Javascript –POO

Bailly Benjamin

Programmation objet – objets littéraux

Jusque-ici nous avons utilisé les objets littéraux comme ceci :

```
let Benjamin = {
        nom : "Bailly",
        prenom : "Benjamin",
        taille : "174 cm",
        saluer : function () {
            console.log ("Hello c'est moi " + this.prenom);
        }
    }
```

On peut voir que pour accéder à la propriété d'un objet au sein de lui-même il faut utiliser « this », les objets littéraux seront plutôt utilisé pour passé des informations (c'est pour cela que je parlais plus tôt de tableau associatif plutôt).

Programmation objet – Créer un constructeur

Un constructeur est très utile quand vous voulez créer de multiples objets avec des propriétés et des methodes similaires.

La convention veux que les développeurs écrivent le constructeur avec la première lettre en majuscule.

```
function JeuVideo(nom,type,support){
    this.nom = nom;
    this.type = type;
    this.support = support;
    this.presentation = () =>{
        console.log(this.nom + " est un jeu de " + this.type + " disponible sur " + this.support)
    }
}
let ageOfEmpire = new JeuVideo("Age of empire", "stratégie", "pc");
```

Programmation objet – TP-25 Constructeur

- Créez un constructeur de “Stagiaire” avec pour attributs : nom, prénom, age, ville de naissance
- Ce constructeur aura pour méthode “sePresenter” qui affichera tous les attributs nom, prénom, age et le nom de la ville de naissance.

- Creéz un constructeur de “Ville” avec pour attribut : nom, nombre d’habitants, pays.
- Créez deux objets de “stagiaire” et créez autant d’objets de “ville” que necessaire pour pouvoir assigner ces objets à l’attribut “ville” de naissance”.

Programmation objet – TP-25 Constructeur

```
function Stagiaire(nom, prenom, age, villeDeNaissance){
    this.nom = nom;
    this.prenom = prenom;
    this.age = age;
    this.villeDeNaissance = villeDeNaissance;

    this.sePresenter = () => {
        console.log("Bonjour, je m'appelle " + this.nom + " " + this.prenom + " j'ai " + thi
s.age + "ans et je viens de " + this.villeDeNaissance.nom);
    }
}

function Ville(nom, pays, nombreHabitant){
    this.nom = nom;
    this.pays = pays;
    this.nombreHabitant = nombreHabitant;
}

var Toulon = new Ville ("Toulon", "France", "59999")

var Benjamin = new Stagiaire("Bailly","Benjamin","27",Toulon);

Benjamin.sePresenter();
```

Programmation objet – Les prototypes

La plupart des langages en POO utilisent les classes , la particularité de Javascript c’est l’utilisation des propotypes. (Bien que les classes soit aussi utilisables mais nous allons pour le moment nous concentrer sur les prototypes).

- Cherchez les differences et particularités entre les deux.

Programmation objet – Héritage

Rappeler les attributs et methodes du parent à l'aide de .call dans le constructeur de l'enfant :

```
function Heros(nom,race,classe){

    this.nom = nom;
    this.race = race;
    this.classe = classe;
    this.sePresenter = function(){
        console.log("hello je suis " + this.nom +" de la race " +
this.race + " et je suis un "+ this.classe)
    }
}
function Humain(nom,race,classe,age){

    Heros.call(this,nom,race,classe);
    this.age = age;
}
```

Programmation objet – Classe

Même si javascript en terme de programmation objets reste un langage prototypal, en 2015 les classes ont été introduite.

En arrière plan le fonctionnement reste exactement le même avec l’heritage via prototype mais la syntaxe s’en voit simplifié et est surtout beaucoup plus ressemblante aux autres langages orientés objets comme java ou php.

Programmation objet – Classe

La syntaxe avec les classes s’en voit beaucoup plus simple et clair qu’avec les prototypes.

Pour faire de l’heritage il suffit de faire un extends de la classe parente au moment de créer la classe enfant.

Et lors du constructeur utiliser la méthode “super()” pour faire passer les paramètres.

Programmation objet – Classe

```
class Animal{
    constructor(race, poids, regime){
        this.race = race;
        this.poids = poids;
        this.regime = regime;
    }

    description(){
        console.log("Cet animal est un " + this.race + " il pèse " + this.poids + "kg et son régime
 alimentaire est : " +this.regime)
    }
}

class Serpent extends Animal{
    constructor(race,poids,regime,frequenceDeMue){
        super(race, poids, regime);
        this.frequenceDeMue = frequenceDeMue;
    }
}

let coki = new Animal("berger belge", "50", "carnivore");
let snake = new Serpent("python", "10","carnivore","2 fois par an")
```

Programmation objet – Les getters et setters

Les getters vont nous permettre de créer une propriété, nous permettant d'accéder à la ou les propriétées liées au getter.

Exemple :

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

    get nomPrenom(){
        return this.nom + " " + this.prenom;
    }
}

console.log(directeurMontpellier.nomPrenom);
```

Programmation objet – Les getters et setters

Les setters vont permettre de changer des propriétés d'un objet, il est possible de faire du traitement sur ces nouvelles valeurs.

Exemple :

```
 set nomPrenom(newValue){
        [this.nom, this.prenom] = newValue.split(" ");
    }

directeurMontpellier.nomPrenom = "Valgean Jean"
```

Programmation objet – TP-26 mini jeu

Pour les consignes de ce tp je vous transmet un pdf et pour la correction je vous transmettrez directement mon fichier.js

Les dates

En programmation vous allez très souvent croiser les dates au format TIMESTAMP, qui correspond au nombre de seconds écoulées depuis le 1er janvier 1970 à minuit , exemple : 1621505938.

En JS Date est un objet et il est très facile d'y accèder.

Date() va afficher ceci : **Thu May 20 2021 12:22:24 GMT+0200 (heure d'été d'Europe centrale).**

Date.Now() va afficher le timestamp actuelle : **1621506212803**

Les dates

Si l'on désire une date bien précise nous allons utiliser le constructeur de l'objet Date.

```
let datePrecise = new Date(2019, 11, 09, 22, 25);
// Au-dessus :
//    - annee (obligatoire)
//    - mois (obligatoire) qui commence par 0 (comme un
tableau)

//    - jour (facultatif) qui commence par 1
//    - heure (facultatif) qui commence par 0
//    - minute (facultatif) qui commence par 0
//    - secondes (facultatif) qui commence par 0
//    - millisecondes (facultatif) qui commence par 0
```

Les dates tp-27

Je vais vous demander de m’afficher la date actuelle dans ce format : jeudi 20 mai 2021

Je vous donne deux indices :

- Une méthode de l’objet date pourra vous être utile.
- L’objet navigator du BOM pourra également vous être utile.

Les dates tp-27

```
let date = new Date();
let dateFrWithLocaleString = date.toLocaleString(navigator.language,{
    day : "numeric",
    year : "numeric",
    month:"long",
    weekday:"long"

})

alert(dateFrWithLocaleString)
```