

Programmation Orientée Objet

(Tour d'horizon)

Orienté Objet

Classe Player





Orienté Objet : Classe/Objet

```
1 class User {  
2     // Corps de la classe  
3 }  
4  
6 let u = new User(); // instantiation
```



Orienté Objet : Attributs

```
1 class User {  
2  
3     username = "admin";  
4     email    = "admin@email.com";  
6  
7 }  
8  
9 let u = new User();  
10  
11 document.write(u.username + " | " + u.email);
```



Orienté Objet : Méthodes

```
1 class User {  
2  
3     username = "admin";  
4     email = "admin@email.com";  
6  
7     display() {  
8         return this.username + " | " + this.email;  
9     }  
10  
11 }  
12  
13 let u = new User();  
14  
15 document.write(u.display());
```



Orienté Objet : Constructeur

```
1 class User {
2
3     constructor(username, email){
4         this.username = username;
5         this.email = email;
6     }
7
8     display() {
9         return this.username + " | " + this.email;
10    }
11
12 }
13
14
15 let u = new User("admin","admin@email.com");
16
17 document.write(u.display());
```



Orienté Objet : Exercice 1

Créez la classe « Person » contenant :

- Trois attributs : name, email et age;
- Un constructeur;
- Une méthode getInfo() retournant toutes les informations d'utilisateur courant.



Orienté Objet : Corrigé 1

```
1  class Person {
2
3      constructor(name, email, age){
4          this.name = name;
5          this.email = email;
6          this.age = age;
7      }
8
9      getInfo(){
10         let msg = this.name + " " + this.email + " " + this.age;
11         return msg;
12     }
13 }
14
15
16
17 let p = new Person("Laroche", "laroche@mail.com", 24);
18
19 document.write(p.getInfo());
```




Orienté Objet : Exercice 2

- Créer une classe voiture;
- La classe voiture contient trois attributs, à savoir, couleur, puissance et vitesse;
- La classe voiture possède également deux méthodes : accélérer()/ralentir() permettant respectivement de augmenter/diminuer la vitesse (de la voiture courante);

Exemple :

1. La vitesse de la voiture courante = 30k/h;
 2. Lorsque vous appelez la méthode accélérer, la vitesse passe à 40k/h;
- Ajouter une méthode freiner() permettant de passer la vitesse à 0;
 - Testez le code.

Orienté Objet : Corrigé 2

```
1 class Car {
2     constructor(brand, color){
3         this.brand = brand;
4         this.speed = 0;
5         this.color = color;
6     }
7     accelerate() {
8         this.speed += 10;
9     }
10    slowDown() {
11        this.speed -= 10;
12    }
13    brake(){
14        this.speed = 0;
15    }
16 }
17
18
19 let city_car = new Car("BMW", "Rouge") ;
20
21 city_car.accelerate(); // 0 --> 10
22 city_car.accelerate(); // 10 --> 20
23 city_car.slowDown();   // 20 --> 10
24 document.write(city_car.speed); // affiche 10
25
26 city_car.brake(); // 10 --> 0
27 document.write(city_car.speed); // affiche 0
```



Orienté Objet : Exercice 3

Créez une classe Calculatrice, contenant deux attributs (nb1, nb2), un constructeur et cinq méthodes : addition, soustraction, division, multiplication et puissance.

Les méthodes créées s'opèrent sur les deux attributs : nb1 et nb2.

Exemple : la méthode addition renvoie : $nb1 + nb2$.

Instanciez la classe et testez les différentes méthodes.

Orienté Objet : Corrigé 3

```
1 class Calculator {
2     constructor(nb1, nb2){
3         this.nb1 = nb1;
4         this.nb2 = nb2;
5     }
6     addition(){
7         return this.nb1 + this.nb2;
8     }
9     multiplication(){
10        return this.nb1 * this.nb2;
11    }
12    subtraction(){
13        return this.nb1 - this.nb2;
14    }
15    division(){
16        return this.nb1 / this.nb2;
17    }
18    pow(){
19        return this.nb1 ** this.nb2;
20    }
21 }
22
23 let obj = new Calculator(3, 7);
24 document.write(obj.pow());
```



Orienté Objet : Exercice 4

Une tâche classique en programmation est de réaliser des calculs sur des suites de nombres afin d'en extraire différentes statistiques.

Dans ce contexte, créez une classe Stat contenant les méthodes :

`entrer(nombre)` : Ajouter un nombre à une suite;

`getSum()` : La somme de toutes les entrées;

`getAvg()` : La somme de toutes les entrées / Le nombre des entrées;

`getMedian()` : $((\text{nombre des entrées} - 1)/2) + 1$;

Orienté Objet : Corrigé 4 (v1)

```
1 class Stat {
2     constructor(){
3         this.count = 0;
4         this.sum = 0;
5     }
6     enter(num){
7         this.count += 1;
8         this.sum += num;
9     }
10    getSum(){
11        return this.sum;
12    }
13    getAverage(){
14        return this.sum/this.count;
15    }
16    getMedian(){
17        return ((this.count - 1)/2) + 1;
18    }
19 }
20
21
22 let calc = new Stat();
23
24 calc.enter(12);
25 calc.enter(10);
26 calc.enter(8);
27 document.write(calc.getAverage()); //10
```

Orienté Objet : Corrigé 4 (v2)

```
1 class Stat {
2     constructor(){
3         this.tab = [];
4     }
5
6     enter(num){
7         this.tab.push(num);
8     }
9
10    getSum(){
11        let sum = 0;
12        for (let i = 0; i < this.tab.length; i++) {
13            sum += this.tab[i];
14        }
15        return sum;
16    }
17
18    getAverage(){
19        return this.getSum() / this.tab.length;
20    }
21
22    getMedian(){
23        return ((this.tab.length - 1)/2) + 1;
24    }
25 }
26
27 let calc = new Stat();
28 calc.enter(12);
29 calc.enter(10);
30 calc.enter(8);
31 document.write(calc.getAverage()); // 10
```



Orienté Objet : Exercice 5

Définissez une classe Rectangle ayant les attributs : Longueur et Largeur.

Ajouter un constructeur d'initialisation.

Ajouter les méthodes suivantes :

getPerimetre() : retourne le périmètre du rectangle.

getAire() : retourne l'aire du rectangle.

afficher() : expose les caractéristiques d'un rectangle comme suit :

Longueur : [...] - Largeur : [...] - Périmètre : [...] - Aire : [...] - C'est un carré / Ce n'est pas un carré

Orienté Objet : Corrigé 5

```
1 class Rectangle{
2     constructor(len, wid){
3         this.len = len;
4         this.wid = wid;
5     }
6     getPerimeter(){
7         return (this.len + this.wid) * 2;
8     }
9     getArea(){
10        return this.len * this.wid;
11    }
12    display(){
13        let msg = this.len + " / " + this.wid;
14        msg += " | Périmètre : " + this.getPerimeter();
15        msg += " | Aire : " + this.getArea();
16        return msg;
17    }
18    isSquare(){
19        if (this.len == this.wid) { return true; }
20        else { return false; }
21    }
22 }
23 }
```



Orienté Objet : Corrigé 5

```
24 let r = new Rectangle(3,7);  
25 document.write(r.display());  
26 document.write(r.isSquare() ? " | C'est un carré." : " | Ce n'est pas un carré.");
```

Orienté Objet : Héritage

```
1 class Vehicle {
2
3     constructor(speed, nb_passagers){
4         this.speed = speed;
5         this.nb_passagers = nb_passagers;
6     }
7
8     display(){
9         return this.speed + " " + this.nb_passagers;
10    }
11
12 }
13
14 class Moto extends Vehicle { }
15
16 let m = new Moto(420, 2);
17 document.write(m.display());
18
```



Orienté Objet : Héritage

```
1 class Vehicle {
2     constructor(speed, nb_passengers){
3         this.speed = speed;
4         this.nb_passengers = nb_passengers;
5     }
6 }
7
8
9 class Moto extends Vehicle {
10     constructor(vitesse, nb_passengers, model){
11         super(vitesse, nb_passengers);
12         this.model = model;
13     }
14 }
15
16 let m = new Moto(420, 2, "Sport");
17
18 document.write(m.speed + " ");
19 document.write(m.nb_passengers + " ");
20 document.write(m.model);
```



Orienté Objet : Statique

```
1 class Car {
2
3     static wheels = 4;
4
5     constructor(speed, nb_passengers){
6         this.speed = speed;
7         this.nb_passengers = nb_passengers;
8     }
9
10    static klaxon(){
11        return "Biiip bip!";
12    }
13 }
14
15
16 document.write(Car.wheels);
17 document.write(Car.klaxon());
```



Orienté Objet : Exercice 6

Créez une classe `Addition_class` contenant une méthode statique `addition(c1,c2,c3)`, prenant trois chiffres en paramètre, afin de retourner leur somme.

Créez une classe `Moyenne_class` contenant une méthode `moyenne(c1,c2,c3)`, prenant également trois chiffres en paramètre afin de retourner leur moyenne.

Faites en sorte d'utiliser la méthode `addition()`, dans le calcul de la méthode `moyenne()`.


Orienté Objet : Corrigé 6

```
1 class AddClass {
2     static add(a, b, c) {
3         return a + b + c;
4     }
5 }
6
7
8 class AvgClass {
9     static avg(a, b, c) {
10         return AddClass.add(a, b, c)/3;
11     }
12 }
13
14 let m = AvgClass.avg(3,3,6)
15 document.write(m)
```

Orienté Objet : Exercice 7

Créez une classe contenant des méthodes statiques, permettant de générer un tableau HTML. Exemple :

```
document.write(
  Table.start(100, 2), // <table>
  // (largeur du tableau en %, épaisseur de la bordure en px)
  Table.header(["Firstname", "Lastname", "Email"]),
  Table.row(["Daniel", "Laroche", "d.laroche@mail.com"]),
  Table.row(["Florence", "Bert", "f.bert@mail.com"]),
  Table.end() // </table>
);
```



Firstname	Lastname	Email
Daniel	Laroche	d.laroche@mail.com
Florence	Bert	f.bert@mail.com

Orienté Objet : Corrigé 7

```
1 class Table {
2
3     static start(width, border){
4         return `




```

Orienté Objet : Corrigé 7

```
17 static row(tab){
18     let row = `|  |
| --- |
|`;
19     for(let key in tab){
20         row += ` ${tab[key]} |`;
21     }
22     row += `
`;
23     return row;
24 }
25 static end(){
26     return ``;
27 }
28 }
29
30 document.write(
31     Table.start(100, 2),
32     Table.header(["Firstname", "Lastname", "Email"]),
33     Table.row(["Daniel", "Laroche", "d.laroche@mail.com"]),
34     Table.row(["Florence", "Bert", "f.bert@mail.com"]),
35     Table.end()
36 );
```



Orienté Objet : Exercice 8

- Créez les classes Etudiant et Professeur héritant de la classe Personne.
- La classe Personne contient les attributs : nom et un constructeur
- La classe Etudiant contient l'attribut ine
- La classe Professeur contient l'attribut spécialité
- Prévoyez pour chaque classe une méthode permettant d'afficher les informations de l'objet courant.
- Créez un tableau ([]) contenant des objets Professeur et Etudiant.
- Prévoyez une boucle affichant les informations des objets de la liste créée.



Orienté Objet : Corrigé 8

```
1 class Person{
2     constructor(name){
3         this.name = name;
4     }
5 }
6
7
8 class Student extends Person{
9     constructor(name, cne){
10         super(name);
11         this.cne = cne;
12     }
13     display(){
14         return `L'étudiant : ${this.name} (${this.cne})`;
15     }
16 }
17
```

Orienté Objet : Corrigé 8

```
18 class Teacher extends Person{
19     constructor(name, speciality){
20         super(name);
21         this.speciality = speciality;
22     }
23     display(){
24         return `Le prof : ${this.name} (${this.speciality})`;
25     }
26 }
27
28 let persons = [];
29
30 let s1 = new Student('Aya', "A12345");
31 persons.push(s1);
32
33 let t1 = new Teacher('Yahia', "DevOps");
34 persons.push(t1);
35
36 for(let key in persons){
37     document.write(persons[key].display() + "<br>");
38 }
```



Orienté Objet : Exercice 9

Créez une classe joueur contenant trois attributs, à savoir, nom, position et buts marqués.

Créez une méthode permettant de comparer le joueur courant avec un autre, par rapport au nombre de buts marqués.

Exemple :

- Joueur 1 : 18 buts marqués, Ronaldo, attaquant
- Joueur 2 : 21 buts marqués, Messi, attaquant

Si nous appelons la méthode de comparaison, celle-ci, retourne :

« Messi » est meilleur que « Ronaldo »

Orienté Objet : Corrigé 9

```
1 class Player {
2
3     constructor(name, goals){
4         this.name = name;
5         this.goals = goals;
6     }
7
8     compar(p){
9         let msg = "Egalité";
10        if (this.goals > p.goals) msg = `${this.name} est meilleur que ${p.goals}`;
11        else if(this.goals < p.goals) msg = `${p.name} est meilleur que ${this.goals}`;
12        return msg;
13    }
14 }
15
16
17 let p1 = new Player("Ronaldo", 2);
18 let p2 = new Player("Messi", 2);
19
20 document.write(p1.compar(p2));
```



Orienté Objet : Exercice 10

- Créez les deux classes `Carnivore` et `Herbivore` héritant de la classe `Animal`.
- La classe `Animal` contient un attribut `point_de_vie` et un constructeur l'initialisant, dont sa valeur par défaut = 100.
- La classe `Animal` contient également une méthode `dormir()`, permettant de rajouter un point à l'attribut `point_de_vie`.
- La classe `Carnivore` contient une méthode `chasser()`, permettant de rajouter 5 points à l'attribut `point_de_vie`, tout en vidant le nombre de points d'un objet de type `Animal` passé en paramètre (l'animal chassé par le carnivore).
- La classe `Herbivore` contient une méthode `paturer()`, permettant de rajouter 5 points de vie à l'objet courant.

Orienté Objet : Corrigé 10

```
1 class Animal {
2     constructor(life_points = 100){
3         this.life_points = life_points;
4     }
5     sleep(){
6         this.life_points += 1;
7     }
8 }
9
10
11 class Carnivore extends Animal {
12     // chasser(proie)
13     hunt(preys){
14         this.life_points += 5;
15         preys.life_points = 0;
16     }
17 }
18
19 class Herbivore extends Animal {
20     // paturer()
21     graze(){
22         this.life_points += 5
23     }
24 }
```



Orienté Objet : Corrigé 10

```
25 let lion = new Carnivore(120)
26 let zebra = new Herbivore()
27
28 lion.hunt(zebra)
29
30 document.write(
31     lion.life_points + "<br>",
32     zebra.life_points
33 )
```



Orienté Objet : Exercice 11

Créez une classe Message contenant les attributs :

- expéditeur;
- destinataire;
- Date d'envoi;

Créez une classe Personne contenant les attributs :

- nom;
- prénom;

Pour la classe Message, Les deux attributs (expéditeur, destinataire) sont de type objet de Personne.

Prévoyez une méthode affichant les détails d'un message.

Exemple: Page suivante.

Orienté Objet : Exercice 11



Orienté Objet : Corrigé 11

```
1 class Person {
2     constructor(firstname, lastname){
3         this.firstname = firstname;
4         this.lastname = lastname;
5     }
6 }
7
8
9 class Mail {
10     constructor(sender, recipient){
11         this.sender = sender;
12         this.recipient = recipient;
13         this.send_date = new Date();
14     }
15     display(){
16         var msg = `De : ${this.sender.lastname} à : ${this.recipient.lastname} <br>
17                     A : ${this.send_date.toLocaleString("fr")}`;
18         return msg;
19     }
20 }
21 let p1 = new Person("David", "Bouneau");
22 let p2 = new Person("Damien", "Claustre");
23
24 let m = new Mail(p1, p2);
25
26 document.write(m.display());
```



Orienté Objet : Exercice 12

Créez une classe Compte bancaire contenant :

- Les attributs : numéro, titulaire, solde, découvert autorisé;
- Un constructeur initialisant les trois attributs;
- Une méthode permettant de créditer le solde du compte courant;
- Une méthode permettant de faire des virements, en vérifiant le découvert du compte;
- Une méthode retournant l'ensemble des valeurs des attributs dans un message.

Orienté Objet : Corrigé 12

```
1 class BankAccount {
2     constructor(num, owner, balance, overdraft) {
3         this.num = num;
4         this.owner = owner;
5         this.balance = balance;
6         this.overdraft = overdraft;
7     }
8     credit(amount){
9         this.balance += amount;
10    }
11    transfer(beneficiary, amount){
12        if((this.balance - amount) > -50){
13            this.balance -= amount;
14            beneficiary.balance += amount;
15            return true;
16        }else{
17            return false;
18        }
19    }
20    toString(){
21        return `Le solde du compte ${this.num} est de ${this.balance} euro(s) <br>`;
22    }
23 }
24 }
```



Orienté Objet : Corrigé 12

```
25 let b1 = new BankAccount("12345", "Matthew DP", 200, 50);
26 let b2 = new BankAccount("54321", "Christel Lavaut", 100, 50);
27
28 b1.credit(50);
29 document.write(b1);
30
31 document.write(b1.transfer(b2, 300) ? "virement accepté" : "virement non accepté", "<br>");
32 document.write(b1, b2);
```