

Lite repetition

Loopar

Arrayer

Objekt

Funktioner

for-loop

Startvärde räknare

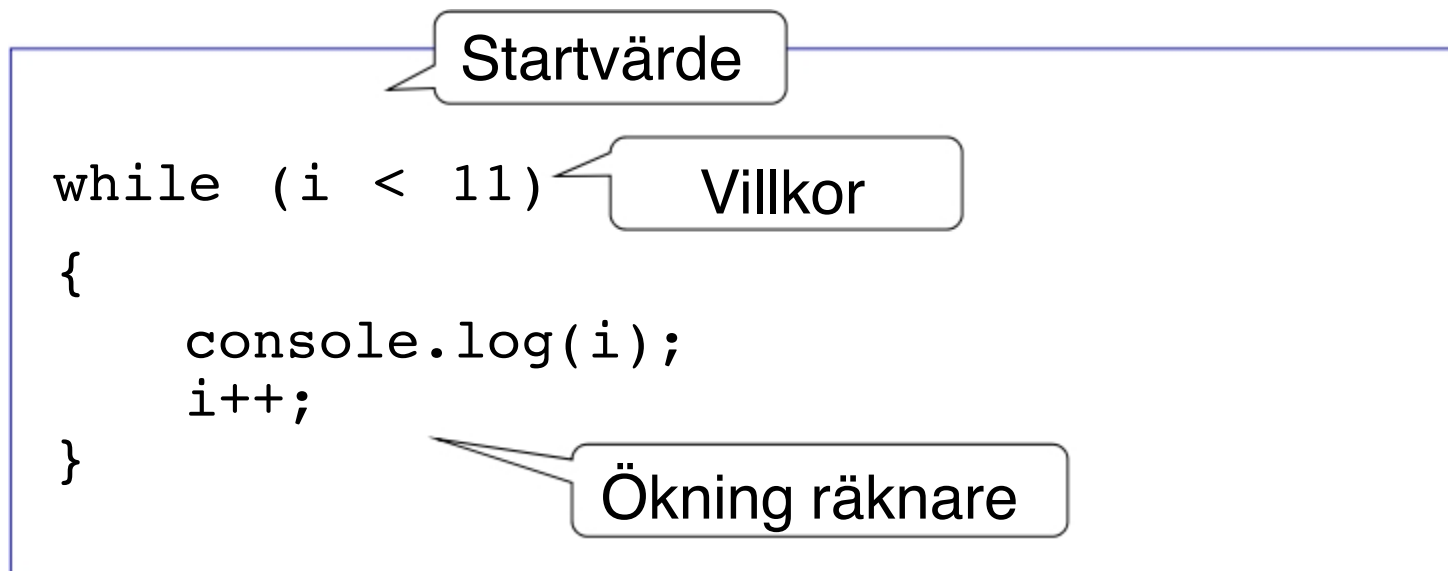
Villkor

Ökning räknare

```
for (let i = 1; i < 11; i++)  
{  
  console.log(i);  
}
```

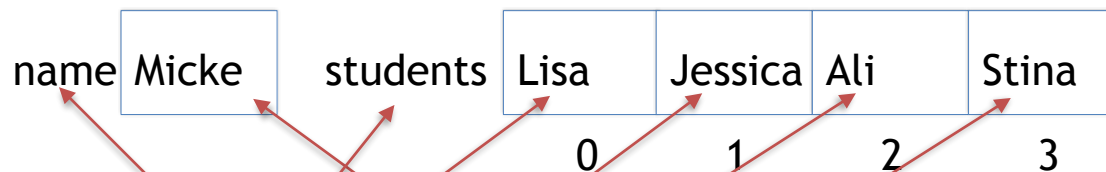
while

- Ett annat sätt att skriva en loop är att använda `while` . Detta exempel gör samma sak som `for` loopen vi skrev tidigare.



Array

- En array är en speciell datatyp som kan innehålla flera värden.
- Varje "låda" i en array har ett index, som alltid börjar på 0.



```
let name = "Micke";  
let students = [  
  "Lisa",  
  "Jessica",  
  "Ali",  
  "Stina"  
];
```

```
console.log(name);  
console.log(studenter[0]);
```

Objekt

- *Objekt* är (i sin enklaste form) variabler med "undervariabler".
- Dessa "undervariabler" kallas *egenskaper*.

```
▶ {name: "Micke", age: 42, shoe_size: 43}
```

```
>
```

```
let person = {};  
  
person.name = "Micke";  
person.age = 42;  
person.shoe_size = 43;  
  
console.log(person);
```

```
let person = {  
  name: "Micke",  
  age: 42,  
  shoe_size: 43  
};  
  
console.log(person);
```

Objekt som egenskap

- Egenskaper kan innehålla olika datatyper, även objekt.

```
let student = {  
  name: {  
    first: "Mikael",  
    last: "Olsson"  
  },  
  class: "JavaScript"  
}  
  
console.log(student);
```

```
[Running] node "/Users/micke/www/projects/ec education/javascript/e  
{ name: { first: 'Mikael', last: 'Olsson' }, class: 'JavaScript' }
```

```
[Done] exited with code=0 in 0.058 seconds
```

Skriva ut en egenskap

```
let student = {  
  name: {  
    first: "Mikael",  
    last: "Olsson"  
  },  
  class: "JavaScript"  
}  
  
console.log(student.class);  
console.log(student.name.first);
```

```
[Running] no  
JavaScript  
Mikael  
  
[Done] exited
```

Funktioner

- Funktionens beståndsdelar

The diagram shows a code snippet on a black background with red boxes highlighting different parts and red arrows pointing to a legend on the right.

```
function addNumbers(a, b) {  
  console.log(a + b);  
}  
  
addNumbers(2, 2);
```

- Namn (Name): Points to `addNumbers` in the function definition.
- Parametrar (Parameters): Points to `a, b` in the function definition.
- Innehåll (Body): Points to `console.log(a + b);` inside the function definition.
- Anrop (Call): Points to `addNumbers(2, 2);` in the code snippet.

Below the legend, a simple calculator interface is shown with a display area containing the number 4 and a blue greater-than sign (>) button.

Funktioner

- En funktion kan även returnera ett värde.

```
function addNumbers(a, b) {  
  return(a + b);  
}  
  
let result = addNumbers(2, 2);  
  
console.log(result);  
console.log(addNumbers(3, 3));
```



```
4  
6  
>
```

Spara en anonym funktion i en variabel

- En anonym funktion är en funktion utan namn.
- Vi kan spara en anonym funktion i en variabel.

```
let addNumbers = function (a, b) {  
  return a + b;  
}  
  
console.log( addNumbers( 3, 4 ) );
```

```
[Running] node  
7  
  
[Done] exited
```

Funktion som egenskap

- En egenskap i ett objekt kan innehålla en funktion. Den kallas då *metod*.

```
let student = {  
  name: "Mikael Olsson",  
  print: function () {  
    console.log("hello");  
  }  
}  
  
student.print();
```

```
[Running] r  
hello  
  
[Done] exit
```

This

- Om vi vill använda en egenskap inuti objektet kan vi använda nyckelordet `this`.

```
let student = {  
  name: "Mikael Olsson",  
  print: function () {  
    console.log(this.name);  
  }  
}  
  
student.print();
```

```
[Running] node "/U.  
Mikael Olsson  
  
[Done] exited with
```

Nu fortsätter vi...

for... of

- En *for... of*-loop loopar igenom alla värden i en array.

```
let students = [  
  "Lisa",  
  "Jessica",  
  "Ali",  
  "Stina"  
];  
  
for(student of students) {  
  console.log(student);  
};
```

Lisa
Jessica
Ali
Stina



Uppgift

- Lägg till metoden *list_students* i ert objekt som skriver ut en lista med alla studenter.

for... in

- *For... in* fungerar som *for... of*, men variabeln kommer då att innehålla det nuvarande indexet istället för det nuvarande värdet.

```
let students = [  
  "Lisa",  
  "Jessica",  
  "Ali",  
  "Stina"  
];  
  
for(student in students) {  
  console.log(student);  
};
```

0

1

2

3

forEach

- *forEach* fungerar som for... of men tar en funktion som parameter.
- Vad kallas denna typ av icke namngivna funktioner?

```
let students = [  
  "Lisa",  
  "Jessica",  
  "Ali",  
  "Stina"  
];  
  
students.forEach(function(element) {  
  console.log(element);  
});
```

Lisa

Jessica

Ali

Stina

Lägga till värde i array

- Arrayer kan använda sig av speciella array-metoder. En av dessa är push som används för att lägga till ett värde, en låda till arrayen.

```
let students = [  
  "Lisa",  
  "Jessica",  
  "Ali"  
];  
  
students.push("Stina");
```

Ta bort värde från array

- `pop` - tar bort från slutet av en array och returnerar värdet.
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/pop
- `shift` - tar bort från början av en array och returnerar värdet.
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/shift
- `splice` - tar bort från en array för ett specifikt index och returnerar värdet / värdena.
 - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice
- `filter` - returnerar en ny array med filtrerade element från en array. (Mer om denna senare.)
 - <https://love2dev.com/blog/javascript-remove-from-array/>

Objekt

- *Objekt* är variabler med "undervariabler".
 - Man kan ha objekt i arrayer.
 - <http://jsfiddle.net/2r1ph08o/>

```
let students = [];
```

```
let person = {  
  name: "Micke",  
  age: 42,  
  shoe_size: 43  
};
```

```
students.push(person);
```

```
person = {  
  name: "Jessica",  
  age: 25,  
  shoe_size: 36  
};
```

```
students.push(person);
```

```
console.log(students);
```

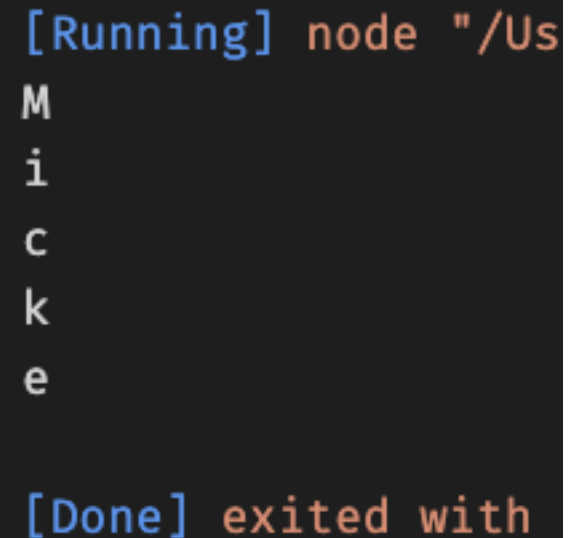
Uppgift

- Skapa en array som innehåller tre bokobjekt.
- Varje bokobjekt ska innehålla en titel, ett sidantal (hur många sidor boken innehåller) och en genre (fakta, roman osv)
- Gör en for-loop som loopar igenom alla arrayens värden och skriver ut titeln för varje bok.

Strings

- En sträng är faktiskt en array av tecken.

```
let name = "Micke";  
  
for ( let char of name ) {  
    console.log(char);  
}
```



```
[Running] node "/Us  
M  
i  
c  
k  
e  
  
[Done] exited with
```

Att hitta en sträng i en sträng

```
var str = "Hello world, welcome  
to the universe.";
var n = str.includes("world");
// true
```

Rensa sträng

- Vi kan kontrollera om ett tecken är tillåtet.

```
let allowed_chars = "abcd";  
if (allowed_chars.includes("b") ) {  
    // ...  
}
```

- *Uppgift:* Gör en funktion som går igenom en sträng och rensar bort alla tecken som inte är tillåtna. Tillåtna tecken ska vara a-ö, siffror samt mellanslag.

Rest

- Ibland vill man kunna ta emot ett okänt antal parametrar. Då kan man använda en `rest`-parameter. Den anges med tre punkter.

```
function sum (...numbers) {  
    console.log(numbers);  
}
```

```
sum (3, 7, 2, 8, 14);
```

```
[Running] node "/User  
[ 3, 7, 2, 8, 14 ]  
  
[Done] exited with co
```

Default-värde

- Parametrar kan innehålla default-värden, värden som används om man inte skickar någon parameter.

```
function add_numbers(a=2, b=5) {  
    return a + b;  
}
```

```
console.log(add_numbers());          // 7  
console.log(add_numbers(4));        // 9  
console.log(add_numbers(5, 7));     // 12
```

Sortering av arrayer

- En array kan sorteras med metoden sort.
Det fungerar lite olika för tal och strängar.

Vad tar sort-metoden för typ
av parameter?

```
var numbers = [9,3,12,5,1,88];  
var sortedNumbers= numbers.sort(function (a, b){ return a - b});
```

```
var names = ["Kalle", "Lisa", "Anna", "Emma"];  
var sortedNames= names.sort( );
```

Sortering av arrayer

- Om vi anger en funktion kan vi själva bestämma hur sorteringen ska fungera.
- Principen för vår jämförelsefunktion är att JS kommer att använda den för att jämföra två värden och vi får själva bestämma vilket som ska komma först.

```
function compare(a, b) {  
  if (a is less than b by some ordering criterion) {  
    return -1;  
  }  
  if (a is greater than b by the ordering criterion) {  
    return 1;  
  }  
  // a must be equal to b  
  return 0;  
}
```

Sortering av arrayer

- `let points = [1, 30, 4, 21, 100000];`
- Skriv en funktion som sorterar talen i numerisk ordning.

- Skapa en array som innehåller tre personobjekt (som har namn, ålder och favoritfärg).
- Skriv en sorteringsfunktion som sorterar objekten efter namn. Om namnen är samma, sortera efter ålder, äldst först.

Länkar

- https://www.w3schools.com/js/js_arrays.asp
- https://www.w3schools.com/js/js_array_methods.asp
- https://www.w3schools.com/js/js_array_sort.asp
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://www.w3schools.com/js/js_functions.asp
- https://www.tutorialspoint.com/javascript/javascript_functions.htm

Mer studiematerial

- https://developer.mozilla.org/sv-SE/docs/Web/JavaScript/Guide/Loops_and_iteration
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/for>
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/while>
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Basics>
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions>

Lambda-funktioner

- Uttryck som skapar funktioner.

- Pre ES6:

```
var anon = function (a, b) { return a + b };
```

- ES6:

```
// Ovanstående exempel:
```

```
var anon = (a, b) => a + b;
```

```
// Eller
```

```
var anon = (a, b) => { return a + b };
```

```
// Om vi bara har en parameter kan vi skippa
```

```
// parenteserna
```

```
var anon = a => a;
```



Lambda-funktioner

```
// Jämför  
[1,2,3,4].filter(function (value)  
{  
    return value % 2 === 0  
});
```

```
// med:  
[1,2,3,4].filter(value => value %  
2 === 0);
```

Map

- `map ()` -metoden skapar en ny array med resultatet av att anropa en funktion för varje array-element.
- `map ()` -metoden anropar den angivna funktionen en gång per element i arrayen i ordning.



```
const myArray = [1,2,3,4];

const myArrayTimesTwo = myArray.map((value, index, array) => {
  return value * 2;
});


console.log(myArray); // [1,2,3,4];
console.log(myArrayTimesTwo); // [2,4,6,8];
```

Map

- Utgå från er array med studenter.
- Med hjälp av map, skapa en ny array som enbart innehåller studenternas namn.

Filter

- `filter` tar emot samma argument som `map` och fungerar liknande. Enda skillnaden är att callback-funktionen måste returnera `true` eller `false`. Om den returnerar `true` kommer arrayen att behålla elementet, om den returnerar `false` kommer det att filtreras bort.



```
const myArray = [1,2,3,4];

const myEvenArray = myArray.filter((value, index, array) => {
  return value % 2 === 0;
});


console.log(myArray); // [1,2,3,4];
console.log(myEvenArray); // [2, 4];
```

Filter

- Utgå från er array med studenter.
- Med hjälp av filter, skapa en ny array som enbart innehåller de studenter som är äldre än 25.

Reduce

- reduce tar en array och reducerar den till ett enda värde. Du kan t ex använda det för att räkna ut medelvärdet av alla värden i arrayen.



```
const myArray = [1,2,3,4];

const sum = myArray.reduce((acc, currValue, currIndex, array) => {
  return acc + currValue;
}, 0);

const avg = sum / myArray.length;

console.log(avg); // 2.5
```

Reduce

- Utgå från er array med studenter.
- Med hjälp av reduce, räkna ut studenternas genomsnittliga ålder genom att först räknas ut summan och sedan dividera den med antalet åldrar.

Klasser

- Klasser är lite som ritningar eller mallar för objekt.
- Varje objekt som skapas från en klass, eller *instancieras*, kommer att innehålla alla de egenskaper och metoder som klassen innehåller. Detta exempel innehåller dock inget.

```
class Vehicle {  
    // Stuff  
}
```


Instansiering

- Man instansierar ett objekt genom nyckelordet *new*.

```
class Vehicle {  
    // Stuff  
}
```

```
let car = new Vehicle();  
console.log(car);
```

Metoder

- Man skriver metoder i en klass lite annorlunda än man gör i objekt.

```
class Vehicle {  
    hello() {  
        console.log("hello!");  
    }  
}
```

```
let car = new Vehicle();  
car.hello();
```

Konstruktor

- Det finns en speciell metod i en klass som heter *constructor*. Det körs alltid när man skapar ett nytt objekt av klassen.

```
class Vehicle {  
    constructor() {  
        console.log("creating an object");  
    }  
}
```

```
let car = new Vehicle();
```

Egenskaper

- Egenskaper skapas i konstruktion i JS.

```
class Vehicle {  
  constructor() {  
    this.make = "Volvo";  
    this.model = "V70";  
  }  
}
```

```
let car = new Vehicle();
```

Konstruktor

- Konstruktor är en vanlig funktion. Den kan ta emot parametrar. Det är vanligt att använda dem till att sätta egenskaper.

```
class Vehicle {  
    constructor(make) {  
        this.make = make;  
    }  
}
```

```
let car = new Vehicle();
```

Uppgift

- Skapa en klass som heter Student och har egenskaperna *name* och *age*.
- Skapa metoden *greeting* som skriver ut "Hello, " och studentens namn.
- Skapa en klass som heter Subject med egenskaperna *title* och *points*.
- Lägg till egenskapen *subjects* i Student-klassen. Låt den innehålla en tom array.
- Lägg till metoden *add_subject* i Student-klassen som lägger till ett objekt av klassen Subject i egenskapen *subjects*.

Getters / setters

- The get syntax binds an object property to a function that will be called when that property is looked up.

```
const obj = {  
  log: ['a', 'b', 'c'],  
  get latest() {  
    if (this.log.length === 0) {  
      return undefined;  
    }  
    return this.log[this.log.length - 1];  
  }  
};
```

```
console.log(obj.latest);
```

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/get>

Getters / setters

- The set syntax binds an object property to a function to be called when there is an attempt to set that property.

```
const language = {  
  set current(name) {  
    this.log.push(name);  
  },  
  log: []  
};
```

```
language.current = 'EN';  
language.current = 'FA';
```

```
console.log(language.log);
```

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/set>

Arv

- Arv låter oss skapa klasser som ärver sitt innehåll från en annan klass.
- Vi kan t ex skapa en klass, Person, som innehåller namn, personnr, hårfärg och sånt som beskriver en person. Vi askan sedan låta klasserna Student, Teacher, Boss, Employee ärva från denna klass.
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes/extends>

Super

- För att använda en metod från föräldraklassen använder vi super.
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/super>