

Les fonctions constructeur

Fonction constructeur d'objet sans prototype, l'inconvénient c'est que a chaque fois que l'on créer une instance, on instancie également la fonction `this.present()`

```
1 function Personne(name, age){
2   this.name = name
3   this.age = age
4   this.present = () => {console.log("Hello my name is "+ this.name)}
5 }
6
7
8 let matt = new Personne("Matt", 32);
9 let franck = new Personne("Franck", 21);
10
11 console.log(matt);
12 console.log(franck);
13
14
```



(index):77

▼ Personne {name: "Matt", age: 32, present: f} ⓘ

- age: 32
- name: "Matt"
- ▶ present: () => {console.log("Hello my name is "+ this.
- ▶ __proto__: Object

(index):78

▼ Personne {name: "Franck", age: 21, present: f} ⓘ

- age: 21
- name: "Franck"
- ▶ present: () => {console.log("Hello my name is "+ this.
- ▶ __proto__: Object

>

Fonction constructeur d'objet avec prototype, l'avantage c'est que la méthode `this.present()` est créée qu'une seule fois dans le prototype de l'objet `Person` et est donc utilisable par toutes les instances de celle-ci.

```
1 function Person(name, age){
2   this.name = name;
3   this.age = age;
4 }
5
6 Person.prototype.present = function(){
7   console.log("Hello my name is " + this.name);
8 }
9
10
11 let matt = new Person("Matt", 32);
12 let franck = new Person("Franck", 21);
13
14 console.log(matt);
15 console.log(franck);
16
17
```



```
< undefined
  ▼ Person {name: "Matt", age: 32} ⓘ (index):80
    age: 32
    name: "Matt"
    ▼ __proto__:
      ► present: f ()
      ► constructor: f Person(name, age)
      ► __proto__: Object
  ▼ Person {name: "Franck", age: 21} ⓘ (index):81
    age: 21
    name: "Franck"
    ▼ __proto__:
      ► present: f ()
      ► constructor: f Person(name, age)
      ► __proto__: Object
> |
```