Kodestil JavaScript

Å følge en bestemt kodestil gjør at koden man skriver ikke bare blir mer leselig, men lettere å vedlikeholde over tid. Hvert programmeringsspråk har ofte sin egen kodestil og inneholder ofte:

* Regler for navngiving og deklarasjon av variabler og funksjoner
* Regler for bruk av «white space» (mellomrom og tab), innrykk og komma.
* Programmeringsprinsipper og beste praksis

## Variabelnavn

Variabelnavn bruker ofte camelCase. Det vil si at variabelnavnet starter alltid med en liten bokstav og består stort sett av små bokstaver. Unntaket er hvis variabelnavnet inneholder to eller flere ord. Da er første bokstav i ord nummer to, tre, fire osv. alltid stor.

let fornavn = "Ola";

let etternavn = "Nordmann";

let pris = 19.90;

let moms = 0.20;

let fullPris = pris + (pris \* moms)

## Mellomrom og operatorer

Alltid ha mellomrom mellom operatorer og etter komma.

let x = y + z;

const mittArray = ["Volvo", "Hyundai", "Mazda"];

## Innrykk

I kode-blokker bør man alltid ha innrykk. I JavaScript er ikke dette nødvendig for at koden skal kjøre, men det gjør koden mer lesbar. Kode-blokk kan være if-blokker, for-løkker, while-løkker, funksjoner, osv.

Hvor stort innrykket skal være er en evig diskusjon i programmeringsmiljøet. Noen mener én tab er greit, mens andre sverger til fire mellomrom. I VS Code, hvis du trykker tab, så gjøres dette automatisk om til fire mellomrom så bare bruk én tab.

if (himmel === "blå") {

    console.log("Alt er greit!");

}

function minFunksjon(x) {

    let y = x \* x;

    return y;

}

## Regler for uttrykk

Et **uttrykk** er én instruksjon med kode som datamaskinen skal gjøre. Dette kan være instruksjoner som å deklarere en variabel, tilordne en variabel en ny verdi, kalle på en funksjon osv.

Alle enkle uttrykk i JavaScript må avsluttes med et semikolon (slik som i eksemplene ovenfor). Blokker er ikke uttrykk og trenger ikke semikolon etter. **Merk at i Svelte så er det lov å skrive JavaScript kode uten å avslutte hvert uttrykk med et semikolon.**

Her er noen regler for mer sammensatte uttrykk:

* Åpningsparentesen puttes på slutten av første linje.
* Bruk et mellomrom før denne første parentesen.
* Sluttparentesen legges på en ny linje uten mellomrom foran.
* Avslutt med semikolon etter sluttparentesen

const person = {

    fornavn: "Ola",

    etternavn: "Nordmann",

    alder: 50,

    øyefarge: "blå"

};

Funksjoner, løkker og valgsetninger følger samme regler, men avsluttes ikke med semikolon etter sluttparentesten.

### Funksjoner

function tilCelsius(fahrenheit) {

    return (5 / 9) \* (fahrenheit - 32);

}

### Løkker

for (let i = 0; i < 5; i++) {

    x += i;

}

### Valgsetninger

if (tid < 20) {

    hilsen = "Good day";

} else {

    hilsen = "Good evening";

}

## Regler for objekter

Objekter i JavaScript følger mange av de samme reglene for sammensatte uttrykk, men har noen ekstra. Følgende regler gjelder:

* Åpningsparentesen legges på slutten av samme linje som objektnavnet
* Bruk kolon og ett mellomrom mellom hver egenskap og dens verdi
* Bruk anførselstegn rundt verdier som er ment å være strenger, men ikke rundt verdier som er ment å være tall.
* Ikke ha komma etter det siste egenskap-verdi paret
* Sluttparentesen legges på en ny linje uten mellomrom foran
* Avslutt alle objektdefinisjoner med et semikolon

let person = {

    fornavn: "Ola",

    etternavn: "Nordmann",

    alder: 50,

    øyefarge: "blå"

};

Hvis objektet ikke har for mange egenskaper, kan det skrives på én linje.

let person = {fornavn: "Ola", etternavn: "Nordmann"}

## Linjelengde < 80

Unngå linjer som overstiger 80 tegn (mellomrom inkludert). Hvis en linje overstiger 80 tegn er det vanlig å dele opp linjen etter et komma eller en operator.

let beskjed = "Hei! Jeg ruller en sekssidet terning på denne måten: " +

(Math.round(6 \* Math.random() + 1));

## Deklarasjon av variabler

Det er god praksis å deklarere og initialisere (gi en verdi til) variabler helt på begynnelsen av koden. Det gjør at man får mer oversikt og man unngår uønsket overskriving av variabler. Hvis vi også initialiserer variablene når dem deklareres unngår man «undefined» verdier.

let fornavn = "Ola";

let etternavn = "Nordmann";

let pris = 19.90;

let moms = 0.20;

let fullPris = pris + (pris \* moms);

## Standardverdier

I mange tilfeller vil vi deklarere og initialisere en variabel, men vi har ikke noen spesiell verdi vi ønsker å tilordne variabelen ennå. F.eks vi ønsker å opprette et tomt array som vi senere skal fylle opp. Her er standardverdier for de ulike datatypene:

let streng = "";                // String

let tall = 0;                   // Number

let boolsk = false;             // Boolean

const objekt = {};              // Object

const array = [];               // Array

const funksjon = function(){};  // Function

## Logiske utsagn

Når man skriver logiske utsagn for eksempel i if-tester eller while-løkker er det foretrukket å bruk «===» i stedet for «==» når man tester om to verdier er like. «==» sjekker om to verdier er like, men ignorerer type.

1 == "1"; // Dette uttrykket er sant

«===» sjekker om to verdier er like både på verdi og type.

1 === "1"; // Dette uttrykket er usant.