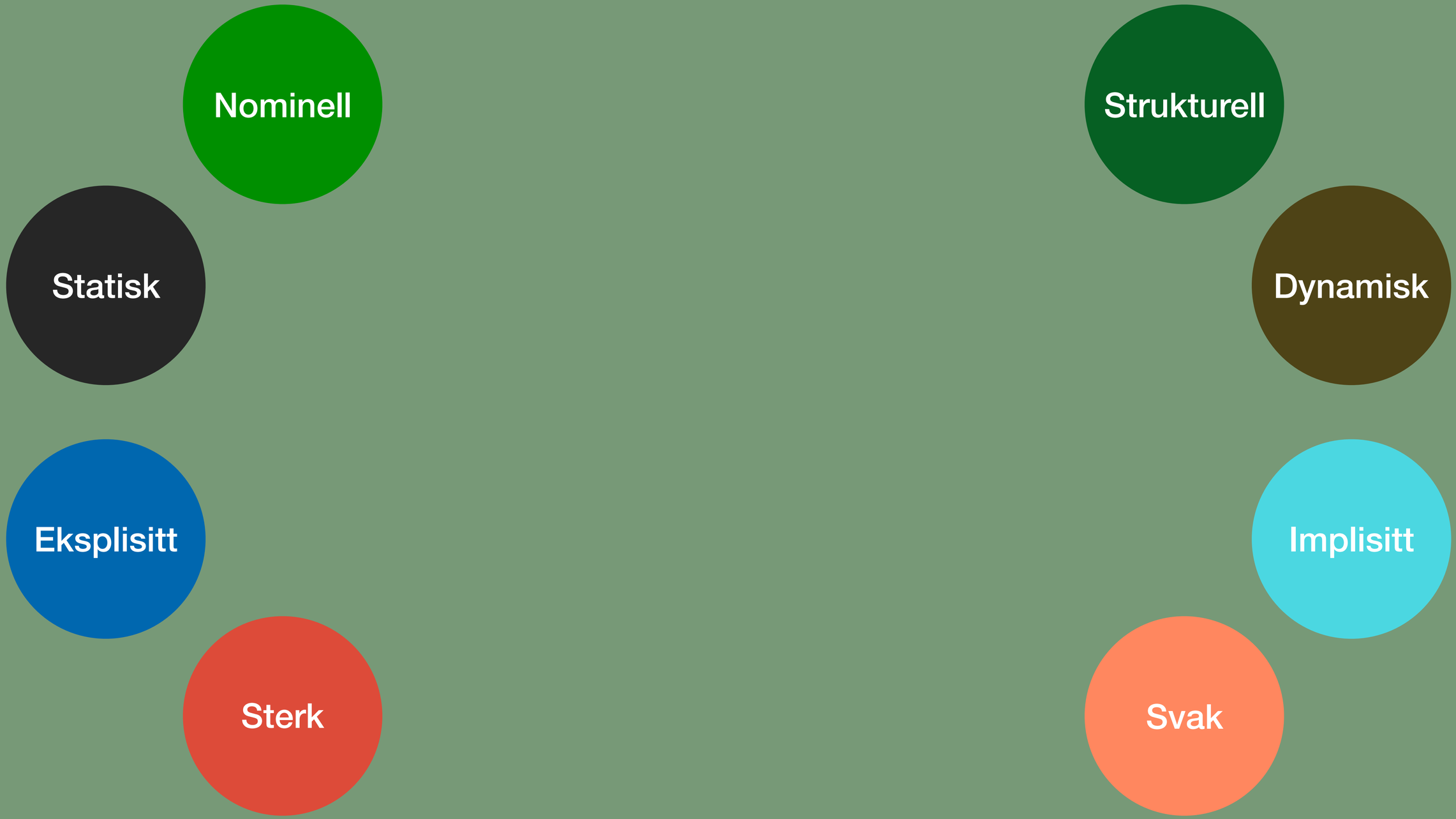
Typesystemer, OCaml og ReasonML

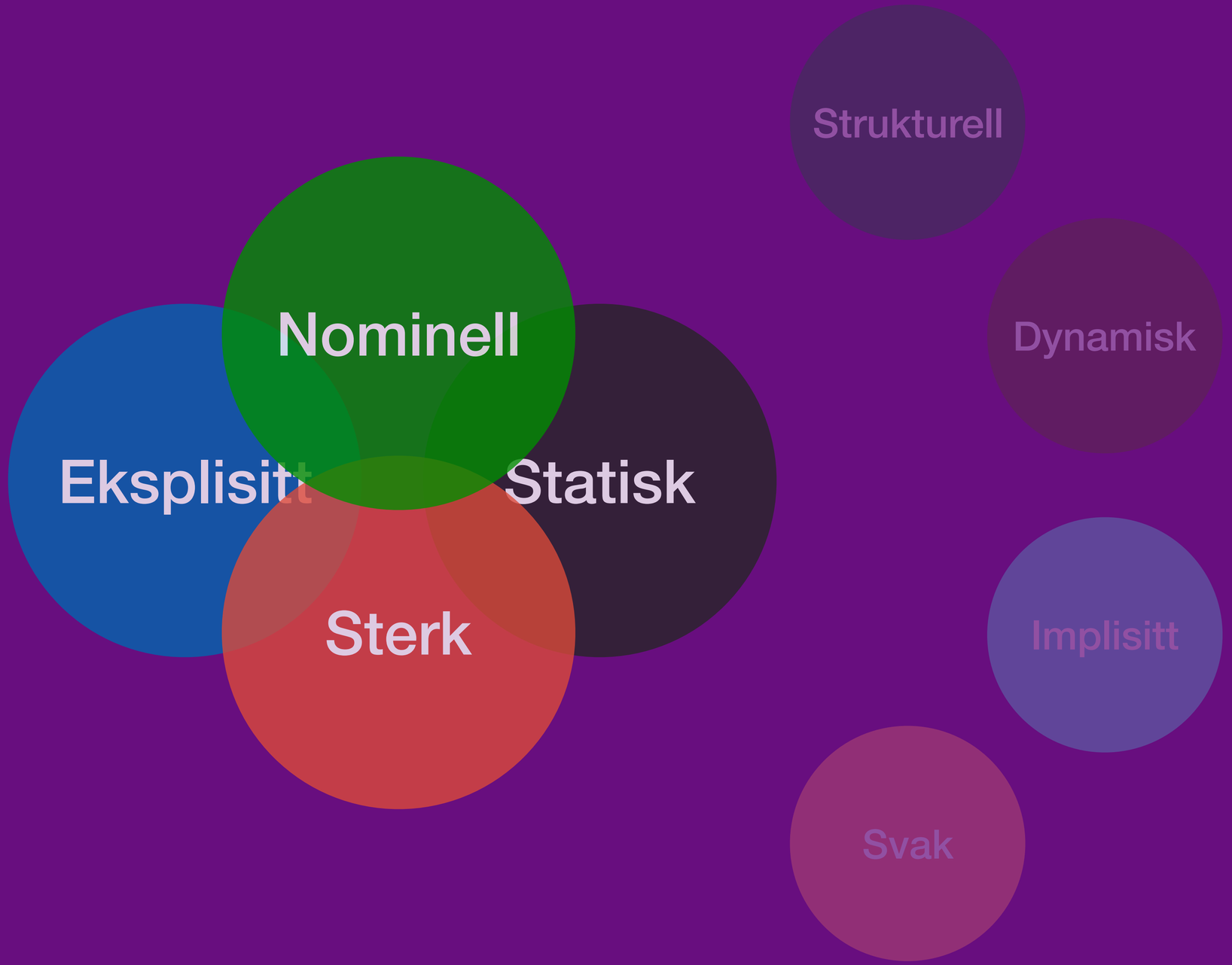
bartjs – april 2018

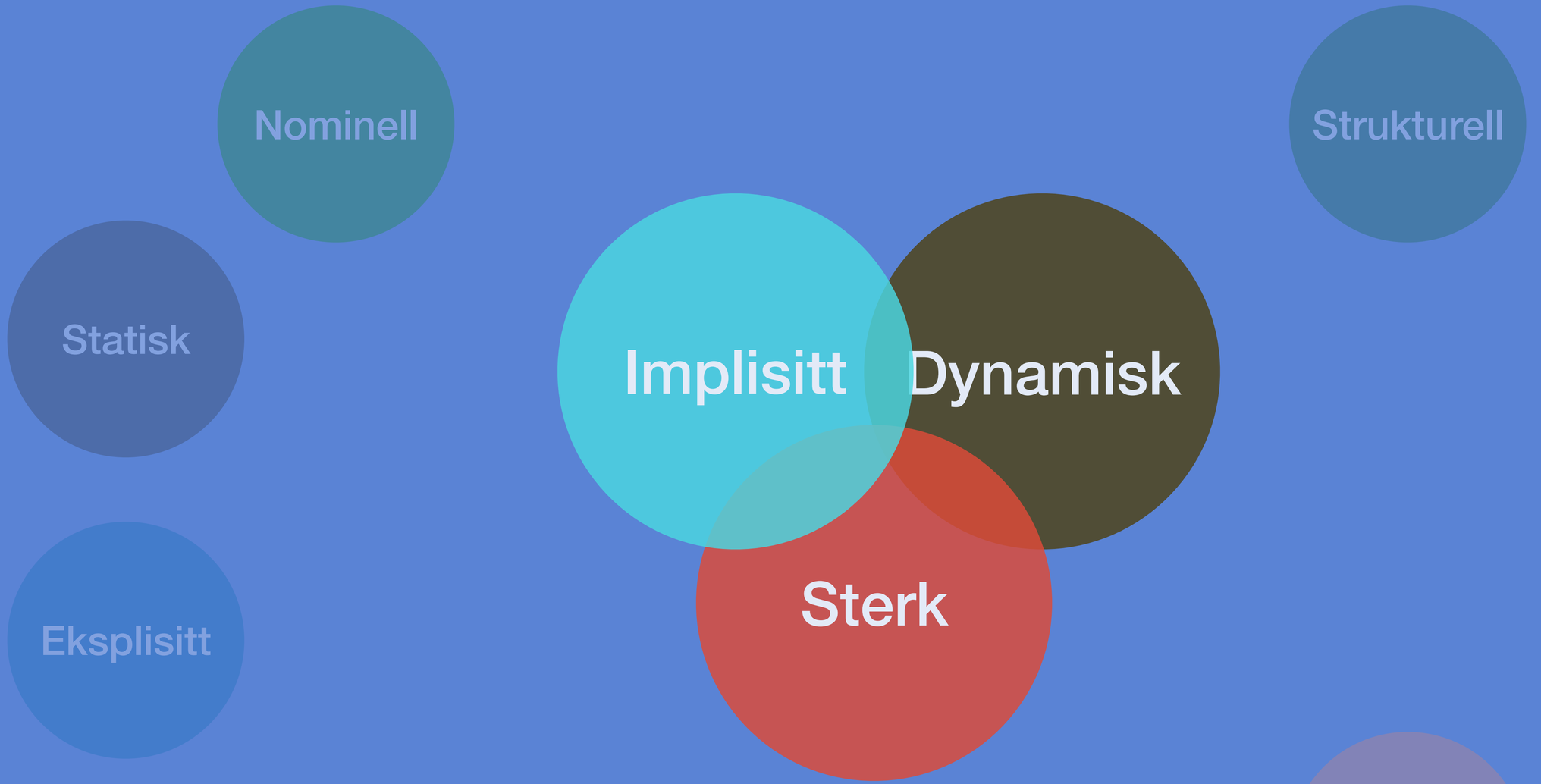
@mikaelbrevik / @kodesnutt

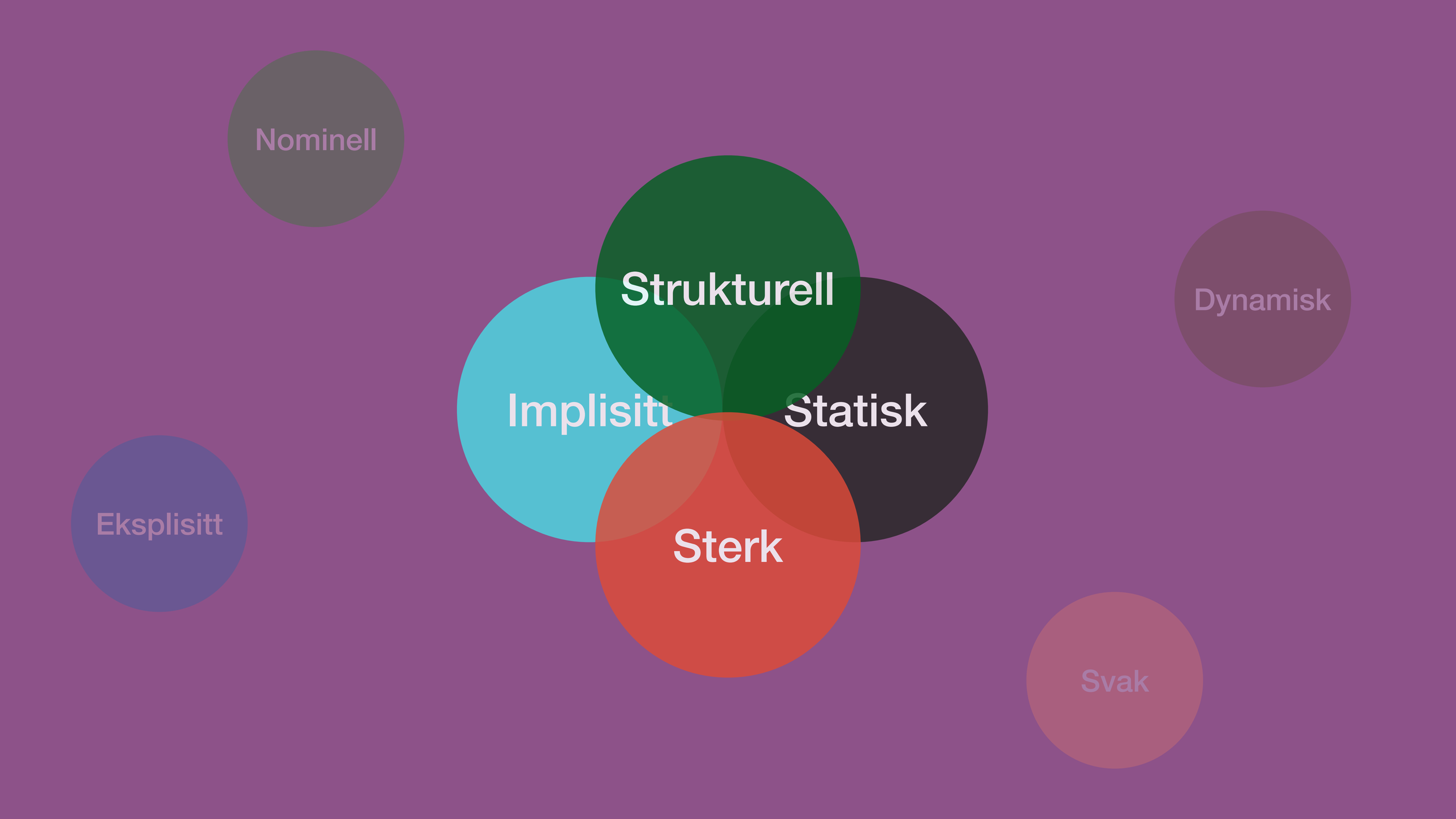
Typer

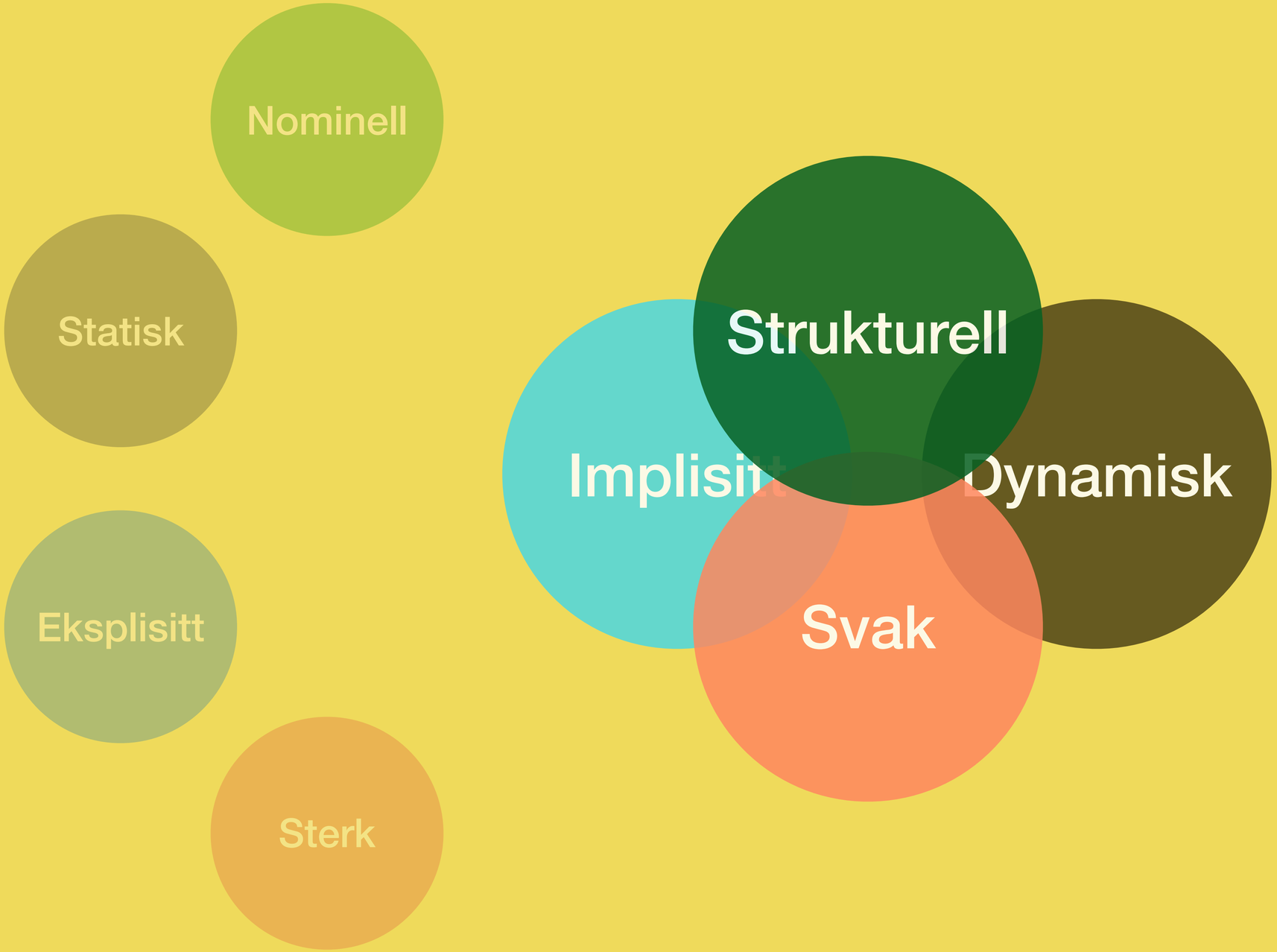
JS

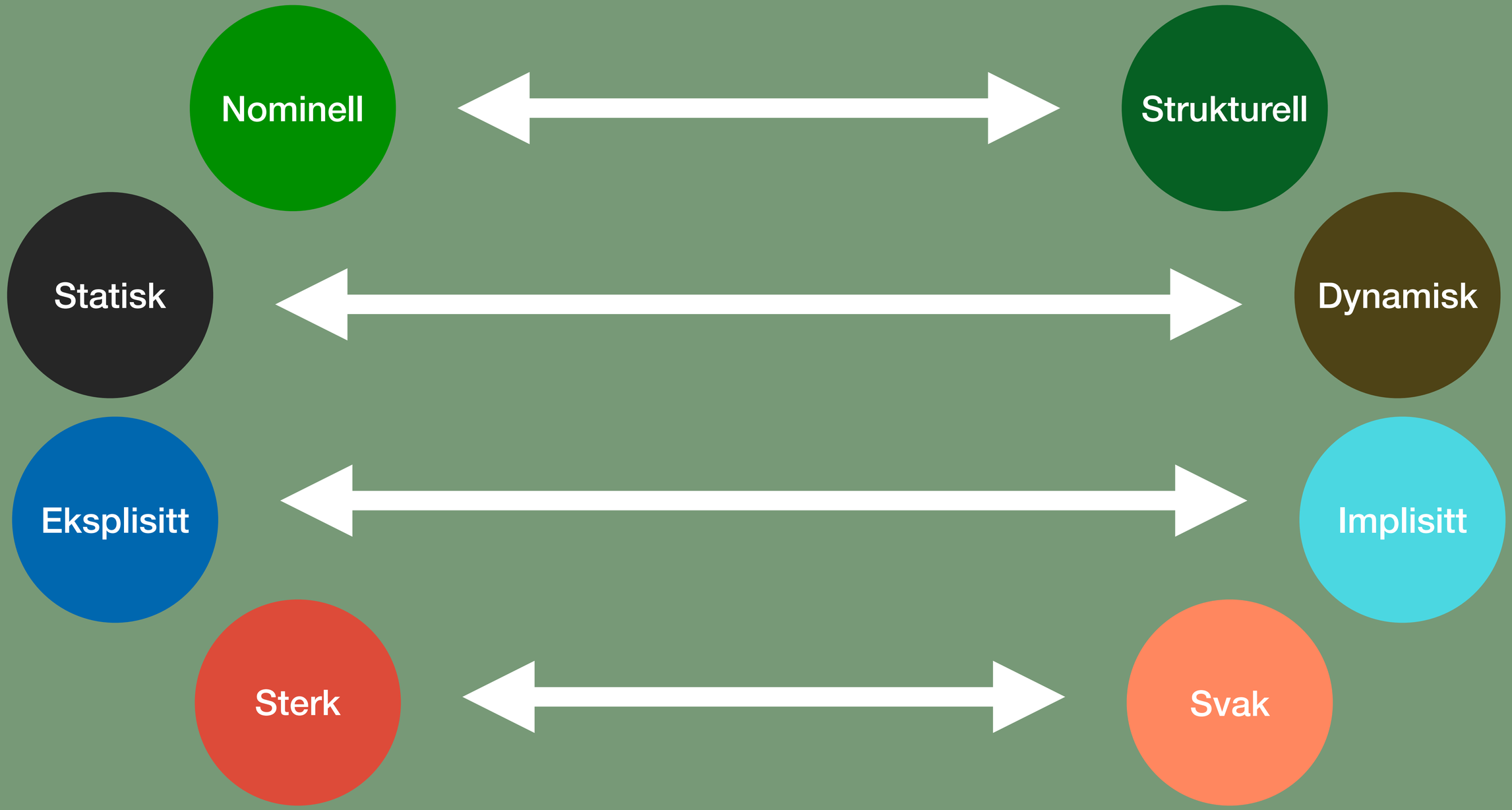

Nominell
Statisk
Eksplisitt
Sterk
Strukturell
Dynamisk
Implisitt
Svak


Nominell
Eksplisitt
Statisk
Sterk
Strukturell
Dynamisk
Implisitt
Svak


Nominell
Strukturell
Statisk
Implisitt
Dynamisk
Eksplisitt
Sterk
Svak


Nominell
Strukturell
Dynamisk
Implisitt
Statisk
Eksplisitt
Sterk
Svak


Nominell
Statisk
Eksplisitt
Sterk
Strukturell
Implisitt
Dynamisk
Svak


Nominell
Strukturell
Statisk
Dynamisk
Eksplisitt
Implisitt
Sterk
Svak


10 lbs
KRUPS
MADE IN THE IRISH REPUBLIC


Ulike avveininger

Noe for alt

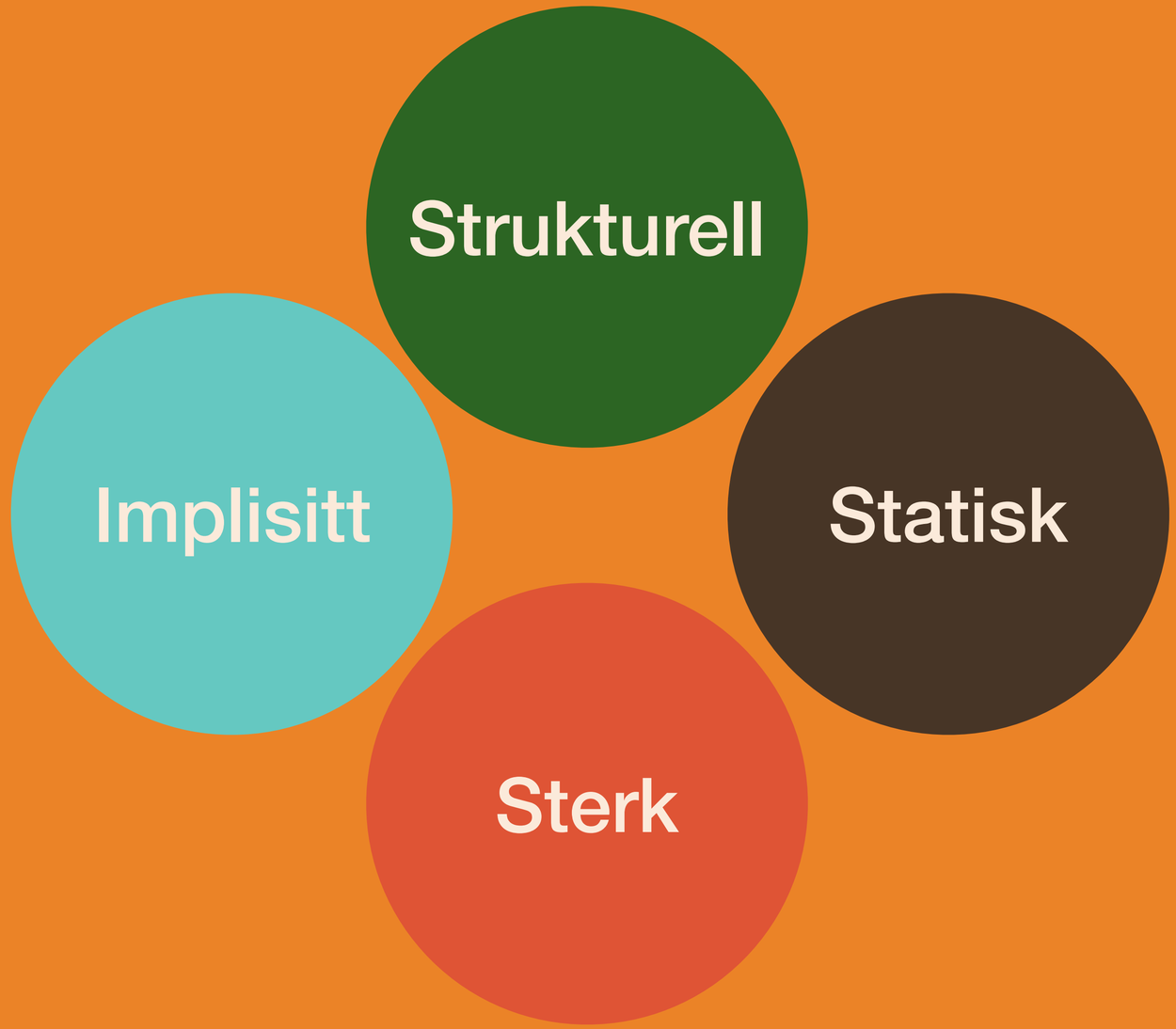
JS


JS
JS
JS
JSCONF.EU
JS
2015
WEB REBEL
WEB REBEL

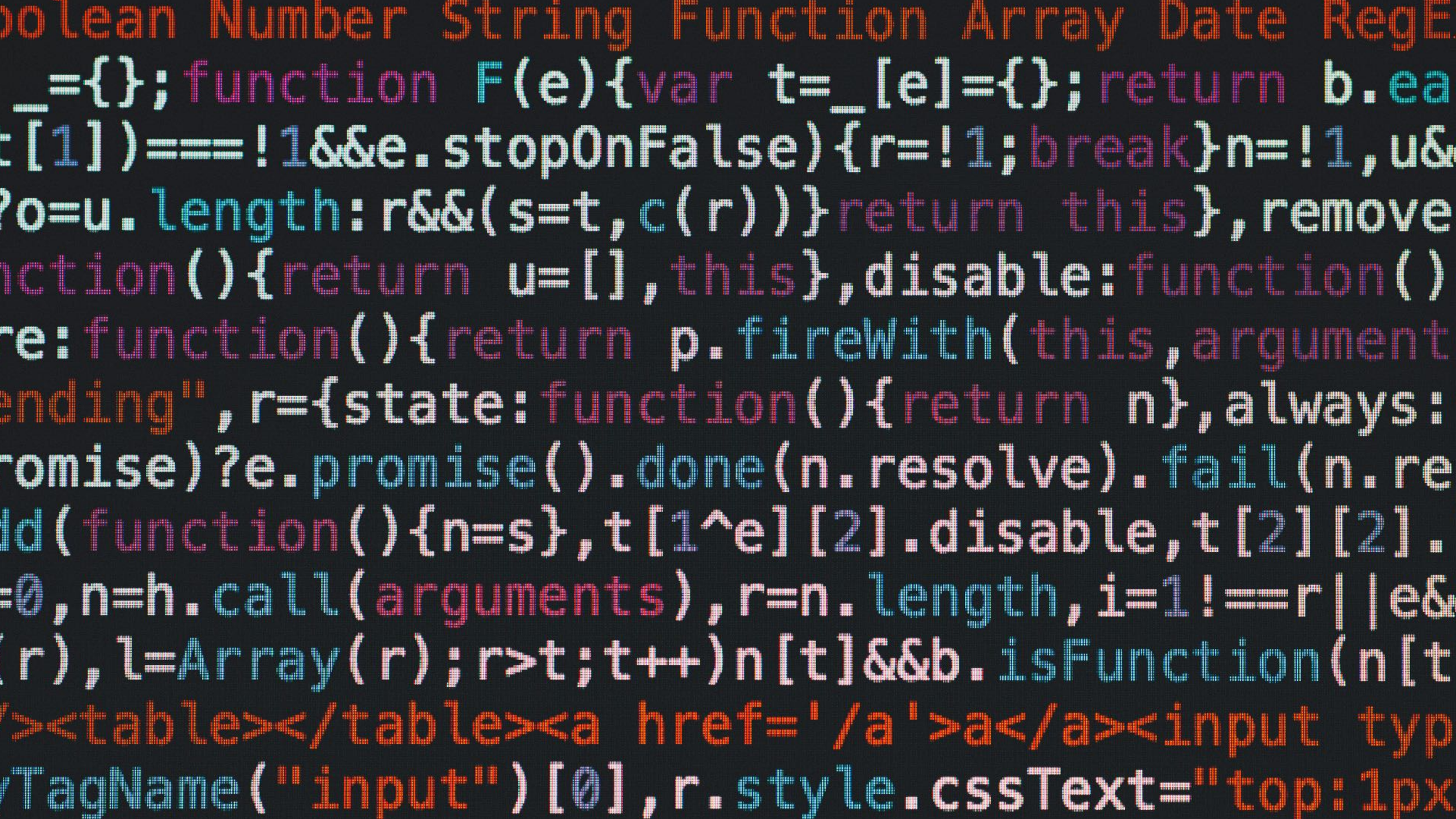
Typer

Typesystemer, OCaml og ReasonML

OCaml


Strukturell
Implisitt
Statisk
Sterk

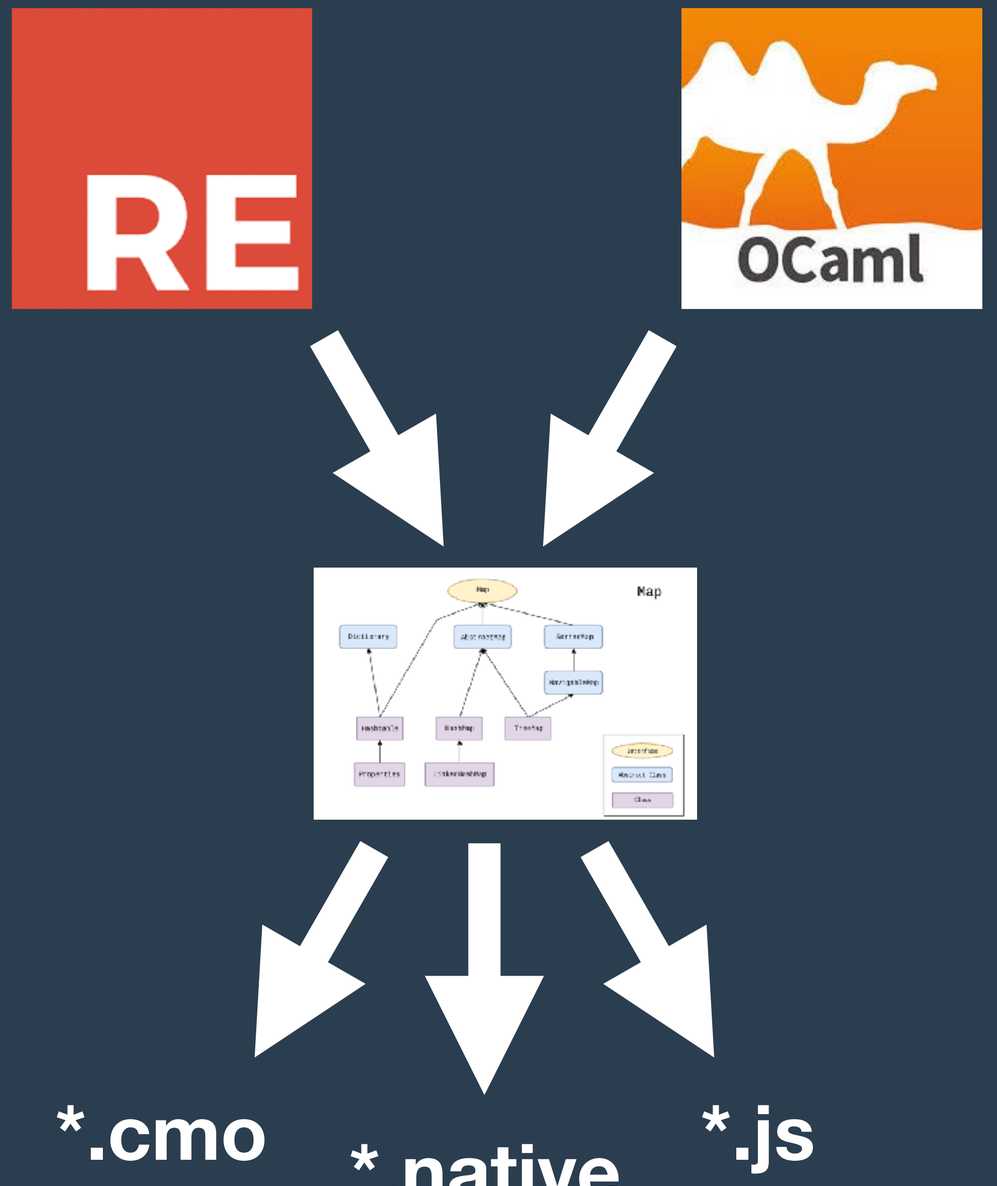

GLOBE TERRESTRE
APARIS
Quay de l'Horloge
Tropique
du
Cancer
MER DU SUD
Apaches


oolean Number String Function Array Date RegE
={};function F(e){var t=[e]={};return b.ea
t[1])===!1&&e.stopOnFalse){r=!1;break}n=!1,u&
?o=u.length:r&&(s=t,c(r))}return this},remove
nction(){return u=[],this},disable:function()
re:function(){return p.fireWith(this,argument
ending",r={state:function(){return n},always:
romise)?e.promise().done(n.resolve).fail(n.re
dd(function(){n=s},t[1^e][2].disable,t[2][2].
=0,n=h.call(arguments),r=n.length,i=1!==r||e&
(r),l=Array(r);r>t;t++)n[t]&&b.isFunction(n[t
/><table></table>a<input typ
yTagName("input")[0],r.style.cssText="top:1px

Syntax

AST

Code gen



OCaml

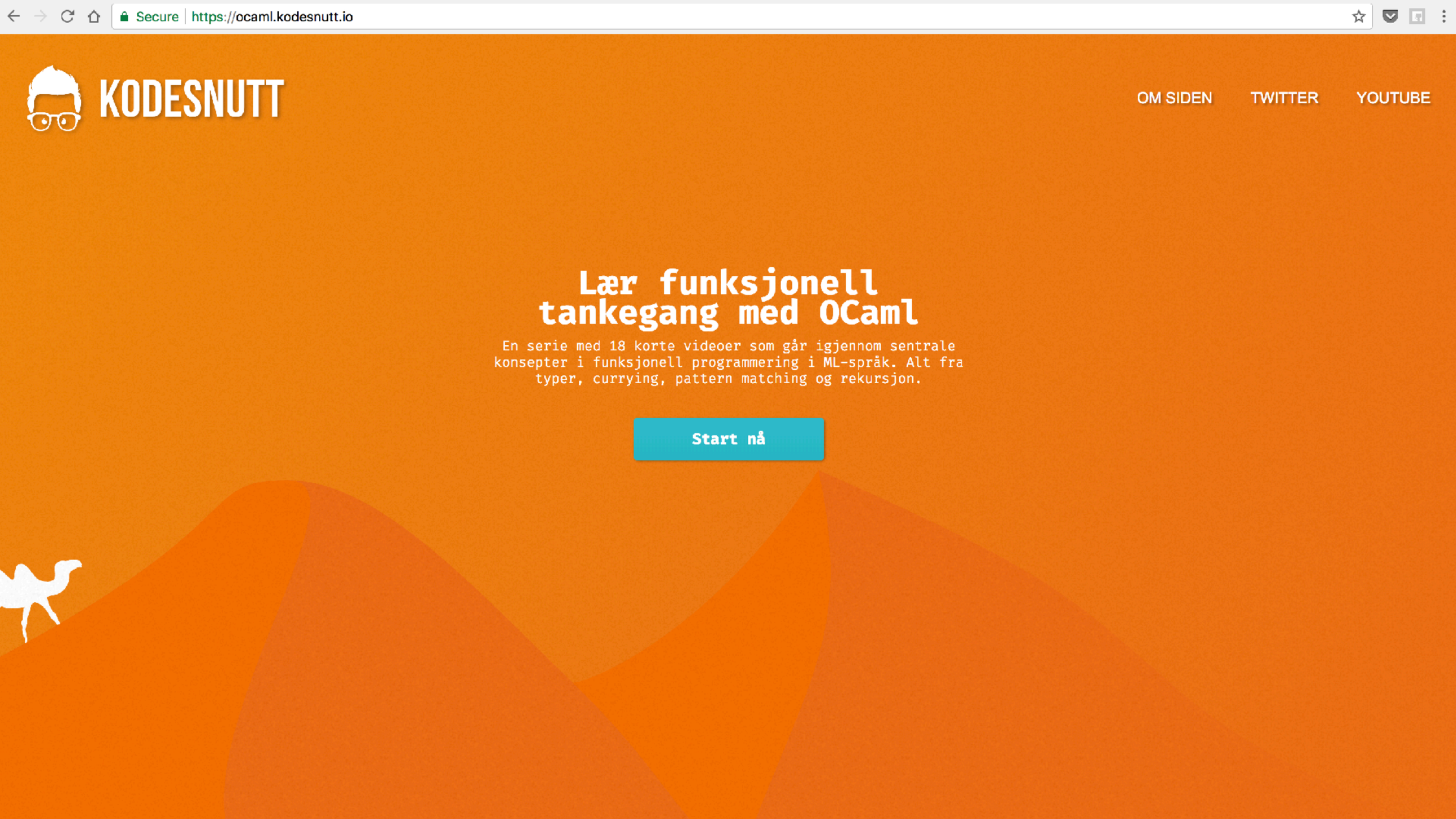
Typesystemer, OCaml og ReasonML


Pause?

Typesystemer, OCaml og ReasonML

ReasonML


McDonald's


Secure
https://ocaml.kodesnutt.io
KODESNUTT
OM SIDEN
TWITTER
YOUTUBE
Lær funksjonell tankegang med OCaml
En serie med 18 korte videoer som går igjennom sentrale konsepter i funksjonell programmering i ML-språk. Alt fra typer, currying, pattern matching og rekursjon.
Start nå

RE

```
type webData('a) =
  | NotAsked
  | Pending
  | Success('a)
  | Error(string);

type action =
  | Loading
  | SetPodcasts(list(Podcast.t))
  | Failure;

type state = {podcasts: webData(list(Podcast.t))};

let component = ReasonReact.reducerComponent("Page");

let str = ReasonReact.stringToElement;

let make = (~podcasts=[], _children) => {
```

Primitiver

```
1   let a: string = "Hello, World!";
2
3   let b: int = "Hello, World!";
4
5   let c: int = 42;
6
7   let d: float = 42.0;
```

```
We've found a bug for you!
/Users/mikaelbrevik/Dropbox/Presentasjoner/BartJS - april 201
-28

1 | let a: string = "Hello, World!";
2 |
3 | let b: int = "Hello, World!";
4 |
5 | let c: int = 42;

This has type:
  string
But somewhere wanted:
  int
```

Ikke statements

```
let a = 1;

let b = 2;

print_int(a + b);
/* - : int = 3 */

/* Same identifiers */
let a = 2;

let b = 3;

let a = 3;

print_int(a + b);
/* - : int = 6 */
```

Egene typer

```
1  type myNumber = int;
2  type myFloat = float;
3  type customTuple = (myFloat, myNumber);
4  type customTuple2 = (myFloat, myNumber);
5
6  let foo: customTuple = (4.2, 4);
7  let bar: customTuple2 = foo;
8
9  let foo: customTuple = (4.2, 4.2);
```

Egene typer

```
1  type myNumber = int;
2  type myFloat = float;
3  type customTuple = (myFloat, myNumber);
4  type customTuple2 = (myFloat, myNumber);
5
6  let foo: customTuple = (4.2, 4);
7  let bar: customTuple2 = foo;
8
9  let foo: customTuple = (4.2, 4.2);
```

Strukturelt

```
1  type myNumber = int;
2  type myFloat = float;
3  type customTuple = (myFloat, myNumber);
4  type customTuple2 = (myFloat, myNumber);
5
6  let foo: customTuple = (4.2, 4);
7  let bar: customTuple2 = foo;
8
9  let foo: customTuple = (4.2, 4.2);
```

Records

```
1  type movie = {
2    title: string,
3    year: int,
4  };
5
6  let infinityWar = {title: "Avengers Infinity War", year: 2017};
7
8  let infinityWar = { ...infinityWar, year: 2018};
9  /* - : movie = {title: "Avengers Infinity War", year: 2018} */
```

Records

```
1  type movie = {
2    title: string,
3    year: int,
4  };
5
6  let infinityWar = {title: "Avengers Infinity War", year: 2017};
7
8  let infinityWar = { ...infinityWar, year: 2018};
9  /* - : movie = {title: "Avengers Infinity War", year: 2018} */
```

Funksjoner & Curry

```
1   let add3 = (a: int, b: int, c: int) ⇒ a + b + c;
2   /* - : (int, int, int) ⇒ int = <fun> */
3
4   add3(3, 4, 5);
5   /* - : int = 12 */
6
7   let add3And4 = add3(3, 4);
8   /* - : int ⇒ int = <fun> */
9
10  add3And4(5);
11  /* - : int = 12 */
```

Funksjoner & Curry

```
1   let add3 = (a: int, b: int, c: int) ⇒ a + b + c;
2   /* - : (int, int, int) ⇒ int = <fun> */
3
4   add3(3, 4, 5);
5   /* - : int = 12 */
6
7   let add3And4 = add3(3, 4);
8   /* - : int ⇒ int = <fun> */
9
10  add3And4(5);
11  /* - : int = 12 */
```

Typeinferens

```
1  let a = 1;
2  let b = 2.5;
3  let myTuple = (a, b);
4
5  let add3 = (a, b, c) ⟹ a + b + c;
6  add3(3, 4, 5);
7  /* - : int = 12 */
```

Infiks

```
1  let (>>) = (f, g, a) => g(f(a));
2
3  let double = ( * )(2);
4
5  let doubleDouble = double >> double;
6
7  4 |> doubleDouble;
8  /* - : int = 16 */
```

Infiks

```
1  let (>>) = (f, g, a) => g(f(a));
2
3  let double = ( * )(2);
4
5  let doubleDouble = double >> double;
6
7  4 |> doubleDouble;
8  /* - : int = 16 */
```

Infiks

```
1  let (>>) = (f, g, a) => g(f(a));
2
3  let double = ( * )(2);
4
5  let doubleDouble = double >> double;
6
7  4 |> doubleDouble;
8  /* - : int = 16 */
```

Infiks

```
1  let (>>) = (f, g, a) => g(f(a));
2
3  let double = ( * )(2);
4
5  let doubleDouble = double >> double;
6
7  4 |> doubleDouble;
8  /* - : int = 16 */
```

Algebraiske datatyper

```
1  type genre =
2    | Documentary
3    | Comedy
4    | Action;
5
6  let infinityGenres = [Documentary, Action];
7  /* - : list(genre) = [Documentary, Action] */
```

Algebraiske datatyper

```
1   type power = string;
2   type name = string;
3   type nemesis = person
4   and person =
5     | Superhuman(name, power, nemesis)
6     | NormalUnknown
7     | Villain(name);
8
9   let drOctopus: nemesis = Villain("Doctor Octopus");
10
11  let spiderman: person = Superhuman("Spiderman", "Agile", drOctopus);
```

Lister

```
1    type hero = string;
2    type heroList = list(hero);
3
4    let heroes: heroList = ["Dr. Strange", "Bruce Banner"];
```

Lister

```
let heroes = ["Shuri", "T'Challa"];

heroes[0];

List.hd(heroes);
/* - : string = "Shuri" */

List.nth(heroes, 1);
/* - : string = "T'Challa" */
```

Lister

```
1  type liste('a) =
2    | []
3    | ::('a, liste('a));
4
5  [1, 2, 3];
6  /* liste(int) = ::(1, ::(2, ::(3, []))) */
```

Option

```
1  type option('a) =
2    | Some('a)
3    | None;
4
5  let hasValue: option(int) = Some(0);
6
7  let doesntHaveValue: option(int) = None;
```

Pattern matching

```
1  let maybeValue41 = Some(41);
2
3  let valuePlus1Or0 = switch (maybeValue41) {
4  | Some(x) ⟹ x + 1
5  | None ⟹ 0
6  };
7  /* - : int = 42 */
```

Pattern matching

```
let maybeValue41 = Some(41);

let valuePlus1Or0 = switch (maybeValue41) {
| Some(x) ⟹ x + 1
| None ⟹ 0
};
/* - : int = 42 */
```

Pattern matching

```
let favouriteNumbers = [1, 2, 3];

let mySecondFavouriteNumber = numbers ⇒
  switch (numbers) {
  | [_, x, _] ⇒ Some(x)
  | _ ⇒ None
  };

mySecondFavouriteNumber(favouriteNumbers);
/* - : option(int) = Some(2) */

mySecondFavouriteNumber([1, 2]);
/* - : option(int) = None */
```

Pattern matching

```
let favouriteNumbers = [1, 2, 3];

let mySecondFavouriteNumber = numbers ⇒
  switch (numbers) {
  | [_, x, _] ⇒ Some(x)
  | _ ⇒ None
  };

mySecondFavouriteNumber(favouriteNumbers);
/* - : option(int) = Some(2) */

mySecondFavouriteNumber([1, 2]);
/* - : option(int) = None */
```

Pattern matching

```
1  let favouriteNumbers = [1, 2, 3];
2
3  let mySecondFavouriteNumber = numbers ⇒
4    switch (numbers) {
5    | [_, x, _] ⇒ Some(x)
6    | _ ⇒ None
7    };
8
9  mySecondFavouriteNumber(favouriteNumbers);
10 /* - : option(int) = Some(2) */
11
12 mySecondFavouriteNumber([1, 2]);
13 /* - : option(int) = None */
```

Pattern matching

```
9   let drOctopus: nemesis = Villain("Doctor Octopus");
10  let spiderman: person = Superhuman("Spiderman", "Agile", drOctopus);
11
12  let heroOrVillain =
13    fun
14    | Superhuman(name, _, _) ⇒ "Hero " ++ name
15    | Villain(name) ⇒ "Villain " ++ name
16    | NormalUnknown ⇒ "NA";
17
18  heroOrVillain(spiderman);
19  /* - : name = "Hero Spiderman" */
```

Polymorfisme

```
1  let four = 2 + 2;
2
3  let four = 2. + 2.;
4
5  let four = 2. +. 2.;
6
7  let two = 4. /. 2.;
```

Polymorfisme

```
1  let four = 2 + 2;
2
3  let four = 2. + 2.;
4
5  let four = 2. +. 2.;
6
7  let two = 4. /. 2.;
```

```
1  let fst = ((a: 'a, _)) => a;
2
3  let tuple = (42, 5);
4
5  fst(tuple);
6  /* - : int = 42 */
```

```reason
let fst = ((a: 'a, _)) => a;

let tuple = (42, 5);

fst(tuple);
/* - : int = 42 */
```

```
1  type option('a) =
2    | Some('a)
3    | None;
4
5  let hasValue: option(int) = Some(0);
6
7  let doesntHaveValue: option(int) = None;
```


BUCKLESCRIPT

```
$ bsb -init project-name -theme basic-reason

$ cd project-name
$ npm run build
$ node src/demo.bs.js
```

```
let myMap = Js.Dict.empty();

Js.Dict.set(myMap, "Allison", 10);
```

```
type movie = {
  .
  "title": string,
  "year": int,
  [@bs.set] "genre": string,
  [@bs.meth] "getPoster": unit ⟹ string,
};

[@bs.val] external infinityWar : movie = "infinityWar";

let year = infinityWar##year;
infinityWar##genre#="Action";
infinityWar##getPoster();
```

Externals

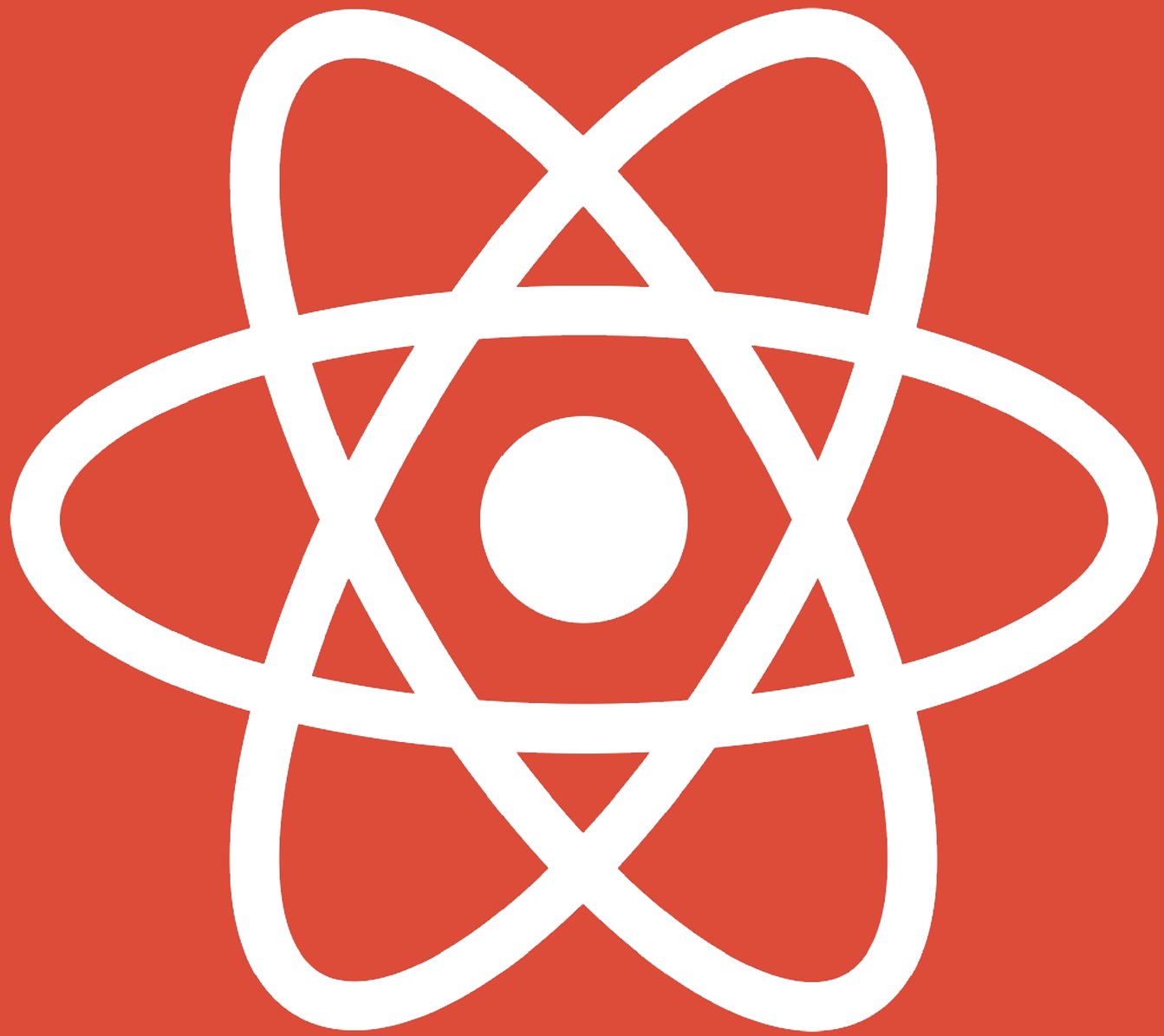
```
type element;

let window: element = [%bs.raw "window"];

[@bs.send]
external addEventListener : (element, string, unit ⇒ unit) ⇒ unit =
  "addEventListener";

[@bs.val]
external requestAnimationFrame : (unit ⇒ unit) ⇒ unit =
  "requestAnimationFrame";
```

```
1  open Js.Promise;
2
3  let myPromise = resolve(2);
4
5  myPromise
6   |> then_(value => resolve(value + 2))
7   |> catch(_err => resolve(-2))
8   |> then_(v => resolve(Js.log(v)))
9   |> ignore;
```



```
1    <button> (ReasonReact.stringToElement("Awesome!")) </button>;
```

```
1   <button> (ReasonReact.stringToElement("Awesome!")) </button>;
```

```
1   // Generated by BUCKLESCRIPT VERSION 3.0.0, PLEASE EDIT WITH CARE
2   'use strict';
3
4   var React = require("react");  7.6K (gzipped: 3.3K)
5
6   React.createElement("button", undefined, "Awesome!");
7
8   /*  Not a pure module */
```

```
1   module Awesome = {
2     let component = ReasonReact.statelessComponent("AwesomeButton");
3     let make = _children => {
4       ...component,
5       render: _self =>
6         <div>
7           <button> (ReasonReact.stringToElement("Awesome!")) </button>
8         </div>,
9     };
10  };
11
12  ReactDOMRe.renderToElementWithId(<Awesome />, "preview");
```

```
1  type state = {
2    active: option(string),
3    items: list(string),
4  };
5
6  type actions =
7    | Active(string)
8    | UpdateList(list(string));
```

```
16  let make = _children ⇒ {
17    ...component,
18    initialState: () ⇒ {active: None, items: []},
19    reducer: (action, state) ⇒
20      switch (action) {
21      | Active(name) ⇒ ReasonReact.Update({...state, active: Some(name)})
22      | UpdateList(l) ⇒ ReasonReact.Update({...state, items: l})
23      },
```

```
30    state.items
31    |> List.map(item =>
32    |  |  |  <li key=item>
33    |  |  |  |  <button onClick=(_event => send(Active(item)))>
34    |  |  |  |  |  (str(item))
35    |  |  |  |  </button>
36    |  |  |  </li>
37    |  |  )
```

ReasonML

Typesystemer, OCaml og ReasonML

bartjs – april 2018

@mikaelbrevik / @kodesnutt