Hi 👋

https://overmind-talk.etienne.tech

OVERMIND

Overmind

State Management library 👷

Manage everything but the view 🧠

Supports React, Vue & Angular ✨

By the team behind CerebralJS 🚀

Made with TS, for TS 🚀

🖐 I'm Etienne

👨‍🎓 Master degree at Gobelins

⚛️ React trainer & dev & consultant @LeReacteurIO

🖥️ Freelance

Mental model

" A mental model is a representation of the surrounding world, the relationships between its various parts and a person's intuitive perception about his or her own acts and their consequences. "

Wikipedia

In Programming

In programming, mental model is about understanding what your program does and how you have to change it to acheive what you want.

Libraries help us shape
the mental model of our apps

React ↦ Components

Redux ↦ Action, Reducer

Naming is important

Presentational ⇔ Container Components

`render` ⇔ `handleInputChange`

Overmind Mental model

State

Derive

Effects

Action

State

One unique state tree (like Redux)

Define what your app look like

When you update the state, the view get updated too

Derive

`(state: State) => DerivedValue`

When you update the state, drived get updated too

In the view, derived are used like state

In fact derived are in the state

Effects

Anything related to side effects

Like fetching data, writting local storage, generating a new id

Overmind does nothing with them...

...except tracking, so they appear in the devtool

Action

Do stuff

Like changing the state and using effects

```
async (ctx: Context) => any
```

Context

...Effects

state

value

myAction(value)

Mutation

Just mutate the state

```
state.isLoading = true
```

Uses Proxy under the hood to track changes

Can't mutate outside of an action

Action

```
const myAction = async ({
  api,
  state,
  value: query
}) => {
  state.isLoading = true;
  const result = await api.getResult(query);
  state.result = result;
  state.isLoading = false;
};

// the the view
myAction("this-is-the-query");
```

Pipe Action

Pipe Action

Operations

Pipe Action

op-op-spec

```
type Operator<Input, Output = Input> = (
  err: Error,
  value: Input,
  next: (err: Error, value: Output) => void,
  final: (err: Error, value: Output) => void
) => void
```

Pipe Action

Operators

Mutate

Run

Map

Filter

When

Debounce

...

Pipe Action

Operators

```ts
const mutate = (
  doMutation: (context: Context) => void
): Operator => {
  // ...
}
```

Pipe Action

Operators

```
const setIsLoading = mutate(({ state, value }) => {
  state.isLoading = value
})
```

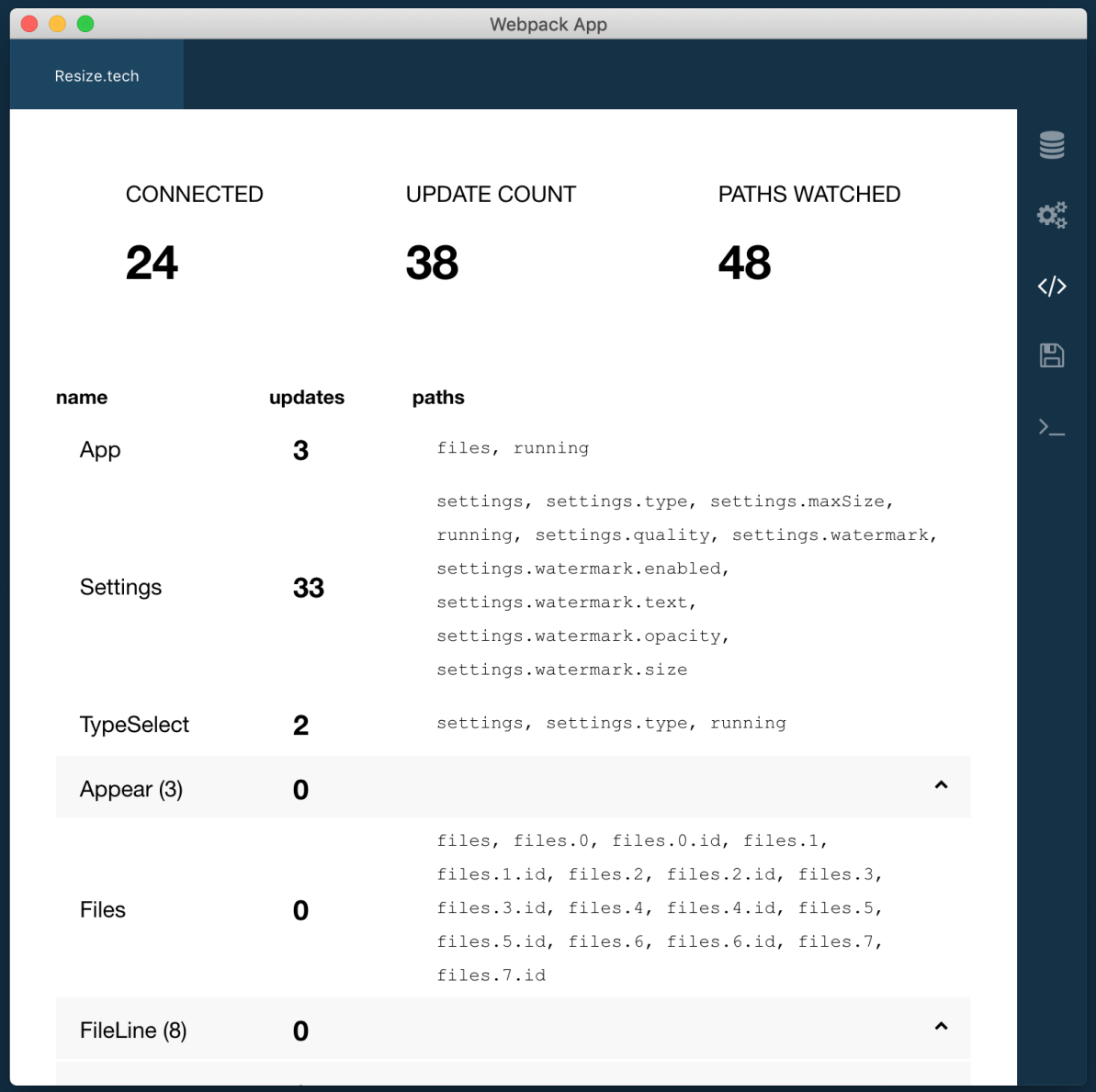
Pipe Operator

```
function pipe(...operators) {
  return (err, context, next, final = next) => {
    if (err) next(err);
    else {
      // run each operator one after the other
      // if an operator return a Promise
      // wait for the Promise to resolve (or reject)
      // before moving to the next
    }
  };
}
```

Pipe Operator

```
pipe(
  setLoading,
  fetchQueryResult,
  setQueryResult,
  setNotLoading
);
```

Devtools 🚀


Webpack App
Resize.tech
CONNECTED
24
UPDATE COUNT
38
PATHS WATCHED
48
name
updates
paths
App
3
files, running
settings, settings.type, settings.maxSize, running, settings.quality, settings.watermark, settings.watermark.enabled, settings.watermark.text, settings.watermark.opacity, settings.watermark.size
Settings
33
TypeSelect
2
settings, settings.type, running
Appear (3)
0
Files
0
files, files.0, files.0.id, files.1, files.1.id, files.2, files.2.id, files.3, files.3.id, files.4, files.4.id, files.5, files.5.id, files.6, files.6.id, files.7, files.7.id
FileLine (8)
0

Running the Devtools

```
npx overmind-devtools
```

Move fast, break things 🏎

☝ but not semver

Versions

October 26, 2018	**4.0.1**
October 23, 2018	**4.0.0**
October 10, 2018	**3.0.0**
September 18, 2018	**2.3.0**
September 15, 2018	**2.2.0**
September 15, 2018	**2.1.1**
September 14, 2018	**2.1.0**
September 13, 2018	**2.0.1**
September 11, 2018	**2.0.0**
September 9, 2018	**1.0.2**
September 9, 2018	**1.0.1**
September 7, 2018	**1.0.0**

Do we have time for more API ?

Yep

Not really

Hell No !

State

Derive

```ts
import { Derive } from "overmind";

const todoCount: Derive<number> = state =>
  state.todos.length;
```

```ts
type State = {
  todos: Array<Todo>;
  todoCount: Derive<number>;
};

const state: State = {
  todos: [],
```

Effects

```
async function fetchData(
  param: string
): Response {
  // ...
}

const effects = {
  fetchData
};
```

Actions

Pipe Actions

Operations

App

```ts
import { Overmind, TApp } from "overmind";

const config = {
  state,
  effects,
  actions
};

declare module "overmind" {
  interface IApp extends TApp<typeof config> {}
}
```

Connecting a view

```
import {
  TConnect,
  createConnect
} from "overmind-react";

const app = new Overmind(config);

export type Connect = TConnect<typeof app>;

export const connect = createConnect(app);
```

Using connect

That's it, thanks

next.overmindjs.org

Questions ?

PS: I'm on 🛰 @Etienne_dot_js