JAVASCRIPT - LA PROGRAMMATION ORIENTÉE OBJET

BASES ET CONCEPTS

blocdroid

RAPPELS

- La **programmation procédurale** est une manière de coder qui se fonde sur le concept d'appel procédural.
- Une procédure, aussi appelée *routine*, *sous-routine* ou *fonction,* contient simplement une série d'étapes à réaliser. N'importe quelle procédure peut être appelée à n'importe quelle étape de l'exécution du programme, y compris à l'intérieur d'autres procédures.

- AVANTAGES :

- La possibilité de réutiliser le même code à différents emplacements dans le programme sans avoir à le dupliquer (principe « DRY »), ce qui a pour effet la réduction de la taille du code source.

blocdroid

DÉFINITIONS

- La **programmation orientée objet** est une manière de coder qui consiste en la définition et l'interaction de briques logicielles appelées *objets* ; un objet représente un concept, une idée ou toute entité du monde physique, comme une voiture, une personne ou encore une page d'un livre.

- AVANTAGES :

- La POO vous permet de coder plus rapidement.
- Coder plus rapidement ne signifie pas écrire moins de lignes de code. Cela signifie que vous pouvez implémenter plus de fonctionnalités en moins de temps sans compromettre la stabilité d’un projet.
- Ce principe rend votre code plus concis et plus lisible.

blocdroid

DÉFINITIONS D'UN OBJET

Un objet vide

```
let personne = {};
```

Un objet avec des propriétés

```
let personne = {
    nom: "Dupont",
    prenom: "Jean",
    age: 30
}
```

Un objet avec des propriétés et des méthodes

```
let personne = {
    nom: "Dupont",
    prenom: "Jean",
    age: 30,
    adresse: {
        rue: "123 rue de Paris",
        ville: "Paris",
        codePostal: "75000"
    },
    afficherNomComplet: function() {
        return `${this.prenom} ${this.nom}`;
    }
};
```

blocdroid

UTILISATION D'UN OBJET

```
let personne = {
    nom: "Dupont",
    prenom: "Jean",
    age: 30,
    adresse: {
        rue: "123 rue de Paris",
        ville: "Paris",
        codePostal: "75000"
    },
    afficherNomComplet: function() {
        return `${this.prenom} ${this.nom}`;
    }
};

console.log(personne.nom); // Affiche "Dupont"
console.log(personne.prenom); // Affiche "Jean" // Affiche 30
console.log(personne.adresse.rue);// Affiche "123 rue de Paris"
console.log(personne.afficherNomComplet()); // Affiche "Jean Dupont"
```

blocdroid

DÉFINITION D'UNE CLASSE (ES6)

```
class Personnage {
  constructor(name, force, life) {
    this.name = name;
    this.force = force;
    this.life = life;
  }
  frapper() {
      console.log(`${this.name} frappe !`);
  }
  deplacer() {
      console.log(`${this.name} se déplace !`);
  }
}
```

LE CONSTRUCTEUR

- Le constructeur est une méthode particulière qui est appelée dès l'instanciation de la classe.
- On peut lui passer des paramètres.

```
constructor(name, force, life) {
    this.name = name;
    this.force = force;
    this.life = life; }
```

blocdroid

DÉFINITIONS

- **Les classes contiennent la définition des objets que l'on va créer par la suite.**

👉 La classe est unique

👉 La classe contient le plan de fabrication d'un objet et on peut s'en servir autant de fois qu'on veut.

- Dans le cas du personnage, la classe contiendra une méthode *frapper()*. Cette méthode (fonction) devra modifier la propriété *life* en fonction de la propriété *force*.

- **Une classe est donc un regroupement logique de propriétés et de méthodes que tout objet issu de cette classe possédera.**

blocdroid

INSTANCIATION ET APPEL

```
class Personnage {
    constructor(name, force, life) {
        this.name = name;
        this.force = force;
        this.life = life; }

    frapper() {
        console.log(`${this.name} frappe !`);
    }

    deplacer() {
        console.log(`${this.name} se déplace !`);
    }
}

$perso = new Personnage("Arthur", 10, 100);
$perso.deplacer();
$perso.frapper();
```

blocdroid

DÉFINITION D’UN OBJET

- **Un objet est une instanciation d'une classe.**

👉 La classe est unique, les objets sont en nombre infini.

- Par exemple, un objet 'personnage' possède des caractéristiques :
  Nom
  Force
  Vie

- Il possède également des capacités ou des compétences :
  Frapper un autre personnage
  Gagner de l'expérience
  Se déplacer

- **Les caractéristiques correspondent aux propriétés, les capacités correspondent aux méthodes.**

blocdroid

INSTANTIATION ET APPEL

```
perso = new Personnage("Arthur", 10, 100);
perso.deplacer();
perso.frapper();
```

L'instanciation de la classe Personnage s'effectue avec le mot-clé new

L'appel des méthodes s'effectue avec la notation pointée

A gauche du point on trouve un objet. A droite du point on trouve une méthode ou une propriété.

blocdroid

```
class Personnage {
    constructor(name, force, life) {
        this.name = name;
        this.force = force;
        this.life = life; }

    frapper(cible) {
        cible.life -= this.force;
        console.log(`${this.name} frappe ${cible.name}
    }
}

arthur = new Personnage("Arthur", 10, 100);
bilbo = new Personnage("Bilbo", 5, 50);
arthur.frapper(bilbo);

console.log(`${bilbo.name} a ${bilbo.life} points de vi
```

INTÉRACTIONS ENTRE OBJETS

- La méthode frapper() demande un paramètre : le personnage à frapper()

- Cette méthode augmente les dégâts du personnage à frapper en fonction de la force du personnage qui frappe.

- On souhaite différencier les personnages en affectant des valeurs spécifiques à chacune de leur propriétés.

blocdroid

L'HÉRITAGE

```
class Guerrier extends Personnage {
    constructor(name, force, life) {
        super(name, force, life);
        this.force = force + 2; // Guerrier a plus de force
    }
}
```

Soient deux classes A et B. Quand on dit que B hérite de A cela signifie que toutes les propriétés et toutes les méthodes de A sont utilisables par B (sauf spécifications).

L'inverse n'est pas vrai.

Toutes instances de la classe B pourra appeler toutes les propriétés publiques et méthodes publiques de la classe A.

Plusieurs classes peuvent hériter d'une même classe mère.

L'héritage permet de spécialiser une classe mère en ajoutant des propriétés et des méthodes aux éléments existants dans la classe dont on hérite.

blocdroid

LES EXCEPTIONS

```
monObjet.addEventListener("click", () => {
    try {
        if (CaSePasseMal === true) {
            throw new Exception ("Erreur détectée");
        }
    } catch (error) {
        alert(error.message);
    }
});
```

- Les exceptions permettent de capturer et gérer les erreurs dans un programme.
- try {} : Bloc d'instructions dans lequel l'erreur peut se produire
  throw new Exception : génère une erreur
  catch {} : Récupère l'erreur et la traite

blocdroid