SYSTEK

HØYNE DIN KODEKVALITET MED STATISK TYPING OG HISSIG LINTING

KODEKVALITET - HVEM BRYR SEG EGENTLIG?

JAVASCRIPT

- Skapt for å validere forms
- Event-basert
- Dynamisk typet
- Utviklet til å støtte moderne behov: ES5, ES6, ES2016, ES2017...
- jQuery, Lodash, Vue, Ember, Angular, React, Elm


SELIS
en del av kretsløpet
08899

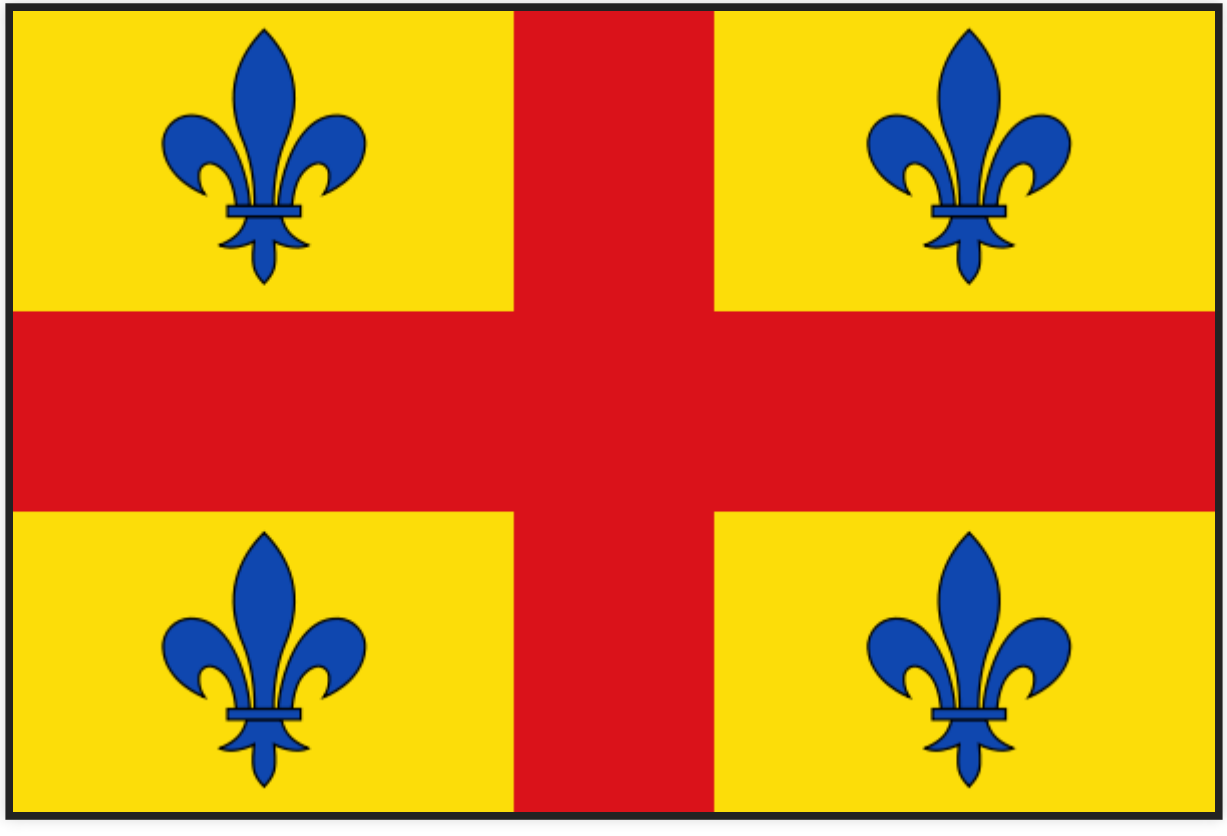
ÅRETS JULEGAVE TIL DINE KOLLEGER

ÅRETS JULEGAVE TIL DEG SELV

LESBAR, FORSTÅELIG OG FORVALTBAR KODE

LINT - HVA ER DET?

En kommune i den belgiske provinsen Antwerpen.



Oxford Dictionary

Short, fine fibres which separate from the surface of cloth or yarn during processing. ‘some fabrics leave tiny specks of lint on the glass’

LO

Statisk analyse av kildekode for å detektere brudd på definerte regler

LINTING I JAVASCRIPT

- **JSLint** (2002) Douglas Crockford. Sjekker om JS-kode gitte koderegler
- **JSHint** (2010) Anton Kovalyov. Fork av JSLint for bedre regeltilpassing
- **ESLint** (2013) Nicholas C. Zakas. Utviklere skal kunne sette opp eget regelsett
- **TSLint** (2015) Palantir Technologies. Linting for TypeScript

BLI KVITT RUSKET I DIN KODE

```
/git/myRepo $ npm install -g eslint
/git/myRepo $ eslint --init
/git/myRepo $ eslint test.js test2.js
```

--init generer *.eslintrc.js*-fil for deg

```
/git/myRepo $ node node_modules/.bin/eslint --init
? How would you like to configure ESLint? Use a popular style
? Which style guide do you want to follow? (Use arrow keys)
❯ Google
  Airbnb
  Standard
```

Eller lage din egen fra scratch

```
module.exports = {
    "env": {
        "browser": true
    },
    "extends": "eslint:recommended",
    "rules": {
        "linebreak-style": [
            "error",
            "unix"
        ],
        "quotes": [
            "error",
            "double"
        ],
        ....
```

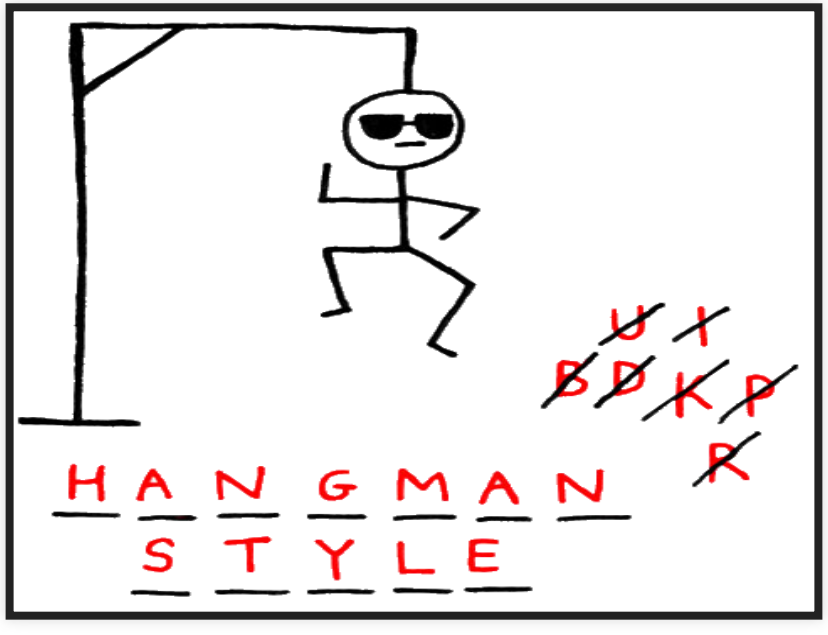
STRATEGI

```
{ "extends": "airbnb-base" }
```

eller

```
{
"extends": "eslint:recommended",
"rules": [.....]
}
```

+ prettier


U I
B D K P
R
HANGMAN
STYLE

JAVASCRIPT / ECMASCRIPT

Syntaks - Sukker - Sirup

- CoffeScript
- LiveScript
- NodeScript (!= Node.JS)
- LispyScript
- ClojureJS/ClojureScript
- Elm
- TypeScript
- Flow

STATISK VS DYNAMISK

- Statisk typede språk: variabeltyper sjekkes compile-time
- Dynamsisk typede språk: sjekkes først runtime
- ▪ større frihet til å mikse typer
- ▪ kræsjer hvis det ikke tas høyde for typekonflikter

FORDELER MED STATISK TYPEDE SPRÅK

- Autofullfør & søk i koden - IDEen leder til funksjonsdefinisjon
- Økt selvdokumentasjon
- Reduserer uforutsette feil - eks typos, manglende parametere og brukerinputtvalidering
- Dokumenterer kontrakten med backend. Spesifiserer domeneobjekter

Statisk typesjekker fra FaceBook

Flow employs the kind of data-flow and control-flow analysis that compilers typically perform to extract semantic information from code. It then uses this information for type inference building on advanced techniques in type theory.

F Code

TYPE UTLEDNING

Using data flow analysis, Flow infers types and tracks data as it moves through your code. You don't need to fully annotate your code before Flow can start to find bugs.

```
// @flow
function concat(a, b) {
  return a + b;
}

concat("A", "B");
concat(1, 2);
```

TYPE ANNOTASJON

```
// @flow
function concat(a: string, b: string) {
  return a + b;
}

concat("A", "B"); // Works!
concat(1, 2); // Error!
```

```
/git/myRepo $  node_modules/.bin/flow
Error: src/error.js:7
  7: concat(1, 2); // Error!
            ^ number. This type is incompatible with the
expected param type of
  2: function concat(a: string, b: string) {
                        ^^^^^^ string
...
```

TYPER

Rikt utvalg av typer og språklige features :

- Primitive typer
- Komplekse typer, klasser, arv
- Enum
- Generics

...

NON-NULLABLE

```
// @flow
function foo(num: number): string {
    if (num > 10) {
        return 'cool';
    }
}
// error: return undefined. This type is
// incompatible with string
```

SUNNHET

```
// @flow

function foo(x: ?number): string {
  if (x) {
    return x;
  }
  return "default string";
}
```

```
/git/myRepo $ node_modules/.bin/flow
Error: src/error.js:5
  5:     return x;
                ^ number. This type is incompatible with the
expected return type of
  3: function foo(x: ?number): string {
                               ^^^^^^ string
```

BABELIFY

```
/git/myRepo $ yarn add --dev babel-cli babel-preset-flow
```

.babelrc

```
{
  "presets": ["flow"]
}
```

OG KJØR!

```
/git/myRepo $ yarn run babel src/
/git/myRepo $ yarn add --dev flow-bin
/git/myRepo $ yarn run flow init
/git/myRepo $ yarn run flow status
```

.flowconfig

- linting for Flow
- use_strict
- suppress_type='anyother' // tillat any-type

Stram til løkka!

Typescript is a typed superset of JavaScript that compiles to plain JavaScript

typescript.org

Utviklet av Microsoft

UTLEDING & ANNOTASJONER

```
let a: string = 'Hello';
let b: number = 22;
let c = false;
let d = c + b; // Operator '+' cannot be applied to types 'num
```

TYPER

- Rikt utvalg av typer og features

  men

- Typer er nullable (må nullsjekkes)

KOMME IGANG

```
/git/myRepo $ npm install typescript
/git/myRepo $./node_modules/.bin/tsc hellovoid.ts
```

hellovoid.ts

```
function greeter(person: string) {
    return "Hello, " + person;
}

let user = [0, 1, 2];

document.body.innerHTML = greeter(user);
```

```
/git/myRepo $  node_modules/.bin/tsc hellovoid.ts
src/test.ts(7,35): error TS2345:
Argument of type 'number[]' is not assignable
to parameter of type 'string'.
```

.tsconfig.json

```
{
    "compilerOptions": {
        "module": "system",
        "noImplicitAny": false,
        "removeComments": true,
        "preserveConstEnums": true,
        "outFile": "../../built/local/tsc.js",
        "sourceMap": true
    },
    "include": ["src/**/*"],
    "exclude": ["node_modules"]
```

STRAM TIL LØKKA

- strictNullChecks
- noImplicitAny
- allowJs
- alwaysStrict

--strict

Compiler Options

FLOW VS TYPESCRIPT

- Får kontroll over typer og struktur i kode
- Lett å opte inn i eksisterende kode
- TypeScript and Flow har sett mye til hverandre
- Begge suporterer React, Angular mv
- TypeScript er en kompilator, Flow er en sjekker
- Flow har bedre sunnhetsforståelse (unngå run-time-feil)
- TypeScript har bedre tooling
- Flow trenger i liten grad å skrive om koden

ANDRE STATISK TYPEDE SPRÅK

Som transpilerer til JavaScript

Elm - Dart - TeJaSasm.js - JavaScript++ - MascaraRoy - Swym - Typecast.js - PureScript - ActionScript - BuckleScript

BLIR KODEN BEDRE?

- Lesbarhet - oversikt
- Syntaks-feil
- Sunnhet
- Garantere for data
- Dokumentasjon
- Skiller data og logikk

TAKK FOR OPPMERKSOMHETEN

- https://flow.org/en/
- https://www.typescriptlang.org/index.html
- https://djcordhose.github.io/flow-vs-typescript/2016_hhjs.html
- https://eslint.org/