Form & Input Methods

Khusus untuk elemen formulir.

submit	blur	checkValidity
reset	select	reportValidity
focus	setCustomValidity	

9. Selection & Range Methods

Digunakan untuk manipulasi teks terpilih.

getSelection	range.setEnd	range.insertNode
createRange	range.selectNode	
range.setStart	range.deleteContents	

10. Window & Screen Methods

Manipulasi jendela browser.

alert	setInterval	scrollBy
confirm	clearTimeout	print
prompt	clearInterval	open
setTimeout	scrollTo	close

Document Methods

getElementById - document.getElementById(id)

Penjelasan: Mengembalikan elemen dengan atribut id yang sesuai.
Syntax: document.getElementById(id)
Contoh:

```
document.getElementById('header');
```

Output:

```
<div id="header">...</div>
```

getElementsByClassName - document.getElementsByClassName(class)

Penjelasan: Mengembalikan koleksi semua elemen dengan nama class tertentu.
Syntax: document.getElementsByClassName(className)
Contoh:

```
document.getElementsByClassName('menu');
```

Output:

```
HTMLCollection [<ul class="menu">...</ul>]
```

getElementsByTagName - document.getElementsByTagName(tag)

Penjelasan: Mengembalikan koleksi semua elemen dengan tag tertentu.
Syntax: document.getElementsByTagName(tagName)
Contoh:

```
document.getElementsByTagName('p');
```

Output:

```
HTMLCollection [<p>...</p>, <p>...</p>]
```

querySelector - document.querySelector(selector)

Penjelasan: Mengembalikan elemen pertama yang cocok dengan selector CSS.
Syntax: document.querySelector(selectors)
Contoh:

```
document.querySelector('.menu');
```

Output:

```
<ul class="menu">...</ul>
```

querySelectorAll - document.querySelectorAll(selector)

Penjelasan: Mengembalikan semua elemen yang cocok dengan selector CSS.
Syntax: document.querySelectorAll(selectors)
Contoh:

```
document.querySelectorAll('li.active');
```

Output:

```
NodeList [<li class="active">Item</li>]
```

Document Methods

createElement - document.createElement(tagName)

Penjelasan: Membuat elemen HTML berdasarkan tag yang diberikan.
Syntax: document.createElement(tagName)
Contoh:

```
let div = document.createElement('div');
```

Output:

```
<div></div>
```

createTextNode - document.createTextNode(text)

Penjelasan: Membuat node teks dengan isi teks yang ditentukan.
Syntax: document.createTextNode(text)
Contoh:

```
let text = document.createTextNode('Hello');
```

Output:

```
#text "Hello"
```

createDocumentFragment - document.createDocumentFragment()

Penjelasan: Membuat node fragment yang dapat menyimpan node sebelum dimasukkan ke DOM.
Syntax: document.createDocumentFragment()
Contoh:

```
let fragment = document.createDocumentFragment();
```

Output:

```
DocumentFragment {}
```

importNode - document.importNode(node, deep)

Penjelasan: Mengimpor node dari dokumen lain ke dokumen saat ini.
Syntax: document.importNode(node, deep)
Contoh:

```
// assume xmlDoc is another document
let imported = document.importNode(xmlDoc.documentElement, true);
```

Output:

```
Node yang diimpor ke dokumen saat ini
```

adoptNode - document.adoptNode(node)

Penjelasan: Mengambil alih kepemilikan node dari dokumen lain ke dokumen saat ini.
Syntax: document.adoptNode(node)
Contoh:

```
// assume foreignNode from iframe
document.adoptNode(foreignNode);
```

Output:

```
Node sekarang dimiliki oleh dokumen saat ini
```

Document Methods

open - document.open()

Penjelasan: Membuka dokumen untuk ditulis ulang (jarang digunakan).
Syntax: document.open()
Contoh:

```
document.open(); document.write('Hello'); document.close();
```

Output:

```
Halaman menampilkan: Hello
```

close - document.close()

Penjelasan: Menutup stream dokumen yang dibuka dengan document.open().
Syntax: document.close()
Contoh:

```
document.open(); document.write('Hi'); document.close();
```

Output:

```
Halaman menampilkan: Hi
```

write - document.write(content)

Penjelasan: Menulis konten ke dokumen secara langsung.
Syntax: document.write(content)
Contoh:

```
document.write('<h1>Hello</h1>');
```

Output:

```
<h1>Hello</h1>
```

writeln - document.writeln(content)

Penjelasan: Menulis konten ke dokumen dan menambahkan newline di akhir.
Syntax: document.writeln(content)
Contoh:

```
document.writeln('Line 1');
```

Output:

```
Line 1
```

Element Methods

setAttribute - element.setAttribute(name, value)

Penjelasan: Menetapkan nilai atribut pada elemen.
Syntax: element.setAttribute(name, value)
Contoh:

```
document.querySelector('div').setAttribute('class', 'highlight');
```

Output:

```
<div class="highlight"></div>
```

getAttribute - element.getAttribute(name)

Penjelasan: Mengambil nilai dari atribut tertentu pada elemen.
Syntax: element.getAttribute(name)
Contoh:

```
<a href="https://example.com"></a>
let link = document.querySelector('a');
link.getAttribute('href');
```

Output:

```
"https://example.com"
```

removeAttribute - element.removeAttribute(name)

Penjelasan: Menghapus atribut dari elemen.
Syntax: element.removeAttribute(name)
Contoh:

```
document.querySelector('div').removeAttribute('style');
```

Output:

```
<div></div>
```

hasAttribute - element.hasAttribute(name)

Penjelasan: Mengembalikan true jika elemen memiliki atribut tertentu.
Syntax: element.hasAttribute(name)
Contoh:

```
<img alt="image" />
document.querySelector('img').hasAttribute('alt');
```

Output:

```
true
```

toggleAttribute - element.toggleAttribute(name)

Penjelasan: Menambah atau menghapus atribut tergantung kondisi keberadaan sebelumnya.
Syntax: element.toggleAttribute(name)
Contoh:

```
const el = document.querySelector('#box');
el.toggleAttribute('hidden');
```

Output:

```
<div id="box" hidden></div> atau <div id="box"></div>
```

Element Methods

insertAdjacentHTML - element.insertAdjacentHTML(position, text)

Penjelasan: Menyisipkan HTML pada posisi relatif terhadap elemen.
Syntax: element.insertAdjacentHTML(position, html)
Contoh:

```
el.insertAdjacentHTML('beforeend', '<p>Hello</p>');
```

Output:

```
<div>...<p>Hello</p></div>
```

insertAdjacentElement - element.insertAdjacentElement(position, element)

Penjelasan: Menyisipkan elemen ke posisi relatif terhadap elemen lain.
Syntax: element.insertAdjacentElement(position, element)
Contoh:

```
el.insertAdjacentElement('afterbegin', document.createElement('span'));
```

Output:

```
Elemen <span> ditambahkan sebagai anak pertama
```

insertAdjacentText - element.insertAdjacentText(position, text)

Penjelasan: Menyisipkan teks ke posisi relatif terhadap elemen.
Syntax: element.insertAdjacentText(position, text)
Contoh:

```
el.insertAdjacentText('beforeend', 'World');
```

Output:

```
teks 'World' disisipkan ke akhir isi elemen
```

closest - element.closest(selector)

Penjelasan: Mengembalikan elemen terdekat (teratas) yang cocok dengan selector.
Syntax: element.closest(selector)
Contoh:

```
el.closest('.container');
```

Output:

```
<div class="container">...</div>
```

matches - element.matches(selector)

Penjelasan: Memeriksa apakah elemen cocok dengan selector yang diberikan.
Syntax: element.matches(selector)
Contoh:

```
el.matches('.selected');
```

Output:

```
true / false
```

Node Methods

appendChild - node.appendChild(child)

Penjelasan: Menambahkan node sebagai anak terakhir dari node induk.
Syntax: node.appendChild(child)
Contoh:

```
let p = document.createElement('p');
document.body.appendChild(p);
```

Output:

```
<body>...<p></p></body>
```

removeChild - node.removeChild(child)

Penjelasan: Menghapus anak node dari node induk.
Syntax: node.removeChild(child)
Contoh:

```
let el = document.getElementById('item');
el.parentNode.removeChild(el);
```

Output:

```
Elemen dihapus dari DOM
```

replaceChild - node.replaceChild(newChild, oldChild)

Penjelasan: Mengganti anak node lama dengan yang baru.
Syntax: node.replaceChild(newChild, oldChild)
Contoh:

```
let newP = document.createElement('p');
parent.replaceChild(newP, oldChild);
```

Output:

```
Node baru menggantikan node lama
```

cloneNode - node.cloneNode(deep)

Penjelasan: Mengkloning node, jika deep true termasuk semua anaknya.
Syntax: node.cloneNode(deep)
Contoh:

```
let clone = document.getElementById('box').cloneNode(true);
```

Output:

```
Salinan elemen dengan semua anak
```

hasChildNodes - node.hasChildNodes()

Penjelasan: Mengembalikan true jika node memiliki node anak.
Syntax: node.hasChildNodes()
Contoh:

```
document.body.hasChildNodes();
```

Output:

```
true
```

Node Methods

normalize - node.normalize()

Penjelasan: Menggabungkan node teks berurutan menjadi satu.
Syntax: node.normalize()
Contoh:

```
element.normalize();
```

Output:
Node teks digabungkan

isEqualNode - node.isEqualNode(otherNode)

Penjelasan: Memeriksa apakah dua node identik secara struktur dan konten.
Syntax: node.isEqualNode(otherNode)
Contoh:

```
node1.isEqualNode(node2);
```

Output:
true / false

isSameNode - node.isSameNode(otherNode)

Penjelasan: Memeriksa apakah dua node merujuk ke objek yang sama.
Syntax: node.isSameNode(otherNode)
Contoh:

```
node1.isSameNode(node2);
```

Output:
true / false

compareDocumentPosition - node.compareDocumentPosition(other)

Penjelasan: Menentukan posisi dokumen relatif terhadap node lain.
Syntax: node.compareDocumentPosition(other)
Contoh:

```
nodeA.compareDocumentPosition(nodeB);
```

Output:
2, 4, 8, 16, dst (bitmask)

contains - node.contains(other)

Penjelasan: Mengembalikan true jika node lain adalah keturunan dari node ini.
Syntax: node.contains(other)
Contoh:

```
parent.contains(child);
```

Output:
true / false

Event Methods

addEventListener - element.addEventListener(type, listener)

Penjelasan: Menambahkan fungsi event listener untuk menangani event tertentu.
Syntax: element.addEventListener(event, function)
Contoh:

```
document.querySelector('button').addEventListener('click', () => alert('Clicked!'));
```

Output:
Menampilkan alert saat tombol diklik

removeEventListener - element.removeEventListener(type, listener)

Penjelasan: Menghapus fungsi event listener yang telah ditambahkan sebelumnya.
Syntax: element.removeEventListener(event, function)
Contoh:

```
const handler = () => alert('Hi');
button.addEventListener('click', handler);
button.removeEventListener('click', handler);
```

Output:
Fungsi handler tidak akan dipanggil saat tombol diklik

dispatchEvent - element.dispatchEvent(event)

Penjelasan: Memicu event secara manual pada elemen.
Syntax: element.dispatchEvent(event)
Contoh:

```
const event = new Event('click');
document.querySelector('button').dispatchEvent(event);
```

Output:
Memicu semua handler yang terkait event tersebut

Traversal & Navigation Methods

parentNode - node.parentNode

Penjelasan: Mengembalikan node induk dari node saat ini.
Syntax: node.parentNode
Contoh:

```
let parent = document.querySelector('li').parentNode;
```

Output:

```
<ul>...</ul>
```

childNodes - node.childNodes

Penjelasan: Mengembalikan NodeList dari semua anak (termasuk teks dan komentar).
Syntax: node.childNodes
Contoh:

```
let children = document.body.childNodes;
```

Output:

```
NodeList(…) [text, div, text, script, ...]
```

firstChild - node.firstChild

Penjelasan: Mengembalikan node anak pertama dari node.
Syntax: node.firstChild
Contoh:

```
let first = document.body.firstChild;
```

Output:

```
#text atau <div>
```

lastChild - node.lastChild

Penjelasan: Mengembalikan node anak terakhir dari node.
Syntax: node.lastChild
Contoh:

```
let last = document.body.lastChild;
```

Output:

```
#text atau <script>
```

nextSibling - node.nextSibling

Penjelasan: Mengembalikan node saudara setelah node ini (termasuk teks).
Syntax: node.nextSibling
Contoh:

```
let next = document.querySelector('div').nextSibling;
```

Output:

```
#text atau elemen
```

Traversal & Navigation Methods

previousSibling - node.previousSibling

Penjelasan: Mengembalikan node saudara sebelum node ini.
Syntax: node.previousSibling
Contoh:

```
let prev = document.querySelector('div').previousSibling;
```

Output:

```
#text atau elemen
```

children - element.children

Penjelasan: Mengembalikan HTMLCollection dari anak elemen saja.
Syntax: element.children
Contoh:

```
let kids = document.body.children;
```

Output:

```
HTMLCollection [header, main, footer]
```

firstElementChild - element.firstElementChild

Penjelasan: Mengembalikan elemen anak pertama.
Syntax: element.firstElementChild
Contoh:

```
let firstEl = document.body.firstElementChild;
```

Output:

```
<header>...</header>
```

lastElementChild - element.lastElementChild

Penjelasan: Mengembalikan elemen anak terakhir.
Syntax: element.lastElementChild
Contoh:

```
let lastEl = document.body.lastElementChild;
```

Output:

```
<footer>...</footer>
```

nextElementSibling - element.nextElementSibling

Penjelasan: Mengembalikan elemen setelah elemen saat ini.
Syntax: element.nextElementSibling
Contoh:

```
let nextEl = document.querySelector('li').nextElementSibling;
```

Output:

```
<li>Item berikutnya</li>
```

Traversal & Navigation Methods

previousElementSibling - element.previousElementSibling

Penjelasan: Mengembalikan elemen sebelum elemen saat ini.
Syntax: element.previousElementSibling
Contoh:

```
let prevEl = document.querySelector('li').previousElementSibling;
```

Output:

```
<li>Item sebelumnya</li>
```

Attribute Methods

getAttribute - element.getAttribute(name)

Penjelasan: Mengambil nilai atribut dari elemen.
Syntax: element.getAttribute(name)
Contoh:

```
<div id="box"></div>
document.getElementById('box').getAttribute('id');
```

Output:

```
"box"
```

setAttribute - element.setAttribute(name, value)

Penjelasan: Menetapkan atau memperbarui nilai atribut.
Syntax: element.setAttribute(name, value)
Contoh:

```
let el = document.getElementById('box');
el.setAttribute('data-type', 'info');
```

Output:

```
<div id="box" data-type="info"></div>
```

removeAttribute - element.removeAttribute(name)

Penjelasan: Menghapus atribut dari elemen.
Syntax: element.removeAttribute(name)
Contoh:

```
let el = document.getElementById('box');
el.removeAttribute('id');
```

Output:

```
<div></div>
```

hasAttribute - element.hasAttribute(name)

Penjelasan: Mengembalikan true jika elemen memiliki atribut tertentu.
Syntax: element.hasAttribute(name)
Contoh:

```
let el = document.getElementById('box');
el.hasAttribute('id');
```

Output:

```
true
```

Style & Class Methods

style - element.style.property

Penjelasan: Digunakan untuk mengatur atau mengambil gaya inline pada elemen.
Syntax: element.style.property
Contoh:

```
document.getElementById('box').style.backgroundColor = 'red';
```

Output:

```
<div id="box" style="background-color: red;"></div>
```

getComputedStyle - getComputedStyle(element)

Penjelasan: Mengembalikan semua gaya CSS yang dihitung dari elemen.
Syntax: getComputedStyle(element)
Contoh:

```
let styles = getComputedStyle(document.getElementById('box'));
styles.backgroundColor;
```

Output:

```
"rgb(255, 0, 0)"
```

classList.add - element.classList.add(className)

Penjelasan: Menambahkan class ke elemen.
Syntax: element.classList.add(className)
Contoh:

```
document.getElementById('box').classList.add('active');
```

Output:

```
<div id="box" class="active"></div>
```

classList.remove - element.classList.remove(className)

Penjelasan: Menghapus class dari elemen.
Syntax: element.classList.remove(className)
Contoh:

```
document.getElementById('box').classList.remove('active');
```

Output:

```
<div id="box"></div>
```

classList.toggle - element.classList.toggle(className)

Penjelasan: Menambahkan class jika belum ada, dan menghapusnya jika sudah ada.
Syntax: element.classList.toggle(className)
Contoh:

```
document.getElementById('box').classList.toggle('hidden');
```

Output:

```
Menambah atau menghapus class "hidden"
```

Style & Class Methods

classList.contains - element.classList.contains(className)

Penjelasan: Memeriksa apakah class tertentu ada pada elemen.
Syntax: element.classList.contains(className)
Contoh:

```
document.getElementById('box').classList.contains('active');
```

Output:

```
true / false
```

classList.replace - element.classList.replace(oldClass, newClass)

Penjelasan: Mengganti class lama dengan class baru.
Syntax: element.classList.replace(oldClass, newClass)
Contoh:

```
document.getElementById('box').classList.replace('old', 'new');
```

Output:

```
<div id="box" class="new"></div>
```

Form & Input Methods

submit - form.submit()

Penjelasan: Mengirim form secara langsung tanpa trigger event submit.
Syntax: form.submit()
Contoh:

```
document.querySelector('form').submit();
```

Output:
Form dikirim ke server

reset - form.reset()

Penjelasan: Mengatur ulang form ke nilai defaultnya.
Syntax: form.reset()
Contoh:

```
document.querySelector('form').reset();
```

Output:
Semua input form direset

focus - element.focus()

Penjelasan: Memberikan fokus pada elemen input.
Syntax: element.focus()
Contoh:

```
document.getElementById('name').focus();
```

Output:
Kursor berpindah ke input dengan id 'name'

blur - element.blur()

Penjelasan: Menghapus fokus dari elemen.
Syntax: element.blur()
Contoh:

```
document.getElementById('name').blur();
```

Output:
Fokus hilang dari input

select - input.select()

Penjelasan: Menyeleksi semua teks dalam input.
Syntax: input.select()
Contoh:

```
document.getElementById('text').select();
```

Output:
Semua teks di dalam input disorot

Form & Input Methods

setCustomValidity - input.setCustomValidity(message)

Penjelasan: Mengatur pesan validasi kustom untuk input.
Syntax: input.setCustomValidity(message)
Contoh:

```
input.setCustomValidity('Kolom ini wajib diisi');
```

Output:
Validasi akan gagal dan pesan ditampilkan

checkValidity - form.checkValidity()

Penjelasan: Memeriksa apakah seluruh form valid.
Syntax: form.checkValidity()
Contoh:

```
document.querySelector('form').checkValidity();
```

Output:
true / false

reportValidity - form.reportValidity()

Penjelasan: Menampilkan pesan validasi jika form tidak valid.
Syntax: form.reportValidity()
Contoh:

```
document.querySelector('form').reportValidity();
```

Output:
true jika valid, jika tidak maka browser akan menampilkan pesan validasi

Selection & Range Methods

getSelection - window.getSelection()

Penjelasan: Mengambil objek selection saat ini (teks yang dipilih pengguna).
Syntax: window.getSelection()
Contoh:

```
let selection = window.getSelection();
```

Output:

```
Selection { anchorNode: ..., focusNode: ..., ... }
```

createRange - document.createRange()

Penjelasan: Membuat objek Range kosong untuk manipulasi DOM.
Syntax: document.createRange()
Contoh:

```
let range = document.createRange();
```

Output:

```
Range {}
```

range.setStart - range.setStart(node, offset)

Penjelasan: Mengatur titik awal range.
Syntax: range.setStart(node, offset)
Contoh:

```
range.setStart(paragraph, 0);
```

Output:

```
Range dimulai dari awal paragraf
```

range.setEnd - range.setEnd(node, offset)

Penjelasan: Mengatur titik akhir range.
Syntax: range.setEnd(node, offset)
Contoh:

```
range.setEnd(paragraph, 1);
```

Output:

```
Range berakhir setelah node pertama
```

range.selectNode - range.selectNode(node)

Penjelasan: Memilih seluruh node sebagai range.
Syntax: range.selectNode(node)
Contoh:

```
range.selectNode(document.querySelector('p'));
```

Output:

```
Range mencakup seluruh elemen <p>
```

Selection & Range Methods

range.deleteContents - range.deleteContents()

Penjelasan: Menghapus semua konten dalam range.
Syntax: range.deleteContents()
Contoh:

```
range.deleteContents();
```

Output:
Konten yang dipilih terhapus

range.insertNode - range.insertNode(node)

Penjelasan: Menyisipkan node ke posisi range.
Syntax: range.insertNode(node)
Contoh:

```
range.insertNode(document.createTextNode('Hello'));
```

Output:
Teks 'Hello' disisipkan pada range

Window & Screen Methods

alert - window.alert(message)

Penjelasan: Menampilkan kotak dialog peringatan dengan pesan.
Syntax: alert(message)
Contoh:

```
alert('Hello, world!');
```

Output:
Kotak dialog dengan pesan 'Hello, world!'

confirm - window.confirm(message)

Penjelasan: Menampilkan kotak dialog konfirmasi dengan tombol OK dan Cancel.
Syntax: confirm(message)
Contoh:

```
confirm('Apakah Anda yakin?');
```

Output:
true jika OK ditekan, false jika Cancel ditekan

prompt - window.prompt(message, default)

Penjelasan: Menampilkan kotak dialog input teks kepada pengguna.
Syntax: prompt(message, defaultText)
Contoh:

```
prompt('Masukkan nama:', 'Anonim');
```

Output:
Nilai yang diketik pengguna, atau null jika dibatalkan

setTimeout - window.setTimeout(function, delay)

Penjelasan: Menjalankan fungsi setelah jeda tertentu (ms).
Syntax: setTimeout(function, milliseconds)
Contoh:

```
setTimeout(() => alert('Delay 2 detik'), 2000);
```

Output:
Menampilkan alert setelah 2 detik

setInterval - window.setInterval(function, interval)

Penjelasan: Menjalankan fungsi berulang tiap interval waktu tertentu.
Syntax: setInterval(function, milliseconds)
Contoh:

```
setInterval(() => console.log('Ulang'), 1000);
```

Output:
Menulis 'Ulang' ke konsol setiap 1 detik

Window & Screen Methods

clearTimeout - window.clearTimeout(id)

Penjelasan: Membatalkan fungsi yang dijadwalkan dengan setTimeout.
Syntax: clearTimeout(id)
Contoh:

```
let id = setTimeout(fn, 3000);
clearTimeout(id);
```

Output:
Fungsi tidak akan dijalankan

clearInterval - window.clearInterval(id)

Penjelasan: Membatalkan eksekusi berulang yang dijadwalkan dengan setInterval.
Syntax: clearInterval(id)
Contoh:

```
let id = setInterval(fn, 1000);
clearInterval(id);
```

Output:
Interval dihentikan

scrollTo - window.scrollTo(x, y)

Penjelasan: Menggulir halaman ke koordinat tertentu.
Syntax: scrollTo(x, y)
Contoh:

```
window.scrollTo(0, 500);
```

Output:
Halaman menggulir ke bawah 500px

scrollBy - window.scrollBy(x, y)

Penjelasan: Menggulir halaman relatif terhadap posisi saat ini.
Syntax: scrollBy(x, y)
Contoh:

```
window.scrollBy(0, 100);
```

Output:
Halaman menggulir turun 100px dari posisi sekarang

print - window.print()

Penjelasan: Membuka dialog cetak untuk halaman saat ini.
Syntax: print()
Contoh:

```
window.print();
```

Output:
Menampilkan dialog cetak browser

Window & Screen Methods

open - window.open(url, name, specs)

Penjelasan: Membuka jendela atau tab browser baru dengan URL.
Syntax: open(url, name, specs)
Contoh:

```
window.open('https://example.com', '_blank');
```

Output:
Tab baru membuka https://example.com

close - window.close()

Penjelasan: Menutup jendela browser yang dibuka oleh script.
Syntax: close()
Contoh:

```
window.close();
```

Output:
Jendela script saat ini akan tertutup

Daftar Event pada addEventListener

Mouse Events

click

Penjelasan:

Terpicu saat pengguna mengklik elemen dengan tombol kiri mouse.

Sintaks:

```
element.addEventListener("click", function);
```

Contoh:

```
document.getElementById("btn").addEventListener("click", function() {
  alert("Tombol diklik!");
});
```

dblclick

Penjelasan:

Terpicu saat pengguna mengklik elemen dua kali secara cepat.

Sintaks:

```
element.addEventListener("dblclick", function);
```

Contoh:

```
document.getElementById("box").addEventListener("dblclick", function() {
  this.style.backgroundColor = "blue";
});
```

mousedown

Penjelasan:

Terpicu saat tombol mouse ditekan (belum dilepas) pada elemen.

Sintaks:

```
element.addEventListener("mousedown", function);
```

Contoh:

```
document.addEventListener("mousedown", function() {
  console.log("Mouse ditekan");
});
```

mouseup

Penjelasan:

Terpicu saat tombol mouse dilepas setelah ditekan.

Sintaks:

```
element.addEventListener("mouseup", function);
```

Contoh:

```
document.addEventListener("mouseup", function() {
  console.log("Mouse dilepas");
});
```

mousemove

Penjelasan:

Terpicu saat mouse digerakkan di atas elemen.

Sintaks:

```
element.addEventListener("mousemove", function);
```

Contoh:

```
document.addEventListener("mousemove", function(e) {
  console.log(`X: ${e.clientX}, Y: ${e.clientY}`);
});
```

mouseenter

Penjelasan:

Terpicu saat mouse masuk ke elemen (tidak bubbling).

Sintaks:

```
element.addEventListener("mouseenter", function);
```

Contoh:

```
document.getElementById("hoverBox").addEventListener("mouseenter", function() {
  this.style.border = "2px solid green";
});
```

mouseleave

Penjelasan:

Terpicu saat mouse keluar dari elemen (tidak bubbling).

Sintaks:

```
element.addEventListener("mouseleave", function);
```

Contoh:

```
document.getElementById("hoverBox").addEventListener("mouseleave", function() {
  this.style.border = "none";
});
```

mouseover

Penjelasan:

Terpicu saat mouse masuk ke elemen atau elemen anaknya (bubbling).

Sintaks:

```
element.addEventListener("mouseover", function);
```

Contoh:

```
document.getElementById("menu").addEventListener("mouseover", function() {
```

```
  console.log("Mouse masuk menu");
});
```

mouseout

Penjelasan:

Terpicu saat mouse keluar dari elemen atau elemen anaknya (bubbling).

Sintaks:

```
element.addEventListener("mouseout", function);
```

Contoh:

```
document.getElementById("menu").addEventListener("mouseout", function() {
  console.log("Mouse keluar dari menu");
});
```

contextmenu

Penjelasan:

Terpicu saat klik kanan dilakukan pada elemen.

Sintaks:

```
element.addEventListener("contextmenu", function);
```

Contoh:

```
document.addEventListener("contextmenu", function(e) {
  e.preventDefault();
  alert("Klik kanan dinonaktifkan");
});
```

Keyboard Events

keydown

Penjelasan:

Terpicu saat tombol keyboard ditekan (saat tombol mulai ditekan).

Sintaks:

```
element.addEventListener("keydown", function);
```

Contoh:

```
document.addEventListener("keydown", function(e) {
  console.log("Tombol ditekan:", e.key);
});
```

keyup

Penjelasan:

Terpicu saat tombol keyboard dilepas setelah ditekan.

Sintaks:

```
element.addEventListener("keyup", function);
```

Contoh:

```
document.addEventListener("keyup", function(e) {
  console.log("Tombol dilepas:", e.key);
});
```

keypress

Penjelasan:

Terpicu saat tombol ditekan dan menghasilkan karakter. Sudah deprecated.

Sintaks:

```
element.addEventListener("keypress", function);
```

Contoh:

```
document.addEventListener("keypress", function(e) {
  console.log("Karakter ditekan:", e.key);
});
```

Form & Input Events

input

Penjelasan:

Terpicu setiap kali nilai input berubah (real-time saat mengetik).

Sintaks:

```
element.addEventListener("input", function);
```

Contoh:

```
document.getElementById("textInput").addEventListener("input", function(e) {
  console.log("Nilai sekarang:", e.target.value);
});
```

change

Penjelasan:

Terpicu saat nilai input selesai diubah (biasanya saat kehilangan fokus atau setelah dipilih).

Sintaks:

```
element.addEventListener("change", function);
```

Contoh:

```
document.getElementById("selectOption").addEventListener("change", function(e) {
  console.log("Pilihan berubah:", e.target.value);
});
```

submit

Penjelasan:

Terpicu saat form dikirim (submit).

Sintaks:

```
formElement.addEventListener("submit", function);
```

Contoh:

```
document.getElementById("myForm").addEventListener("submit", function(e) {
  e.preventDefault();
  alert("Form dikirim!");
});
```

reset

Penjelasan:

Terpicu saat form di-reset ke nilai awal.

Sintaks:

```
formElement.addEventListener("reset", function);
```

Contoh:

```
document.getElementById("myForm").addEventListener("reset", function() {
  alert("Form direset!");
});
```

focus

Penjelasan:

Terpicu saat elemen mendapatkan fokus. Tidak bubble.

Sintaks:

```
element.addEventListener("focus", function);
```

Contoh:

```
document.getElementById("textInput").addEventListener("focus", function() {
  this.style.backgroundColor = "#e0f7fa";
});
```

blur

Penjelasan:

Terpicu saat elemen kehilangan fokus. Tidak bubble.

Sintaks:

```
element.addEventListener("blur", function);
```

Contoh:

```
document.getElementById("textInput").addEventListener("blur", function() {
  this.style.backgroundColor = "";
});
```

focusin

Penjelasan:

Terpicu saat fokus masuk ke elemen. Bisa bubble.

Sintaks:

```
element.addEventListener("focusin", function);
```

Contoh:

```
document.getElementById("formWrapper").addEventListener("focusin", function() {
  console.log("Fokus masuk ke dalam form");
});
```

focusout

Penjelasan:

Terpicu saat fokus keluar dari elemen. Bisa bubble.

Sintaks:

```
element.addEventListener("focusout", function);
```

Contoh:

```
document.getElementById("formWrapper").addEventListener("focusout", function() {
  console.log("Fokus keluar dari form");
});
```

Drag & Drop Events

drag

Penjelasan:

Terpicu saat elemen sedang di-drag.

Sintaks:

```
element.addEventListener("drag", function);
```

Contoh:

```
document.getElementById("draggable").addEventListener("drag", function() {
  console.log("Sedang di-drag...");
});
```

dragstart

Penjelasan:

Terpicu saat drag dimulai.

Sintaks:

```
element.addEventListener("dragstart", function);
```

Contoh:

```
document.getElementById("draggable").addEventListener("dragstart", function(e) {
  e.dataTransfer.setData("text/plain", e.target.id);
  console.log("Drag dimulai");
});
```

dragend

Penjelasan:

Terpicu saat drag selesai, baik berhasil atau dibatalkan.

Sintaks:

```
element.addEventListener("dragend", function);
```

Contoh:

```
document.getElementById("draggable").addEventListener("dragend", function() {
  console.log("Drag selesai");
});
```

dragenter

Penjelasan:

Terpicu saat elemen draggable memasuki area target drop.

Sintaks:

```
element.addEventListener("dragenter", function);
```

Contoh:

```
document.getElementById("dropzone").addEventListener("dragenter", function() {
  this.style.backgroundColor = "#e0f7fa";
});
```

dragleave

Penjelasan:

Terpicu saat elemen draggable keluar dari area drop.

Sintaks:

```
element.addEventListener("dragleave", function);
```

Contoh:

```
document.getElementById("dropzone").addEventListener("dragleave", function() {
  this.style.backgroundColor = "";
});
```

dragover

Penjelasan:

Terpicu saat elemen draggable berada di atas area drop.

Sintaks:

```
element.addEventListener("dragover", function);
```

Contoh:

```
document.getElementById("dropzone").addEventListener("dragover", function(e) {
  e.preventDefault();
});
```

drop

Penjelasan:

Terpicu saat elemen draggable dilepaskan di dalam area drop.

Sintaks:

```
element.addEventListener("drop", function);
```

Contoh:

```
document.getElementById("dropzone").addEventListener("drop", function(e) {
  e.preventDefault();
  const data = e.dataTransfer.getData("text/plain");
  const dragged = document.getElementById(data);
  this.appendChild(dragged);
});
```

Animation & Transition Events

animationstart

Penjelasan:

Terpicu saat animasi CSS dimulai.

Sintaks:

```
element.addEventListener("animationstart", function);
```

Contoh:

```
document.getElementById("box").addEventListener("animationstart", function() {
  console.log("Animasi dimulai");
});
```

animationend

Penjelasan:

Terpicu saat animasi CSS selesai.

Sintaks:

```
element.addEventListener("animationend", function);
```

Contoh:

```
document.getElementById("box").addEventListener("animationend", function() {
  console.log("Animasi selesai");
});
```

animationiteration

Penjelasan:

Terpicu setiap kali animasi CSS mengulang.

Sintaks:

```
element.addEventListener("animationiteration", function);
```

Contoh:

```
document.getElementById("box").addEventListener("animationiteration", function()
{
  console.log("Animasi diulang");
});
```

transitionstart

Penjelasan:

Terpicu saat transisi CSS dimulai.

Sintaks:

```
element.addEventListener("transitionstart", function);
```

Contoh:

```
document.getElementById("box").addEventListener("transitionstart", function() {
  console.log("Transisi dimulai");
});
```

transitionend

Penjelasan:

Terpicu saat transisi CSS selesai.

Sintaks:

```
element.addEventListener("transitionend", function);
```

Contoh:

```
document.getElementById("box").addEventListener("transitionend", function() {
  console.log("Transisi selesai");
});
```

transitionrun

Penjelasan:

Terpicu saat transisi mulai berjalan.

Sintaks:

```
element.addEventListener("transitionrun", function);
```

Contoh:

```
document.getElementById("box").addEventListener("transitionrun", function() {
  console.log("Transisi mulai berjalan");
});
```

transitioncancel

Penjelasan:

Terpicu saat transisi dibatalkan sebelum selesai.

Sintaks:

```
element.addEventListener("transitioncancel", function);
```

Contoh:

```
document.getElementById("box").addEventListener("transitioncancel", function() {
  console.log("Transisi dibatalkan");
});
```

Touch Events (Mobile)

touchstart

Penjelasan:

Terpicu saat satu atau lebih jari mulai menyentuh layar.

Sintaks:

```
element.addEventListener("touchstart", function);
```

Contoh:

```
document.getElementById("touchArea").addEventListener("touchstart",  function(e)
{
  console.log("Sentuhan dimulai:", e.touches.length, "jari");
});
```

touchmove

Penjelasan:

Terpicu saat jari digerakkan di atas layar setelah touchstart.

Sintaks:

```
element.addEventListener("touchmove", function);
```

Contoh:

```
document.getElementById("touchArea").addEventListener("touchmove", function(e) {
          console.log("Sentuhan     bergerak     ke:",     e.touches[0].clientX,
e.touches[0].clientY);
});
```

touchend

Penjelasan:

Terpicu saat jari diangkat dari layar (mengakhiri sentuhan).

Sintaks:

```
element.addEventListener("touchend", function);
```

Contoh:

```
document.getElementById("touchArea").addEventListener("touchend", function(e) {
  console.log("Sentuhan selesai");
});
```

touchcancel

Penjelasan:

Terpicu saat sistem menghentikan sentuhan (misalnya karena interupsi).

Sintaks:

```
element.addEventListener("touchcancel", function);
```

Contoh:

```
document.getElementById("touchArea").addEventListener("touchcancel",  function()
{
  console.log("Sentuhan dibatalkan oleh sistem");
```

```
});
```

Media Events

play

Penjelasan:

Terpicu saat media mulai diputar.

Sintaks:

```
mediaElement.addEventListener("play", function);
```

Contoh:

```
document.getElementById("video").addEventListener("play", function() {
  console.log("Video mulai diputar");
});
```

pause

Penjelasan:

Terpicu saat media dijeda.

Sintaks:

```
mediaElement.addEventListener("pause", function);
```

Contoh:

```
document.getElementById("video").addEventListener("pause", function() {
  console.log("Video dijeda");
});
```

ended

Penjelasan:

Terpicu saat media selesai diputar.

Sintaks:

```
mediaElement.addEventListener("ended", function);
```

Contoh:

```
document.getElementById("video").addEventListener("ended", function() {
  alert("Video selesai diputar");
});
```

volumechange

Penjelasan:

Terpicu saat volume media diubah.

Sintaks:

```
mediaElement.addEventListener("volumechange", function);
```

Contoh:

```
document.getElementById("audio").addEventListener("volumechange", function() {
```

```
  console.log("Volume berubah:", this.volume);
});
```

timeupdate

Penjelasan:

Terpicu secara berkala saat waktu pemutaran media berubah.

Sintaks:

```
mediaElement.addEventListener("timeupdate", function);
```

Contoh:

```
document.getElementById("video").addEventListener("timeupdate", function() {
  console.log("Waktu saat ini:", this.currentTime);
});
```

seeking

Penjelasan:

Terpicu saat pengguna mulai mencari posisi (seek) pada media.

Sintaks:

```
mediaElement.addEventListener("seeking", function);
```

Contoh:

```
document.getElementById("video").addEventListener("seeking", function() {
  console.log("Pengguna sedang mencari posisi...");
});
```

seeked

Penjelasan:

Terpicu saat proses pencarian posisi (seek) selesai.

Sintaks:

```
mediaElement.addEventListener("seeked", function);
```

Contoh:

```
document.getElementById("video").addEventListener("seeked", function() {
  console.log("Posisi baru ditemukan:", this.currentTime);
});
```

loadeddata

Penjelasan:

Terpicu saat data pertama media dimuat.

Sintaks:

```
mediaElement.addEventListener("loadeddata", function);
```

Contoh:

```
document.getElementById("video").addEventListener("loadeddata", function() {
  console.log("Data media siap diputar");
```

```
});
```

loadedmetadata

Penjelasan:

Terpicu saat metadata media tersedia.

Sintaks:

```
mediaElement.addEventListener("loadedmetadata", function);
```

Contoh:

```
document.getElementById("video").addEventListener("loadedmetadata", function() {
  console.log("Durasi video:", this.duration);
});
```

error

Penjelasan:

Terpicu saat terjadi kesalahan saat memuat media.

Sintaks:

```
mediaElement.addEventListener("error", function);
```

Contoh:

```
document.getElementById("video").addEventListener("error", function() {
  console.error("Gagal memuat video");
});
```

Window & Document Events

load

Penjelasan:

Terpicu saat seluruh halaman (termasuk gambar, skrip, dan style) selesai dimuat.

Sintaks:

```
window.addEventListener("load", function);
```

Contoh:

```
window.addEventListener("load", function() {
  console.log("Halaman selesai dimuat");
});
```

resize

Penjelasan:

Terpicu saat ukuran jendela browser diubah.

Sintaks:

```
window.addEventListener("resize", function);
```

Contoh:

```
window.addEventListener("resize", function() {
```

```
  console.log("Ukuran jendela:", window.innerWidth, "x", window.innerHeight);
});
```

scroll

Penjelasan:

Terpicu saat pengguna menggulir (scroll) halaman atau elemen yang bisa di-scroll.

Sintaks:

```
window.addEventListener("scroll", function);
```

Contoh:

```
window.addEventListener("scroll", function() {
  console.log("Scroll posisi Y:", window.scrollY);
});
```

beforeunload

Penjelasan:

Terpicu sebelum halaman ditutup atau dimuat ulang.

Sintaks:

```
window.addEventListener("beforeunload", function);
```

Contoh:

```
window.addEventListener("beforeunload", function(e) {
  e.preventDefault();
  e.returnValue = "";
});
```

unload

Penjelasan:

Terpicu saat halaman ditutup.

Sintaks:

```
window.addEventListener("unload", function);
```

Contoh:

```
window.addEventListener("unload", function() {
  console.log("Halaman ditutup");
});
```

error

Penjelasan:

Terpicu saat terjadi kesalahan runtime (bisa pada skrip, gambar, dsb.).

Sintaks:

```
window.addEventListener("error", function);
```

Contoh:

```
window.addEventListener("error", function(e) {
```

```
  console.error("Kesalahan:", e.message);
});
```

DOMContentLoaded

Penjelasan:

Terpicu saat struktur DOM selesai dimuat.

Sintaks:

```
document.addEventListener("DOMContentLoaded", function);
```

Contoh:

```
document.addEventListener("DOMContentLoaded", function() {
  console.log("DOM siap digunakan");
});
```

Clipboard Events

copy

Penjelasan:

Terpicu saat pengguna menyalin (copy) konten dari halaman.

Sintaks:

```
element.addEventListener("copy", function);
```

Contoh:

```
document.addEventListener("copy", function(e) {
  console.log("Konten disalin");
});
```

cut

Penjelasan:

Terpicu saat pengguna memotong (cut) konten dari halaman.

Sintaks:

```
element.addEventListener("cut", function);
```

Contoh:

```
document.addEventListener("cut", function(e) {
  console.log("Konten dipotong");
});
```

paste

Penjelasan:

Terpicu saat pengguna menempelkan (paste) konten ke dalam elemen.

Sintaks:

```
element.addEventListener("paste", function);
```

Contoh:

```
document.addEventListener("paste", function(e) {
  const pasted = (e.clipboardData || window.clipboardData).getData("text");
  console.log("Konten ditempelkan:", pasted);
});
```

Security & Network Events

online

Penjelasan:

Terpicu saat browser kembali terhubung ke internet.

Sintaks:

```
window.addEventListener("online", function);
```

Contoh:

```
window.addEventListener("online", function() {
  console.log("Kembali online");
});
```

offline

Penjelasan:

Terpicu saat koneksi internet terputus.

Sintaks:

```
window.addEventListener("offline", function);
```

Contoh:

```
window.addEventListener("offline", function() {
  console.log("Koneksi terputus");
});
```

message

Penjelasan:

Terpicu saat menerima pesan dari iframe, worker, atau koneksi WebSocket.

Sintaks:

```
window.addEventListener("message", function);
```

Contoh:

```
window.addEventListener("message", function(e) {
  console.log("Pesan diterima:", e.data);
});
```

popstate

Penjelasan:

Terpicu saat pengguna melakukan navigasi kembali/maju.

Sintaks:

```
window.addEventListener("popstate", function);
```

Contoh:

```
window.addEventListener("popstate", function(e) {
  console.log("Navigasi kembali/maju:", e.state);
});
```

Custom Events

CustomEvent

Penjelasan:

Event yang dibuat secara manual oleh developer menggunakan new CustomEvent().

Sintaks:

```
const event = new CustomEvent("namaEvent", {
  detail: { ...dataTambahan }
});
element.dispatchEvent(event);
element.addEventListener("namaEvent", function);
```

Contoh:

```
// Buat event kustom
const myEvent = new CustomEvent("produkTerpilih", {
  detail: {
    id: 123,
    nama: "Laptop"
  }
});

// Dengarkan event
document.addEventListener("produkTerpilih", function(e) {
  console.log("Produk dipilih:", e.detail);
});

// Dispatch event
document.dispatchEvent(myEvent);
```