

Arrayer

En ordnad lista med värden



Detta verk är licensierad under en
[Creative Commons Erkännande 4.0](https://creativecommons.org/licenses/by/4.0/)
[Internationell licens](https://creativecommons.org/licenses/by/4.0/).

Hur hantera flera värden genom en variabel eller ett returvärde?

let myValue = 42

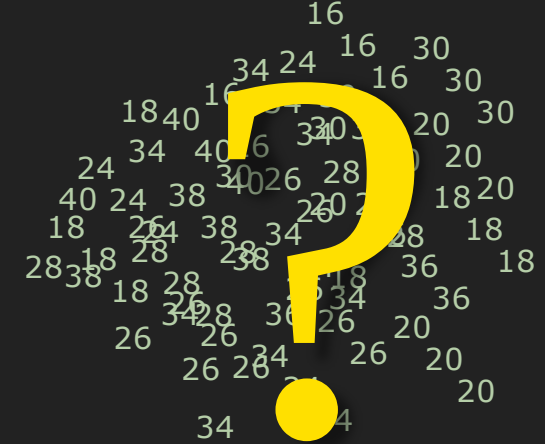


✓ Problem

- En variabel kan bara tilldelas ett värde.
 - Vilka alternativ finns det om "flera" värden ska kunna tilldelas en variabel?
- En funktion kan bara returnera ett värde.
 - Vilka alternativ finns det om "flera" värden ska returneras?

✓ Lösning

- Oavsett om det är "flera" värden som ska tilldelas en variabel eller om en funktion ska returnera "flera" värden är lösningen EN referens till en **array** eller ett **objekt**, där arrayen eller objektet kan innehålla flera värden.



En ordnad lista med värden

- ✓ Genom att skapa en enkel handlingslista har du ett exempel på en ordnad lista med värden – en array!
 - En lista som är ordnad innebär att den behåller en viss ordning mellan raderna i listan (inte att dess värden är sorterade).
 - En array benämns ibland även som fält, vektor.

handlingslista

	0. mjölk
	1. ägg
	2. pasta
	3. ketchup
index	4. nudlar
värde	5. havregryn

- Pappersslappen kallas "handlingslista".
- Listan innehåller rader med enskilda värden.
- De enskilda värdena numreras från och med 0 och uppåt.

Vad står det i
handlingslistan på
raden med index 4?

nudlar

En array som handlingslista

- ✓ Genom att använda hakparenteser skapas en array som representerar handlingslistan.
 - Konstanten `groceries` refererar till arrayen.

```
const groceries = [  
  'mjölk',  
  'ägg',  
  'pasta',  
  'ketchup',  
  'nudlar',  
  'havregryn'  
]
```

- Arrayen refereras till med "groceries".
- Arrayen innehåller element med värden.
- Elementen numreras från och med 0 och uppåt.

- ✓ För att komma åt ett värde i arrayen måste dess index anges.

```
console.log(groceries[4])
```

```
// OUTPUT: 'nudlar'
```

Vad har elementet för värde på index 4?

nudlar

Vad är en array?

- ✓ Du kan samla flera värden i en array.
 - En array innehåller element där varje element innehåller ett värde.
 - Med hjälp av index kan du ange vilket element i arrayen du vill hämta ett värde från eller tilldela ett värde.
 - En array har ett 0-baserat index, d.v.s. första elementet i en array har alltid index 0.

värde:	28	18	26	28	38
index:	0	1	2	3	4

- ✓ Värdena i en array kan vara av olika typer. (Bör undvikas! Till exempel Object lämpligare att använda i så fall.)

värde:	42	'är meningen med livet'	true
index:	0	1	2

Detta är ingen bra idé!

Så skapar du en array

- ✓ Arrayliteral - det rekommenderade sättet.

```
// En tom array
```

```
const values = []
```

```
// En array med fem element
```

```
const someFemalePioneersInComputerScience = ['Lovelace', 'Hopper', 'Johnson', 'Hamilton', 'Goldberg']
```

- ✓ Med nyckelordet `new` - *inte rekommenderat!*

```
const numbers = new Array(28, 18, 26, 38, 24) // [28, 18, 26, 38, 24]
```

```
console.log(numbers[0]) // OUTPUT: 28
```

```
console.log(numbers.length) // OUTPUT: 5
```

```
const anotherArray = new Array(10) // [undefined, undefined, undefined, ..., undefined]
```

```
console.log(anotherArray[0]) // OUTPUT: undefined
```

```
console.log(anotherArray.length) // OUTPUT: 10
```

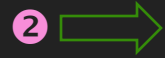
Så arbetar du med en array

```
// Array med fem element.  
const temperatures = [12.6, 18.7, 9.8, 13.2, 14.4]  
  
// Hämta värdet för det fjärde elementet.  
console.log(temperatures[3]) // OUTPUT: 13.2  
  
// Tilldela det andra elementet ett nytt värde.  
temperatures[1] = 4.2  
  
// Antalet element i en array.  
console.log(temperatures.length) // OUTPUT: 5  
  
// Iterera igenom en array från första till sista elementet.  
let output = ''  
for (let i = 0; i < temperatures.length; i++) {  
  output += temperatures[i] + ' '  
}  
  
console.log(output) // OUTPUT: 12.6 4.2 9.8 13.2 14.4
```

Så lägger du till och tar bort

- ✓ Du kan lägga till nya saker att handla i slutet eller i början av listan.

0. mjölk
1. ägg
2. pasta
3. ketchup
4. nudlar
5. havregryn
6. bönor



0. bönor
1. mjölk
2. ägg
3. pasta
4. ketchup
5. nudlar
6. havregryn

② unshift

① push

④ shift

③ pop

splice

Tar bort, ersätter eller lägger till element i befintlig array.

- ✓ Du kan också ta bort saker från slutet eller i början av listan.

0. mjölk
1. ägg
2. pasta
3. ketchup
4. nudlar
5. havregryn
6. bönor

④ 0. bönor
1. mjölk
2. ägg
3. pasta
4. ketchup
5. nudlar
6. havregryn

```
const groceries = [  
  'mjölk',  
  'ägg',  
  'pasta',  
  'ketchup',  
  'nudlar',  
  'havregryn'  
]
```

- ① `groceries.push('bönor')` // lägger till i slutet
- ② `groceries.unshift('bönor')` // lägger till i början
- ③ `let lastItem = groceries.pop()` // tar bort sista
- ④ `let firstItem = groceries.shift()` // tar bort första

Värdetyper och referenstyper

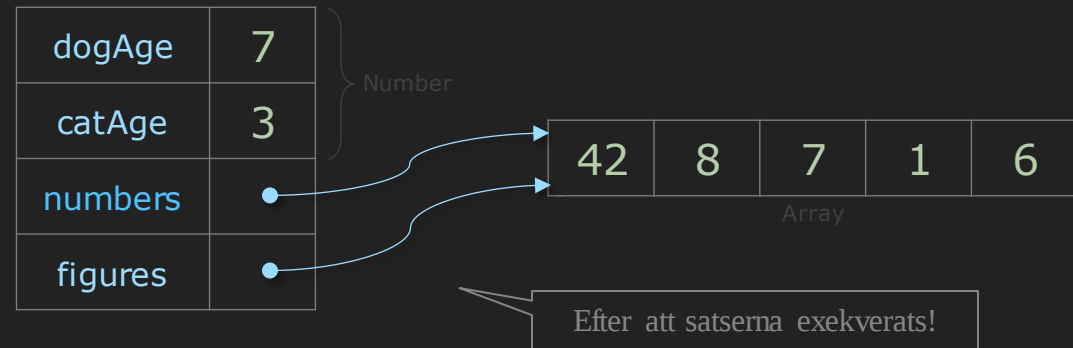
✓ Värdetyp (t.ex. Number)

```
let dogAge = 7  
let catAge
```

```
console.log(dogAge) // OUTPUT: 7
```

```
catAge = dogAge  
catAge = 3
```

```
console.log(catAge) // OUTPUT: 3  
console.log(dogAge) // OUTPUT: 7
```



✓ Referenstyp (t.ex. Array som är av typen Object)

```
const numbers = [5, 8, 7, 1, 6] // tilldelar en konstanten en referens till en array  
let figures
```

```
console.log(numbers) // OUTPUT: [ 5, 8, 7, 1, 6 ]
```

```
figures = numbers // kopierar referensen till arrayen till en variabel  
figures[0] = 42 // ändrar det första elementets värde genom att använda kopian av referensen
```

```
console.log(figures) // OUTPUT: [ 42, 8, 7, 1, 6 ]  
console.log(numbers) // OUTPUT: [ 42, 8, 7, 1, 6 ], (numbers och figures refererar till samma array!)
```

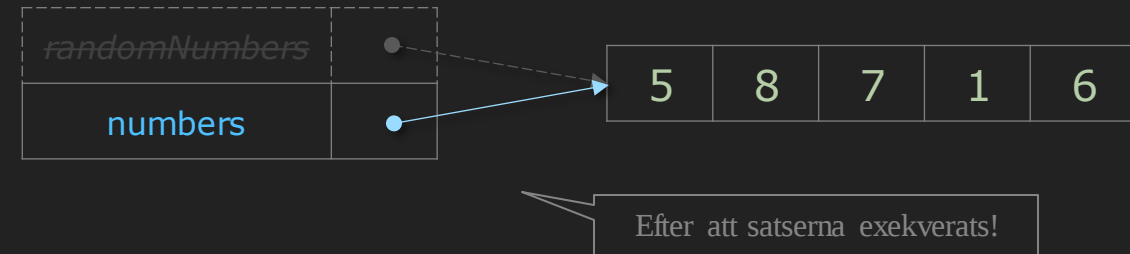
En referens till en array som returvärde

- ✓ Det är referensen som returneras. Inte själva arrayen i sig.

```
/**  
 * Returns an array of numbers in the closed interval between 0 and 9.  
 *  
 * @param {number} count - Number of numbers to randomize.  
 * @returns {number[]} An array with random numbers between 0 and 9.  
 */
```

```
const getRandomNumbers = function (count) {  
  const randomNumbers = []  
  
  for (let i = 0; i < count; i++) {  
    randomNumbers.push(Math.floor(Math.random() * 10))  
  }  
  
  // Return reference to the array with count elements.  
  return randomNumbers  
}
```

```
// Set a constant to refer to the returned array.  
const numbers = getRandomNumbers(5)  
console.log(numbers) // OUTPUT: [ 5, 8, 7, 1, 6 ]
```



Referens till array som parameter

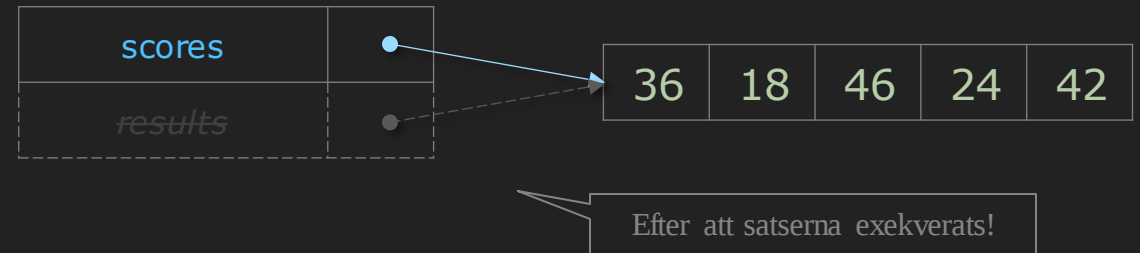
- ✓ Använder du en array som argument kan du få oönskade "sidoeffekter".

```
const logDoubledUp = function (results) {  
  for (let i = 0; i < results.length; i++) {  
    results[i] *= 2  
  }  
  console.log(results) // OUTPUT: [ 36, 18, 46, 24, 42 ]  
}
```

```
const scores = [18, 9, 23, 12, 21]  
console.log(scores) // OUTPUT: [ 18, 9, 23, 12, 21 ]
```

```
logDoubledUp(scores)
```

```
console.log(scores) // OUTPUT: [ 36, 18, 46, 24, 42 ]
```



- ✓ Konstanten `scores` refererar till exakt samma array som parametern `results` gör, varför samma array påverkas oavsett vilken av referenserna som används.

Så skapar du en kopia av en array

- ✓ Det finns flera sätt att skapa en kopia av en array.

```
const animals = ['🐶', '🐱', '🐭', '🐹', '🐮']
```

```
// Old way
```

```
const animalsCopy = animals.slice()
```

```
// ES6 way
```

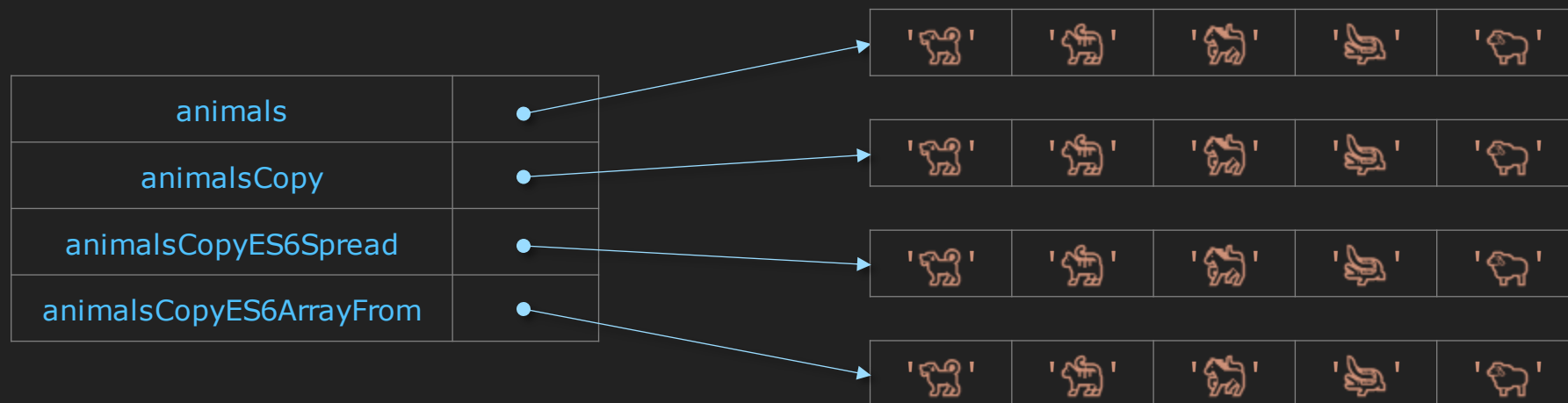
```
const animalsCopyES6Spread = [...animals]
```

Läs om "[spread syntax \(...\)](#)" på MDN.

```
// Yet another ES6 way
```

```
const animalsCopyES6ArrayFrom = Array.from(animals)
```

[Array.from\(\)](#) kan mycket mer än att "bara" skapa en kopia.



Arbeta med en modifierad kopia av en array

- ✓ För att undvika oönskade sidoeffekter skapas en kopia av arrayen parametern refererar till.

```
const logDoubledUp = function (results) {  
  const copy = Array.from(results) // skapar en kopia av arrayen results refererar till  
  
  for (let i = 0; i < copy.length; i++) {  
    copy[i] *= 2  
  }  
  
  console.log(copy) // OUTPUT: [ 36, 18, 46, 24, 42 ]  
}  
  
const scores = [18, 9, 23, 12, 21]  
console.log(scores) // OUTPUT: [ 18, 9, 23, 12, 21 ]  
  
logDoubledUp(scores)  
  
console.log(scores) // OUTPUT: [ 18, 9, 23, 12, 21 ]
```

Sortera en array

- ✓ Metoden `sort` sorterar, om ingen sorteringsfunktion anges, genom att först konvertera elementens värden till strängar för att sedan jämföra strängarna (lexikografisk sortering).

```
const values = [2, 42, 10, 1, 8, 24, 4]

values.sort()
console.log(values) // OUTPUT: [ 1, 10, 2, 24, 4, 42, 8 ], 10 kommer före 2!
```

- ✓ Genom att ange en sorteringsfunktion i form av ett funktionsuttryck kan en egen sorteringsordning bestämmas.

```
const values = [2, 42, 10, 1, 8, 24, 4]

values.sort(function (a, b) {
  return a - b
})

console.log(values) // OUTPUT: [ 1, 2, 4, 8, 10, 24, 42 ]
```

Några fler metoder Array har

- ✓ Iterera igenom värden i en array ([forEach](#)).

```
const values = [2, 1, 8]

values.forEach(function (value) {
  console.log(value)
})
```

```
// OUTPUT: 2
// OUTPUT: 1
// OUTPUT: 8
```

- ✓ Reducera värdena i en array genom att t.ex. summera dem ([reduce](#)).

```
const values = [2, 1, 8]

const sum = values.reduce(function (a, b) {
  return a + b
}, 0)

console.log(sum) // OUTPUT: 11
```

Metoden `Array#forEach` är exempel på det som kallas "higher-order functions".

En "higher-order function" är en funktion som tar emot en funktion som argument eller returnerar en funktion.

”Arrow functions” - ännu ett sätt att skriva funktioner

- ✓ Transformera en array till en annan med [map](#), som skapar en helt ny array.

```
const values = [2, 1, 8]

const doubledValues = values.map(function (value) {
  return value * 2
})

console.log(doubledValues) // OUTPUT: [ 4, 2, 16 ]
```

- ✓ Istället för ett funktionsuttryck kan en ”arrow function” användas.

```
const values = [2, 1, 8]

const doubledValues = values.map(value => {
  return value * 2
})

console.log(doubledValues) // OUTPUT: [ 4, 2, 16 ]
```

En ”arrow function” kan ha en eller flera satser. Satserna placeras i ett block precis som en ”vanlig” funktions kropp, och en return-sats används för att returnera ett värde.

- Alternativt uttryck (kort och läsbart!)

```
const values = [2, 1, 8]
const doubledValues = values.map(value => value * 2)

console.log(doubledValues) // OUTPUT: [ 4, 2, 16 ]
```

Då en ”arrow function” endast innehåller en sats kan block och return uteslutas.

