# JS avancé

## Fonctions

Il est possible de créer des fonctions anonymes.

Par exemple on peut déclarer :

var f = function (i) {  
 console.log("La valeur est "+i);

} ;

Et l’appel sera :

f(2) ;

avec comme résultat :

La valeur est 2

## Objets

Pour un objet, on peut créer des attributs ou des fonctions à la volée.

Par exemple :

var o = new Object() ;  
o.name="jean" ;  
o.age=12 ;

Ici on a créé deux attributs name et age pour l’objet o ;

On peut aussi écrire comme ceci :

var o = new Object() ;  
o["name"]="jean" ;  
o["age"]=12 ;

On peut créer un objet en l’initialisant à la JSON :

var o = {name : "jean",age: 12} ;

On peut parcourir un objet pour découvrir quels sont ses attributs :

**for** (key **in** o) {

}

key correspond au nom de l’attribut (ici *name* ou *age*) et o[key] donne la valeur

## Arrays

On peut initialiser un array directement :

var ar = [1,2,3] ;

ou le déclarer vide au départ :

var ar = new Array() ;

Pour parcourir un array :

### Méthode classique :

for (var i = 0 ; i < ar.length ; i++) {  
}

### Methode forEach

ar.forEach(**function** (entry) {

});

### Methode for … of

**for** (elem of ar) {

}

## Classes comme fonctions

En JS les fonctions sont des objets. Des objets qui contiendrait du code executable.

Et donc, les classes (en tant qu’objet spécialisé) peuvent être implémentées comme des fonctions.

**function** PseudoClass (bidule) {

**this**.bidule = bidule;

**this**.log = **function**() {

**console.log(this.**bidule);

};

}

Là, on crée une fonction avec un attribut (bidule) et une méthode (log()).

Pour appeler, on fait :

var pc = new PseudoClass(4) ;  
pc.log() ; // là, ça affiche la valeur de bidule, donc 4

A noter, qu’on peut définir les méthodes à l’extérieur de la fonction en utilisant le prototype.

PseudoClass.prototype.log = **function**() {

**console.log(this.**bidule);

}

## Classes « pures »

Les dernières versions de JS implémentent des classes à la Java (mais qui en pratique utilise la méthode précédente de façon cachée)

Si on reprend l’exemple plus haut :

**class** PseudoClass {

constructor (bidule) {

**this**.bidule = bidule;

}

log() {

**console.log(this.**bidule);

}

}

## Fonctions (2)

On peut très bien définir une fonction avec zéro paramètre et lui en passer quand même.

function func() { // définition

….  
}

….  
func(2) ; // appel  
  
Ca ne génèrera pas d’erreur.  
Et si on veut connaitre les paramètres effectivement passés lors de l’appel, on parcourt le Array *arguments*.