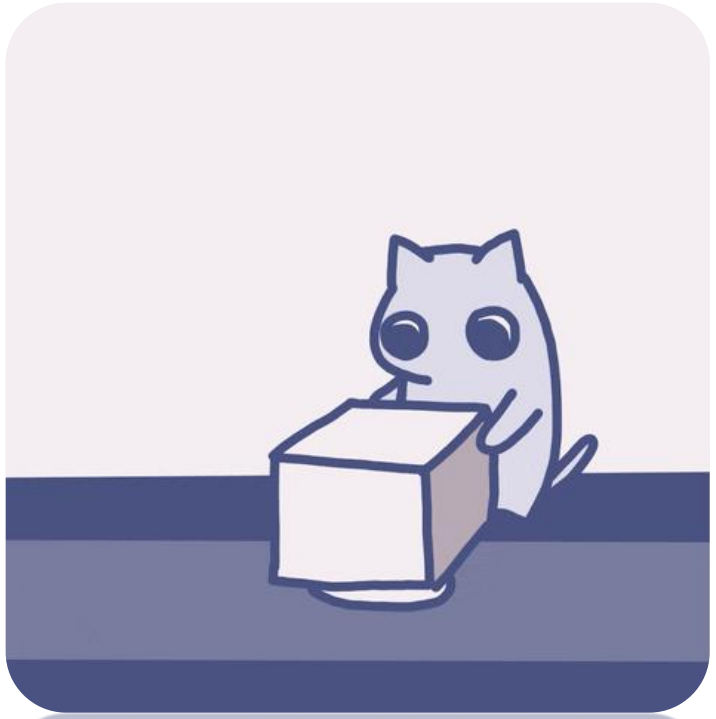
MODEL

MODÈLE DE DONNÉE

Le principe



Un moule mais des objets uniques

Class

Objet

Objet

Objet

L’intérêt en programmation en générale ?

Dans Angular ?

TypeScript

Typage, Sécurité dans un 1er Temps

```
salaire: number;
slogan: string;
vérité:boolean;
```

```
cars: Car[] = [
  {
    name: "pagani h
    pays: "italie",
    coverImage:"./a
    power: 765,
    perf: 3.2
  },
```

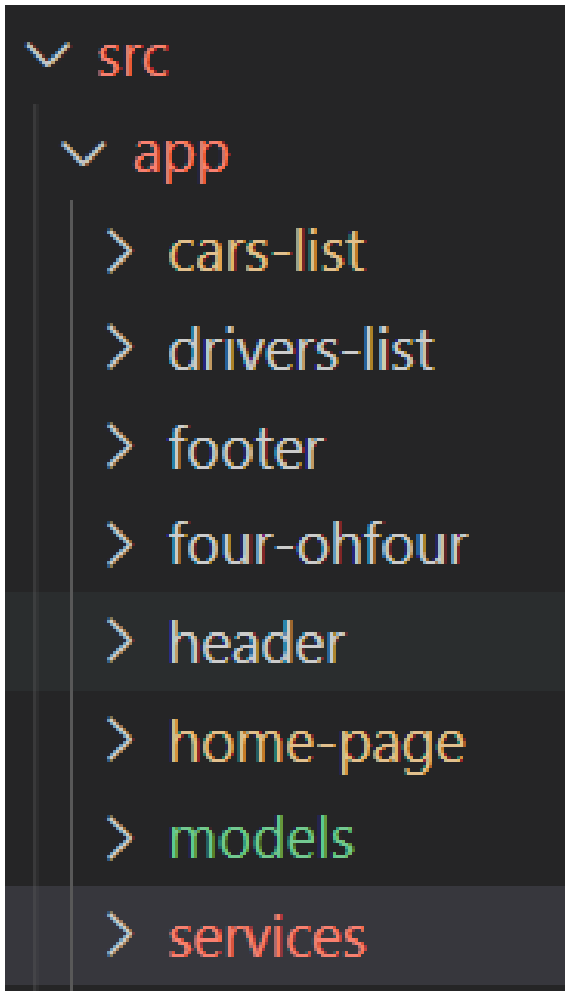
Typage

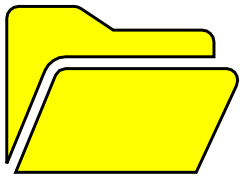
Sécurité exemple

```
cars: Car[] = [
  {
    name: "pagani huayra",
    pays: "italie",
    coverImage:"./assets/i
    power: 765,
    perf: 3.2,
    truc: "Un machin"
  },
```

```
Type '{ name: string; pays: string; coverImage: string; power:
number; perf: number; truc: string; }' is not assignable to type
'Car'.
  Object literal may only specify known properties, and 'truc' does
not exist in type 'Car'. ts(2322)

View Problem (Alt+F8)    No quick fixes available
truc: "Un machin"
```

Mise en pratique des models





Créer un dossier models dans app

Créer un fichier Car.ts dans models

Rappel

Convention:

- une Class type model nom du fichier en majuscule
- Dossier peut s'appeler models, model ou autres

POO – CLASS

```
export class Car {
```

Créer la class Car

Propriétés de la class

name : string

pays: string

coverImage: string

power: number

perf: number

Car Model

```
export class Car {

    public name:string;
    public pays: string;
    public coverImage: string;
    public power: number;
    public perf: number;

    constructor(name:string,pays:string,coverImage:string,power:number,perf:number){
        this.name = name;
        this.pays = pays;
        this.coverImage = coverImage;
        this.power = power;
        this.perf = perf;
    }

}
```

Typer grâce au model

```
import { Car } from '../models/Car';
```

```
class DataService {
```

```
cars: Car[] = [
  {
```

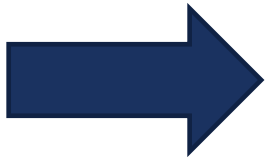
```
export class CarItemComponent

  @Input()car: Car;
```

```
export class CarsListComponent

  carUpdate:any;
      You, 6 months ago • Mini
  cars: Car[];
```

Utiliser la class pour créer une voiture

```
export class CarsListComponent implements OnInit {

  carUpdate:any;

  cars: Car[];

  voiture: Car = new Car("peugeot L750 r","France","http://",750, 2.4);
```

```
ngOnInit(): void {
  this.carUpdate = new Date();
  console.log(this.voiture);
}       You, 6 months ago • Sta
```



```
▼ Car i
    coverImage: "https://ww
    name: "peugeot L750 r"
    pays: "France"
    perf: 2.4
    power: 750
```

Si je veux voir cette voiture à la suite des autres ?

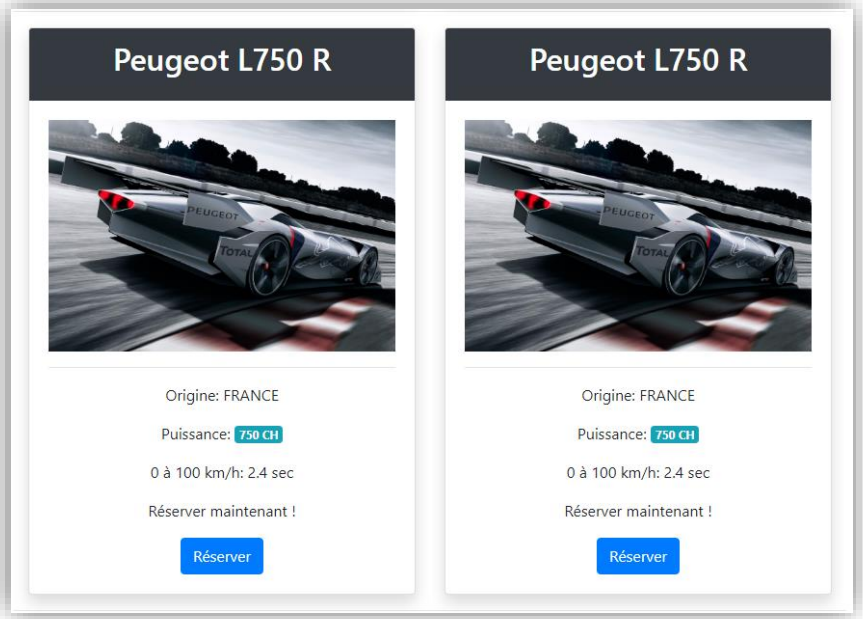
Plusieurs
Solutions
Possibles

Spoil pour la suite…

Solutions…

CarsListComponent

```
constructor(private data:DataService) {
  this.data.cars.push(this.voiture);
  this.cars = this.data.getAllCars();
```




```
export class DataService {

  voiture: Car = new Car("pe
```

```
  },
  this.voiture
];
```

Retour sur la class Car – Syntaxe TypeScript - Raccourcis

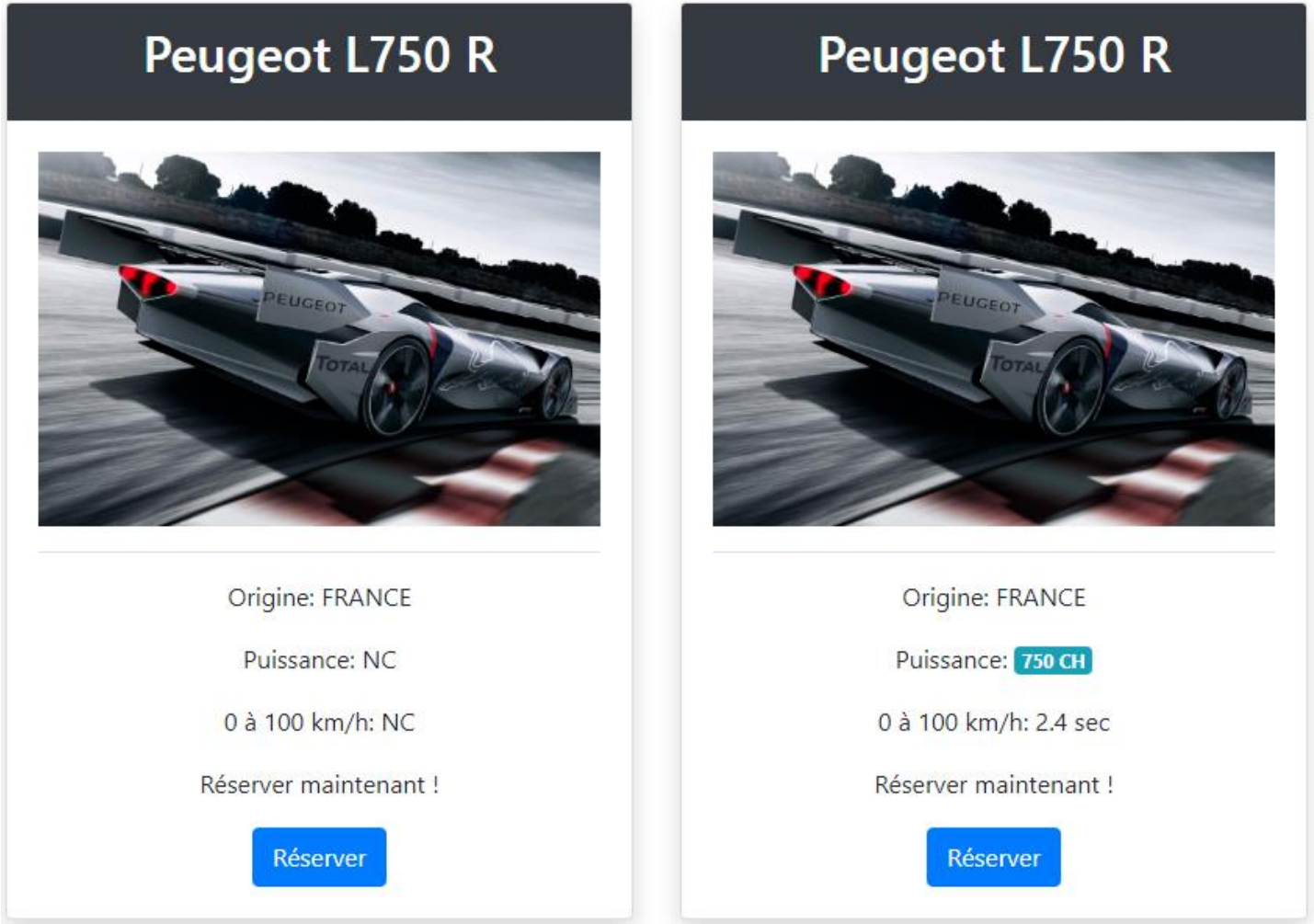
```
export class Car {

    constructor(public name:string,
                public pays:string,
                public coverImage:string,
                public power:number,
                public perf:number){}

}
```

Rendre facultatif une props du constructor

```
public power?:number,
public perf?:number){}
```

Tester

Rappel Component Reactif – Directive Structurelle

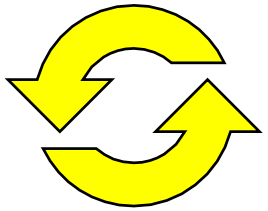


FORMULAIRE

TEMPLATE & REACTIVE

Créer des Cars dynamiquement

Mais pas encore persister en bdd…



Angular Form
2 Méthodes

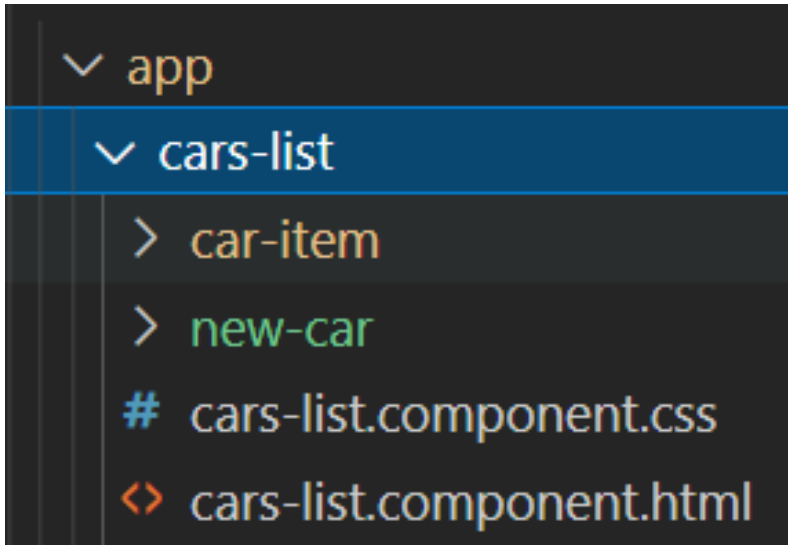
Template
Code uniquement
dans le template
Plus simple

Reactive Form
Class + Template
Plus avancée

Méthode Template

Dans le terminal

```
Drive-X> ng g c cars-list/newCar
```



Créer le component qui va accueillir le formulaire

Ps: dans cars-list simple suggestion
On peut s'organiser autrement

Détails visuelle

Sur la page des véhicules

Donc dans le routing…

```
const routes: Routes = [
  {path: '', component: H
  {path: 'cars', componen
  {path: 'new-car', compo
```

```
<a  class="badge badge-success mx-5 p-2" routerLink="/new-car">+ Ajouter une voiture</a>
```


Ne pas oublier le slash pour la route, pour repartir de la racine…

Un formulaire bootstrap

‹hackerthemes›

Forms

On récupère on fignole

Code snippet

```
<form>
  <div class="form-group">
    <label for="formGroupExam
    <input type="text" class=
```

Copy

Créer une voiture

Nom de la voiture

Pays d'origine

Photo de la voiture (url)

Puissance

Temps de 0 à 100 km/h

Enregister

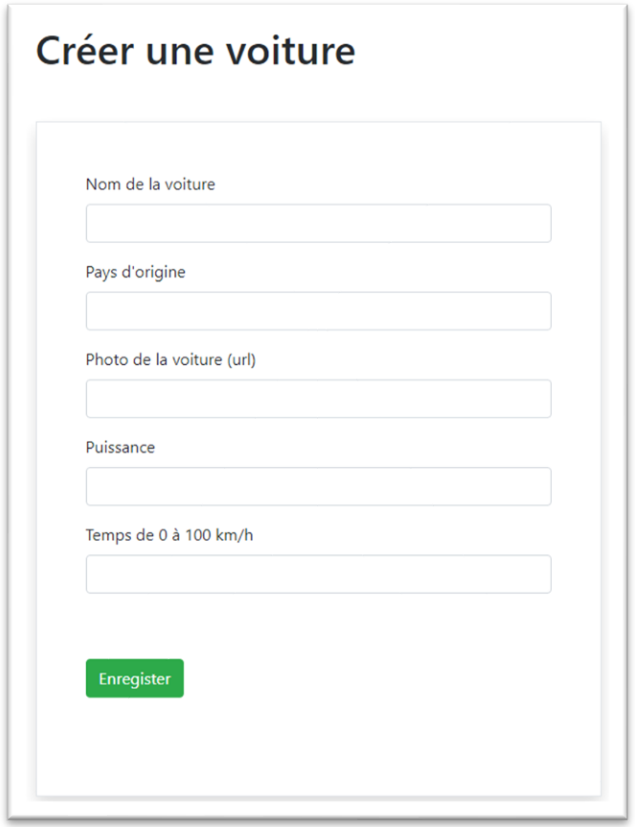
Adapter à la class Car

Simple formulaire

Sans attr action ni method

```html
<h1 class="my-5">
  Créer une voiture
</h1>
<div class="col-6 border shadow my-5 p-5">
  <form >
    <div class="form-group">
      <label for="name">Nom de la voiture</label>
      <input type="text" class="form-control" id="name">
    </div>
    <div class="form-group">
      <label for="pays">Pays d'origine</label>
      <input type="text" class="form-control" id="pays">
    </div>
    <div class="form-group">
      <label for="coverImage">Photo de la voiture (url)</label>
      <input type="text" class="form-control" id="coverImage">
    </div>
    <div class="form-group">
      <label for="power">Puissance</label>
      <input type="number" class="form-control" id="power">
    </div>
    <div class="form-group">
      <label for="perf">Temps de 0 à 100 km/h</label>
      <input type="number" class="form-control" id="perf">
    </div>
    <button class="btn btn-success my-5">Enregister</button>
  </form>
</div>
```

Permettre a Angular récupérer le formulaire



Mot Clés Angular


```html
<h1 class="my-5">
  Créer une voiture
</h1>
<div class="col-6 border shadow my-5 p-5">
  <form >
    <div class="form-group">
      <label for="name">Nom de la voiture</label>
      <input type="text" class="form-control" id="name">
    </div>
    <div class="form-group">
      <label for="pays">Pays d'origine</label>
      <input type="text" class="form-control" id="pays">
    </div>
    <div class="form-group">
      <label for="coverImage">Photo de la voiture (url)</label>
      <input type="text" class="form-control" id="coverImage">
    </div>
    <div class="form-group">
      <label for="power">Puissance</label>
      <input type="number" class="form-control" id="power">
    </div>
    <div class="form-group">
      <label for="perf">Temps de 0 à 100 km/h</label>
      <input type="number" class="form-control" id="perf">
    </div>
    <button class="btn btn-success my-5">Enregister</button>
  </form>
</div>
```

1er dans chaque input

```
<div class="form-group">
  <label for="name">Nom de la voiture</label>
  <input type="text" class="form-control" id="name" name="name" ngModel>
</div>
<div class="form-group">
  <label for="pays">Pays d'origine</label>
  <input type="text" class="form-control" id="pays" name="pays" ngModel>
</div>
```

```
name="coverImage" ngModel>
```

```
name="power" ngModel>
```

```
name="perf" ngModel>
```

MDN Web Docs
moz://a

<input>

```
$_POST['name']
```

name	Le nom du champ qui sera rattaché à la donnée envoyée via le formulaire.

API > @angular/forms

NgModel DIRECTIVE

Creates a `FormControl` instar

2e event (ngSubmit)

```
<form (ngSubmit)="onSubmit()" >
  <div class="form-group">
```

On écoute event ngSubmit
(directive NgForm)

```
<button type="submit"
```

On passe le button en type submit

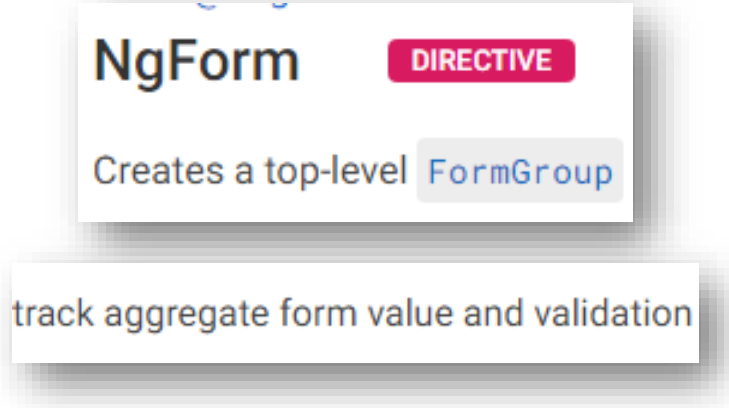
```
onSubmit(){
  console.log("Ok c'est simple");
}
```

On test , c'est good !

3e variable de référence locale

```
<form (ngSubmit)="onSubmit(myForm)" #myForm="ngForm">
```

On créer une ref locale (#truc #f #variableName)
Sorte de query selector, cible le formulaire ici
On lui attribue la directive ngForm… la doc →




```
onSubmit(myForm: NgForm){
  console.log(myForm.value);
}
```

On affiche les valeurs récupérer…FIN

Pour la ref locale si besoin
input #truc
{{truc.placeholder}}

Résultat

Créer une voiture

Nom de la voiture

Ford Mustang GT500

Pays d'origine

USA

Photo de la voiture (url)

https://auto.cdn-rivamedia.com/photos/annonce

Puissance

760

Temps de 0 à 100 km/h

3.5

Enregistrer

```
new-car.component.ts:26
{name: "Ford Mustang GT500", pays: "USA",
coverImage: "https://auto.cdn-rivamedia.c
om/photos/annonce/big/…lby-gt500-v8-52l-s
upercharged-760hp-118218052.jpg", power:
760, perf: 3.5}
```

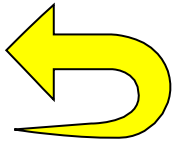
Class

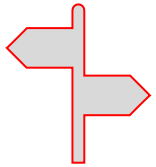
On peut faire quoi ensuite ?

Mini Exo – Medium – Faire l'ajout et la redirection

Class

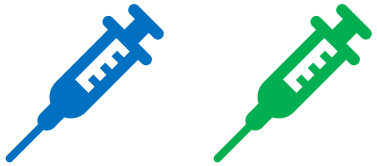
Utiliser le model pour créer une voiture





```
console.log(myForm.value['name'],
            myForm.value['perf'])
```

Dans le data service
Créer une méthode qui ajoute une voiture au tableau des voitures



Après l'ajout de la voiture rediriger l'utilisateur vers la liste des voitures
Utiliser le service de router comme pour la 404 (cf class)

Mini Exo – Medium – Correction

```
onSubmit(myForm: NgForm){
  const car = new Car(myForm.value['name'],
                      myForm.value['pays'],
                      myForm.value['coverImage'],
                      myForm.value['power'],
                      myForm.value['perf']);
  this.data.addCar(car);
  this.router.navigate(['cars']);
}
```

```
addCar(car:Car){
  this.cars.push(car);
}
```

```
export class NewCarComponent implements On

  constructor(private data : DataService,
              private router: Router) { }
```

Amélioration légère de la méthode Template-Driven

RequiredValidator DIRECTIVE

A directive that adds the `required` validator

Description

Adding a required validator using template-driven forms

```
<input name="fullName" ngModel required>
```

Required Angular pas html

`invalid: boolean` **Read-Only**

A control is `invalid` when its `status` is `INVALID`.

See also:

- `status`

NgForm DIRECTIVE

Creates a top-level `FormGroup`

Form is Valid

New Car Template

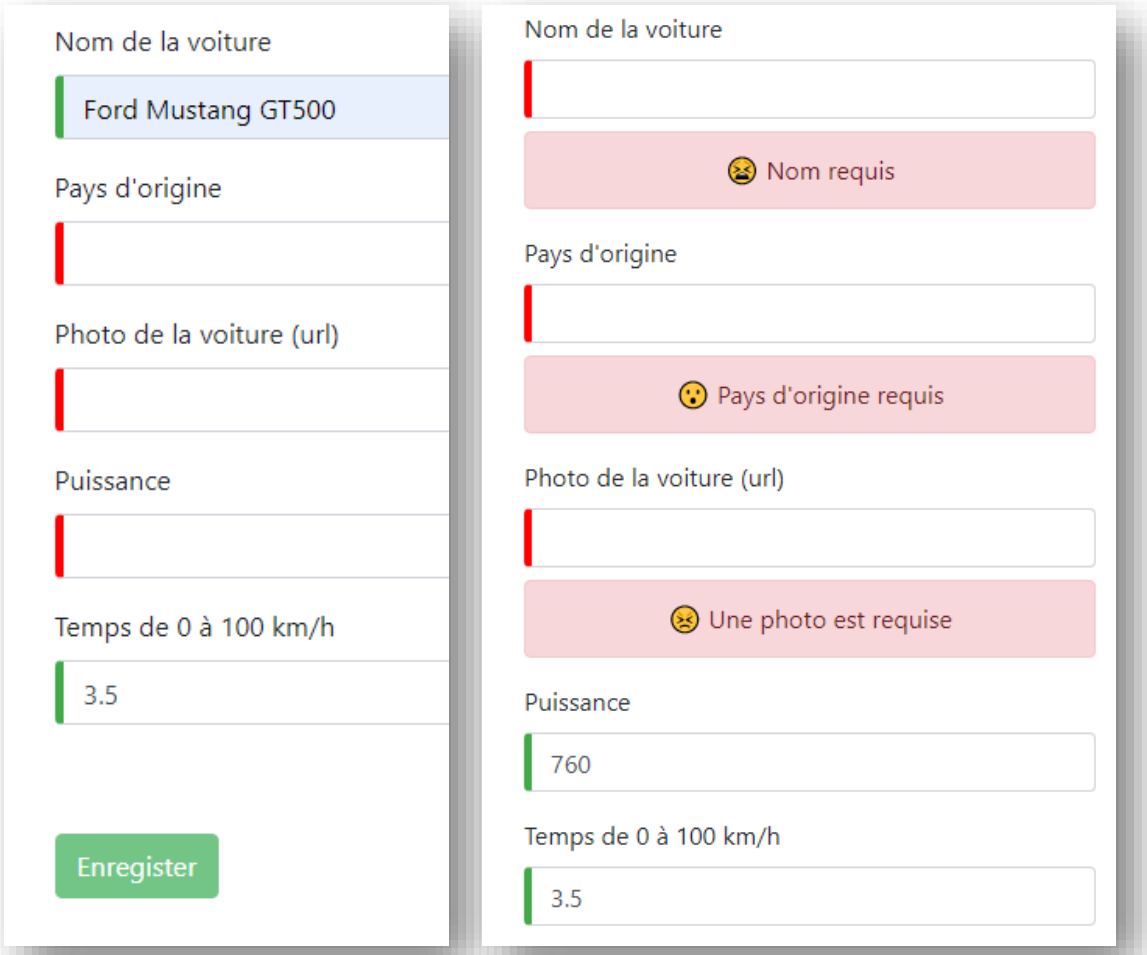
```
name="name" ngModel required>
```

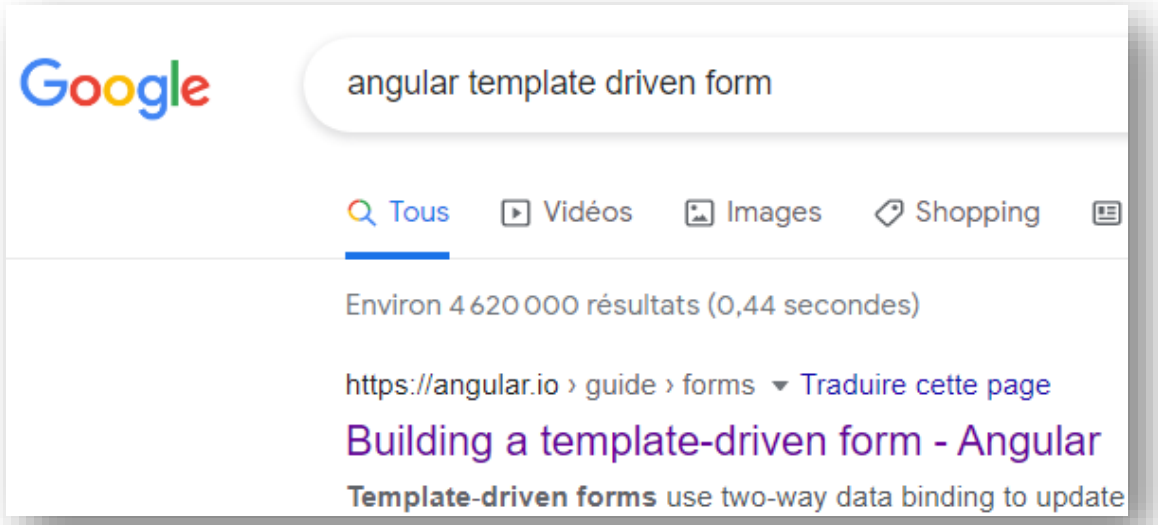
Dans tous les inputs

```
[disabled]="myForm.invalid">Enregister</button>
```

Sur le bouton

Mini Exo Easy - Utiliser la DOC Angular





Reproduire ce comportement
en consultant la doc
Champs valid en vert sinon en rouge,
Puis gestion des messages erreurs

Correction – Tout simplement dans le CSS

```css
.ng-invalid:not(form)  {
    border-left: 5px solid red; /* red */
  }
.ng-valid[required], .ng-valid.required  {
    border-left: 5px solid #42A948; /* green */
  }
```

Fonctionne avec la directive required – supprimer en une du input dans le template et constater

Ps: Inutile d'ajouter la class ng-valid ou autres dans le template, s'ajoute automatiquement

Correction – Message erreur

```
<div class="form-group">
  <label for="pays">Pays d'origine</label>
  <input type="text" class="form-control" id="pays" name="pays" ngModel required #pays="ngModel">
  <div [hidden]="pays.valid || pays.pristine" class="alert alert-danger my-2 text-center">
    😮 Pays d'origine requis
  </div>
</div>
```

Faire pareil pour tous les inputs

😎

N'apparait pas la 1er fois, sinon trop agressif, mais si l'utilisateur vide le champ et l'oubli, Oui

On aurait pu laisser le champs power et perf
Non requis…On aurait pu…

```
export class Car {

    constructor(public name:string,
                public pays:string,
                public coverImage:string,
                public power?:number,
                public perf?:number){}

}
```

Rappel:

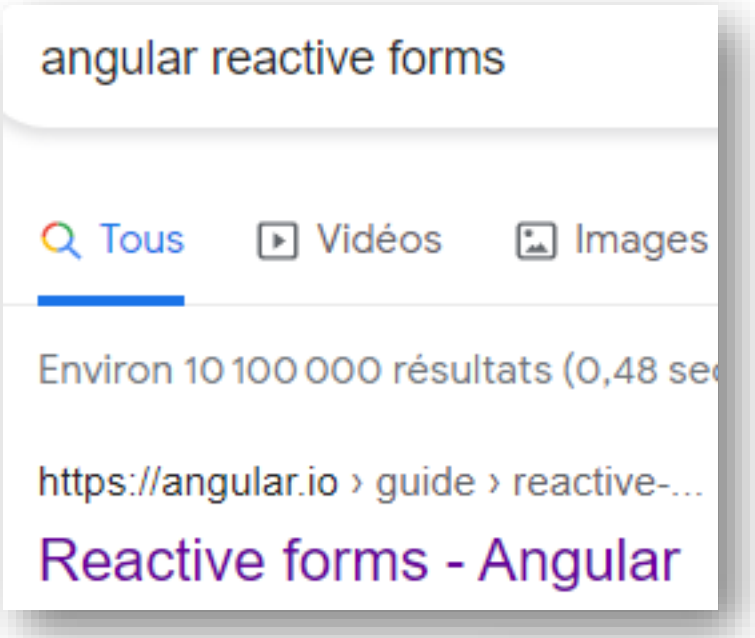
Constructeur syntaxe shortcut grace à typescript

Et le ? Signifie propriété de l'objet optionnel

Form Template Driven
Formulaire Simple

Manière simple mais moins poussée

Reactive Forms



ANGULAR

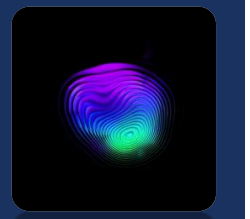
Overview of reactive forms

Reactive forms differ from template-driven forms in distinct ways.

Un peu plus de contrôle et possibilités async pour plus tard…

AppModule

TS app.module.ts



Prévenir notre projet qu'on va utiliser le ReactiveFormsModule

```
imports: [
  BrowserModule,
  AppRoutingModule,
  FormsModule,
  ReactiveFormsModule
],
```

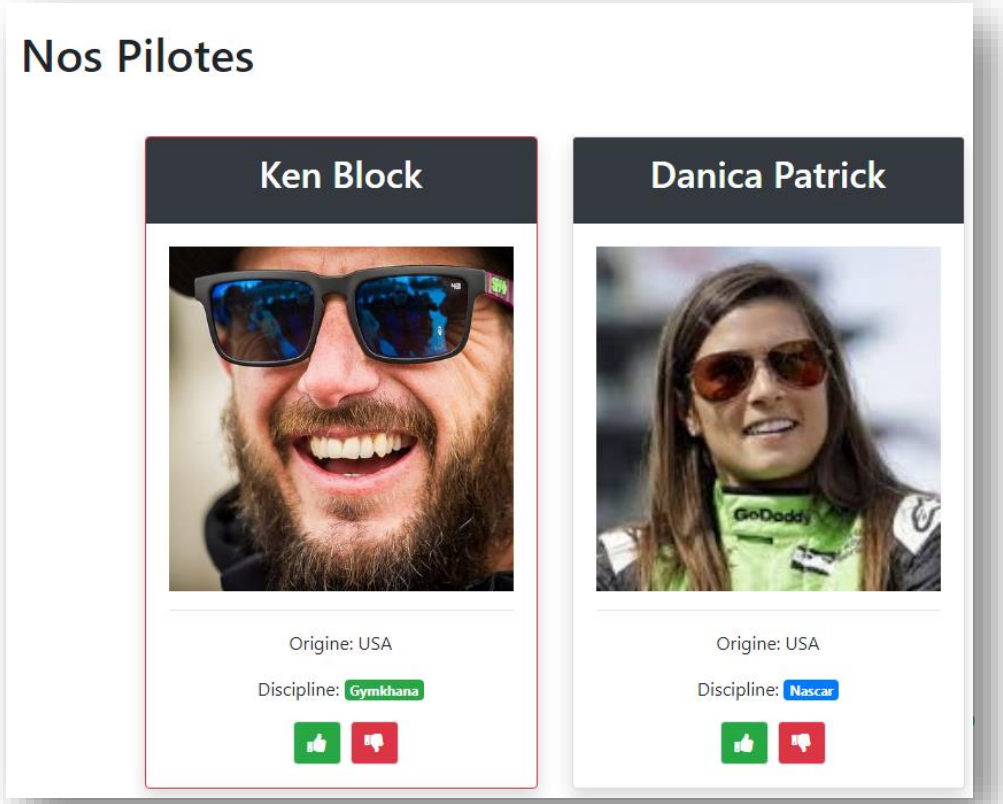
```
import { ReactiveFormsModule } from '@angular/forms';
```



```
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
```

Préparation du terrain avant

Utilisation du ReactiveForm sur les Drivers



Quelles sont les étapes ? 🤔

Liste des étapes avant le formulaire

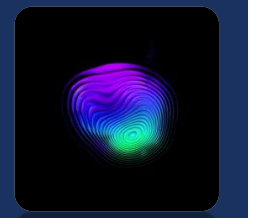
Ajouter un button qui renvoie vers le component de création (futur form)
Créer le component new-driver dans drivers-list et sa route

On créer un model Driver basé sur les datas drivers du data-services
Avec les likes (note) par défaut à 0

On utilise la class pour créer un pilote en dur et on le rajoute aux drivers

On crée la méthode qui addDrive(driver: Driver) dans le data-service
Et on la test dans le component qui l'utilisera avec un formulaire

Correction



Ajouter un button qui renvoie vers le component de création (futur form)
Créer le component new-driver dans drivers-list et sa route

```
<h1 class="my-5">
    Nos Pilotes
</h1>
<a  class="badge badge-success shadow my-5 p-2" routerLink="/new-driver">+ Ajouter un(e) pilote</a>
```

```
ng g c drivers-list/new-driver
```

```
{path: 'new-car', component: NewCarComponent},
{path: 'drivers', component: DriversListComponent},
{path: 'new-driver', component: NewDriverComponent},
```

On test on est goodyear

new-driver works!

Correction

On créer un model Driver basé sur les datas drivers du data-services
Avec les likes (note) par défaut à 0

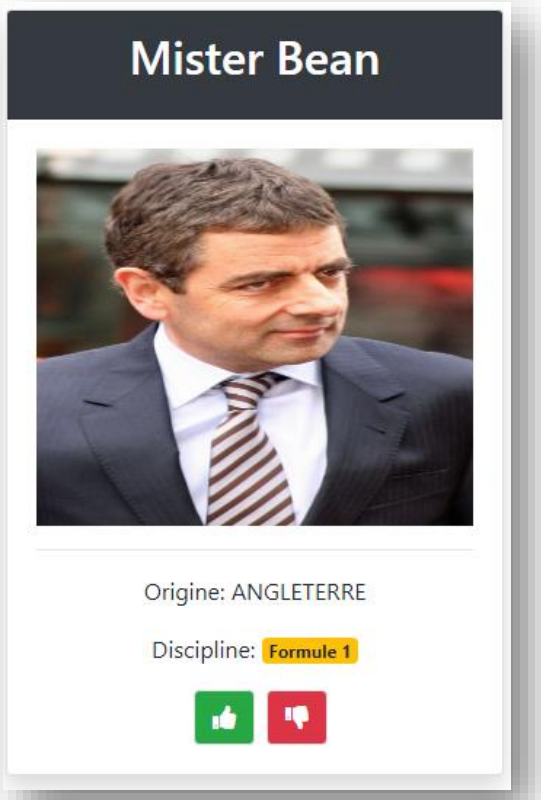
```
export class Driver {

    constructor(public fullName:string,
                public pays:string,
                public coverImage:string,
                public category?:string,
                public likeIts:number = 0){}

}
```

Correction

On utilise la class pour créer un pilote en dur et on le rajoute aux drivers

```
export class DataService {

  voiture: Car = new Car("peugeot L750 r","France","https:

  pilote: Driver = new Driver("mister bean","angleterre","
      You, 6 months ago • Service - TP Service
```

```
  {
    fullName: "erica enders",
    pays: "usa",
    coverImage: "./assets/img/
    category: "drag",
    likeIts: 0
  },
  this.pilote
;
```



On test toujours

Correction

On crée la méthode qui addDrive(driver: Driver) dans le data-service
Et on la test dans le component qui l'utilisera avec un formulaire

```
addDriver(driver:Driver){
  this.drivers.push(driver);
}
```

TS data.service.ts

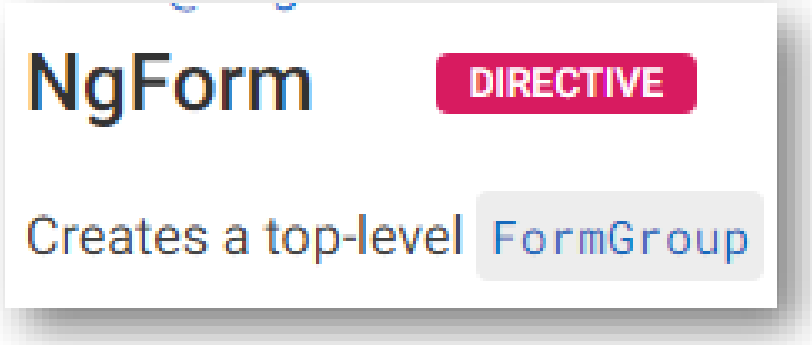
```
export class NewDriverComponent implements

  constructor(private data: DataService) {

  pilote: Driver = new Driver("michael kni

  ngOnInit(): void {
    this.data.addDriver(this.pilote);
  }
```

Donc plus qu'a
récupérer les données
via un formulaire

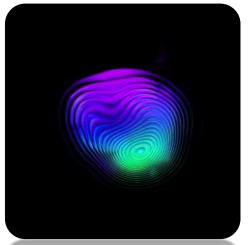
Forms

Template Driven

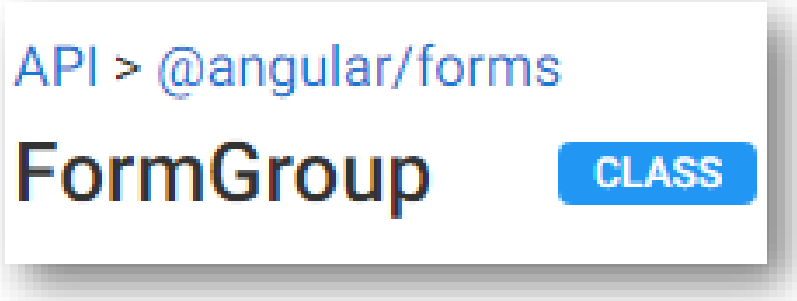
On code principalement dans le template et utilise des mots clé pour que Angular lise et récupère



Reactive



On construit le formulaire dans la class et on le connecte au template
On utilise FormGroup et cie



Ie Etape

```
import { FormBuilder, FormGroup } from '@angular/forms';
```

Une props FormGroup
Et le service FormBuilder

```
driverForm: FormGroup;

constructor(private data: DataService,
            private formBuilder: FormBuilder) { }
```

La doc 2 manières de faire

src/app/profile-editor/profile-editor.component.ts (instances)

```ts
profileForm = new FormGroup({
  firstName: new FormControl(''),
  lastName: new FormControl(''),
  address: new FormGroup({
    street: new FormControl(''),
    city: new FormControl(''),
    state: new FormControl(''),
    zip: new FormControl('')
  })
});
```

```ts
profileForm = this.fb.group({
  firstName: [''],
  lastName: [''],
  address: this.fb.group({
    street: [''],
    city: [''],
    state: [''],
    zip: ['']
  }),
});
```

fb = formBuilder (on peut mettre nimp)

FormGroup un groupe de donnée FormControl un champ

2e Etape Donc

```
ngOnInit() {
  this.createForm();
}

createForm(){
  this.driverForm = this.formBuilder.group({
    fullName: [''],
    pays: [''],
    coverImage:[''],
    category:['']
  })
};
```

ngOnInit pour que la méthode charge à init

On stock dans le FormGroup, les FormControl
FormControl = champ individuel voulu

Le driverForm contiendra les données reçues (valeurs)

3e Etape Enfin

```
onSubmit(){
  const formValue = this.driverForm.value;
  const driver = new Driver(
    formValue['fullName'],
    formValue['pays'],
    formValue['coverImage'],
    formValue['category']
  )
  console.log(driver);
}
```

On créer une variable pour alléger la syntaxe

Puis on créer notre objet Driver grace au formulaire

Avant d'aller plus loin on test

C'est tout pour la partie CLASS

Coté Template 1er

```
export class NewDriverComponent

  driverForm: FormGroup;
```

Reactive form

```
<form (ngSubmit)="onSubmit()" [formGroup]="driverForm">
```

Avant Template form

```
<form (ngSubmit)="onSubmit(myForm)" #myForm="ngForm">
```

On se lie (bind) à la propriété Angular formGroup qui attend un objet de type formGroup

On utilise formGroup et plus ngForm donc onSubmit ne reçoit plus de paramètre

Coté Template 2eme

API > @angular/forms

FormControlName DIRECTIVE

Reactive form

```
<div class="form-group">
  <label for="name">Nom complet du Pilote</label>
  <input type="text" class="form-control" id="name" formControlName="fullName">
</div>
```

```
formControlName="pays">
```

```
formControlName="coverImage">
```

```
formControlName="category">
```

FormControlName prend le nom du groupe de contrôle donner dans la Class, ce input rempli cette ce champ

(ps: tout le reste c'est juste du html)

```
this.driverForm = this.formBuilder.group({
  fullName: [''],
```

Avant Template form

```
<div class="form-group">
  <label for="name">Nom de la voiture</label>
  <input type="text" class="form-control" id="name" name="name" ngModel required #name="ngModel">
</div>
```

validators

Coté Template 3eme et fin

Reactive form

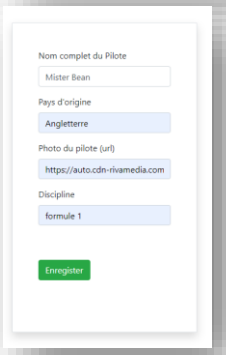
```
    <button type="submit" class="btn btn-success my-5"
            [disabled]="driverForm.invalid">
        Enregister
    </button>
</form>
```

```
driverForm: FormGroup;
```

```
this.driverForm = this.formBuilder.group({
```

On envoie
on est good

```
Driver {fullName: "Mist
coverImage: "https://au
nonce/big/…lby-gt500-v8
052.jpg", category: "fo
```



Avant Template form

```
[disabled]="myForm.invalid">
```

```
#myForm="ngForm">
```

Template Complet

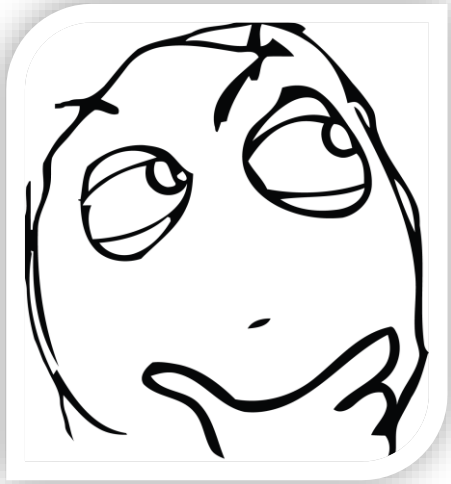
New-driver

```
<div class="col-6 border shadow  my-5 p-5">
    <form (ngSubmit)="onSubmit()" [formGroup]="driverForm">
        <div class="form-group">
            <label for="name">Nom complet du Pilote</label>
            <input type="text" class="form-control" id="name"
                formControlName="fullName">
        </div>
        <div class="form-group">
            <label for="pays">Pays d'origine</label>
            <input type="text" class="form-control" id="pays"
                formControlName="pays">
        </div>
        <div class="form-group">
            <label for="coverImage">Photo du pilote (url)</label>
            <input type="text" class="form-control" id="coverImage"
                formControlName="coverImage">
        </div>
        <div class="form-group">
            <label for="category">Discipline</label>
            <input type="text" class="form-control" id="category"
                formControlName="category">
        </div>
        <button type="submit" class="btn btn-success my-5"
            [disabled]="driverForm.invalid">
            Enregister
        </button>
    </form>
</div>
```

Class Complete
New-driver

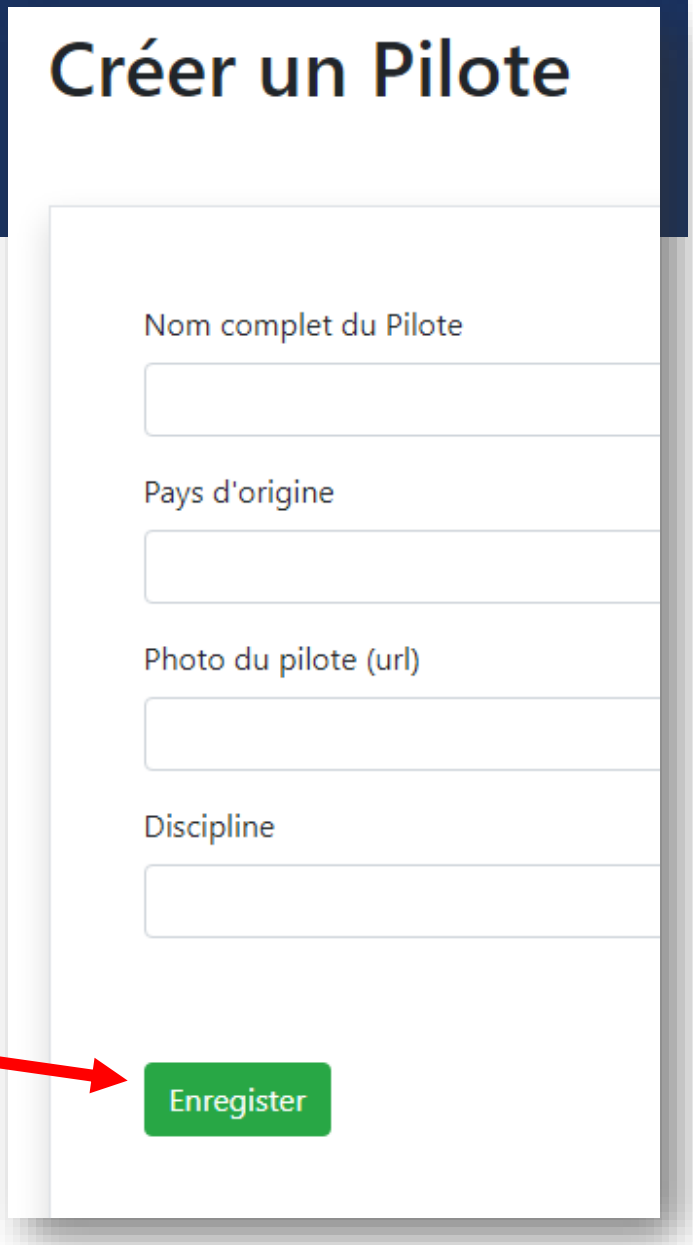
```
driverForm: FormGroup;
constructor(private data: DataService,
                    private formBuilder: FormBuilder) { }

ngOnInit() {
  this.createForm();
}
createForm(){
  this.driverForm = this.formBuilder.group({
    fullName: [''],
    pays: [''],
    coverImage:[''],
    category:['']
  })
};
onSubmit(){
  const formValue = this.driverForm.value;
  const driver = new Driver(
    formValue['fullName'],
    formValue['pays'],
    formValue['coverImage'],
    formValue['category']
  )
  console.log(driver);
}
```

Un soucis…Encore



Comment le bouton devrait être ?
Pourquoi il ne l'est pas ?

```
    <button type="submit" class="btn btn-success my-5"
            [disabled]="driverForm.invalid">
        Enregister
    </button>
</form>
```



Mise en place des conditions
VALIDATORS

API > @angular/forms

Validators CLASS

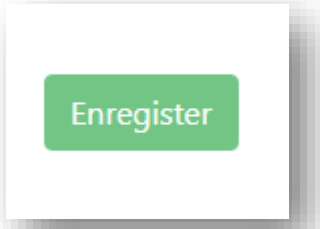
```
class Validators {
  static min(min: number): ValidatorFn
  static max(max: number): ValidatorFn
  static required(control: AbstractControl): ValidationErrors | null
  static requiredTrue(control: AbstractControl): ValidationErrors | null
  static email(control: AbstractControl): ValidationErrors | null
  static minLength(minLength: number): ValidatorFn
  static maxLength(maxLength: number): ValidatorFn
  static pattern(pattern: string | RegExp): ValidatorFn
  static nullValidator(control: AbstractControl): ValidationErrors | null
  static compose(validators: ValidatorFn[]): ValidatorFn | null
  static composeAsync(validators: AsyncValidatorFn[]): AsyncValidatorFn | null
}
```

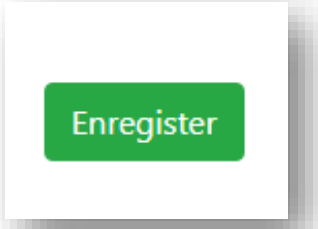
Voir la DOC

Form
I VALIDATORS ou Plusieurs [ ]

```
createForm(){
  this.driverForm = this.formBuilder.group({
    fullName: ['' , Validators.required ],
    pays: ['', [Validators.required, Validators.minLength(2)]],
    coverImage:['', Validators.required],
    category:['', Validators.required]
  })
};
```

Voir la DOC





Form
VALIDATORS.required ou via HTML

```
.ng-invalid:not(form)  {
    border-left: 5px solid red; /* red */
  }
.ng-valid[required], .ng-valid.required  {
    border-left: 5px solid #42A948; /* green */
  }
```

```
<input type="text" class="form-control" i
    formControlName="fullName" required>
```

Nom complet du Pilote

Amar BOUTAYEB

Nom complet du Pilote

Amar BOUTAYEB

← Sans le required dans le html

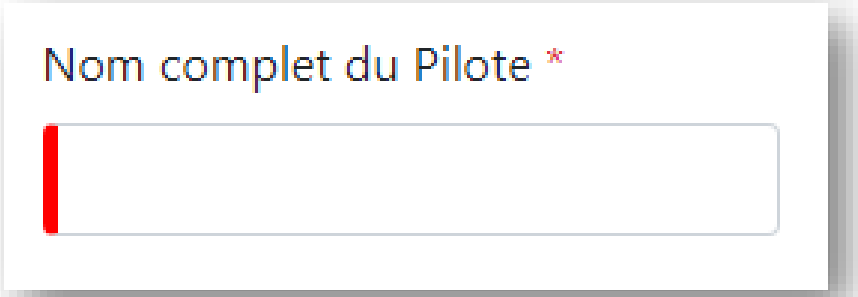
Nom complet du Pilote

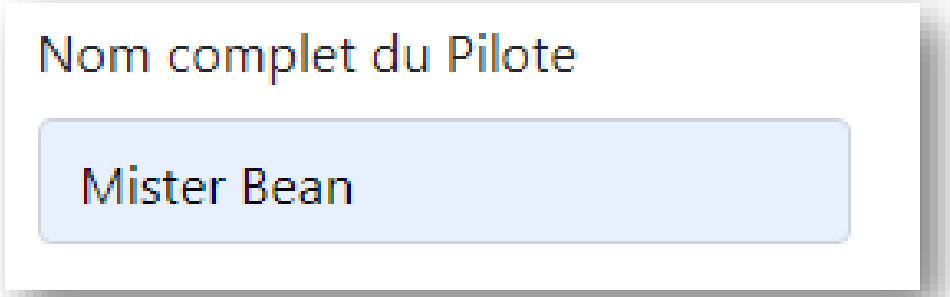
Pays d'origine

Photo du pilote (url)

Discipline

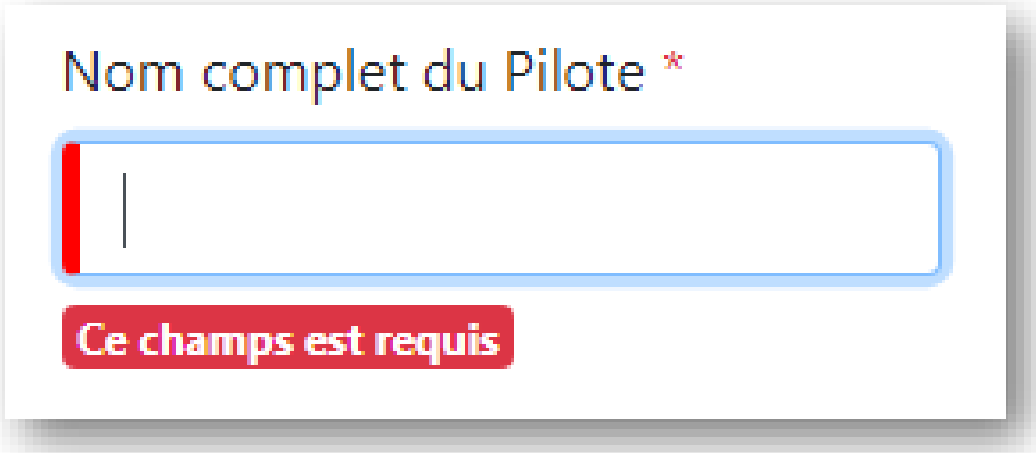
FormGroup
VALIDATORS et UX ++






```
<label for="name">Nom complet du Pilote

    <span class="text-danger" *ngIf="driverForm.get('fullName').errors &&
        driverForm.get('fullName').hasError('required')">
        *
    </span>

</label>
```

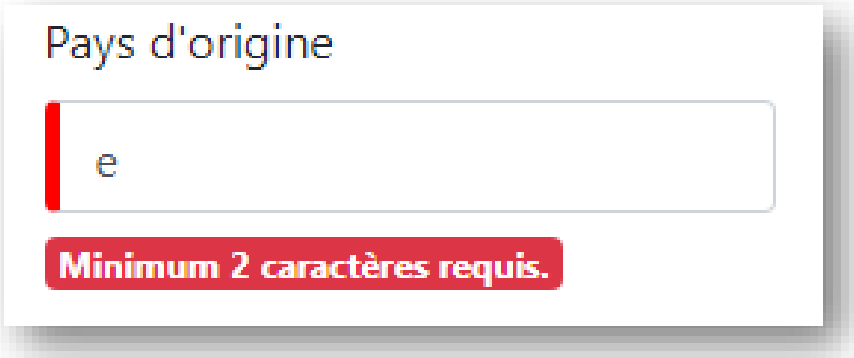
FormGroup
VALIDATORS et UX ++



Apparait si le user écris
puis supprime

```
<input type="text" class="form-control" id="name" formControlName="fullName" #fullName>

<small class="badge badge-danger my-2" *ngIf="driverForm.get('fullName').invalid &&
    driverForm.get('fullName').errors &&
    (driverForm.get('fullName').dirty || driverForm.get('fullName').touched)">
    Ce champs est requis
</small>
```

FormGroup
VALIDATORS et UX ++



Si le nom du pays est trop court

```
<input type="text" class="form-control" id="pays" formControlName="pays">

<small class="badge badge-danger my-2" *ngIf="driverForm.get('pays').hasError('minlength')">
    Minimum {{driverForm.get('pays').errors.minlength.requiredLength}} caractères requis.
</small>
```

FormGroup
VALIDATORS et UX ++

Cette syntaxe est possible aussi…

```
<small class="badge badge-danger my-2"
    *ngIf="driverForm.controls['coverImage'].invalid">
    Photo requis
</small>
```

On récupére bien le pilote, validation ok, ux ok , next ?

Nom complet du Pilote

Mister Bean

Pays d'origine

Angletterre

Photo du pilote (url)

https://auto.cdn-rivamedia.com

Discipline

formule 1

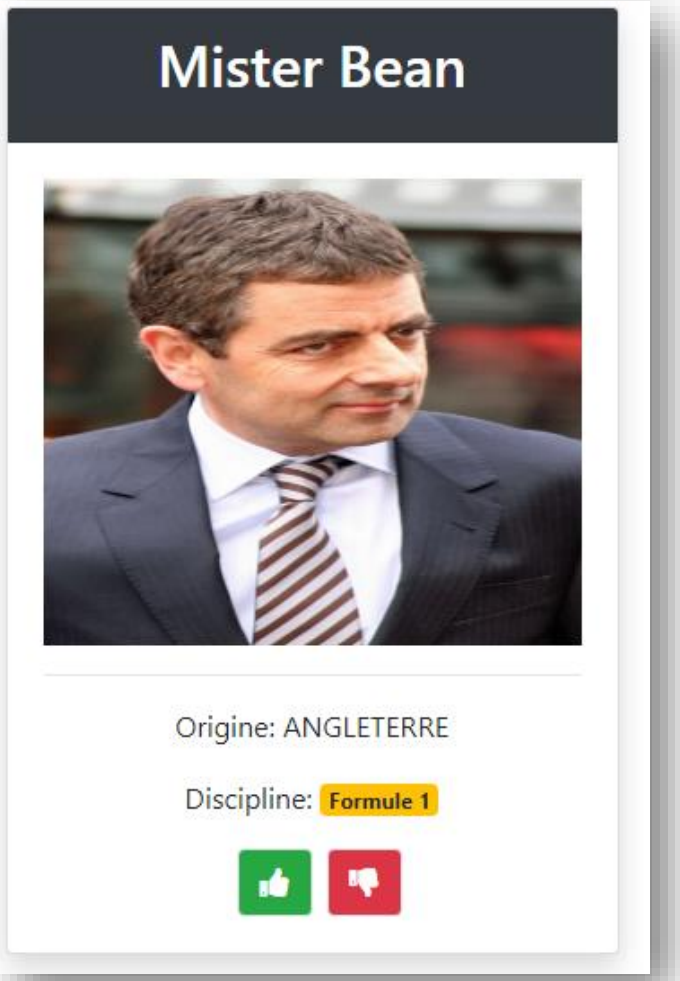
Enregister

...

```
new-driver.component.ts:36
Driver {fullName: "Mister Bean", pays: "Angletterre", coverImage: "https://auto.cdn-rivamedia.com/photos/annonce/big/…lby-gt500-v8-52l-supercharged-760hp-118218052.jpg", category: "formule 1", likeIts: 0}
```

```
console.log(driver);
```

Ajouter le driver au tableau…
Rediriger…



…

```
driverForm: FormGroup;
constructor(private data: DataService,
            private formBuilder: FormBuilder,
            private router: Router ) { }
```

```
onSubmit(){
  const formValue = this.driverForm.value;
  const driver = new Driver(
    formValue['fullName'],
    formValue['pays'],
    formValue['coverImage'],
    formValue['category']
  )
  this.data.addDriver(driver);
  this.router.navigate(['drivers']);
}       You, 7 hours ago • Presque Done
```

Mais si…
Quelle amélioration possible ?

Nom complet du Pilote

Mister Freeze

Pays d'origine

Norvége

Photo du pilote (url)

https://static.wikia.nocookie.net/b

Discipline

machin truc

Enregister

Rallye

Nascar

Formule 1

Gymkhana

Drag

…

Mister Freeze

Origine: NORVÉGE

Discipline: **Machin Truc**

Amélioration possible…

Nascar

Rallye

Formule 1

Drag

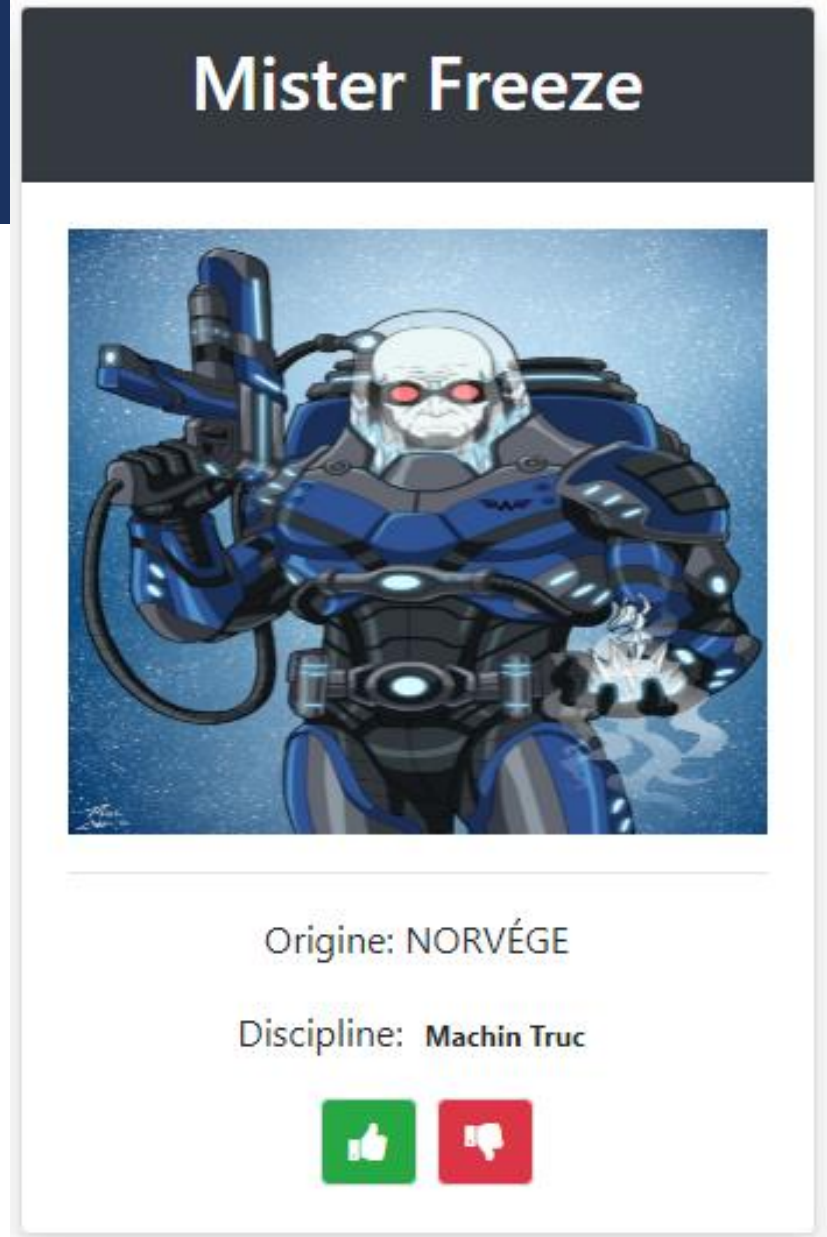
Gymkhana

Dans le template un badge par défaut…ça ne bride pas bcp…

Dans la class un valeur par défaut…Pas toujours vrai donc…

D'autres solutions (pattern, pas de choix, regex, validators…)

Une liste déroulante…



Mini exo Easidium

Nascar

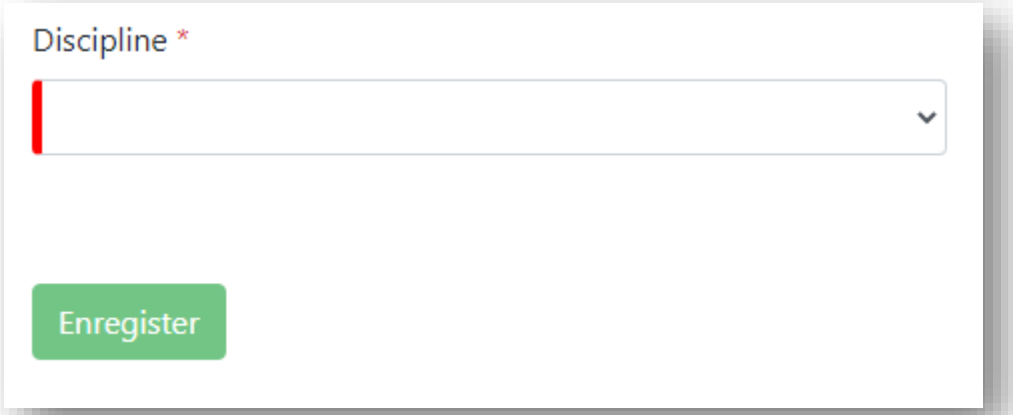
Rallye

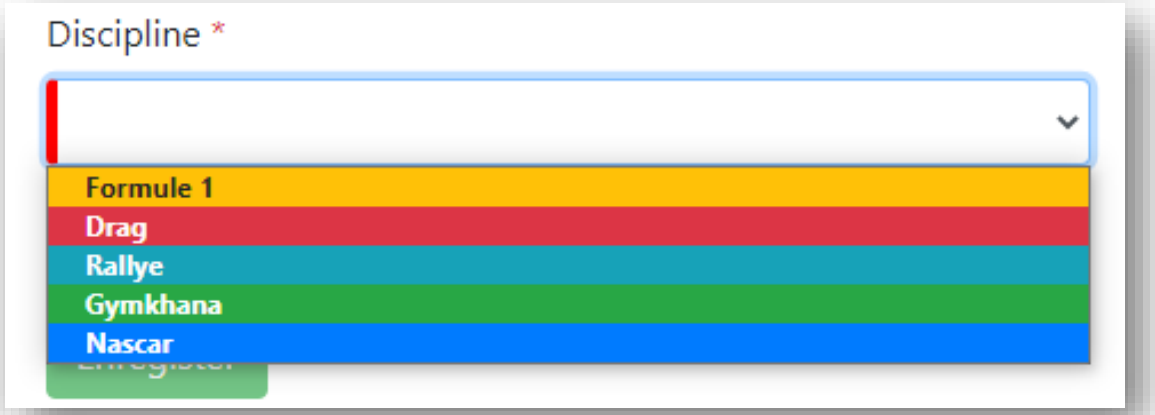
Formule 1

Drag

Gymkhana

Dans le dataservice stocker la liste des disciplines
Faire une méthode pour la récupérer
Se servir de ça pour faire une liste déroulante qui affiche ces options
Adapter le formulaire





CORRECTION

Dans le dataservice stocker la liste des disciplines
Faire une méthode pour la récupérer

```
@Injectable()
export class DataService {
```

```
categories: string[] = ["formule 1","drag","rallye","gymkhana","nascar"];
```

```
getAllCategories(){
  return this.categories;
}
```

CORRECTION

Se servir de ça pour faire une liste déroulante qui affiche ces options

Adapter le formulaire

```
driverForm: FormGroup;
categories:any;
constructor(private data: DataService,
            private formBuilder: FormBuilder,
            private router: Router ) {}

ngOnInit() {
  this.categories = this.data.getAllCategories();
  this.createForm();      You, 8 hours ago • Pres
}
```

CORRECTION

Adapter le formulaire

```
<select class="form-control" name="category" id="category" formControlName="category">
    <option class="badge" *ngFor="let category of categories" [value]="category" [ngClass]="{
        'badge-success': category == 'gymkhana',
        'badge-warning': category == 'formule 1',
        'badge-info': category == 'rallye',
        'badge-primary': category == 'nascar',
        'badge-danger': category == 'drag'}">      You, 2 minutes ago • Uncommitted changes

        {{ category | titlecase }}
    </option>
</select>
```

TP – Form – Service - Models

Créer un model user
Firstname lastName email motdepasse et une méthode getFullName
Formulaire pour créer un compte avec validators erreur ect
Et page de login logout qui utilise un AuthService qui renvoie un boolean
Pour la connexion, avec dans la navigation les boutons nécessaires et l'apparition ou non ce ces derniers
But du AuthService → Guard But du User → Auth (FireBase)

Bon courage