
Adieu RxJS ! Vive les Signals ! Oh wait...
Anthony Pena

Anthony Pena

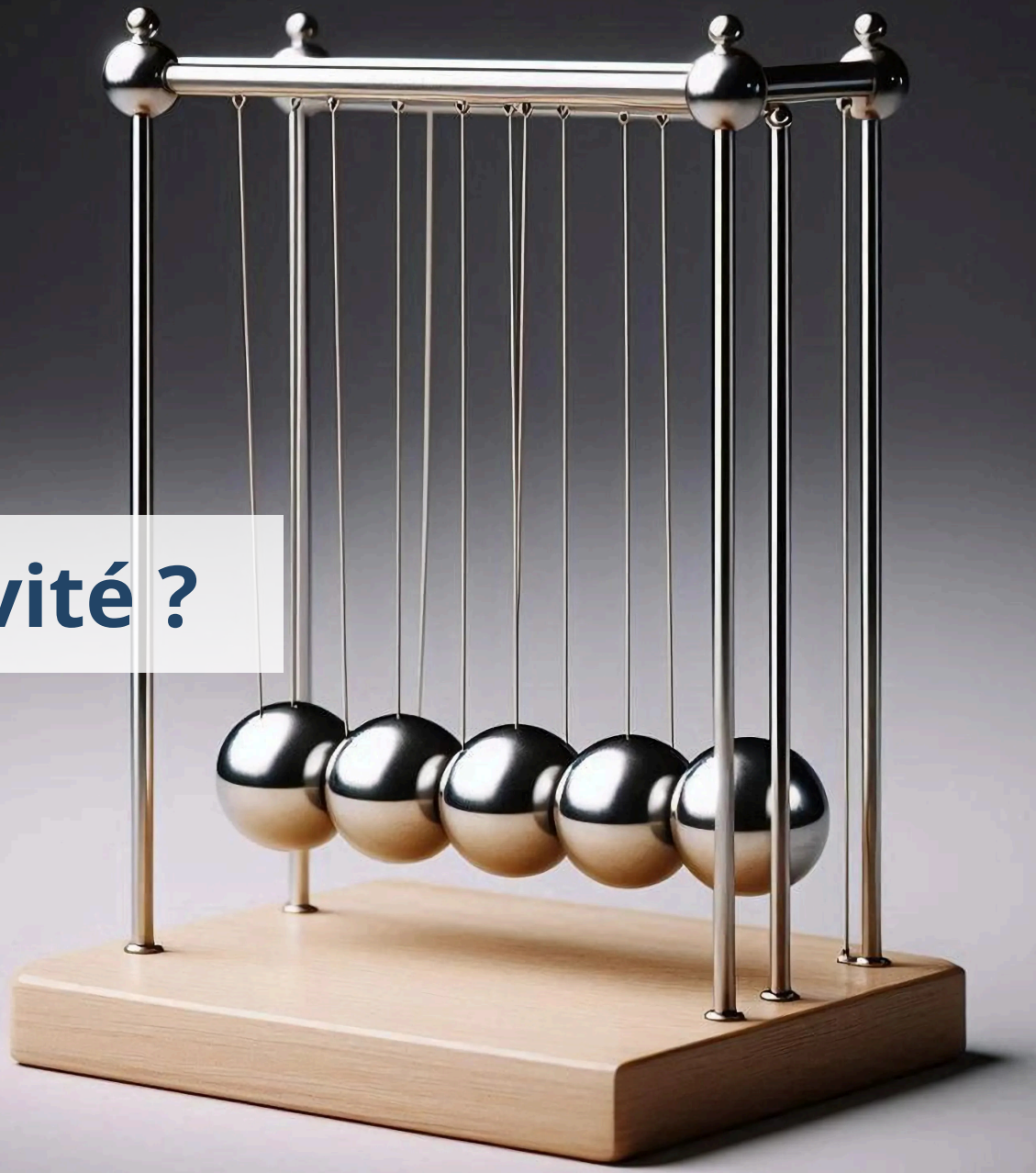
Développeur Web Fullstack @ [sf≡ir]

Women only Workshops

Avant les frameworks

© Anthony_Pena

La Réactivité ?



> "A declarative programming model for updating based on changes to state."
>
> -- Kristen / pzuraq

https://www.pzuraq.com/blog/what-is-reactivity

Angular.JS et ses watchers

@_Anthony_Pena

Angular 2 et Zone.js

@_Anthony_Pena

```
@Component({
  template: `
    <p>{{ text }}</p>
  `,
})
export class PlaygroundComponent {
  text = "";

  ngOnInit() {
      setInterval(() => this.text += '!', 1_000)
  }
}
```

```
@Component({
  template: `
    <input (change)="setText($event)"/>
    <p>{{ text }}</p>
  `,
})
export class PlaygroundComponent {
  text = "";

  setText(event: Event) {
    this.text = (event.target as HTMLInputElement).value;
  }
}
```

```
@Component({
  template: `
    <input [(ngModel)]="text"/>
    <p>{{ text }}</p>
  `,
})
export class PlaygroundComponent {
  text = "";

  ngOnInit() {
      setInterval(() => this.text += '!', 1_000)
  }
}
```

```
@Component({
  selector: 'app-root',
  standalone: true,
  template: `
    <p>{{ text }}</p>
  `,
})
export class PlaygroundComponent {
  text = "waiting...";

  ngOnInit() {
      this.asyncHello().then(text => this.text = text);
  }

  private asyncHello(): Promise<string> {
    // ...
  }
}
```

```
@Component({
  template: `
    <p>{{ text | async }}</p>
  `,
})
export class PlaygroundComponent {
  text = this.asyncHello();

  private asyncHello(): Promise<string> {
    // ...
  }
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  @Input() text: string;
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text  }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  @Input() text: string;
  @Output() textChange = new EventEmitter<string>();
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  @Input() text: string;
  @Output() textChange = new EventEmitter<string>();
  @ViewChild(Alert) alerts: Alert;
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  @Input() text: string;
  @Output() textChange = new EventEmitter<string>();
  @ViewChild(Alert) alerts: Alert;
  @ViewChildren(CustomCard) cards: QueryList<CustomCard>;
}
```


C'est cool tout ça non ?
@_Anthony_Pena


ZONE
spc
Oui mais Zone.js
@_Anthony_Pena


Angular 17 et les Signals
@_Anthony_Pena

```ts
@Component({
  template: ` <p>{{ text() }}</p> `,
})
export class PlaygroundComponent {
  text = signal('');

  constructor() {
    setInterval(() => {
      this.text.set(this.text() + '!');
      // or
      // this.text.update((actual) => actual + '!')
    }, 1_000);
  }
}
```

```
@Component({
  template: `
    <input (change)="setText($event)" />
    <p>{{ text() }}</p>
  `,
})
export class PlaygroundComponent {
  text = signal('');

  setText(event: Event) {
    this.text.set((event.target as HTMLInputElement).value);
  }
}
```

```
@Component({
  template: `<p>{{ text() }}</p> `,
})
export class PlaygroundComponent {
  text = signal('waiting...');

  constructor() {
    this.asyncHello().then((text) => this.text.set(text));
  }

  private asyncHello(): Promise<string> {
    // ...
  }
}
```

```
@Component({
  template: `
    <p>{{ text() }}</p>
    <p>{{ questionText() }}</p>
  `,
})
export class PlaygroundComponent {
  text = signal('');
  questionText = computed(() => this.text().replaceAll('!', '?'));

  constructor() {
    setInterval(() => {
      this.text.set(this.text() + '!');
    }, 1_000);
  }
}
```

```
@Component({
  template: ` <p>{{ text() }}</p> `,
})
export class PlaygroundComponent {
  text = signal('');

  constructor() {
    setInterval(() => this.text.set(this.text() + '!'), 1_000);
    effect(() => {
      console.log('text =', this.text());
    });
  }
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  @Input() text: string;
  @Output() textChange = new EventEmitter<string>();
  @ViewChild(Alert) alerts: Alert;
  @ViewChildren(CustomCard) cards: QueryList<CustomCard>;
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text() }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  text = input('');
  @Output() textChange = new EventEmitter<string>();
  @ViewChild(Alert) alerts: Alert;
  @ViewChildren(CustomCard) cards: QueryList<CustomCard>;
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text() }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  text = input('');
  textChange = output<string>();
  @ViewChild(Alert) alerts: Alert;
  @ViewChildren(CustomCard) cards: QueryList<CustomCard>;
}
```

```
@Component({
  template: `
    <app-alert>Alert</app-alert>
    <p>{{ text() }}</p>
    <app-card />
    <app-card />
  `,
})
export class PlaygroundComponent {
  text = input('');
  textChange = output<string>();
  alerts = viewChild(Alert);
  cards = viewChildren(CustomCard);
}
```

Signals ❤️

tc39 / **proposal-signals**

Type / to search

Code | Issues 101 | Pull requests 11 | Actions | Security | Insights

proposal-signals

Public

generated from tc39/template-for-proposals

Unwatch 97 | Fork 54 | Starred 3k

main | 8 Branches | 0 Tags | Go to file t | Add file | Code

littledan Simpler logo (#229) b608efa · 4 days ago 146 Commits

README | Code of conduct | MIT license | Security

About

A proposal to add signals to JavaScript.

- Readme
- MIT license
- Code of conduct
- Security policy
- Activity
- Custom properties
- 3k stars
- 96 watching
- 54 forks

Report repository

Contributors 20

JavaScript Signals standard proposal

Stage 1 (explanation)

TC39 proposal champions: Daniel Ehrenberg, Yehuda Katz, Jatin Ramanathan, Shay Lewis, Kristen Hewell Garrett, Dominic Gannaway, Preston Sego, Milo M, Rob Eisenberg

Original authors: Rob Eisenberg and Daniel Ehrenberg

Bientôt un standard

This document is currently a champion of JavaScript, similar to the Promises/A+ effort which preceded the Promises standardized by TC39 in ES2015. Try it for yourself, using a polyfill.

Similarly to Promises/A+, this effort focuses on aligning the JavaScript ecosystem. If this a[illegible] then a standard could emerge, based on that experience. Several framework authors are collaborating here on a

@_Anthony_Pena

https://github.com/tc39/proposal-signals

Zone.js c'est fini ?

Oups j'ai oublié RxJS dans tous ça 🙊 (non)

Mais au fait... C'est quoi RxJS ?


RxJS
@_Anthony_Pena

ReactiveX for JavaScript

@_Anthony_Pena


ReactiveX
An API for asynchronous programming
with observable streams
@_Anthony_Pena


Parlons Observable
@_Anthony_Pena

```ts
import { toObservable, toSignal } from '@angular/core/rxjs-interop';

@Component({})
export class PlaygroundComponent {
  text = signal('Hello');
  text$ = toObservable(this.text);
  textSignal = toSignal(this.text$);
}
```

Observables

Composable functions with guarantees.

Use observables for cancellation and event coordination.

- **Generally stateless**
  process event and clean up
- **Multiple values**
  a collection of events or values over time
- **Push N values**
- **Convert to Signal? Yes**

https://twitter.com/BenLesh/status/1775207971410039230

Signals

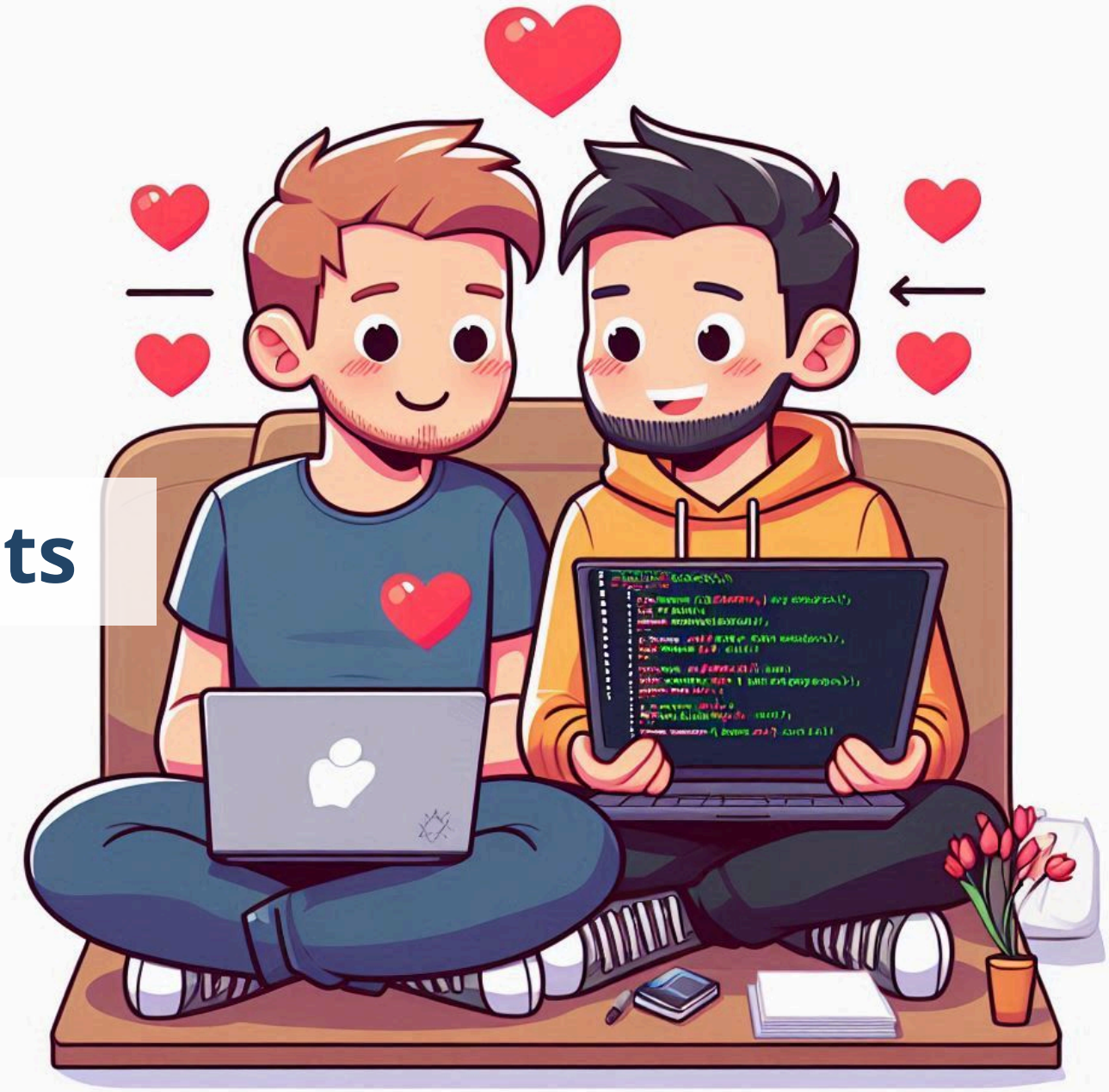
A dependency graph of values and logic around notification, invalidation, computation, and memoization.

Use signals for state management.

- **Maintain state**
  retain things in memory
- **Single value**
  a value that changes over time
- **Read 1 value**
- **Convert to Observable? Avoidable**
  the event that set the signal can also notify the observabl


Mais du coup les Signals...
@_Anthony_Pena

Quelques cas concrets



HttpClient

```
@Component({
  template: ` <button (click)="clicked()">Get</button> `,
})
export class PlaygroundComponent {
  http = inject(HttpClient);

  clicked() {
    this.http.get('/axolotl').subscribe();
  }
}
```

… et interceptor

```
export function authInterceptor(
  req: HttpRequest<unknown>,
  next: HttpHandlerFn
): Observable<HttpEvent<unknown>> {
  return next(
    req.clone({
      headers: req.headers.set('X-Quiz', 'Axolotl'),
    })
  );
}
```

État interne des composants

```
@Component({
  template: ` <p>{{ text() }}</p> `,
})
export class PlaygroundComponent {
  text = signal('');
}
```


Reactive Forms
@_Anthony_Pena

Services


Global state management
(NgRx, NgXs)
@_Anthony_Pena

The NgXs way

Classic way

```
@Component({ ... })
export class ZooComponent {
  animals$: Observable<string[]> = this.store.select(ZooState.getAnimals);

  constructor(private store: Store) {}
}
```

Signal way

```
@Component({ ... })
export class ZooComponent {
  animals = select(ZooState.getAnimals);
}
```

The NgRx way

```
type BooksState = {
  books: Book[];
  isLoading: boolean;
  filter: { query: string; order: 'asc' | 'desc' };
};

const initialState: BooksState = {
  books: [],
  isLoading: false,
  filter: { query: '', order: 'asc' },
};

export const BooksStore = signalStore(
  withState(initialState),
  withComputed(/* ... */),
  withMethods((store, booksService = inject(BooksService)) => ({/* ... */})))
);
```

```
@Component({
  providers: [BooksStore],
})
export class BooksComponent {
  readonly store = inject(BooksStore);
}
```

The NgRx way

```
export const BooksStore = signalStore(
  withMethods((store, booksService = inject(BooksService)) => ({
    loadByQuery: rxMethod<string>(
      pipe(
        debounceTime(300),
        distinctUntilChanged(),
        tap(() => patchState(store, { isLoading: true })),
        switchMap((query) => {
          /* ... */
        })
      )
    ),
  }))
);
```

En résumé


Les Signals c'est pour gérer les états et la réactivité dans les composants
@_Anthony_Pena

RxJS est là pour gérer tous vos flux de données

@_Anthony_Pena

https://x.com/BenLesh/status/1775207971410039230

Anthony Pena

Développeur Web Fullstack @ [sf≡ir]

@_Anthony_Pena_

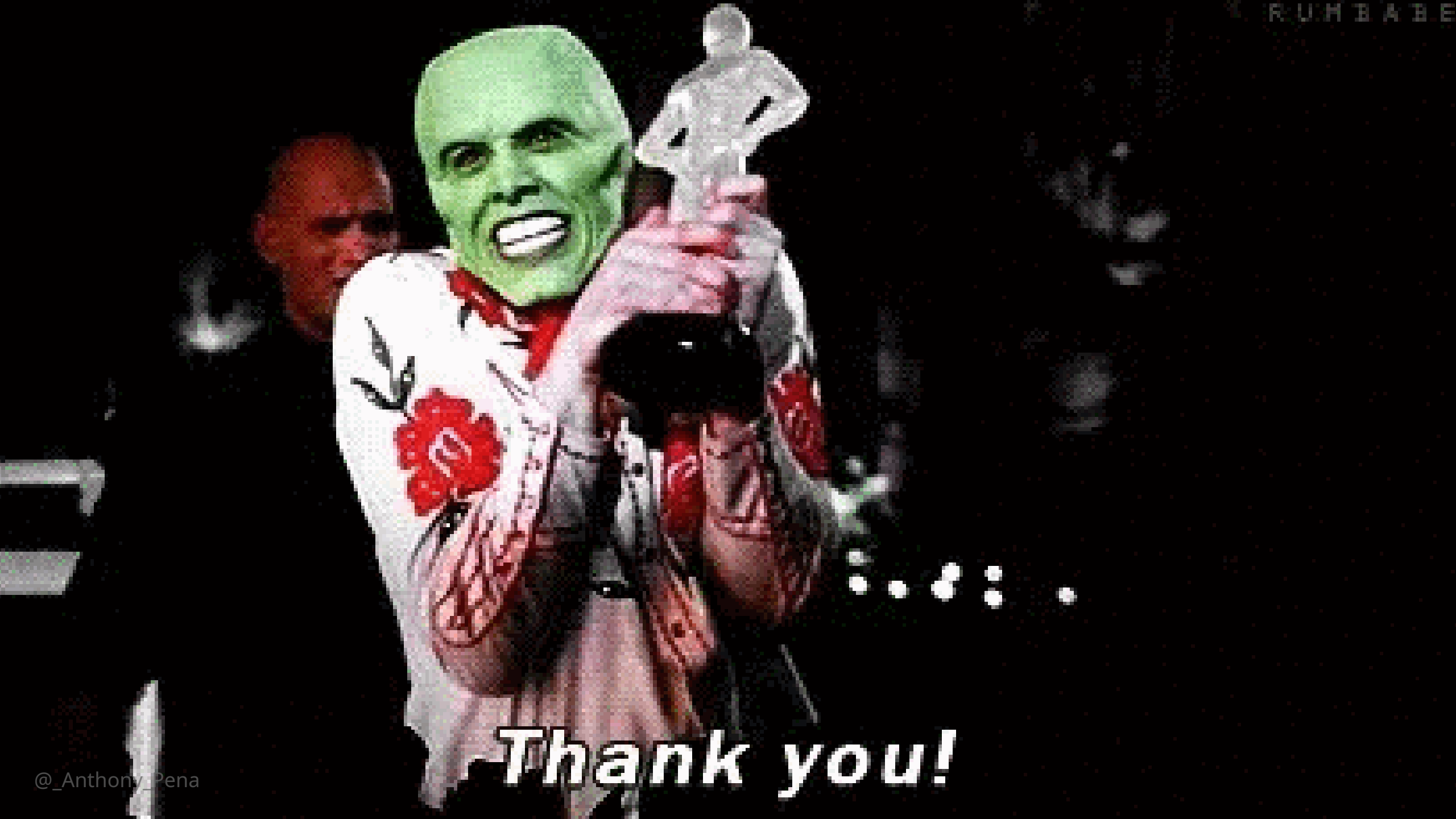
@kuroidoruido

@penaanthony

https://k49.fr.nf

https://github.com/kuroidoruido/talks

Women only Workshops


Thank you!
@_Anthony_Pena