Angular 2+
Jour 4

RxJs : Observable / Observer

Pour réagir à des événements ou à des données de manière **asynchrone** (Http) **:**

Attendre qu'une tâche soit terminée avant de passer à la ligne de code suivante

Nous utilisons les **observable** et les **observer**.

RxJs : Observable

un **Observable** est un objet qui émet des informations auxquelles on souhaite réagir.

Dans le **service** ⟶ personneSubject =new **Subject**<Personne>();

Objectif : émettre des événements : méthode **next**()

⟶ this.personneSubject.**next**(this.personne)

RxJs : Observer

Code qui sera exécuté à chaque fois que **l'Observable** émet une information

pour observer l'obserbable on utilise la fonction **subscribe**() :

Grace à un objet de type **Subscription**

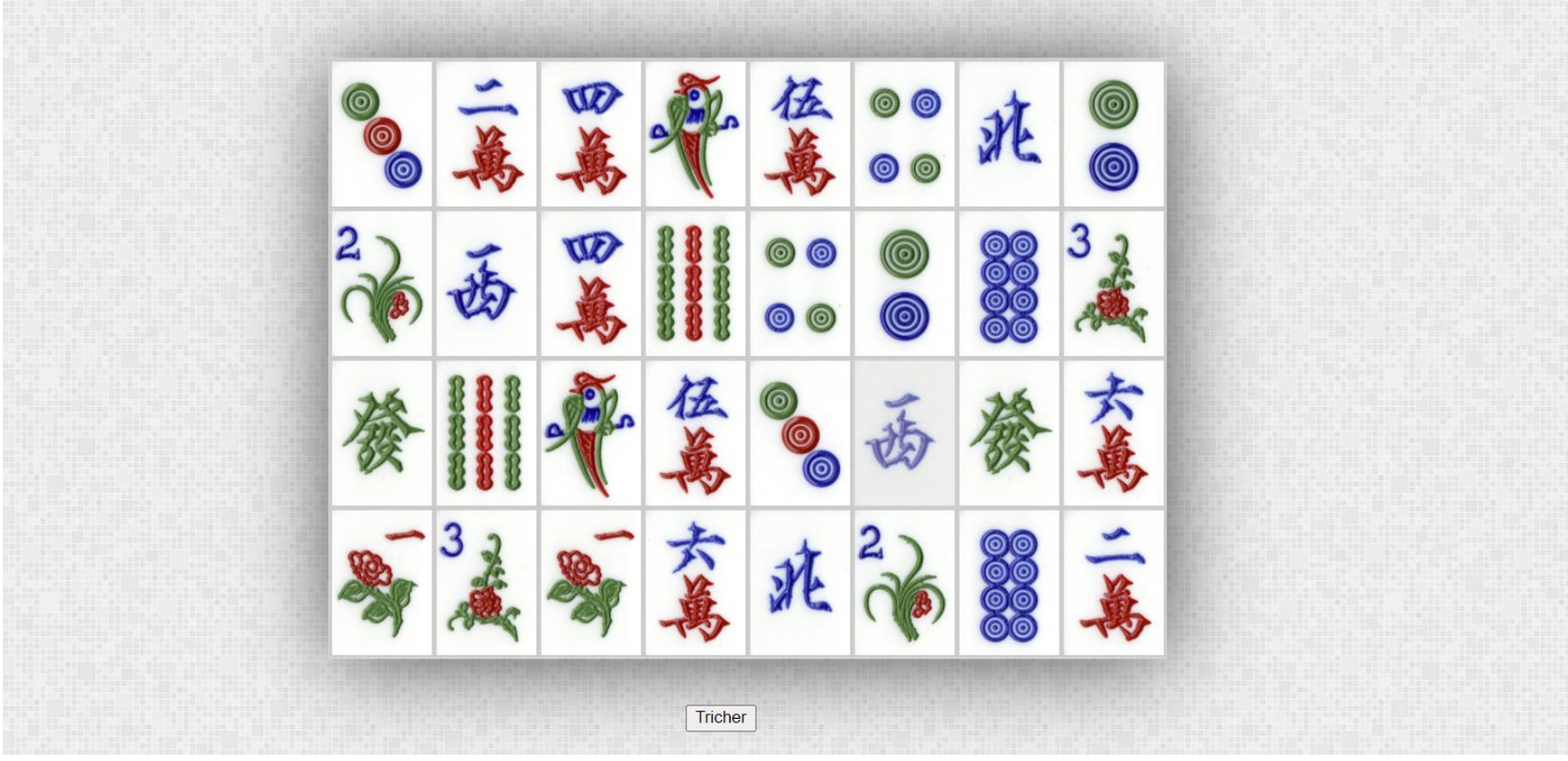
personneSubscription:**Subscription**

this.personneSubscription = this.personneService.personneSubject.**subscribe**(…)

assets

```
// Dans le component HTML

<img src= '' . /assets/img1.jpg ''>
```

assets : atelier



Les routes 1

```
// Dans app.module.ts  OU app-routing.module.ts

import { Routes } from '@angular/router';

let appRoutes:Routes = [
        { path: 'appareils', component: HomeComponent },
        { path: 'appareils/:id', component: AppareilViewComponent },
        { path: '', component: HomeComponent }
        { path: '**', component: HomeComponent }
];
//..
imports: [ RouterModule.forRoot(appRoutes) ];
```

Les routes 2

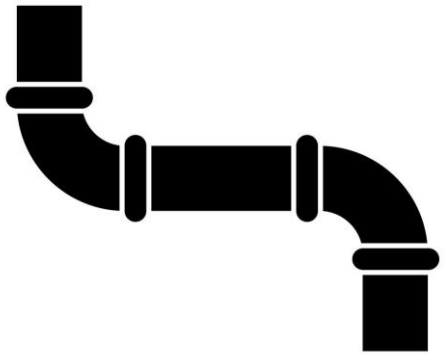
```
// dans le component root HTML

// routerLinkActive="active"  ⟶  class="active"


<a routerLink="/">Home</a>
<a routerLink="/contact">Contact</a>

<router-outlet></router-outlet>
```

Les routes 3

Component TS

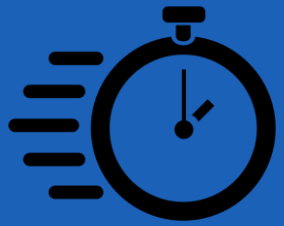
```
import { Router} from '@angular/router’;
import { ActivatedRoute } from '@angular/router';
import { Location } from '@angular/common’;

constructor(
                private route: ActivatedRoute,
                private router,
                private location: Location
) { }

this.router.navigate(['appareils’]);      // aller sur 1 page

this.location.back();  //go back

let id = +this.route.snapshot.paramMap.get('id’);  // get id
```

Les pipes



```
// https://angular.io/guide/pipes

@Component({
 selector: 'date-pipe',
 template: `<div>
   <p>Today is {{today | date}}</p>
   <p>Or if you prefer, {{today | date:'fullDate'}}</p>
   <p>The time is {{today | date:'h:mm a z'}}</p>
 </div>`
})
// Get the current date and time as a date-time value.
export class DatePipeComponent {
  today: number = Date.now();
}
```

Les promises



```
// code qui ne marche pas ...
function getInfo(){
        setTimeout(function(){
            return 'hello';
         }, 1000);
}

Let x= getInfo();
Console.log(x);
```

```
function getInfo2(){
                return new Promise(
                                (resolve, reject) =>
                                {
                                                setTimeout(function(){
                                                resolve("hello");
                                }, 1000);
                });
}
```

```
getInfo2().then(
                                (info) => {
                                console.log(info);
});
```

Form builder dans le component TS

```
// app.module.ts -> ReactiveFormsModule

 userForm: FormGroup;
 constructor(private formBuilder: FormBuilder){}

initForm() { // add to ngOnInit
   this.userForm = this.formBuilder.group({
     firstName: '',
     lastName: ''
   });
  }

initForm() {
   this.userForm = this.formBuilder.group({
     firstName: ['', Validators.required],
     lastName: ['', Validators.required]
});
}
```

```
onSubmitForm() {
   let formValue = this.userForm.value;
   let newUser = new User(
     formValue['firstName'],
     formValue['lastName']
   );
// envoyer au service
//changer de page
}
```

Form builder dans le component HTML

```
<form [formGroup]="userForm" (ngSubmit)="onSubmitForm()“>

  <div class="form-group">
    <label for="firstName">Prénom</label>
    <input type="text" id="firstName" class="form-control"  formControlName="firstName">
  </div>
  <div class="form-group">
    <label for="lastName">Nom</label>
    <input type="text" id="lastName" class="form-control" formControlName="lastName">
  </div>

  <button type="submit" class="btn btn-primary">Soumettre</button>

  <!--<button type="submit" class="btn btn-primary" [disabled]="userForm.invalid">Soumettre</button>-->
</form>
```