

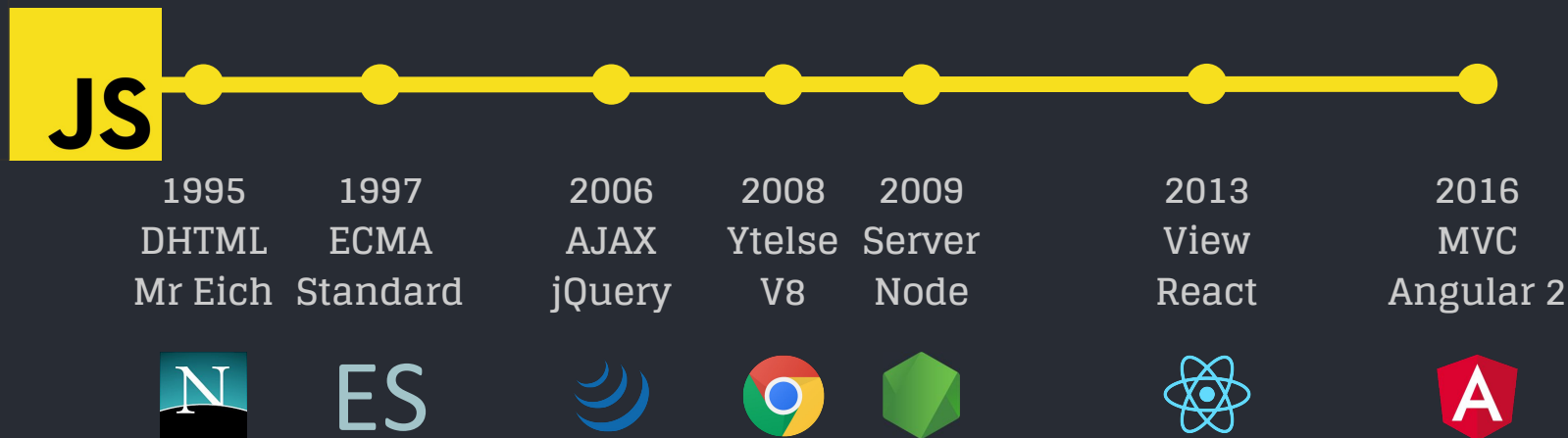


JavaScript



Tilbakeblikk

Fortellingen om JS



Hva betyr så dette?

JavaScript ble opprinnelig utviklet for å håndtere skjemavalidering og for å gjøre websider mer dynamiske.

I dag kan man utvikle avanserte applikasjoner i JS. Vi kan bruke språket både på klient- og serversiden.

JS kommer dessuten til IoT (f.eks. JerryScript: en JS-engine for mikrokontrollere)

*Any application that can be written
in JavaScript, will eventually be
written in JavaScript.*

Jeff Atwood

Løsevet fra nettleseren

I dag er JavaScript løsevet fra nettleseren gjennom såkalte “JS engines”.

Men ulike nettlesere benytter seg av disse:

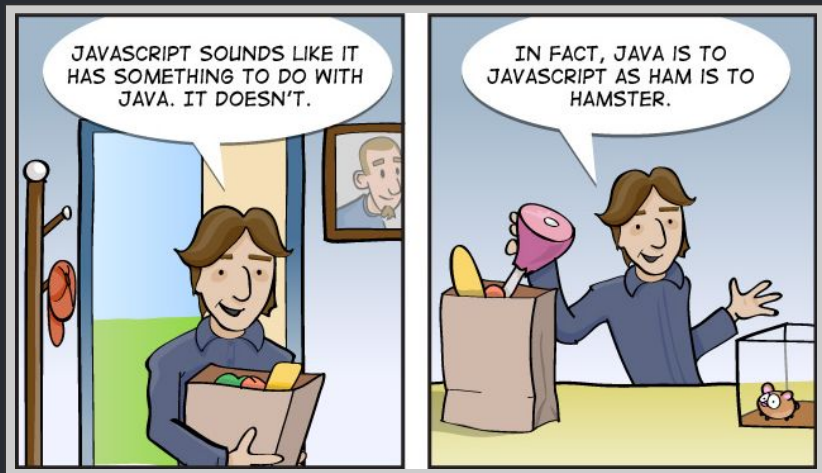
- V8: Google Chrome
- Chakra: Microsoft IE/Edge
- Nitro: Apple Safari

Merk: JavaScript \neq Java

Skiftet navn til JavaScript fordi det ga publisitet.

Har syntaks som likner litt på java sin.

Kuriositet: Java prøvde seg i nettleseren med applets, men det gikk ikke så bra.



Legge til JavaScript

Vi anbefaler å legge koden i en separat js-fil. Vi referer til denne i HTML-dokumentet.

```
<head>  
  <meta charset="utf-8">  
  <title>Tittel her!</title>  
  <link rel="stylesheet" href="css/screen.css" media="screen" />  
  <script src="js/scripts.js"></script>  
</head>
```


Syntaks

```
if (top != self) {  
    function calcWidth() {  
        var wW = 0;  
        if (typeof window.innerWidth == 'number')  
            wW = window.innerWidth;  
        } else if (document.documentElement) {  
            wW = document.documentElement.clientWidth;  
        } else if (document.body) && document.body.clientWidth {  
            wW = document.body.clientWidth;  
        }  
        if (3H == document.documentElement) {  
            n = window.innerHeight;  
            All &  
        }  
    }  
}
```

JS er casesensitivt

Vi skiller mellom små og store bokstaver.

Det betyr at hei og Hei er forskjellig.

Kommentarer

Vi kan kommentere linje for linje:

```
//Kommentar.  
//Linje for linje...
```

Vi kan også kommentere ut flere linjer på en gang:

```
/*  
Kommentar over  
flere linjer...  
*/
```

Uttrykk

Hvert uttrykk avsluttes med et semikolon.

Koden er en sekvens med uttrykk og kjøres i den rekkefølgen den står.

```
c = a * b;
```

```
c = c/b;
```

Variabler

En variabel refererer til og lagrer en verdi i minnet.

```
var age = 50;
```

Variabler

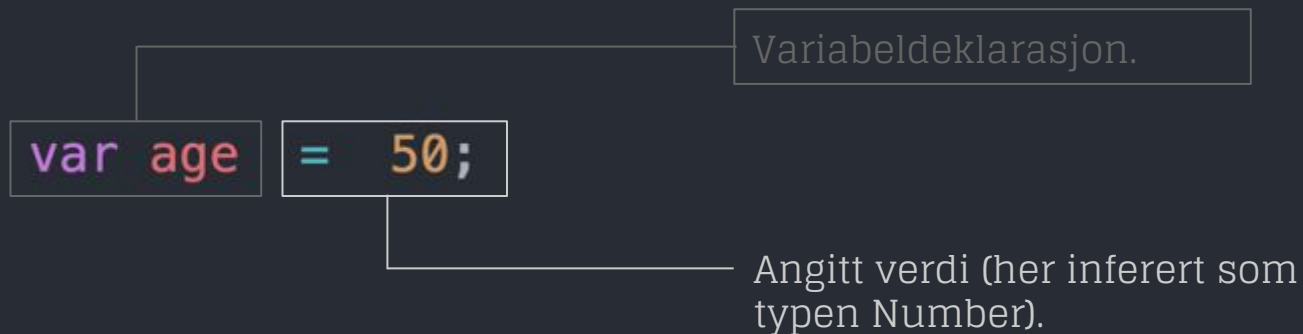
En variabel refererer til og lagrer en verdi i minnet.

— Variabeldeklarasjon.

```
var age = 50;
```

Variabler

En variabel refererer til og lagrer en verdi i minnet.



Typer

En type bestemmer hva slags verdi variabelen kan ha.

```
var age = 50; //Number, kan også ha desimaler  
var name = "Ola"; // String, kan også bruke enkeltfnutter  
var male = true; //Boolean, true eller false
```


Typer

JS er løst typet; vi definerer ikke variabeltypen.

```
var a = 50;           a = "0la ";  
var b = 150;          b = "Nordmann";  
var sum = a+b; //200   var navn = a+b; //0la Nordmann  
  
var derforTrengerViSterkeTyper = 5 + "Hei"; //5Hei
```

Arrays

Array er en spesiell variabel som kan holde flere verdier på en gang. En array er et objekt.

De opprettes slik:

```
var cars = ["Toyota", "BMW", "Volkswagen"];
```

//or with the new keyword

```
var cars = new Array("Toyota", "BMW", "Volkswagen");
```

Arrays

Vi henter ut en verdi ved å angi indeks:

```
var car = cars[0]; //Toyota
```

Vi kan sette/overskrive en verdi slik:

```
cars[0] = "Kia"; //var Toyota, er nå Kia
```

Nøkkelord

Et nøkkelord er en identifikator med en spesiell mening.

JS har en haug med slike nøkkelord. De er reserverte, som betyr at de ikke kan brukes til å navngi variabler eller funksjoner.

var er et slik nøkkelord: det angir en ny variabel. Andre eksempler er **if**, **else** og **function**.



Funksjoner

Funksjoner

En funksjon er en blokk med kode som utfører noe og returnerer en verdi.

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

Funksjoner

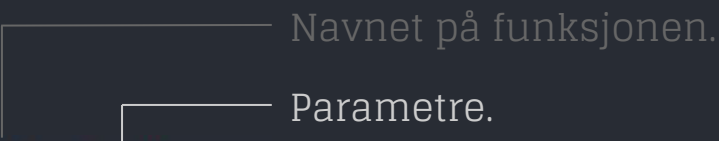
En funksjon er en blokk med kode som utfører noe og returnerer en verdi.

Navnet på funksjonen.

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


Funksjoner

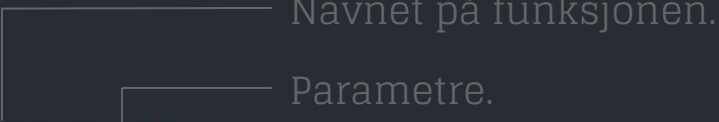
En funksjon er en blokk med kode som utfører noe og returnerer en verdi.



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

Funksjoner

En funksjon er en blokk med kode som utfører noe og returnerer en verdi.



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

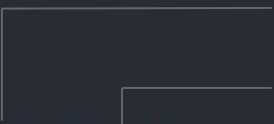
Navnet på funksjonen.

Parametre.

Kode som utfører noe.

Funksjoner

En funksjon er en blokk med kode som utfører noe og returnerer en verdi.



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

Navnet på funksjonen.

Parametre.

Kode som utfører noe.

En verdi returneres.

Funksjoner

Etter at vi har definert funksjonen kan vi kalle den.

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

```
console.log("50 + 100 is " + add(50, 100));
```

his & OBJECT
PROTOTYPES

Objekter

DON'T KNOW

DON'T KNOW

Objekter

Objekter er variabler med nøkkel:verdi par.

```
var person = {  
  age: 50,  
  firstName: "Ola",  
  lastName: "Nordmann",  
  fullName: function() { return this.firstName + " " + this.lastName; }  
};
```

Objekter

Når vi har definert et objekt er det en variabel, og vi kan referere til dets egenskaper.

```
var personAge = person.age;
```

```
var personFullName = person.fullName();
```

```
var another = person; //another og person peker til samme objekt
```

Objekter

Merk at en verdi kan være en primitiv type (tall, tekst osv), andre objekter eller funksjoner.

```
var car = {  
  name: "Aston Martin DB9",  
  owner: {  
    firstName: "Atle",  
    lastName: "Ølsø"  
  },  
  sell: function(priceInDollars) { return 10000000000; }  
};
```


Date

Date er et innebygd objekt i JavaScript med flere nyttige funksjoner. Noen eksempler:

```
var date = new Date(); //ny dato med tid
var timeInMs = date.getTime(); //tid i ms etter 01.01.1970
var day = date.getDay(); //man=0, søn=6

console.log(new Date("2016-09-16 12:15:00")); //input i ISO-format
//Fri Sep 16 2016 12:15:00 GMT+0200 (CEST)
```

Arrays

Et array er et objekt. Det har flere tilknyttede funksjoner. Noen eksempler:

```
var cars = ["Toyota", "BMW", "Volkswagen"];
```

```
cars.sort(); //sortert: BMW, Toyota, Volkswagen
```

```
cars.pop(); //fjerner siste element: BMW, Toyota
```

```
cars.push("Kia"); //legger til element: BMW, Toyota, Kia
```




The background features a repeating pattern of Venn diagrams with three overlapping circles. Each diagram is associated with a logical expression in a light blue font. The expressions include: 'A' (top left), 'Not A' (top center), 'A And B' (top right), 'A Or B' (far right), 'Or B Or C' (middle left), '(A Or B) and Not C' (bottom left), 'C And Not A and Not B' (bottom center), and 'B Or (C And A)' (bottom right). A dark blue horizontal rectangle is centered across the middle of the image, containing the word 'Operatorer' in white.

Operatorer

Aritmetiske

```
z = x + y; //addisjon
```

```
x = x - y; //divisjon
```

```
x = x * y; //multiplikasjon
```

```
x = x / y; //divisjon
```

```
x = x % y; //modulus: rest etter divisjon. 5 % 2 = 1
```

```
x++; //inkrement: øker x med 1
```

```
x--; //dekrement: minker x med 1
```

Tilordning

```
x = y;
```

```
x += y; //tilsvarer x = x + y
```

```
x -= y; //tilsvarer x = x - y
```

```
x *= y; //tilsvarer x = x * y
```

```
x /= y; //tilsvarer x = x / y
```

```
x %= y; //tilsvarer x = x % y
```

Sammenlikning

```
x == y //lik  
x != y //ulik  
x > y  //større enn  
x >= y //større eller lik  
x < y  //mindre enn  
x <= y //mindre eller lik
```

```
if (age < 18) {  
    console.log("For ung for førerkort");  
}
```

Logiske

```
x && y //x og y
```

```
x || y //x eller y
```

```
!x      //ikke x
```

```
if (x>=1 && x<=10) {  
    console.log("x er mellom 1 og 10");  
}
```




If, Else og Switch

If og Else

If utfører noe hvis en betingelse er sann.

```
if (age < 18) {  
  console.log("For ung for førerkort");  
}
```

Else utfører noe hvis betingelsen under if-uttrykket er usann.

```
if (age < 18) {  
  console.log("For ung for førerkort");  
} else {  
  console.log("Kjør i vei");  
}
```

Else-If

Vi kan bruke `else if` hvis vi ønsker å spesifisere nye betingelser etter `if`.

```
if (season == "summer") {  
    console.log("Det er sommer");  
} else if (season == "autumn") {  
    console.log("Det er høst");  
} else if (season == "winter") {  
    console.log("Det er vinter");  
} else {  
    console.log("Det er vår");  
}
```


Switch

Har vi mange ulike betingelser kan vi bruke **switch**.

Husk **break** etter hver kodeblokk, det stopper videre eksekvering.

```
switch (season) {  
  case ("summer"):  
    console.log("Det er sommer");  
    break;  
  case ("autumn"):  
    console.log("Det er høst");break;  
  case ("winter"):  
    console.log("Det er vinter");  
    break;  
  case ("spring"):  
    console.log("Det er vår");  
    break;  
  default: console.log("Ukjent årstid");  
}
```


A low-angle shot of a roller coaster with red tracks and blue supports. The coaster features two large loops. The background is a sky with soft, golden clouds from a setting or rising sun. A semi-transparent grey rectangle is centered over the image, containing the word 'Løkker' in white text.

Løkker

For-løkke

For-løkka itererer over noe så lenge betingelsen er sann.

```
var cars = ["Toyota", "BMW", "Volkswagen"];
```

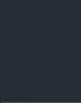
```
for (i = 0; i < cars.length; i++) {  
  console.log(cars[i]);  
}
```

For-løkke

For-løkka itererer over noe så lenge betingelsen er sann.

```
var cars = ["Toyota", "BMW", "Volkswagen"];
```

Teller starter på 0.



```
for (i = 0; i < cars.length; i++) {  
  console.log(cars[i]);  
}
```


For-løkke

For-løkka itererer over noe så lenge betingelsen er sann.

```
var cars = ["Toyota", "BMW", "Volkswagen"];
```

Teller starter på 0.

Betingelse for fortsatt
eksekvering.

```
for (i = 0; i < cars.length; i++) {  
  console.log(cars[i]);  
}
```

For-løkke

For-løkka itererer over noe så lenge betingelsen er sann.

```
var cars = ["Toyota", "BMW", "Volkswagen"];
```

Teller starter på 0.

Betingelse for fortsatt
eksekvering.

Øk teller med 1 for hver iterasjon.

```
for (i = 0; i < cars.length; i++) {  
  console.log(cars[i]);  
}
```

While-løkke

While-løkka gjentar seg helt til betingelsen blir usann.

```
var i = 0;

while (i < 10) {
  console.log("Value: " + i);
  i++;
}
```


A large, gnarled tree with thick, dark brown branches dominates the foreground. The tree's leaves are a mix of green and yellow, suggesting an autumn setting. In the background, a black metal park bench sits on a grassy area. The scene is bathed in warm, golden light, likely from the setting or rising sun. A semi-transparent dark rectangle is centered over the image, containing the text "Document Object Model" in white, serif font.

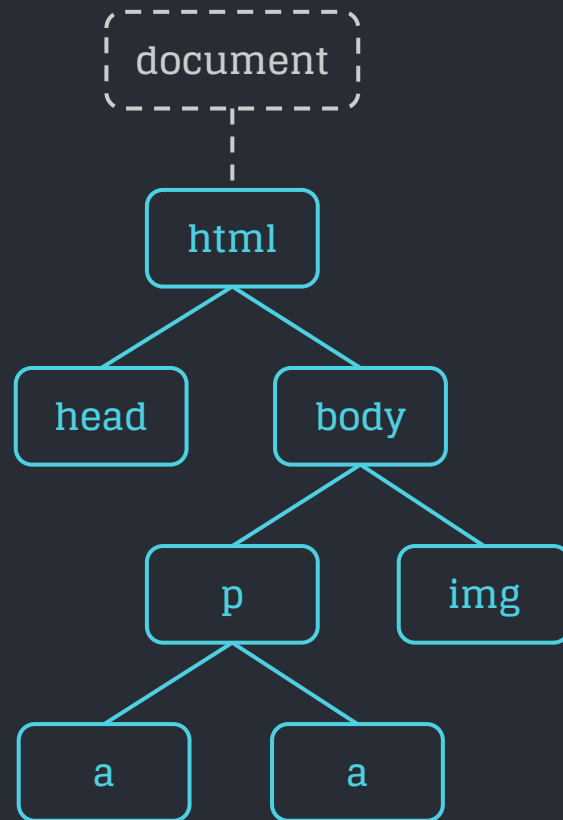
Document Object Model

DOM

Når websida lastes inn genererer nettleseren en DOM.

En DOM er HTML som en veldefinert trestruktur av objekter.

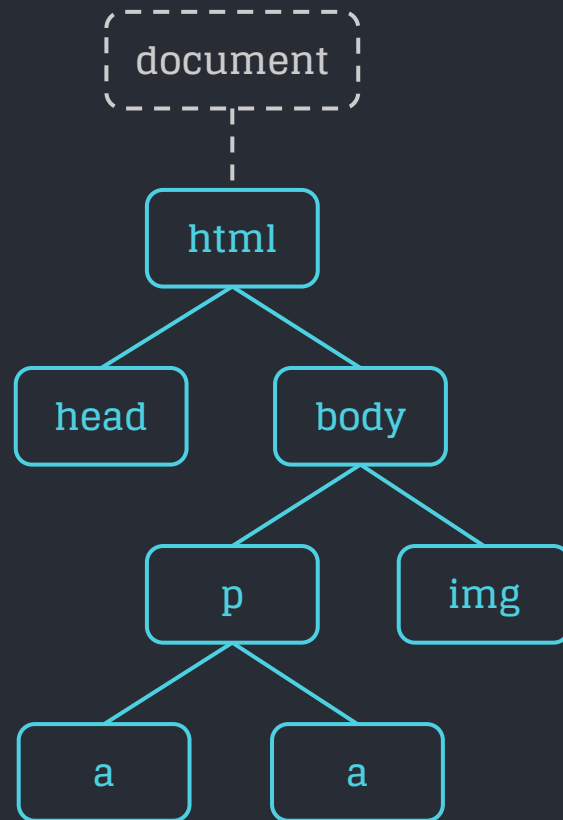
html er rotelementet i strukturen.



DOM

I JS har vi metoder for å operere på DOMen, og dermed endre HTMLen. Vi kan:

- Hente ut og slette elementer.
- Endre verdier i HTMLen.
- Skifte stiler (CSS).
- Respondere på hendelser.



Basert på Id

Vi kan hente ut et unikt element ved å angi id-attributtet.

I HTML:

```
<div id="firmanavn">Firmanavn AS</div>
```

I JavaScript:

```
var firmaNavn = document.getElementById("firmanavn");
```


Basert på klasse

Vi kan hente ut elementer basert på klassenavn.

I HTML:

```
<ul>
  <li class="list">Element 1</li>
  <li class="list">Element 2</li>
</ul>
```

I JavaScript (gir tilbake en array):

```
var listElementer = document.getElementsByClassName('list');
```

Basert på selektor

Vi kan hente ut med selektor.

Basert på id:

```
var element = document.querySelector("#myId");
```

Basert på klasse:

```
var elements = document.querySelectorAll(".myClass");
```

innerHTML

Vi kan endre HTMLen til et element vi har hentet ut.

```
var firmanavn = document.getElementById("firmanavn");  
firmanavn.innerHTML = "Navn på firma via JS";
```

Endre stil

Vi kan endre stil. Først henter vi ut ett element basert på id:

```
document.querySelector("#myId").id = "newId";
```

Så flere basert på klasse. Da må vi iterere over elementene:

```
var elements = document.querySelectorAll(".myClass");

for (var i = 0; i < elements.length; i++) {
  elements[i].className = "newClass";
}
```

Hendelser

Vi kan kalle funksjoner ved hendelser (events). En hendelse er noe som skjer med et HTML-element.

Eksempler:

- Siden lastes inn første gang: `onLoad`.
- Bruker klikker på en knapp: `onClick`.
- Muspeker går over et element: `onMouseOver`.
- Data blir sendt til tjeneren: `onSubmit`.

Hendelser

I HTMLen definerer vi hendelsen som et attributt på elementet. Verdien er javascript-koden som skal kjøres.

Eksempler:

```
<button onclick="doOnClick()">My Button</button>
```

```
<div onmouseover="doOnMouseOver" onmouseout="doOnMouseOut">  
  My element  
</div>
```




Til slutt

God lesbarhet

Det er viktig at koden har god lesbarhet:

http://www.w3schools.com/js/js_conventions.asp

Spørsmål?

Ta gjerne kontakt på e-post: atle.olsen@ntnu.no eller via it`s Learning.