# JAVASCRIPT NOTION D'OBJET

#### LES PARADIGMES DE PROGRAMMATION

- La programmation procédurale
- La programmation fonctionnelle
- La programmation orientée objet

Un paradigme de programmation, c'est une méthodologie que l'on va suivre pour construire son code. Il s'agit plus d'une façon de penser, d'organiser son travail.

#### LA PROGRAMMATION PROCÉDURALE

La manière *classique* de coder, c'est un enchainement des procédures de haut en bas.

#### LA PROGRAMMATION FONCTIONNELLE

On donne une plus grande importance sur les fonctions *pures*, l'immutabilité des données...

Et on verra ça plus tard

# LA PROGRAMMATION ORIENTÉE OBJET (POO)

On réfléchit son code autour d'une entité, qui possède ses propriétés et ses méthodes.

## AVANT DE VOIR LA POO, QUELQUES RÉVISIONS SUR L'OBJET

```
// Création d'un objet film
let film = {
   titre: "Jurassik Park",
   realisateur: "Steven Spielberg",
   annee: 1993,
   serie: ["Jurassik Park", "Le Monde perdu : Jurassic Park", .
   presentation : ??? // TODO
}
film.titre // "Jurassik Park"
film['annee'] // 1993
film.presentation // "Jurassik Park, réalisé par Steven Spielb
```

#### UN PEU DE VOCABULAIRE

titre, realisateur, annee et serie sont des propriétés de l'objet film

presentation est une méthode de l'objet film

```
document.querySelector('#monBouton')
```

querySelector est une méthode de l'objet document

# EST-CE QUE LA VARIABLE HELLO EST UN OBJET ?

let hello = "Bonjour"

#### POURTANT...

```
let hello = "Bonjour"
hello.length // 7
hello.toUpperCase() // "BONJOUR"
```

# POUR CRÉER UNE *PRIMITIVE*, TROIS FAÇONS DE FAIRE:

```
// Littérale
let hello = "Bonjour" // "Bonjour"

// Constructeur
let hello = String("Bonjour") // "Bonjour"

// new + Constructeur
let hello = new String("Bonjour") // un objet de type String
```

#### **OBJET DE TYPE STRING**

```
let hello = new String("Bonjour")

// à tester dans une console

// hello

// String {"Bonjour"}

// => __proto__: String

// =>anchor: f anchor()

// ...
```

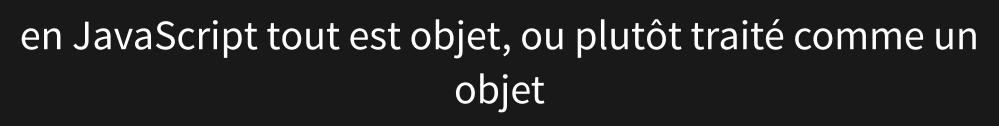
### alors pourquoi en littérale, on a aussi accès à des propriétés et des méthodes ?

```
let hello = "Bonjour"
hello.toUpperCase()
```

#### **AUTO-BOXING**

```
let hello = "Bonjour"
```

À l'exécution de hello.toUpperCase(), JS va caster "hello" en new String(hello). On accède donc à la méthode demandée, puis l'objet est supprimé.



#### LE PROTOTYPE

```
let a = {}
a // a est vide... vraiment ?
a.__proto__
```

### ET COMME TOUT EST OBJET

```
let b = 1
b. __proto__ // proto de type Number()
b. __proto__ . __proto__ // proto de type Object()
```

Il s'agit là de la notion d'héritage de JavaScript. b est une variable de type **Number** qui hérite donc de ses propriétés et de ses méthodes. Cette variable hérite également des propriétés et des méthodes du type **Object** 

Cette notion d'héritage est un des principes de la programmation orientée objet.

Généralement les langages qui utilisent la POO le font à traver le système des classes (Java, C...).

JavaScript a la particularité de baser sa POO avec les prototypes.

C'est seulement à partir de 2015, avec ES6, qu'un système de classe, similaire aux autres langages va apparaître.

### BIBLIOGRAPHIE / RESSOURCES

- https://developer.mozilla.org/fr/docs/Learn/JavaScrip
- https://medium.com/@alexandre.cibot/parlons-un-perlongered
   e10fb31029f5