

Upprepning

- Att äta mat kan vara ett exempel på upprepning. Man gör samma rörelse och tar mat från tallriken och stoppar i munnen. Detta upprepas tills man är mätt eller tills tallriken är tom.



Vad är en loop ?

- Iteration betyder upprepning. I programkod finns många exempel på när man behöver upprepa något flera gånger.
- En loop innebär att man upprepar något till dess att ett visst villkor är uppfyllt eller att man väljer att avbryta.
- Avbryta en loop görs med kommandot `break;`

Typer av loopar

- Vi kommer nu att titta på två typer av loopar.
- `for` och `while` loopar fungerar på ungefär samma sätt.

for

- Grunden för en for loop är

```
for ( startvärde; villkor för loopen; uppräkning )
```

- Uppfylls inte villkoret går koden aldrig in i koden i loopen.
- Följande loop upprepas 10 gånger. För varje gång ökas värdet på variabeln counter med 1.

```
for ( let counter = 1; counter < 11; counter++ ) {  
  // Kod som körs i loopen  
}
```

- Uppgift: Gör en loop som räknar från 0-30 och skriver ut de tal som är jämnt delbara med 3.

for-loop

Startvärde räknare

Villkor

Ökning räknare

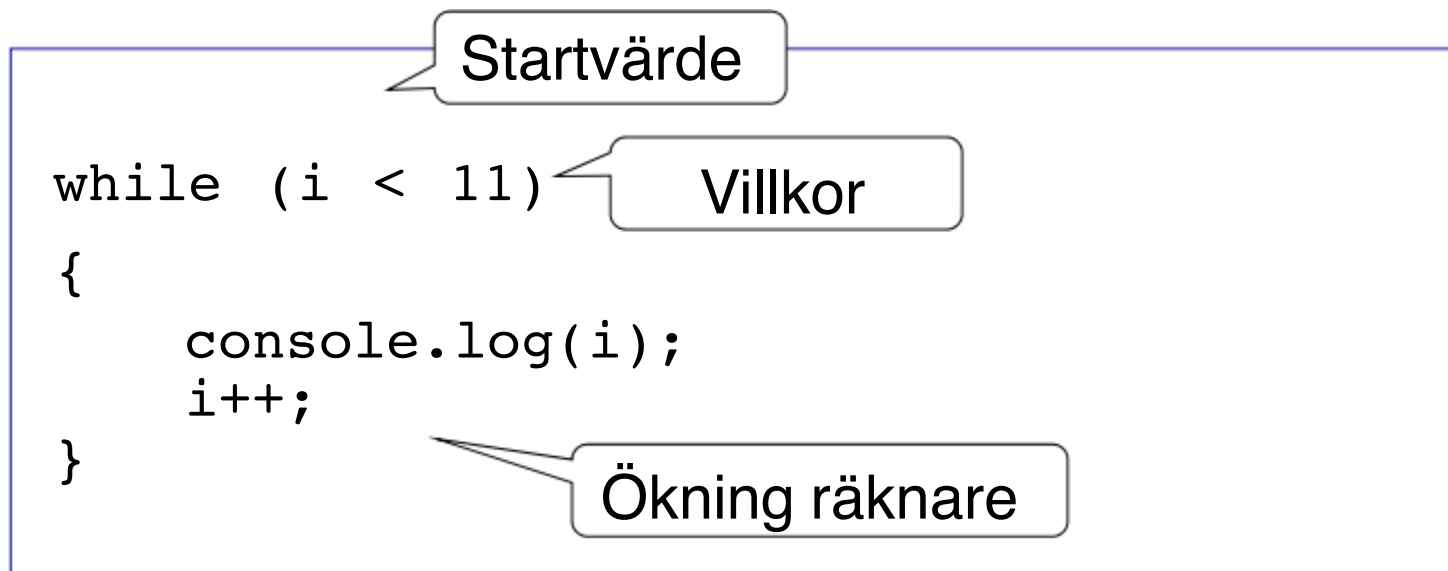
```
for (let i = 1; i < 11; i++)  
{  
  console.log(i);  
}
```

Kort om debugging

- VS Code har en inbyggd debugger.
 - <https://code.visualstudio.com/docs/editor/debugging>
- Nu kan vi följa koden och se vad olika variabler och uttryck innehåller.

while

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




Länkar

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Loops_and_iteration

https://www.w3schools.com/js/js_loop_for.asp

Värden som hör ihop

- Hur ska man göra om man har flera studenter?
- Begränsande!
 - Varje gång vi vill lägga till en student måste vi lägga till en ny variabel.

```
let student = "Kalle";  
let student1 = "Lisa";  
let student2 = "Jessica";  
let student3 = "Lisa";
```

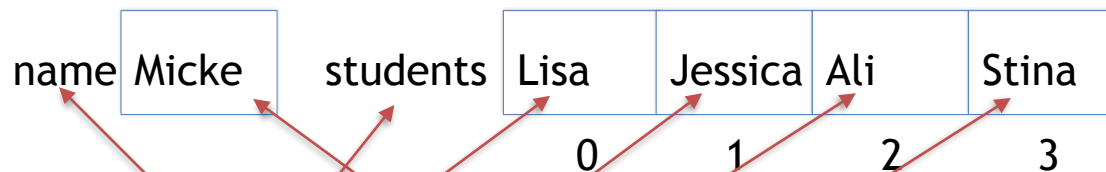
Array

En array är en speciell datatyp som kan innehålla flera värden.

```
let students = ["Lisa", "Jessica"];  
  
// Kan stå på flera rader  
  
let students = [  
    "Lisa",  
    "Jessica"  
];
```

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]);
```

Array

- *Uppgift:* Skapa en array och skriv ut varje värde i den.

```
let students = [  
  "Lisa",  
  "Jessica",  
  "Ali",  
  "Stina"  
];  
  
console.log(students[0]);  
// osv...
```

Lisa



Skapa en array

- Vi kan skapa en array med ett visst antal värden.
- Vi kan använda metoden `fill()` för att fylla arrayen med ett visst värde.

```
1  let arrA = new Array(7);  
2  
3  console.log(arrA);  
4  
5  arrA.fill(3);  
6  
7  console.log(arrA);  
8
```

```
[Running] node "/Users/  
[ <7 empty items> ]  
[  
  3, 3, 3, 3,  
  3, 3, 3  
]  
  
[Done] exited with code
```

Mer om array

- Vi kan använda egenskapen `length` för att ta reda på hur många värden vår array innehåller.
- *Uppgift:* Skapa en array och skriv ut varje värde i den med hjälp av en `for`-loop.

```
let students = [  
  "Lisa",  
  "Jessica",  
  "Ali",  
  "Stina"  
];  
  
let l = students.length;  
  
console.log(students[0]);
```

Objekt

- Saker i riktiga världen består ofta av olika variabler.
- Hänger de ihop på något sätt?
 - Ja, för oss människor. Inte för en dator.

```
let name = "Micke";  
let age = 42;  
let shoe_size = 43;
```

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

- Skapa en variabel med namnet `vehicle` med egenskaperna `make`, `model`, `color` och `speed`.

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
```

Avancerade variabler

- *Uppgift*: Skapa ett objekt som håller reda på en *course* med några lämpliga egenskaper, som *students*, *teacher*, *schedule* och/eller liknande. Students måste finnas och ska bestå av en array.
- Skriv ut några egenskaper.

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

Funktions beståndsdelar

- Funktioner är en kod-block som utförs när vi vill att den ska utföras.
- Funktioner kan ta emot data i så kallade parametrar.
- Funktioner kan kallas av JavaScript-kod eller när användaren gör något, t.ex. klickar på en knapp.

Funktioner

- Funktionens beståndsdelar

The diagram illustrates the components of a JavaScript function. It features a black rectangular area containing two code snippets. The first snippet is a function definition: `function addNumbers(a, b) { console.log(a + b); }`. The second snippet is a function call: `addNumbers(2, 2);`. Red boxes highlight specific parts of the code: `addNumbers` in the function name, `a, b` in the parameter list, `console.log(a + b);` in the function body, and `addNumbers(2, 2)` in the function call. Red arrows point from these boxes to a list of components on the right:

- Namn (Name)
- Parametrar (Parameters)
- Innehåll (Content)
- Anrop (Call)

 Below the list, a light gray box contains the number `4` in blue, and a blue greater-than sign `>` is positioned below it.

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

- Namn
- Parametrar
- Innehåll
- Anrop

4

>

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  
>
```

Funktioner

- Ser vi något mönster?
- Vad händer om vi måste ändra i formeln?

```
let temp_f = [80, 75, 88];  
let temp_c;  
let tmp;
```

```
tmp = temp_f[0];  
temp_c = (5/9) * (tmp-32)  
console.log(temp_c);
```

```
tmp = temp_f[1];  
temp_c = (5/9) * (tmp-32)  
console.log(temp_c);
```

```
tmp = temp_f[2];  
temp_c = (5/9) * (tmp-32)  
console.log(temp_c);
```


Funktioner

```
function toCelcius(fahrenheit) {  
  let temp_c = (5/9) * (fahrenheit-32);  
  console.log(temp_c);  
}  
  
let temp_f = [80, 75, 88];  
let tmp;  
  
tmp = temp_f[0];  
toCelcius(tmp);  
  
toCelcius(temp_f[1]);  
toCelcius(temp_f[2]);
```

26.666666666666668

23.888888888888889

31.111111111111114

>

Funktioner

- *Uppgift:* Gör en funktion som räknar ut arean av en rektangel och returnerar resultatet.



$$\text{Area} = a * b$$

```
function toCelcius(fahrenheit) {  
  let temp_c = (5/9) * (fahrenheit-32);  
  console.log(temp_c);  
}  
  
toCelcius(80);
```

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
```

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* fungerer som for... of men tar en funktion som parameter.

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

Lisa

Jessica

Ali

Stina

Lägga till värde i array

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

Ta bort värde från array

- `pop` - tar bort från slutet av en array och returnerar värdet.
- `shift` - tar bort från början av en array och returnerar värdet.
- `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  
};
```

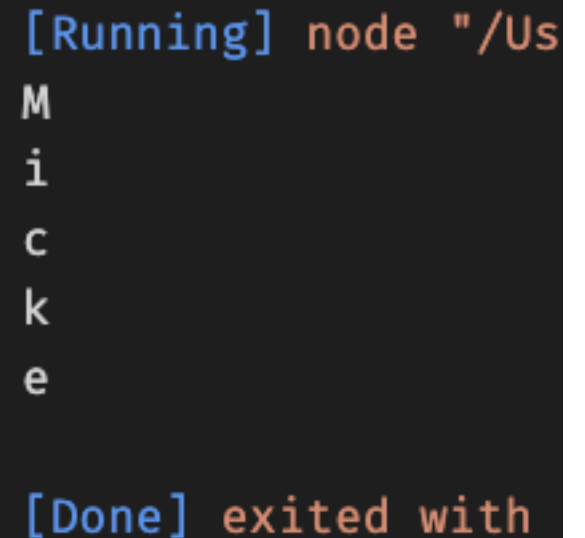
```
students.push(person);
```

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

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") ) {  
    // ...  
}
```

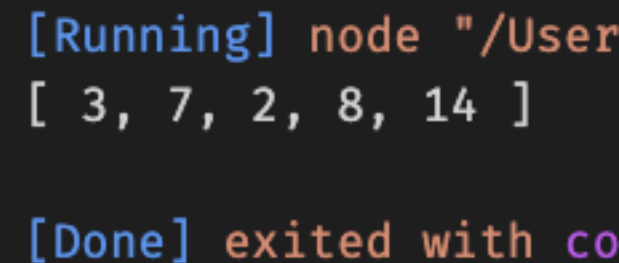
- *Uppgift:* Gör en funktion som går igenom en sträng och rensar bort alla tecken som inte är tillåtna. Tillåtna tecken ska vara a-ö, siffror samt mellanslag.

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
```


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

Länkar

- https://www.w3schools.com/js/js_functions.asp
- https://www.tutorialspoint.com/javascript/javascript_functions.htm