Javascript

Programmer objet

Prototype

Javascript est un langage basé sur les prototypes

```
function Personne(nom, age) {
  this.nom = nom;
  this.age = age;
}

Personne.prototype.vieillir = function() { this.age++; }


let toto = new Personne("Toto", 25);
toto.vieillir();
```

Le modèle objet JavaScript en détails

Classes

Introduites en ES6 pour simplifier l'écriture du modèle prototypal.

```
class Personne {
  constructor(nom, age) {
    this.nom = nom;
    this.age = age;
  }
  vieillir() {
    this.age++;
  }
}

let toto = new Personne("Toto", 25);
toto.vieillir();
```

Déclaration des propriétés

```
class MaClasse {

  propPublique = "coucou";

  constructor(msg) {
    if (msg) this.propPublique = msg;
  }
}

let obj = new MaClasse();
obj.propPublique; // "coucou"
```

Propriétés/méthodes privées

```
class MaClasse {

  #propPrivee = true;

  #methodePrivee() {
    /* ... */
  }
}

let obj = new MaClasse();
obj.#propPrivee; // SyntaxError
obj.#methodePrivee(); // SyntaxError
```

Propriétés/méthodes privées

Avant ES2022, on utilisait par convention "_"

```
class MaClasse {

  _propPrivee = true;

  _methodePrivee() {
    /* ... */
  }
}

let obj = new MaClasse();
obj._propPrivee; // true
obj._methodePrivee(); // pas d'erreur
```

Méthodes statiques

```
class Personne {

  constructor(nom,age) {
    this.nom = nom || "inconnu";
    this.age = age || 0;
  }

  static test(obj) {
    return obj instanceof Personne;
  }
}

let toto = {nom:"Toto",age:26};

Personne.test(toto); //false
```

Héritage

```
class Personne {
  constructor(nom, age) {
    this.nom = nom;
    this.age = age;
  }
  vieillir() {
    this.age++;
  }
}

class Homme extends Personne {
  constructor(nom,age) {
    super(nom,age); //appel du constructeur parent
    this.sexe="M";
  }
}
```

Surcharge d'une méthode

```
class Personne {
  constructor(nom,age) {
    this.nom = nom;
    this.age = age;
    this.memoire = 2000;
  }
  vieillir() {
    this.age++;
  }
}

class Alzheimer extends Personne {
  vieillir() {
    super.vieillir();
    this.memoire--;
  }
}

let totoMalade = new Alzheimer("Toto",60);
totoMalade.vieillir();
```

Surcharge d'une méthode de conversion

```
class Personne {
  constructor(nom,age) {
    this.nom = nom;
    this.age = age;
  }
  toString() { return this.nom; }
  toJSON() { return '{"nom":"'+this.nom+'"}'; }
}

let toto = new Personne("Toto",25);

console.log("je m'appelle " + toto); //je m'appelle Toto
console.log( JSON.stringify(toto) ); //'{"nom":"Toto"}'
```

Accesseurs et mutateurs

```
class Personne {

  #age = 0;

  constructor(nom) {
    this.nom = nom;
  }

  get age() {
    return this.#age;
  }

  set age(val) {
    if (typeof val !== "number") throw new TypeError("pas un nombre");
    this.#age = val;
  }
}

let toto = new Personne("Toto");
toto.age; // 0
toto.age = 25;
toto.age = "titi"; // TypeError
```