



# Université Assane Seck de Ziguinchor

## JavaScript – Les objets

UFR ST – Département Informatique  
Licence 2 Ingénierie Informatique 2021/2022

**Marie NDIAYE**

# LOO basés sur les classes vs LOO basés sur les prototypes

---

- ▶ Il existe deux grands types de langages orientés objet : ceux basés sur les classes, et ceux basés sur les prototypes.
- ▶ La majorité des langages orientés objets sont basés sur les classes.
- ▶ Les langages objets basés sur les classes et ceux basés sur les prototypes fonctionnent différemment :
  - ▶ Dans les langages orientés objet basés sur les classes, tous les objets sont créés à partir de classes et vont hériter des propriétés et des méthodes définies dans la classe.
  - ▶ Dans les langages orientés objet utilisant des prototypes comme le JavaScript, tout est objet et il n'existe pas de classes et l'héritage va se faire au moyen de prototypes.

# Les objets en JavaScript

---

- ▶ EN JavaScript, les objets sont aussi des variables. Mais ils peuvent contenir de nombreuses valeurs.
- ▶ Création d'un objet
  - ▶ `const voiture = {type:"Fiat", model:"500", couleur:"white"};`
  - ▶ `const personne = {  
 prenom: "Mamadou",  
 nom: "Sène",  
 age: 50,  
 sexe: "masculin"  
};`
- ▶ Accès aux valeurs de l'objet
  - ▶ `personne.nom`
  - ▶ `personne["nom"]`

# Les méthodes

---

- ▶ Les objets peuvent aussi avoir des méthodes.
- ▶ Les méthodes sont des actions qui peuvent être effectuées sur des objets.
- ▶ Les méthodes sont stockées dans les propriétés en tant que définitions de fonction.

```
▶ const personne = {  
  prenom: "Mamadou",  
  nom: "Sène",  
  age: 50,  
  sexe: "masculin"  
  nomComplet : function() {  
    return this.prenom + " " + this.prenom;  
  }  
};
```

# Accès aux méthodes d'un objet

---

- ▶ L'accès à une méthode d'un objet se fait avec la syntaxe suivante :

- ▶ `nomObjet.nomMethode()`

- ▶ *Exemple*

- ▶ `nom = personne.nomComplet();`

- ▶ Si vous accédez à une méthode sans les parenthèses (), elle renverra la définition de la fonction :

```
nom = personne.nomComplet;
```



```
nomComplet : function() {  
    return this.prenom + " " + this.prenom;  
}
```

# Le mot clé `this`

---

- ▶ En JavaScript, le mot clé `this` fait référence à un objet.
- ▶ L'objet référencé dépend de la façon dont le mot clé `this` est utilisé :
  - ▶ Dans une méthode objet, `this` fait référence à l'objet.
  - ▶ Seul, cela fait référence à l'objet global.
  - ▶ Dans une fonction, cela fait référence à l'objet global.
  - ▶ Dans une fonction, en mode strict, ceci est indéfini.
  - ▶ Dans un événement, cela fait référence à l'élément qui a reçu l'événement.

# Les constructeurs d'objets

---

- ▶ Pour construire des objets à partir d'une fonction constructeur, nous allons devoir suivre deux étapes :
  - ▶ définir une fonction constructeur;
  - ▶ appeler ce constructeur en utilisant le mot clefs new.
- ▶ **Exemple : Définition et appel**
  - ▶ 

```
function Personne(prenom, nom, age, sexe) {  
    this.prenom = prenom;  
    this.nom = nom;  
    this.age = age;  
    this.sexe = sexe;  
}
```
  - ▶ 

```
const pere = new Personne("Mamadou", "SENE", 50,  
    "masculin");  
const mere = new Personne("Khady", "BA", 48,  
    "féminin");
```

# Propriétés et méthodes de fonction

---

- ▶ les fonctions sont en fait un type particulier d'objets en JavaScript !
- ▶ Comme tout autre objet, une fonction peut donc contenir des propriétés et des méthodes.
- ▶ Exemple
  - ▶ 

```
function Personne(prenom, nom, age, sexe) {  
    this.prenom = prenom;  
    this.nom = nom;  
    this.age = age;  
    this.sexe = sexe;  
    this.nomComplet = function(){  
        return this.prenom + " " + this.nom;  
    };  
}
```



# Ajout de propriétés et de méthodes

---

- ▶ Parfois, vous souhaitez ajouter de nouvelles propriétés (ou méthodes) à tous les objets existants d'un type donné.
- ▶ Parfois, vous souhaitez ajouter de nouvelles propriétés (ou méthodes) à un constructeur d'objet.
- ▶ Ajouter de propriétés
  - ▶ La propriété JavaScript `prototype` vous permet d'ajouter de nouvelles propriétés aux constructeurs d'objet.
  - ▶ `Personne.prototype.nationalite = "Sénégalaise";`
- ▶ Ajout de méthodes
  - ▶ La propriété JavaScript `prototype` permet également d'ajouter de nouvelles méthodes aux constructeurs d'objets.
  - ▶ 

```
Person.prototype.nomComplet = function() {  
    return this.prenom + " " + this.nom;  
};
```

# Un petit exemple complet (1 / 2)

---

## ► JavaScript

```
► function Personne(prenom, nom, age, sexe) {  
    this.prenom = prenom;  
    this.nom = nom;  
    this.age = age;  
    this.sexe = sexe;  
    this.nomComplet = function(){  
        return this.prenom + " " + this.nom;  
    };  
}  
  
► let pere = new Personne("Mamadou", "SENE", 50, "masculin");  
let mere = new Personne("Khady", "BA", 48, "féminin");  
  
► document.getElementById("p1").innerHTML =  
document.getElementById("p1").innerHTML + " "  
+pere.nomComplet() + " " + pere.age + " " + pere.sexe;  
document.getElementById("p2").innerHTML =  
document.getElementById("p2").innerHTML + " " +  
mere.nomComplet() + " " + mere.age + " " + mere.sexe;
```

# Un petit exemple complet (2/2)

---

## ► HTML

```
<!DOCTYPE HTML>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title></title>
</head>
<body>
  <p id = "p1"><b>Le père : </b> </p>
  <p id = "p2"><b>La mère : </b></p>
  <script type="text/javascript"
src="js/script.js"></script>
</body>
</html>
```

# Les classes en JavaScript

---

- ▶ JavaScript a, dans ses dernières versions, introduit un mot clef `class` qui va nous permettre de créer des architectures objets similaires à ce qu'on rencontre dans les LOO basés sur les classes.
- ▶ **Attention** : le JavaScript est toujours un langage orienté objet à prototypes et, en tâche de fond, il va convertir nos « classes » selon son modèle prototype.

# Création d'une classe

---

- ▶ Utilisez le mot clé `class` pour créer une classe et ajoutez toujours une méthode nommée `constructor(...)`.
- ▶ 

```
class Personne {  
    constructor(prenom, nom, age, sexe) {  
        this.prenom = prenom;  
        this.nom = nom;  
        this.age = age;  
        this.sexe = sexe;  
    }  
}
```
- ▶ Le constructeur est une méthode spéciale :
  - ▶ Elle doit avoir le nom exact "constructeur".
  - ▶ Elle est exécuté automatiquement lorsqu'un nouvel objet est créé.
  - ▶ Elle est utilisé pour initialiser les propriétés de l'objet.
  - ▶ Si le constructeur n'est pas défini, JavaScript ajoutera une méthode constructeur vide.

# Utilisation d'une classe

---

## ► La classe est utilisée pour créer des objets

```
► let pere = new Personne("Mamadou", "SENE", 50,  
    "masculin");  
    let mere = new Personne("Khady", "BA", 48,  
    "féminin");
```

# Les méthodes d'une classe

---

- ▶ Les méthodes de classe sont créées de la même façon que les constructeur.
- ▶ 

```
class Personne {  
    constructor(...) {...}  
    nomComplet() {  
        return this.prenom + " " +  
            this.nom;  
    }  
}
```

# L'héritage

---

- ▶ En Javascript, une classe peut hériter d'une autre classe. Pour créer une classe héritière, on utilise le mot clé `extends`.
- ▶ Une classe créée avec un héritage de classe hérite de toutes les méthodes d'une autre classe.

```
class Etudiant extends Personnel {  
    constructor(prenom, nom, age, sexe, matricule) {  
        super(prenom, nom, age, sexe);  
        this.matricule = matricule;}  
}
```

```
    afficheMatricule(){  
        alert("Le matricule : "+this.matricule); }  
}
```

```
let etu = new Etudiant("Mohamed", "SENE", 20,  
"masculin", 20197685);  
document.getElementById("p3").innerHTML =  
document.getElementById("p3").innerHTML + " " +  
etu.nomComplet() + " " + etu.age + " " + etu.sexe;  
etu.afficheMatricule();
```



# Références

---

- ▶ [http://www.mcours.net/cours/pdf/info/Support\\_de\\_cours\\_Javascript.pdf](http://www.mcours.net/cours/pdf/info/Support_de_cours_Javascript.pdf)
- ▶ <https://www.w3schools.com/js/>
- ▶ <https://www.pierre-giraud.com/javascript-apprendre-coder-cours/>