



JavaScript 1
alert("Lektion 4");

Utbildare: Mahmud Al Hakim

NACKADEMIN

Lektionstillfällets mål

Mål med lektionen

- Lite mer om Git och Github
- Lite mer om objekt och metoder
- JavaScript syntax
- Satser (statements)
- Kommentarer
- Variabler och värde
- Datatyper (typ)

Arbetsmetod

- Teori och praktik varvas under lektionen

NACKADEMIN

Kort summering av föregående lektion

Föregående lektion:

- Vi har gått igenom hur man länkar JavaScript till HTML.
- Jobbat lite med objektet console.
- Exekverat JavaScript-kod via terminalen med NodeJS.
- Vi har laborerat lite med Git och Github

Repetition: Git Basics med av Gregg Pollack

<https://youtu.be/ltzQbZrWLds>



Objektet document

- När ett HTML-dokument laddas i en webbläsare blir det ett dokument-objekt.
- Detta objekt kallas **document**.
- **write()** är metod i detta objekt.
- OBS! Metoden write() används enbart vid testning i webbläsaren (ej node)
- Exempel

```
document.write("Hello World!");
```

Metoden getElementById()

- `getElementById()` är en viktig metod som finns i objektet `document`
- Metoden hämtar ett HTML-element som har ett specifikt **id**
- Exempel: HTML-kod

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

- Exempel: JS-kod

```
document.getElementById("demo");
```

Objektet Element

- Ett HTML-objekt representerar ett HTML-element t.ex. body eller div
- En viktig egenskap (property) i detta objekt är `innerHTML`

```
let el = document.getElementById("demo");
// OBS! el blir ett objekt av typen HTML-element
el.innerHTML = "<h1>TEST</h1>";
```

- En annan viktig egenskap i detta objekt är `textContent`
- ```
el.textContent = "OBS! Enbart text, ej HTML";
```

# Syntax

---

Syntax är språkets regler och grammatik

---

Syntax handlar om att skriva korrekt källkod

---

En duktig programmera måste känna till  
språkets syntax!

# Satser (statements)

- JavaScript-applikationer består huvudsakligen av satser med lämplig syntax.
- Enkla satser avslutas vanligen med semikolon men detta är inte obligatoriskt i JavaScript.
- JavaScript har "**Automatic semicolon insertion**".
  - \* se länken nedan
- Flera satser kan förekomma på en enda rad men då måste varje sats avslutas med ett semikolon.

\* [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical\\_grammar#Automatic\\_semicolon\\_insertion](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#Automatic_semicolon_insertion)

# Satser – Exempel

```
alert("Hello World!")
```

Semikolon  
saknas här!  
Helt okej!

```
console.log("Hello World");
```

```
console.log("Hello 1"); console.log("Hello 2");
```

Obs!  
Semikolon  
obligatoriskt här



## Kommentarer

- Att kommentera källkod är en konst.
- Skriv i kommentaren VAD som görs och inte HUR det görs.
- Kommentera i en sammanhängande längre kommentar före ett avancerat block vad som görs.
- Undvik ”Papegoja -kommentarer”!

# Kommentarer – Exempel

```
// Detta är en kommentar på en rad

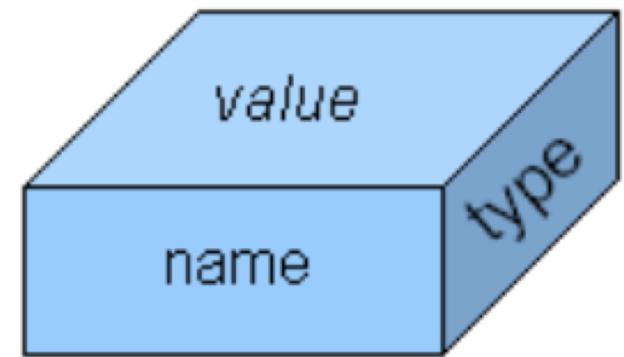
// Nedanstående sats är bortkommenterad
// alert("Hello World!");

alert("Hello World!"); // visar en alert
// Ovanstående kommentar är en "Papegoja-kommentarer"

/* Detta är en kommentar som sträcker sig
 över flera rader.
*/
```

# Variabler och värde

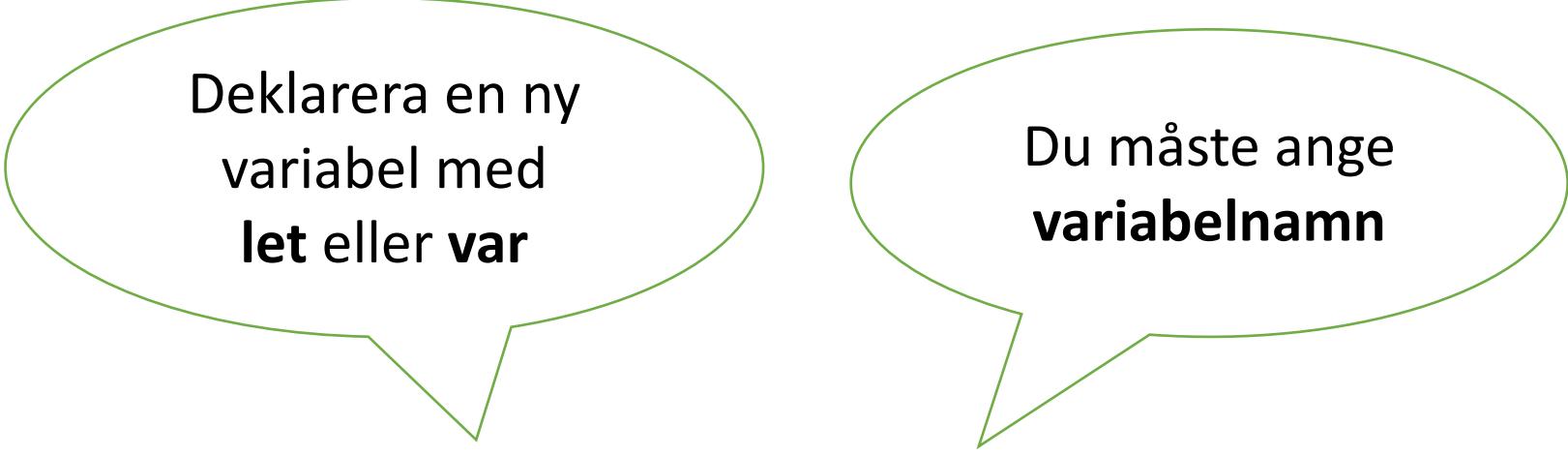
- En **variabel** är en platshållare för ett **värde** som kan ändras under programmets gång.
- I datorprogram använder man **variabler** för att lagra data.
- Data kan vara av olika slag (typer) t.ex. text eller tal.
- I JavaScript kan en variabel innehålla data av en valfri typ.
- Innan man använder en variabel i ett program måste man **deklarera** den.



# Deklarera en variabel

Deklarera en ny  
variabel med  
**let eller var**

Du måste ange  
**variabelnamn**



```
graph TD; A([Deklarera en ny variabel med let eller var]); B([Du måste ange variabelnamn]); C[let namn;]; D[var namn;]
```

**let namn;**  
**var namn;**



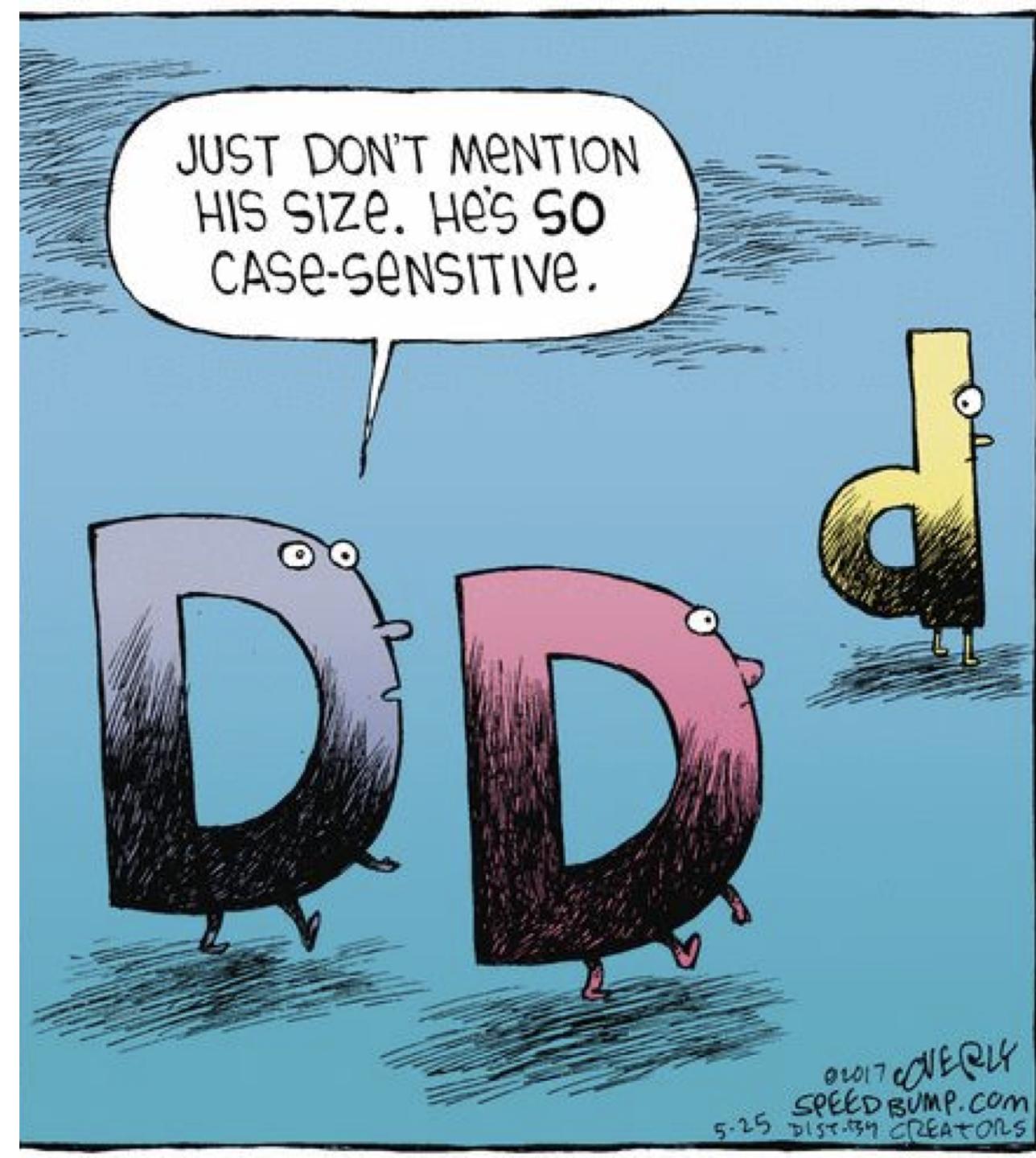
Du måste ge dina  
variabler **unika** namn

Kan du ge  
mig  
kassen?

Vilken?

# JavaScript är Case Sensitive

```
// OBS! tre olika variabler
let firstname;
let firstName;
let FIRSTNAME;
```



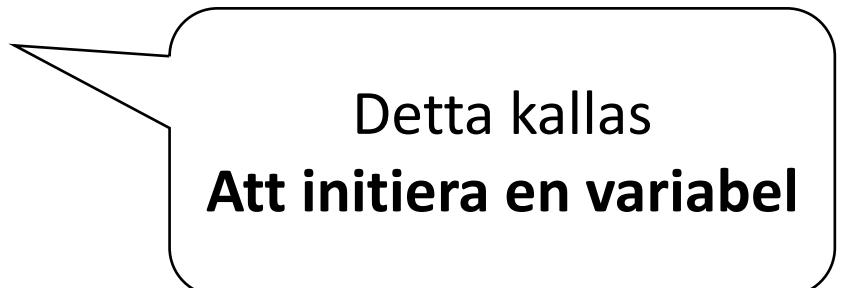
# Tilldelning

- Vill du ge en variabel ett värde använder du likhetstecknet (=) som kallas **tilldelningsoperator (assignment operator)**.
- Då du ger en variabel ett värde kallas det att du **tilldelar (assign)** variabeln ett värde t.ex.

```
namn = "Mahmud";
```

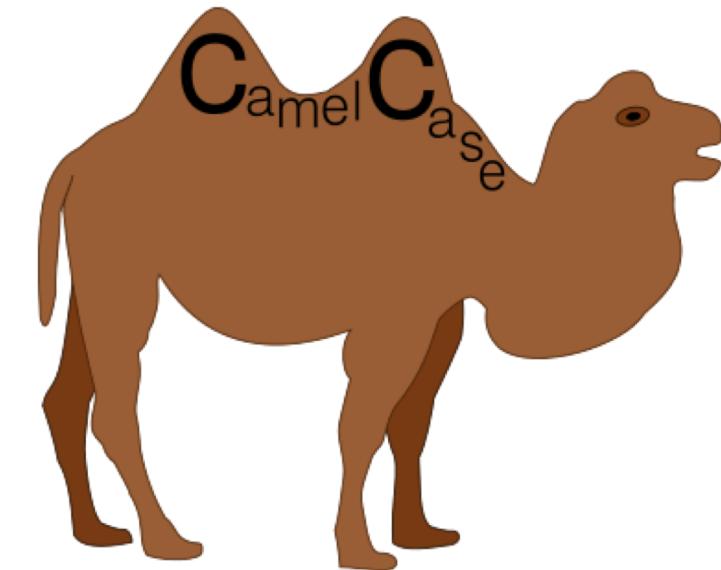
- Du kan deklarera och tilldela en variabel ett värde på en och samma gång t.ex.

```
let age = 45;
```



Detta kallas  
**Att initiera en variabel**

# camelCase (*kamelNotation*)



I JavaScript används camelCase för att skriva variabelnamn utan bindestreck eller mellanslag.

```
let firstname; // Ok men...
```

```
let firstName; // mycket bättre variabelnamn
```

# Variabler – Exempel

```
let firstName;
firstName = "Mahmud";
```

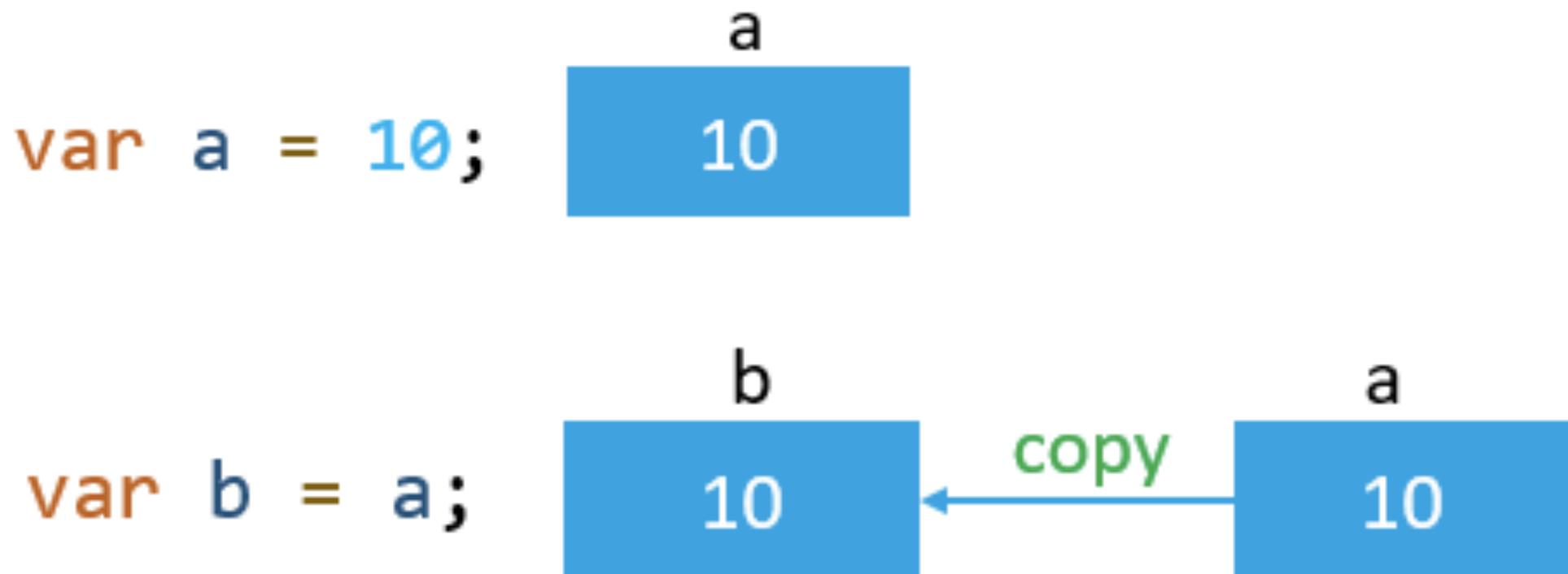
```
let lastName = "Al Hakim";
```

```
var tel , mobile;
tel = "08-1234567";
mobile = "073-123456";
```

# Variabler – Flera exempel

```
let firstName = "Mahmud" ,
lastName = "Al Hakim",
tel = "08-1234567",
mobile = "073-123456";
```

# Kopiera en variabel



Bildkälla: <http://www.javascripttutorial.net/javascript-primitive-vs-reference-values/>

# ReferenceError

- Vad händer om du försöker använda en **odeklarerad** variabel?

```
console.log(x);
```

✖ ➔ Uncaught ReferenceError: x is not defined  
at j.js:1



Vad händer om du glömt let eller var ?

```
x = 10;
// OBS! Ingen let eller var före variabelnamnet

console.log(x);
// skriver ut 10
```

# Strict mode

- Strict mode hindrar dig från att använda odeclarerade variabler.
- Du kan använda strict mode i alla dina skript genom att lägga till direktivet `"use strict";`
- Detta hjälper dig att skriva bättre källkod.

```
"use strict";
x = 10;
console.log(x);
// ReferenceError: x is not defined
```

# \$ och \_

- Variabelnamn måste börja med en bokstav, \$ eller \_

```
let x = "x är en variabel";
```

```
let $ = "dollar kan användas i variabelnamn";
```

```
let _ = "underscore kan användas i variabelnamn ";
```

```
let $_TEST = "detta är ett giltigt variabelnamn";
```

```
let MAX_VALUE = 10;
```

# Siffror i variabelnamn

- Variabelnamn får innehålla siffror men får inte börja med en siffra!

```
let x1;
```

```
let x2;
```

```
let 123; // OBS! SyntaxError!
```

— och .

- Variabelnamn får inte innehålla minustecken (-) eller punkt (.)

```
let first-name; // Unexpected token -
```

```
let first.name; // SyntaxError: Unexpected token .
```

// Mellanslag och komma då? Testa själv!

```
let first name;
```

```
let first,name;
```

# Reserverade ord

- Variabelnamn får inte vara nyckelord t.ex. **var** eller reserverade ord t.ex. **super**
- Här finns en lista över alla reserverade ord

[https://www.w3schools.com/js/js\\_reserved.asp](https://www.w3schools.com/js/js_reserved.asp)

- OBS! Var försiktig med att använda namn på inbyggda JavaScript-objekt t.ex. console

```
let console = "TEST"; // OBS!!!
console.log("TEST"); // console.log is not a function
https://www.w3schools.com/jsref/default.asp
```

# Tips! JavaScript variable name validator

<https://mothereff.in/js-variables>

A screenshot of a web browser window showing the "JavaScript variable name validator" tool. The title bar says "JavaScript variable name validator". The address bar shows the URL "https://mothereff.in/...". The main content area has a heading "JavaScript variable name validator" and a subtext: "Wondering if you can use a given string as a variable name in JavaScript? [Learn how it works](#), or just use this tool." Below this is a form with a label "Enter a variable name:" followed by a text input field containing the string "σ\_σ". To the right of the input field is a link "permalink". Below the input field is a green button with the same string "σ\_σ". At the bottom, a message says "That's a valid identifier according to ECMAScript 6 / Unicode 8.0.0." At the very bottom, it says "Made by [@mathias](#) — [fork this on GitHub!](#)".

JavaScript variable name validator

Wondering if you can use a given string as a variable name in JavaScript? [Learn how it works](#), or just use this tool.

Enter a variable name: [permalink](#)

σ\_σ

That's a valid identifier according to ECMAScript 6 / Unicode 8.0.0.

Made by [@mathias](#) — [fork this on GitHub!](#)

# Övningar

[https://www.w3schools.com/js/exercise\\_js.asp?filename=exercise\\_js\\_variables1](https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables1)

[https://www.w3schools.com/js/exercise\\_js.asp?filename=exercise\\_js\\_variables2](https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables2)

[https://www.w3schools.com/js/exercise\\_js.asp?filename=exercise\\_js\\_variables3](https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables3)

[https://www.w3schools.com/js/exercise\\_js.asp?filename=exercise\\_js\\_variables4](https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables4)

[https://www.w3schools.com/js/exercise\\_js.asp?filename=exercise\\_js\\_variables5](https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables5)

# Datatyper (typ)

## Vad är en datotyp?

- En datotyp (eller bara **typ**) är ett attribut för data som berättar för datorn (och programmeraren) vilken sorts typ data bär på t.ex. tal eller text.

## Primitiva datatyper

- Med **primitiva datatyper** menas de grundläggande typer som tillhandahålls direkt av programspråket.

## Dynamiska typer (dynamic typing )

- JavaScript har **dynamiska typer**, vilket innebär att samma variabel kan användas för att lagra olika typer.

# Primitiva datatyper i JavaScript

String

Number

Boolean

Undefined

Null

# Typen string

```
let firstName = "Mahmud";
let lastName = 'Al Hakim';
console.log(firstName,lastName);

console.log("It's nice");
```

# Typen number

```
let price = 5.5;
let quantity = 10;
let total = price * quantity;
console.log(total);
```

# Typen boolean

```
let ok = false;
let again =true;
if(again)
 console.log("Fortsätt...");
```

# Typen undefined

```
// En variabel som har deklarerats
// men inte tilldelats ett värde
// får ett odefinierat värde (undefined)
let name;
console.log(name); // ger undefined
```

# Let vs var

```
console.log(x1); // undefined
console.log(x2); // ReferenceError
var x1 = "En variabel deklarerad med var";
let x2 = "En variabel deklarerad med let";
```

# Typen null

```
let nothing = "Something"
nothing = "";
console.log(nothing);
nothing = null;
console.log(nothing);
// OBS!
// En variabel med värdet "" är inte samma sak som NULL
// Värdet "" är en tom sträng medan null representerar ingenting.
// Förväxlingen kan i vissa fall leda till logiska fel!
```

# Typeof

Typeof är en operator som returnerar typen.

```
let name = "Mahmud";
console.log(typeof name); // string
```

```
let age = 45;
console.log(typeof age); // number
```

```
age = "45";
console.log(typeof age); // OBS! string
```

# Typeof – Fortsättning

```
let again = true;
console.log(typeof again); // boolean
```

```
var declaredButUndefined;
console.log(typeof declaredButUndefined); // undefined
```

```
var nothing = null;
console.log(typeof nothing); // object
```

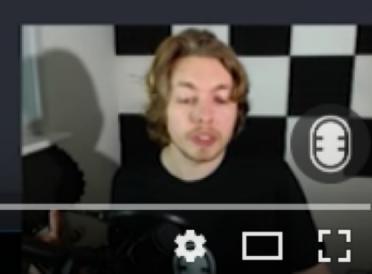
OBS! Typen borde vara null och inte object. Detta är en bugg i JS

Läs här: [https://www.w3schools.com/js/js\\_datatypes.asp](https://www.w3schools.com/js/js_datatypes.asp)

# Bra sammanfattning

## How to Create Variables in JavaScript

<https://youtu.be/9aGIAL16DL4>



The screenshot shows a video player interface with a video frame in the bottom right corner. The video frame displays a person with long hair and a beard, wearing a black shirt, sitting in front of a computer screen. The computer screen shows a browser window and an Atom code editor side-by-side.

The browser window title is "JavaScript tutorial" and the address bar shows "127.0.0.1:3000". The code editor window title is "index.html" and the path is "C:\YouTube\RECORDING\PROJECTS\JS\js7". The Atom interface includes a "Console" tab with the message "Live reload enabled." and file paths "(index):37" and "(index):54".

The code in the editor is:

```
8 </head>
9 <body>
10
11 </body>
12 <script>
13
14 var a = 20;
15 var b = 20;
16 var sum = a + b;
17
18 var a, b, sum;
19 a = 20;
20 b = 20;
21 sum = a + b;
22
23 console.log(sum);
```

# Summering av dagens lektion

- Vi har idag jobbat med grundläggande JavaScript syntax
- Satser (statements) och kommentarer
- Variabler, värde och datatyper (typer)
- Reflektioner kring dagens lektion?
  - Vad tar du med dig från dagens lektion?
  - Finns det något som var extra svårt att förstå?
  - Finns det något som vi behöver repetera?
  - Hur upplevde du dagens arbetsmetoder?

# Framåtblick inför nästa lektion

- Läs: Bok 2, sid. 53–69
- Under nästa lektion kommer vi att arbeta med arrayer, uttryck (expressions) och operatorer