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 vont fonctionner différemment.

Pour information, une classe est un plan général qui va servir à créer des objets similaires. Une classe va généralement contenir des propriétés, des méthodes et une méthode constructeur.

Cette méthode constructeur va être appelée automatiquement dès qu'on va créer un objet à partir de notre classe et va nous permettre dans les langages basés sur les classes à initialiser les propriétés spécifiques des objets qu'on crée.

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.

Orienté Objet basé sur les prototypes

Un objet, en informatique, est un ensemble cohérent de données et de fonctionnalités qui vont fonctionner ensemble. Pour le dire très simplement, un objet en JavaScript est un conteneur qui va pouvoir stocker plusieurs variables qu'on va appeler ici des propriétés. Lorsqu'une propriété contient une fonction en valeur, on appelle alors la propriété une méthode.

Un objet est donc un conteneur qui va posséder un ensemble de propriétés et de méthodes qu'il est cohérent de regrouper.

Création d'un objet "personne "

```
let personne = {
    //nom, age et mail sont des propriétés de l'objet "personne"
    nom : ['Pierre', 'Richard'],
    age : 40,
    mail : 'pierre.richard@gmail.com',

    //appel est une méthode de l'objet personne
    appel: function(){
        alert('Bonjour, je suis ' + this.nom[0] + ', j\'ai ' + this.age + ' ans');
    }
};
```

```
//on accède à la méthode appel de l'objet personne
personne.appel();
```

Accéder aux membres d'un objet

Pour accéder aux propriétés et aux méthodes d'un objet, on utilise le caractère point . que l'on appelle également un accesseur.

On accède aux propriétés d'un objet dans le fichier html à l'aide de la commande (text_HTML correspond à la variable HTML) :

```
document.getElementById('text_HTML').innerHTML = 'Nom : ' + pierre.nom;
```

A noter : "document" est en fait aussi un objet, getElementById() une méthode et innerHTML une propriété de l'API "DOM"!

Les fonctions constructeur

```
//personne() est une fonction constructeur
function personne(n, a, m){
    this.nom = n;
    this.age = a;
    this.mail = m;

    this.appel = function(){
        alert('Bonjour, je suis ' + this.nom[0] + ', j\'ai ' + this.age + ' ans');
    }
}

let pierre = new personne(['Pierre', 'Richard'], 40, 'pierre.richard@gmail.com');

pierre.appel();
```

Orienté Objet basé sur les classes

On crée une nouvelle classe grâce au mot clef class. Dans cette classe, on définit une méthode constructor() qui va nous servir à initialiser les propriétés des objets créés par la suite à partir de la classe avec les valeurs courantes des objets.

Sous la méthode constructeur, nous allons définir des méthodes de classe auxquelles les objets auront accès.

```
class Perso{
    constructor(nom, age){
        this.nom = nom;
        this.age = age;
    }

    annee(){
        alert('Monsieur ' + this.nom + ' a ' + this.age + ' ans' )
    };
```

```
}

let perso1 = new Perso('Durand', 30);
let perso2 = new Perso('Dupont', 45);
perso1.annee();
perso2.annee();
```

Une fois la classe définie, on crée des objets à partir de celle-ci en utilisant le mot clef new suivi du nom de la classe. On dit qu'on instancie la classe. Dans le cas présent, on crée deux objets perso1 et perso2.

Classe étendue et héritage

Pour étendre une classe, c'est-à-dire pour créer une classe enfant qui va hériter des propriétés et des méthodes d'une classe parent, nous allons utiliser le mot clef extends.

```
class Personnel extends Perso{
    constructor(nom, age, taille){
        super(nom, age); //Appelle le constructeur parent
        this.taille = taille;
    }

    phrase(){
        alert('Monsieur ' + this.nom + ' a ' + this.age + ' ans et mesure ' + this.taille + '
        cm');
    };
}

let perso3 = new Personnel('Raymond', 50, 180);
perso3.annee();
perso3.phrase();
```

Ici, on crée une classe Personnel qui vient étendre notre classe de base Perso avec la syntaxe classe Personnel extends Perso.

Nous devons utiliser le mot clef super() qui permet d'appeler le constructeur parent dans le constructor() de notre classe fille afin que les propriétés soient correctement initialisées.

On peut ensuite créer des objets à partir de notre classe fille. Les objets vont également avoir accès aux propriétés et méthodes de la classe mère.