

REASON



@alejandronanez

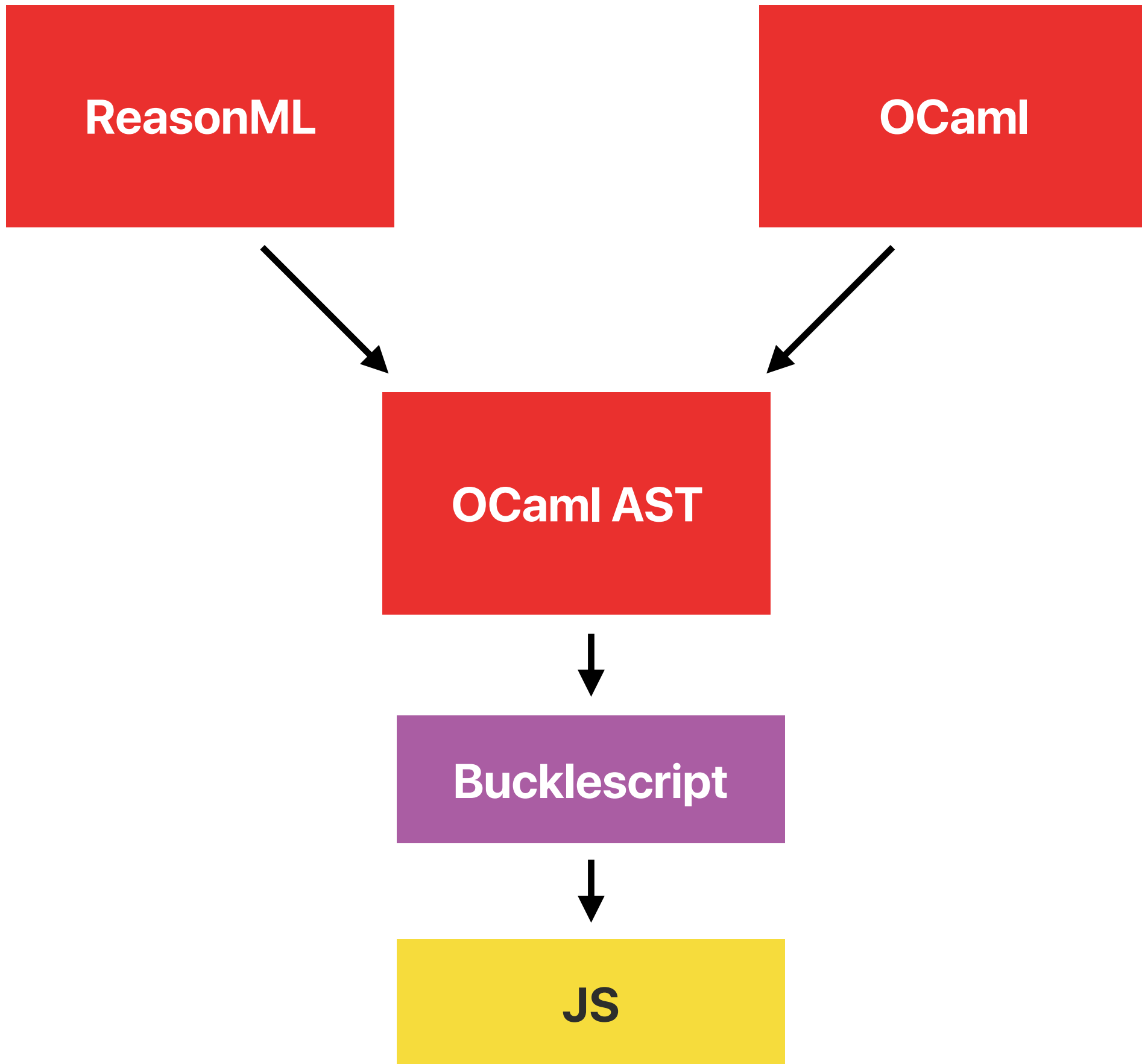
**Sintaxis MUY
parecida a Javascript**

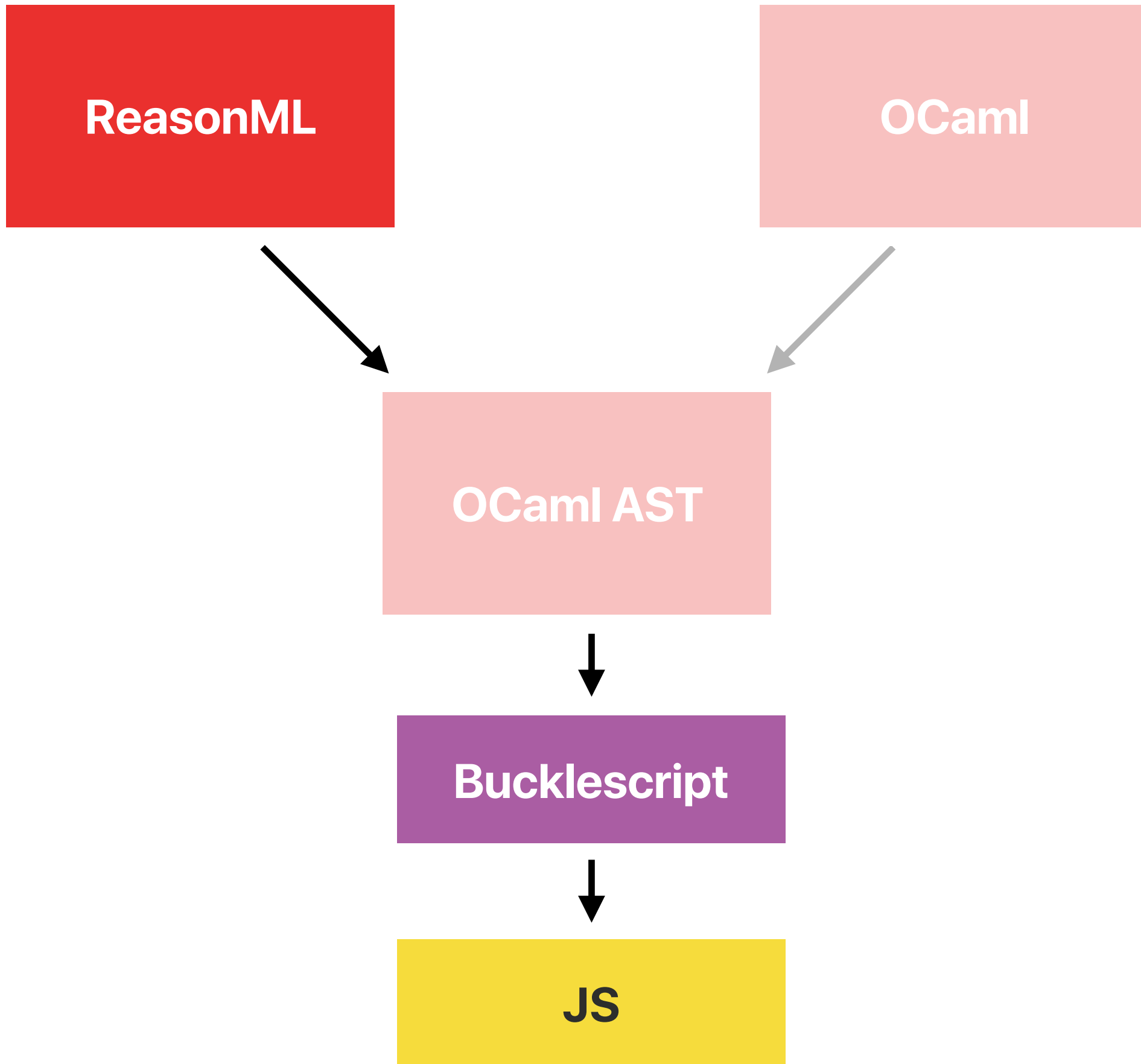
let edad = 32;

```
let getName = (name) => "Welcome " ++ name;  
getName("React MDE");
```

```
let coworkings = [  
    "Factoria",  
    "Seedspace",  
    "WeWork"  
];
```

```
if (true) {  
    Js.log("Hola!");  
};
```





Records

```
let meetup = {nombre: "React Medellin", edad: 2};
```

```
35 | let meetup = {nombre: "React Medellin", edad: 2};
```

✖ Index.re 1 of 1 problem

Error: Unbound record field nombre

```
let meetup = {nombre: "React Medellin", edad: 2};
```

```
type meetup = {  
  nombre: string,  
  edad: int,  
};
```

```
let meetup = {nombre: "React Medellin", edad: 2};
```

Variants

```
type estado =  
  | Abierto  
  | Cerrado;
```

```
let coworking = Abierto;
```


Javascript

```
const state = {  
  loading: true,  
  error: false,  
  data: {}  
};
```

Javascript

```
state.loading // true  
state.error // true  
state.data // {}
```

```
state.loading // false  
state.error // true  
state.data // { ... }
```

```
state.loading // true  
state.error // false  
state.data // { ... }
```



@alejandronanez

```
type data = {coworkings: list(string)};
```

```
type state =  
  | Loading  
  | Error  
  | Success(data);
```



Pattern Matching

```
switch (true) {  
| true => "verdadero"  
| false => "falso"  
| _ => "No se :("  
};
```

```
let condition = Success({coworkings: ["workia"]});
```

```
let resultado =  
  switch (condition) {  
    | Loading => "Esta cargando"  
    | Error => "Erro :("  
    | Success(data) =>  
      "Hay " ++ string_of_int(Belt.List.length(data.coworkings)) ++ " datos"  
  };
```


Optionals

```
let coworkings = None;  
let coworkings = Some(3);
```

```
let coworkings =  
  switch (coworkings) {  
    | None => "No hay coworkings"  
    | Some(qty) => "Hay " ++ string_of_int(qty)  
  };
```

Functions

```
let getName = (name) => "Welcome " ++ name;  
getName("React MDE");
```

```
let name = (~firstName, ~lastName) => firstName ++ " " ++ lastName;
```

```
name(~firstName="React", ~lastName="Medellin")
```

```
name(~lastName="Medellin", ~firstName="React")
```

Módulos

Todos los archivos son módulos


```
/** Greeting.re */  
[@react.component]  
let make = () =>  
    <div>{React.string("Hey")}</div>;  
  
let nombre = "ReactMDE";
```

```
/** OtroArchivo.re */  
<Greeting />
```

```
let valor = Greeting.name;
```

**Puedes crear módulos dentro de
los archivos**

```
/** Greeting.re */  
module ModuloA = {  
    let name = "ReactMDE";  
};
```

/ OtroArchivo.re */**
Greeting.ModuloA.name

Pipe & Fast Pipe

```
let fullName = (firstName, middleName, lastName) =>
  firstName ++ " " ++ middleName ++ " " ++ lastName;

/** Alejandro Nanez Ortiz */
let normal = fullName("Alejandro", "Nanez", "Ortiz");

/** Alejandro Nanez Ortiz */
let fastPipe = "Alejandro"->fullName("Nanez", "Ortiz");

/** Alejandro Nanez Ortiz */
let pipe = "Ortiz" |> fullName("Alejandro", "Nanez");
```


ReasonReact


```
<div>  
  React.string("ReactMDE")  
</div>
```

```
/** App.re */  
[@react.component]  
let make = () => <div> {React.string("Hola")} </div>;
```

```
/** Index.re */  
ReactDOMRe.renderToElementWithId(<App />, "app");
```

Prop

```
/** App.re */  
[@react.component]   
let make = (~mensaje) =>  
    <div> {React.string(mensaje)} </div>;
```

<App mensaje="ReactMDE" />

Prop opcional

```
/** App.re */  
[@react.component]  
let make = (~mensaje="default") =>  
    <div> {React.string(mensaje)} </div>;
```



<App />

Gracias :)