do your thing

The Netherlands

Argentina

Córdoba


3x

Who I'm?

Matias Fernandez Martinez {

- 9 years Front-end development
- Back in the days of jQuery
- Angular, ExpressJS & SSR React
- Lit

}

matias.Fernandez.Martinez@ing.com
LinkedIn.com/in/matias-fernandez-martinez
Github.com/thematho

Things I like

Matias Fernandez Martinez {

- Role, board & video games
- Spend time with my kids
- “JavaScript powered” things

}

Unreal.js

Bangle.js

PixiJS

Event-driven Dependency Injection

(Dependency resolution in Custom Elements)

Matias Fernandez Martinez

October 2022

Wholesale banking – ING Princing Architecture

do your thing

agenda.html

agenda.css

```
Agenda {

    01    What is Dependency Injection
          What and why?

              02    Custom Elements
                    The basics

                        03    The conflict

}
```

Spoilers

di.html

di.css

01 {

[Dependency Injection]

< What and why? >

}

with-dependency.js

with-dependency-injection.js

What is Dependency Injection?

```
class Cooker {


    constructor() {
        this.shop = new PizzaShop();
    }


}

let pizza = new Pizza();

```

with-dependency.js | with-dependency-injection.js

What is Dependency Injection?

```
class Cooker {


    constructor(pizzaShop) { <== Pushed into the constructor
        this.shop = pizzaShop;
    }
}
import { injector } from '@food-injector';


let pizza = injector.get('pizza') <== Pushed

```

di.txt

ic.txt

Dependency Injection < /1 > {

< Program for a interface >
< Decouple dependencies >

}

Inversion of Control < /2 > {

< Inverts the flow >
< The Framework / container instantiate what you need >
< Tell don't ask principle >

}

Concepts

quotes.html

quotes.css

```
< “A 25-dollar term for a 5-cent concept” >

  – Dependency Injection Demystified ‘James Shore’

< “A Setter with bells & whistles” >

  – this conference ‘I’
```

Great and not so great quotes

simple.js

benefits.css

```
Fancy name for a ‘Simple concept’ {

    Decouple

    Scoped implementations

    Runtime config

    Testing

    Override behaviour

}
```

custom-elements.js

custom-elements.html

```
02 {

  [Custom Elements]

}
```

custom-elements.js

custom-elements.html

Custom Elements

```
1
2
3   class PopUpInfo extends HTMLElement {
4
5     constructor() {
6
7       super();
        console.log('Hello Popup’);
8
9     }
10  }
11
12  customElements.define('popup-info-element', PopUpInfo);
13
14
```

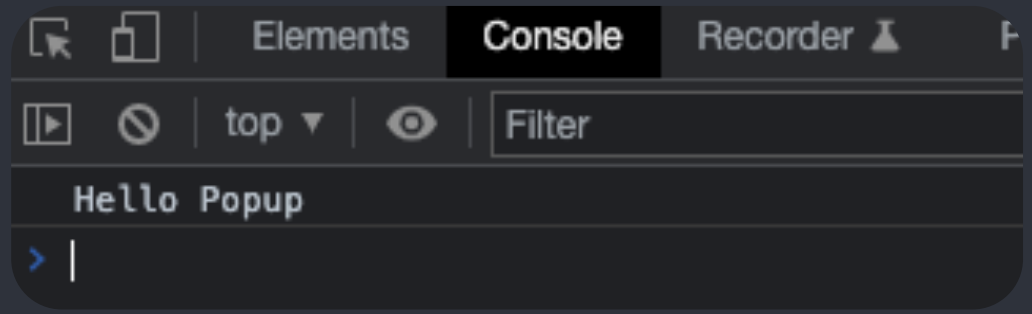
custom-elements.js

custom-elements.html

Custom Elements

```
<body>

  <popup-info-element></popup-info-element>

</body>
```

The Browser calls the constructor



confict.html

confict.css

```
03 {
    [The conflict]
}
```

confict.html

confict.css

```
1   < DI relies on >
2
3       < A container >
4
5       < Call the constructor with dependencies>
6           constructor(myService)
7
8
9    < Custom Elements >
10
11      < Are instanciated by the browser >
12          <my-element></my-element>
13
14      < Calls the constructor without arguments >
            constructor() ← ~~myService?~~
```

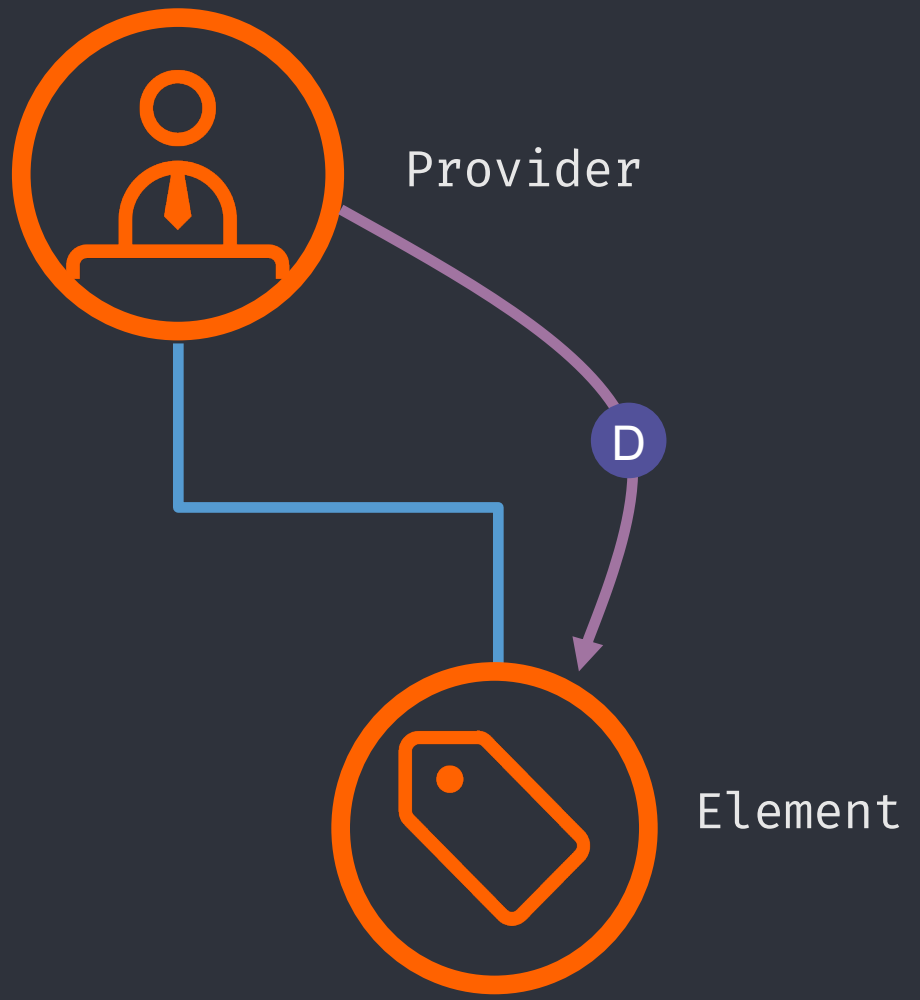
resolution.html

resolution.css

```
01.1 {

    [Dependency ~~Injection~~]
                resolution

}
```

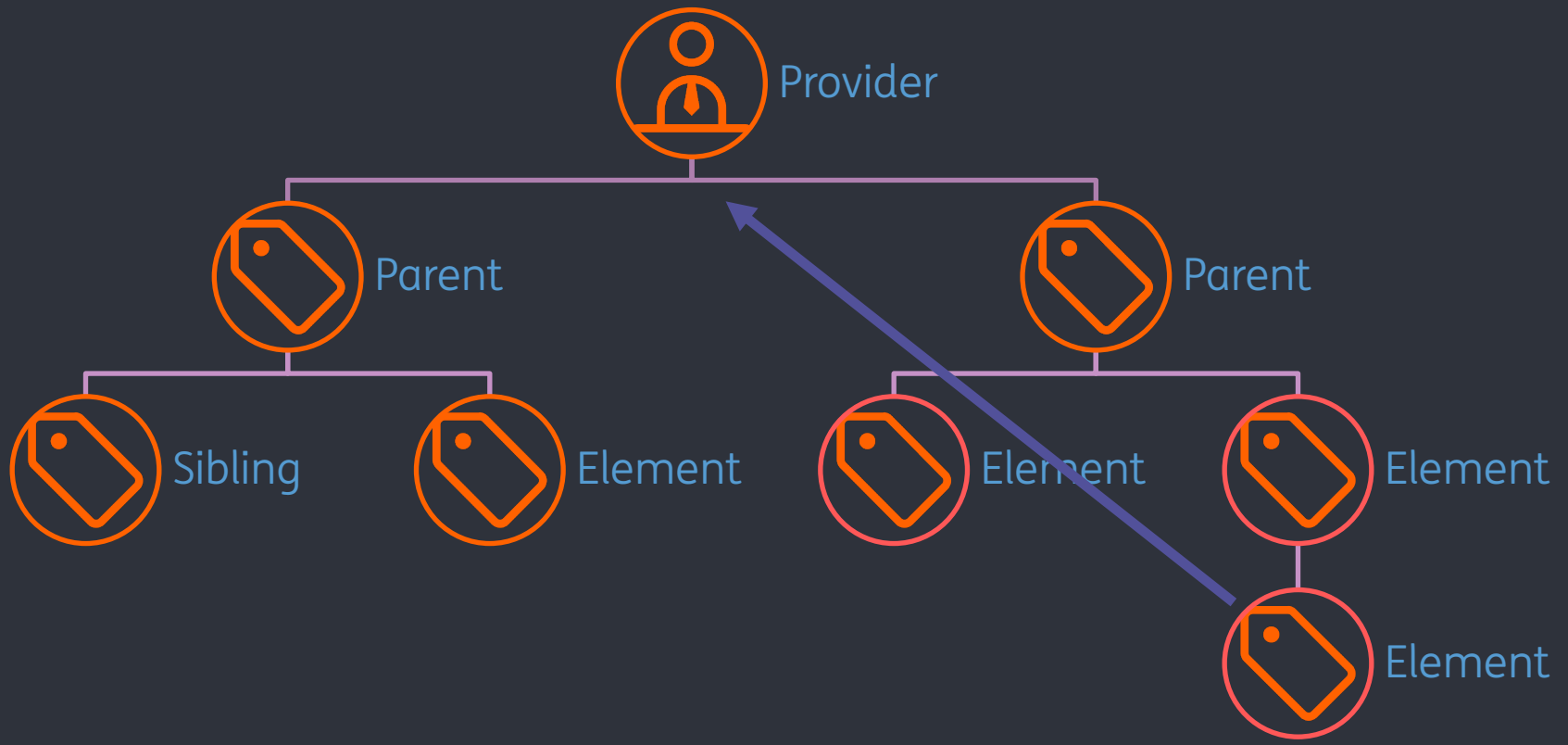
resolve-dependencies.html

resolve-dependencies.css



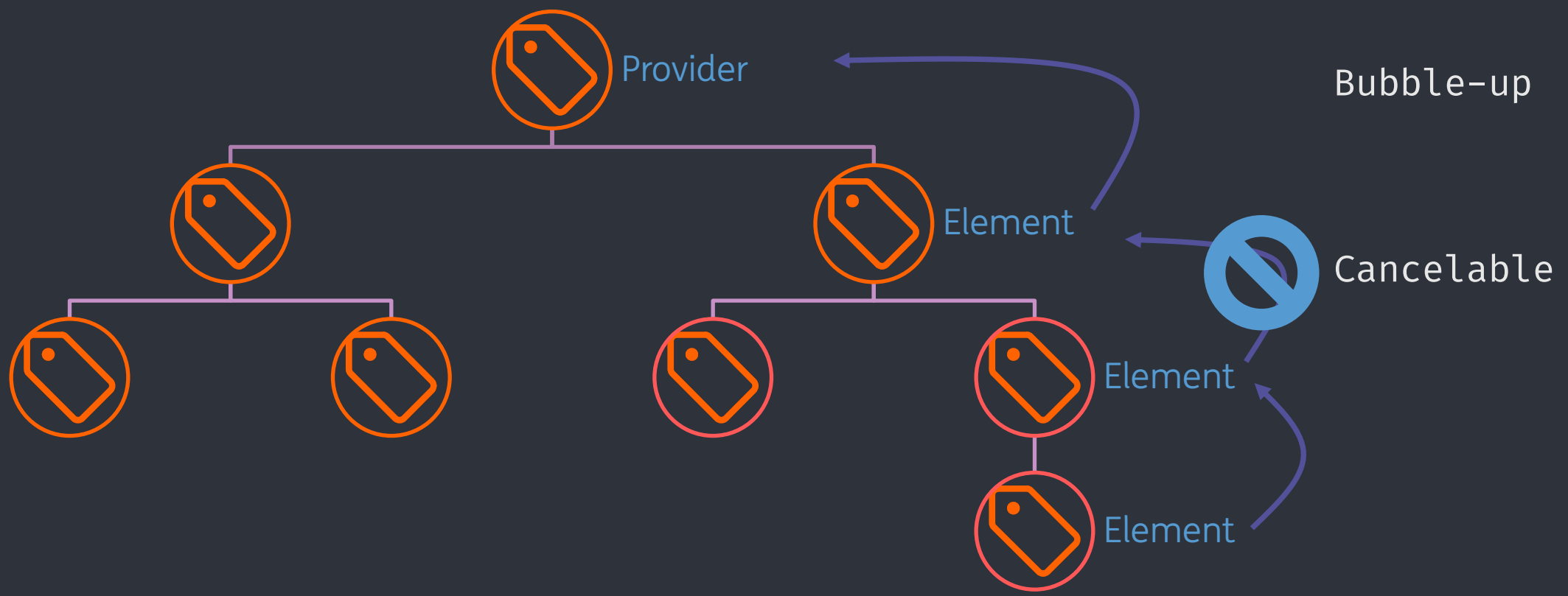
resolve-dependencies.html

resolve-dependencies.css



resolve-dependencies.html

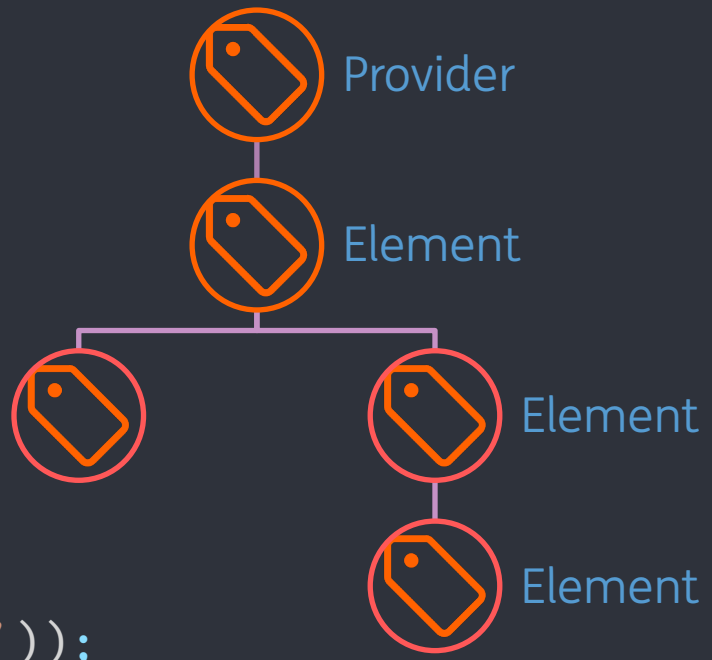
resolve-dependencies.css



resolve-dependencies.html

resolve-dependencies.css

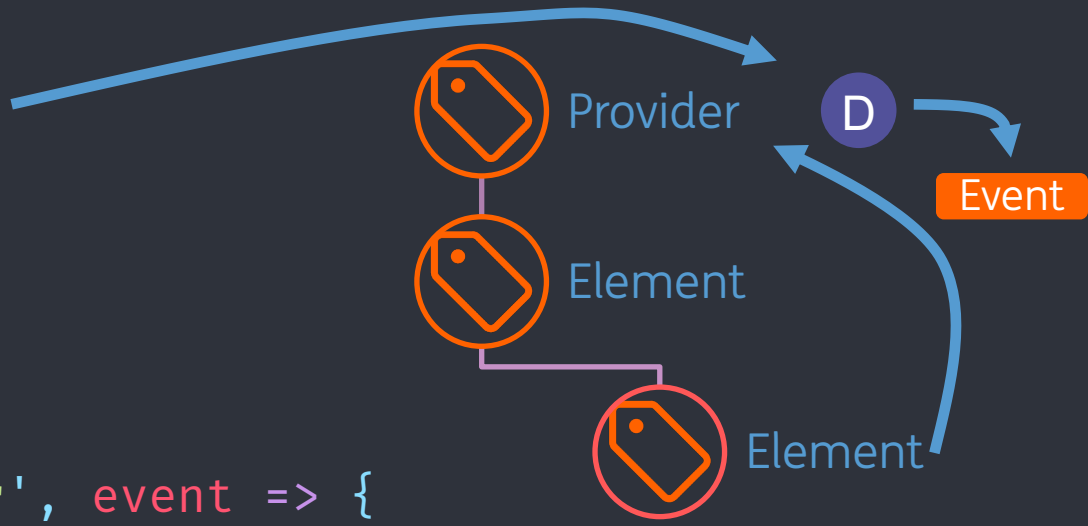
```
Mutate the Event Object

addEventListener('', event => {
  event.payload = 'foo'; // Mutate
})

Synchronous

this.dispatchEvent(event); // sync!
Provider
addEventListener('inject', event => {
  event.dependency = DateServiceFactory('DD/MM/YYYY'));
  event.stopPropagation();
});
Element
this.dateService = event.dependency;
```



provider.js

my-element.js

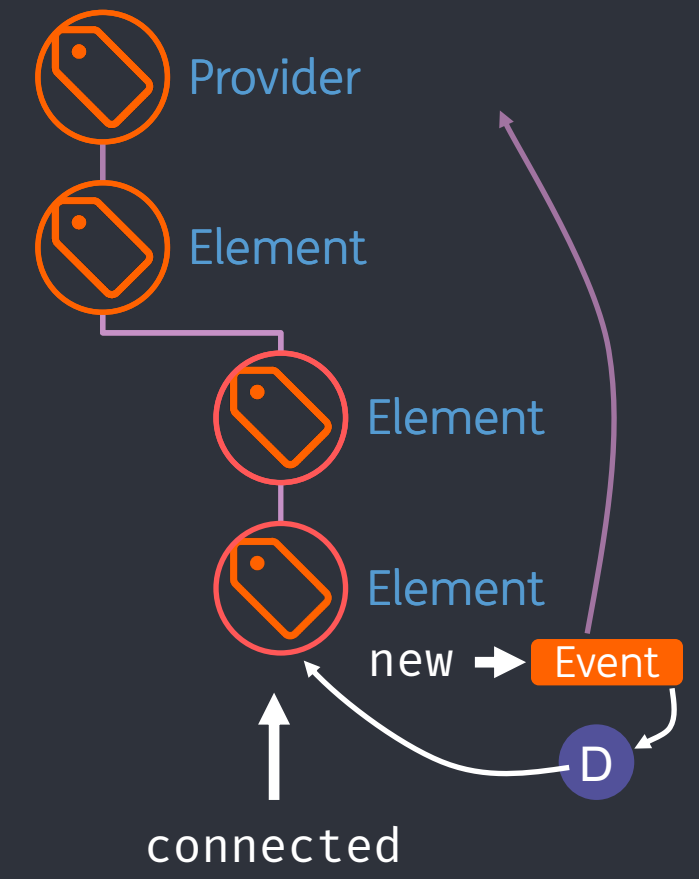
```
constructor() {
  this._providers = new Map();
  this._providers.set('dateService',
    DateServiceFactory('DD/MM/YYYY')
  );

}

connectedCallback(){
  this.addEventListener('inject-provider', event => {
    const { dependencyName } = event;
    if (this._providers.has(dependencyName)) {
      event.provider = this._providers.get(dependencyName);
      event.preventDefault();
      event.stopPropagation();
    }
  });

}
```



provider.js

my-element.js

```
1   connectedCallback() {
2
3     const event = new CustomEvent('inject-provider', {
4       bubbles: true,
5       cancelable: true,
6     });
7     event.dependencyName = dependencyName;
8
9     this.dispatchEvent(event); // synchronous
10
11    if (event.defaultPrevented) {
12      this.dateService = event.dependency;
13    };
14
  }
```



Demo

thematho.github.io/lit-elements-dependency-injection-demo

github.com/thematho/lit-elements-dependency-injection-demo

provider.js

my-element.js

```
import { DependencyProviderMixin } from '@di/DependencyProvider';
import { DateServiceFactory } from '../factories/DateServiceFactory';

class Europe extends DependencyProviderMixin(LitElement) {


  // ...
  connectedCallback() {
    // ...
    this._providers.set('DateService', DateServiceFactory('DD/MM/YYYY'));

  }


}
```

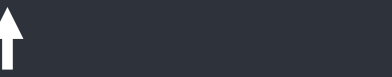
Event listener for ‘inject’ events & injects dependencies

Instatiate the dependency

provider.js

my-element.js

```
import { InjectMixin } from '@di/Inject.js';

export class Product extends InjectMixin(LitElement) {

  // ...
  connectedCallback() {
  // ...
  this._dateService = this.inject('DateService');

 }

}
```



Dispatches an Events

Declares the service name

before.js

after.js

Coupled to the DateServiceFactory implementation

```
import { DateServiceFactory } from '../factories/DateServiceFactory.js';

export class Product extends LitElement {

  constructor() {
    this._dateService = DateServiceFactory('DD/MM/YYYY');
  }
}
```

Decoupled from implementation

```
import { InjectMixin } from '@di/Inject.js'
export class Product extends InjectMixin(LitElement) {

  connectedCallback() {
    this._dateService = this.inject('DateService');
  }
}
```

Some credits

Inspired on *“DI with Custom elements”* by Justin Fagnani

youtu.be/6o5zaKHedTE

“Dependency Injection desmystified” article by Justin Fagnani

jamesshore.com

Others:

- No you can’t art by Julia Grumt — deviantart.com/julia-grumt
- Open-Source Hackable smartwatch — banglejs.com

Resources

Source code

github.com/thematho/lit-elements-dependency-injection

Presentation

Documentation of APIs

developer.mozilla.org

lit.dev/docs


Questions?

Thanks!

do your thing

matias.Fernandez.Martinez@ing.com

Github.com/thematho

LinkedIn.com/in/matias-fernandez-martinez