Kickoff Mario

Trucs & astuces

Appliquée au JS (< ES 5.1)

La POO

Intérêt

- La programmation orientée objet a pour but de permettre une plus grande flexibilité et maintenabilité du code.
- Le code orienté objet est :
  - Plus simple à développer
  - Plus facile à reprendre
  - Plus facile à analyser
  - Et permet de répondre à des situations complexes

Introduction

- La POO en JS est appelée « programmation orientée prototype »
- CAD que :
  - L'héritage est effectué via des objets qui seront des prototypes pour d'autres objets.

Les espaces de noms

- Les espaces de noms en JS (équivalent des classes dans d'autres langages) peuvent contenir des attributs et des méthodes.
- Ils permettent de regrouper tout ou partie des fonctionnalités d'une application.
- JS utilise des fonctions comme constructeurs pour définir un objet.

```
var Personne = function () { };
var p = new Personne();
```

Les propriétés et les méthodes

- Les propriétés sont définies au sein du prototype afin que tous les objets qui en héritent puissent disposer de cette propriété via la chaîne de protoypes.
- Les méthodes et les attributs sont des propriétés.

```
function Personne(nom) {
    this.nom = nom;
    this.afficherNom = function () {
        console.log(this.nom);
    }
}

Personne.prototype.sePresenter = function() {
    console.log("Je suis " + this.nom);
};

var p = new Personne('Gustave');
p.afficherNom(); // 'Gustave'
p.sePresenter(); // 'Je suis Gustave'
```

L'héritage

- Pour indiquer un héritage, on assigne une instance de l'objet parent à la propriété prototype de l'objet fils.

```
// Le constructeur Personne
var Personne = function(nom) {
    this.nom = nom;
};

Personne.prototype.marcher = function(){
    console.log("Je marche !");
};
Personne.prototype.direBonjour = function(){
    console.log("Bonjour, je suis " + this.nom);
};

// Le constructeur Étudiant
function Étudiant(nom, sujet) {
    // On appelle le constructeur parent
    // pour profiter des propriétés définies dans la
fonction
    Personne.call(this, nom);
    this.sujet = sujet;
}

// on remplace la méthode direBonjour pour l'étudiant
Étudiant.prototype.direBonjour = function(){
    console.log("Bonjour, je suis un " + this.nom + ".
J'étudie " + this.sujet + ".");
};

var étudiant1 = new Étudiant("Jean", "la physique
appliquée");
étudiant1.direBonjour();
étudiant1.marcher();
```

Mécaniques

Le jeu

La boucle de jeu

- La boucle de jeu est un évènement qui se répète à intervalle défini (delta) tant que dure le jeu. Elle permet :
  - De récupérer les inputs
  - De calculer les déplacements
  - De checker les collisions
  - D'animer les objets

```
var delta = {
    ANIMATIONS: 50,
    DEPLACEMENTS_MARIO : 100,
    GLOBAL: 25,
    animations: 0,
    mario: 0,
    animations : 0
};
setInterval(function () {
    delta.mario += delta.GLOBAL;
    delta.animations += delta.GLOBAL;
    if (delta.animations >= delta.ANIMATIONS) {
        mario.animate();
        delta.animations = 0;
    }
    if (delta.mario >= delta.DEPLACEMENTS_MARIO) {
        mario.move();
        delta.mario = 0;
    }
}, delta.GLOBAL);
```

Les collisions

- Vous devez pouvoir vérifier si un élément en heurte un autre.
- Basiquement cela arrive quand la hitbox d'un élément intersecte celle d'un autre.
- Attention le moteur de collisions peut être gourmand en ressources, attention à votre conception!

```
function Cell (x, y) {
    this.x = x;
    this.y = y;
    this.checkCollision = function (cell) {
        return cell.x === this.x && cell.y === this.y;
    }
}
```

Les autres éléments

- Le moteur de gravité
- Les inputs
- …

Des questions?