Variabler

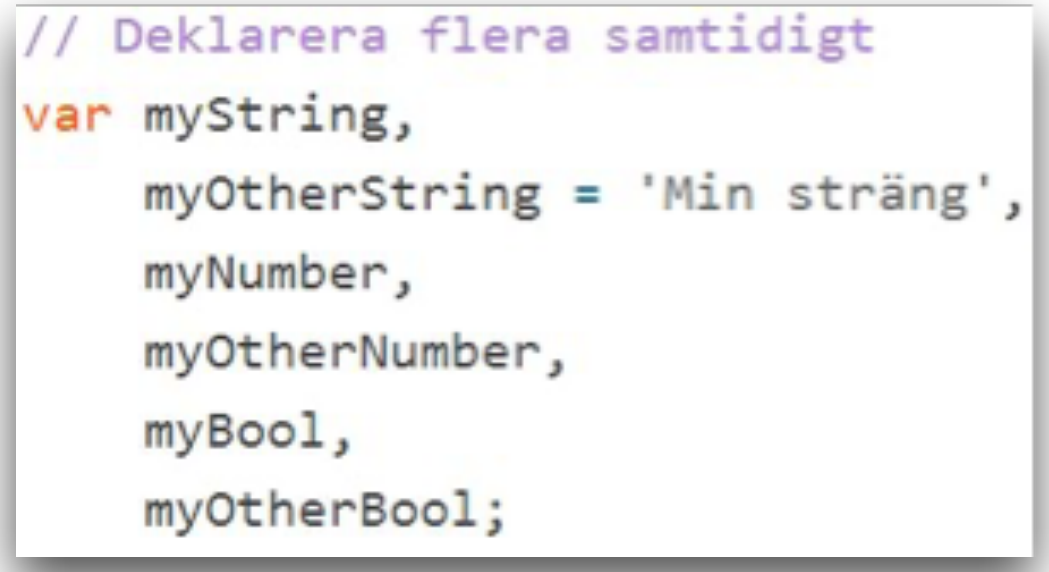
- Man deklarerar variabler med **var let const**
- Man kan välja att ge ett värde direkt eller vänta med det
- Man kan ersätta variabeln med ett värde av annan typ
- Strängar
- Numbers
  - Innefattar både heltal och flyttal
  - Ersätter Integer och Double
- Boolean
- (Functions)
  - En funktion kan också sparas i en variabel
- (Arrays)
- Vi kommer till dessa lite senare
- (Objects)
- Vi kommer till dessa ännu senare och inte i denna kurs

```
var myVariable = 5;  myVariable får värdet 5
console.log(myVariable); Skriver ut värdet i consolen

myVariable = "But now it is a string!”; Får ett nytt värde
console.log(myVariable); Skriver ut det nya värdet

var minVariabel; // Deklaration
var minAndraVariabel = 'Lite text'; // Deklaration +
initiering

minVariabel = 1337; // Initiering
minAndraVariabel = 555; // Tilldelning, exempel 2
// Detta är okej i JavaScript -
// variabeln ändrar typ själv i denna situationen!

//Strängar
var myString = ”Min sträng”;
```

```
//Numbers
var myNumber = 123;
var myOtherNumber = 123.456;

//Boolean
var myBool = true;
var myOtherBool = false;
```

Variabler - deklarera flera samtidigt

När ni gör såhär. Var uppmärksamma på varje enskilt kommatecken. Detta = , Det är det som separerar variablerna och för att avsluta deklarationen så används semicolon = ;
Det går också utmärkt att skriva detta efter varandra på en och samma rad, men då blir det svårare att se vad koden ska göra. Samt ännu svårare att debugga = felsöka.



Strängar i javascript

- Strängar i JavaScript kan skrivas både med **single quotes** och **double quotes**
- Men använd samma i början som i slutet!
- ”Hello World’ <- Fel
- ”Hello World” eller ’Hello World’

- När man sätter ihop en sträng med en annan variabel så är det precis som i andra programspråk, man använder sig av +

```
var a = ”Hello World”;
Var b = ” nice to meet”;

console.log(a + b); => ”Hello World nice to meet”
```

Typer i javascript

De primitiva typerna som finns i JavaScript är:

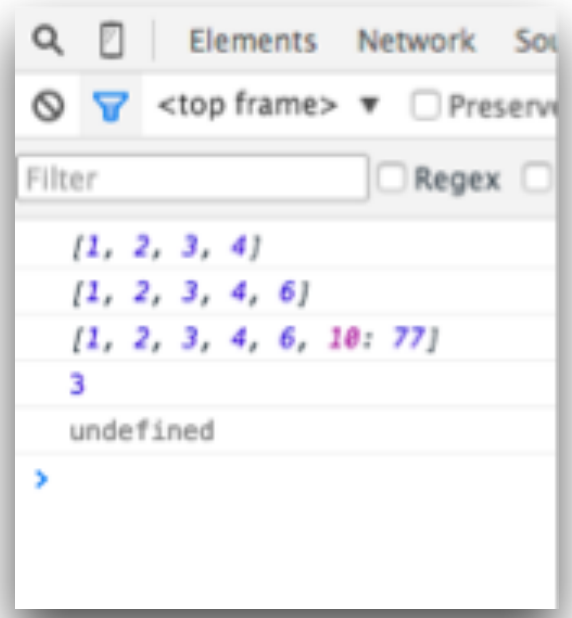
- **integer** - heltal
- **floating** number - decimaltal
- **string** - sträng
- **boolean** - sant eller falskt
- **undefined** - variabel som aldrig fått ett värde
- **null** - variabel som fått värdet ingenting

```
var myInt = 5;
var myFloat = 12.34;
var myString = "Hello World!";
var myBoolean = true;
var myNull = null;
var myUndefined;
```

Arrays i javascript

- Arrays skapar man med hakparenteser dvs [och]
- Man hämtar längden på samma sätt som i andra c-baserade språk, dvs **arrayName.length**

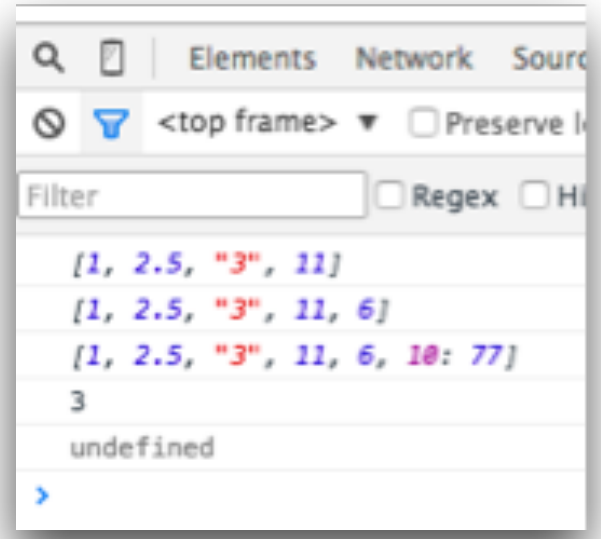
```
var emptyArray = [];
var myArray = [1,2,3,4];
console.log(myArray);

myArray.push(6);
console.log(myArray);

myArray[10] = 77;
console.log(myArray);

console.log(myArray[2]); //
returns 3
console.log(myArray[5]); //
returns undefined
```



Arrays i javascript

- Man kan också ha olika typer i samma array!

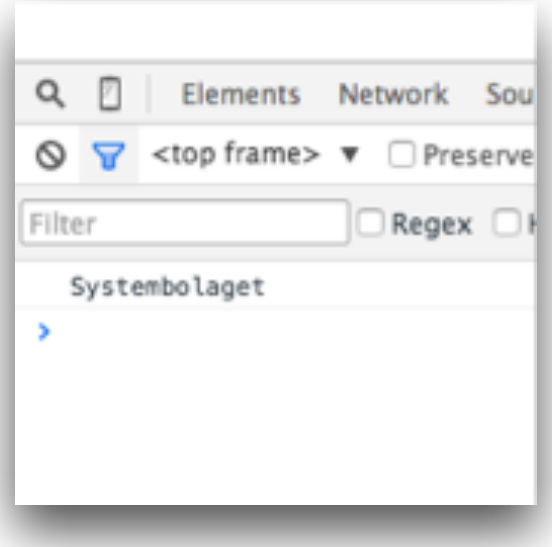
```
var emptyArray = [];

var myArray = [1,2.5,”3”,4+7];
console.log(myArray);


myArray.push(6);
console.log(myArray);


myArray[10] = 77;
console.log(myArray);
console.log(myArray[2]); // returns 3
console.log(myArray[5]); // returns undefined
```



If-satser

- Liknar if-satser från andra c-liknande språk väldigt mycket

```
var myAge = 26;

if(myAge > 67) {

  console.log(‘Retirement');

} else if(myAge > 20) {


  console.log(‘Systembolaget');

} else {

  console.log('Nothing :/’);

}
```



```
var myAge = 26;
var myAgeString = ”26";

if(myAge === myAgeString) {

  console.log('Same value and type!’);

}

if(myAge == myAgeString) {

  console.log('Same value!’);

}
```

Vad menas med “===” ??

DET BETYDER SAMMA TYP OCH SAMMA VÄRDE!

=== TESTAR SÅ ATT TYPERNA ÄR SAMMA

== TESTAR SÅ ATT VÄRDET KAN BLI SAMMA EFTER TYPKONVERTERING

Operationer - enkel matematik

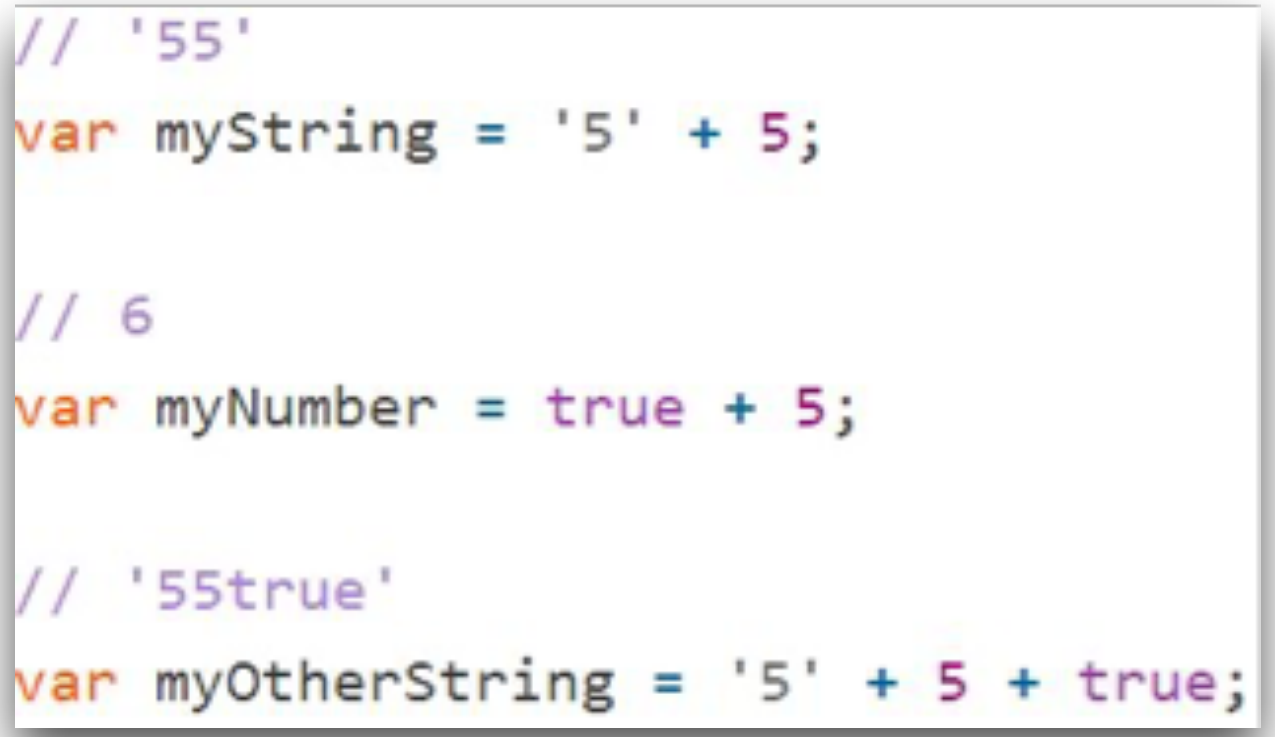
- Operationer fungerar i regel på samma sätt som i andra C-språk
- Företräde fungerar som i matematiken (Precedence)
  - * och / kommer före + och -
  - Paranteser () kan användas för att komma runt detta

```
var apples = 5,
    pears = 10,
    oranges = 15;

var fruits = apples + pears; // 15
var fruitPies = fruits/10; // 1.5

var wronglyWeightedFruits = 2*apples + oranges + pears; // 35
var weightedFruits = 2*(apples + oranges) + pears; // 50
```

Operationer - klurigare utan typing

- + används också för att konkatenera strängar

- Dynamisk typning skapar kluriheter….
- Typer konverteras “uppåt” så mycket som behövs
- a. Strängar
- b. Numbers
- c. Boolean



Logik

- A === B
- Kolla om A är detsamma som B

- OBS! Trippel === för att jämföra värden!
  Jämför även typ
  !==
  Även trippeltecken för “inte lika med”

- Större än och mindre än
- > och <
- Större/mindre än eller lika med
- >= och <=

```
55 === 55;      // true
55 !== 55;      // false
'55' === 55;    // false
'55' !== 55;    // true
55 > 54;        // true
55 <= 55;       // true
```

If, If-else, else

```
if(55 === 55) {
  // Yes!
} else {
  // Nix!
}
```

```
if(1 === 2) {
  // Nix!
} else {
  // Yes!
}
```

```
if(1 === 2) {
  // Nix!
} else if(2 === 2) {
  // Yes!
} else {
  // Nix!
}
```

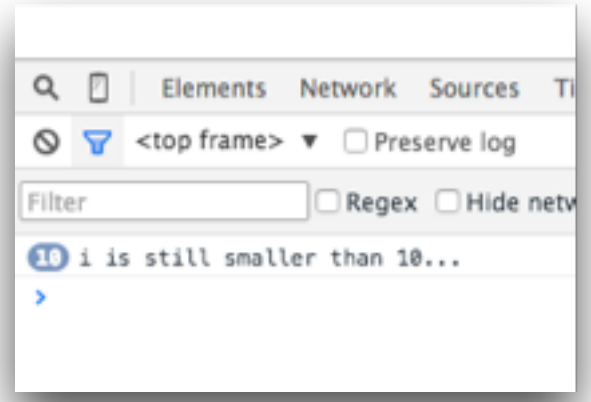
For-loopar

- Fungerar likadant som i de flesta c-språk

```
for(let i=0; i<10; i++) {

 console.log(i);

}
```

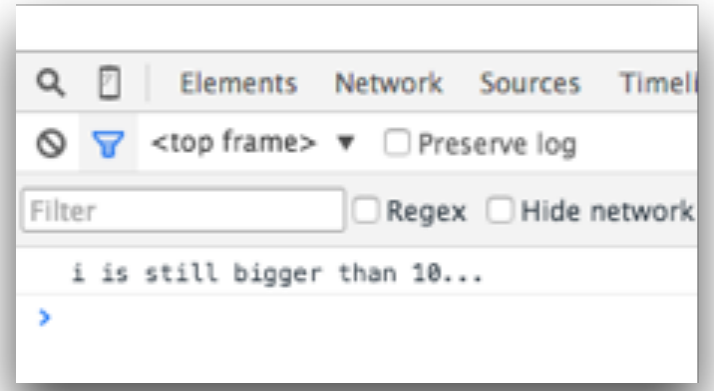
While-loopar

```
let i = 0;

while(i < 10) {

  console.log('i is still smaller than 10...');

  i++;

}
```



Do-While-loop

```
let i = 10;

do {

  console.log('i is still bigger than 0...');

  i --;

} while(i > 0);
```



Funktioner

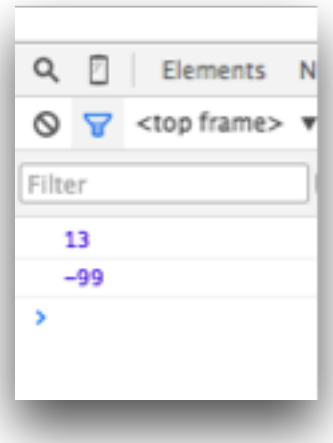
Liknar metoder i andra C-språk, men de kan vara helt självstående

- Argumenten har ingen typ utan endast namn
- Ingen returtyp heller!
- I JavaScript behöver argumenten inte skickas med
  - Men det kan ju bli lite fel...

```
function addTwoNumbers(a, b) {
  return a + b;
}
console.log(addTwoNumbers(6, 7));
```

- Man kan spara funktioner i variabler
- Man kan också spara funktioner i varabler på objekt!
- När man sparar så skriver man nameOfVariable = function(args) { ... }

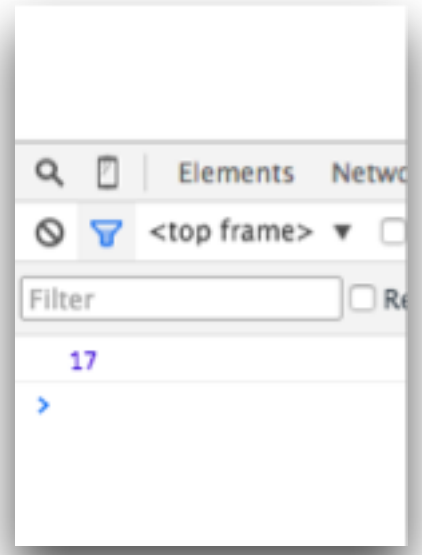
```
const myObject = { myValue: 45 };
myObject.niceFunction = function(a, b) { return a + b;
}
var anotherFunction = function(a) { return a * -1;
}
console.log(myObject.niceFunction(6, 7));
console.log(anotherFunction(99));
```



Funktioner som argument

- Eftersom man kan spara och lagra funktioner kan man
- skicka dem till andra funktioner som argument!
- Man kallar ofta en argument-funktion för **callback**
- Eftersom man kan spara och lagra funktioner kan man skicka dem till andra funktioner som argument!

```
let firstFunction = function(a) {
  return a+5;
}

let secondFunction = function(b,c) {
  return b*2 + c(b);
}
console.log(secondFunction(4,
firstFunction));
```



- Ett annat sätt att skicka in en funktion som är vanligt i JavaScript

```
let secondFunction = function(b,c) {
  return b*2 + c(b);
}

let result = secondFunction(4, function(a) {
  return a+5;
});
console.log(result);
```

Bonus Inmatning

- En metod för att mata in data i JavaScript är prompt
  - Skapar en popupruta där användaren får mata in en sträng
  - Man kan skicka med ett sträng-argument som visas i popup'n
  - Returnerar strängen därefter

```
// Sparar det användaren skriver in i variabeln name
var name = prompt('Skriv ditt namn!');
```