

Le développement côté client avec JavaScript

Module 3 - Développer des librairies en JavaScript



Objectifs

- Connaître les mécanismes du développement objet en JavaScript
- Savoir créer des librairies en JavaScript

Développer des librairies en JavaScript

Créer un objet

Conteneur
d'attributs
et de fonctions

- Déclaration

```
var objet = {cle: "valeur", cle2: 256};
```

- Manipulation

```
objet.cle  
objet['cle2']
```

- Structure dynamique

```
objet.cle3=10;
```

- Tableau d'objets

```
var objets = [{}, {}, ...];
```

Développer des librairies en JavaScript

Démonstration



Développer des librairies en JavaScript

Créer des classes

Descripteur
d'attributs
et de fonctions

- Déclaration

```
var UneClasse = function(param1, param2) {  
    //ajout d'attributs  
    this.att1 = param1 || "";  
    this.att2 = param2 || 10;  
    //ajout de fonctions  
    this.uneFonction = function() {  
        //ajout de fonctions  
    };  
}
```

- Utilisation

```
var unObjet = new UneClasse("valeur param1", 5);
```

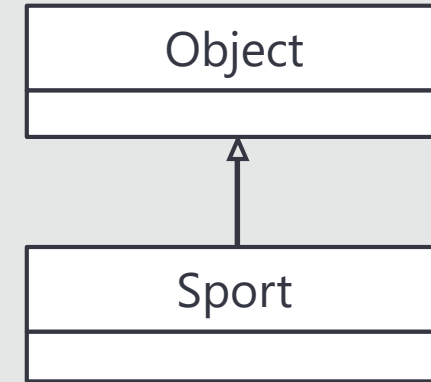
Développer des librairies en JavaScript

Démonstration



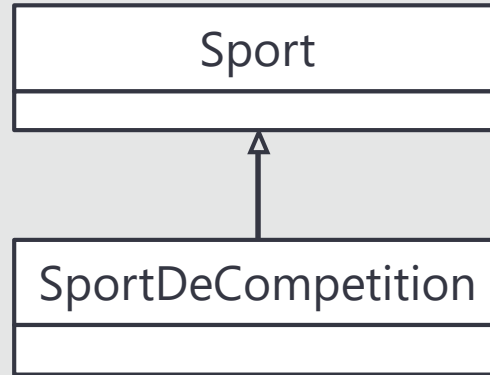
Le mécanisme d'héritage

```
var Sport = function(nom,description) {  
    //...  
}
```



```
▼ Sport {afficher: f, nom: "squash!", description: "un sport cardio!"}  
  ► afficher: f ()  
    description: "un sport cardio!"  
    nom: "squash!"  
  ► __proto__: Object
```

L'héritage des attributs



```
var SportDeCompetition = function(nom,description,niveau) {  
    Sport.call(this,nom,description);  
    this.niveau = niveau || "";  
}
```

```
▼ SportDeCompetition {nom: "badminton", description: "ça vole", afficher: f, niveau: "National"}  
  ► afficher: f ()  
    description: "ça vole"  
    niveau: "National"  
    nom: "badminton"  
  ► __proto__: Object
```


Développer des bibliothèques en JavaScript

L'héritage des fonctions

- Par partage du prototype
- Mise en œuvre

```
SportDeCompétition.prototype = Object.create(Sport.prototype);
```

- Nécessite la déclaration des fonctions par prototype

```
Sport.prototype.jouer = function() { console.log("..."); };
```

```
▼ SportDeCompétition {nom: "Rugby", description: "Un jeu de ballon ovale", afficher: f, niveau: "International"}  
  ► afficher: f ()  
    description: "Un jeu de ballon ovale"  
    niveau: "International"  
    nom: "Rugby"  
  ▼ __proto__: Sport  
    ▼ __proto__:  
      ► jouer: f ()  
      ► constructor: f (nom,description)  
      ► __proto__: Object
```

La substitution des fonctions

- Par la réécriture d'un prototype

```
SportDeCompétition.prototype.jouer = function() {console.log("C'est parti!");};
```

```
▼ SportDeCompétition {nom: "Rugby", description: "Un jeu de ballon ovale", afficher: f, niveau: "International"}
  ► afficher: f ()
    description: "Un jeu de ballon ovale"
    niveau: "International"
    nom: "Rugby"
  ▼ __proto__: Sport
    ► jouer: f ()
      ▼ __proto__:
        ► jouer: f ()
        ► constructor: f (nom,description)
        ► __proto__: Object
```

Développer des librairies en JavaScript

Démonstration



Développer des librairies en JavaScript

class

- Mot clé : **class**

```
class Sport{
  constructor(nom, description){
    this.nom = nom || "pas de nom";
    this.description = description || "";
  }

  afficher() {
    console.log(`${this.nom} ${this.description}`);
  }
}
```

Développer des librairies en JavaScript

extends

- Mot clé d'héritage : **extends**

```
class SportDeCompétition extends Sport{
  constructor(nom, description, niveau) {
    super(nom, description);
    this.niveau = niveau || "";
  }

  afficher() {
    console.log(`${this.nom} ${this.description} ${this.niveau}`);
  }
}
```

Développer des librairies en JavaScript

Démonstration



Développer des librairies en JavaScript

TP



Les conflits entre librairies

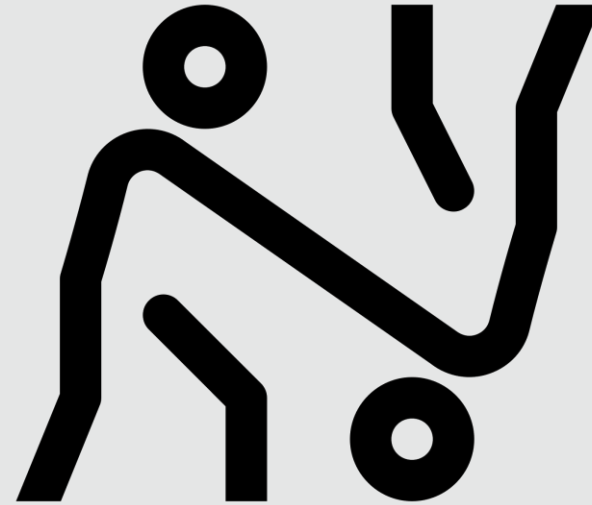
Perd en cas de conflit...



```
<script src="uneLibrairie.js"></script>  
<script src="uneAutreLibrairie.js"></script>
```



Gagne en cas de conflit...



Développer des librairies en JavaScript

Démonstration



Développer des librairies en JavaScript

Les contextes d'exécution

Encapsulation

Fonctions
anonymes

Les fonctions anonymes

- Déclaration

```
var uneFonction = function ([unParametre,...]) {  
    //le code de la fonction  
};
```

- Appel

```
uneFonction ([valeurParametre,...]);
```

Les fonctions anonymes

- Déclaration et appel consécutif

```
(function ([unParametre,...]) {  
    //le code de la fonction  
}) ([valeurParametre,...]);
```

Développer des librairies en JavaScript

L'encapsulation

- Première déclaration possible

```
var librairieA =  
(  
    function () {  
        var librairieA={};  
        librairieA.data = "data a";  
  
        librairieA.uneFonction = function ()  
        {  
            alert("Une fonction a");  
        }  
        return librairieA;  
    }  
)();
```

Développer des librairies en JavaScript

L'encapsulation

- Deuxième déclaration possible

```
(  
    function () {  
        var librairieA={};  
        librairieA.data = "data a";  
  
        librairieA.uneFonction = function ()  
        {  
            alert("Une fonction a");  
        }  
        window.librairieA=librairieA;  
    }  
) ();
```

Développer des librairies en JavaScript

Démonstration



La spécialisation de méthodes

- Par encapsulation

```
Sport.prototype.afficher =  
    function(separateur, avecDetail){...}  
  
Sport.prototype.afficherDefaut =  
    function(){this.afficher(", ", false);}  
Sport.prototype.afficherDefautAvecSeparateur =  
    function(separateur){this.afficher(separateur, false);}  
Sport.prototype.afficherAvecDetail =  
    function(avecDetail){this.afficher(", ", avecDetail);}
```

- Inconvénients

- Déclaration statique
- Verbeux
- Maintenance coûteuse

La closure

- Retourner une fonction contextualisée

```
Sport.prototype.afficher = function(avecDetail){  
    return function(separateur) {  
        if (avecDetail) {...}  
        else {...}  
    };  
}
```

- Ajouter des méthodes dynamiquement

```
var afficherDetail = Sport.prototype.afficher(true);  
var afficherGeneral = Sport.prototype.afficher(false);
```

Le pattern factory par closure

La closure

La factory

Le choix



```
var librairie = (function() {  
  var librairie={};  
  librairie.enregistrer = function(cible)  
  {  
    if(cible.toLowerCase()=== "memoire") {  
      return function(data) {  
        console.log("J'enregistre en mémoire '%s'",data);}  
      }  
    else if(cible.toLowerCase()=== "localStorage") {  
      return function(data) {  
        console.log("J'enregistre dans le localStorage '%s'",data);}  
      }  
    else if(cible.toLowerCase()=== "rest") {  
      return function(data) {  
        console.log("J'enregistre avec l'API rest '%s'",data);}  
      }  
    }  
  }  
  return librairie;  
})();  
librairie.enregistrerData = librairie.enregistrer("rest");
```

Développer des librairies en JavaScript

Démonstration



Le stockage local



LocalStorage

sessionStorage

```
setItem(cle, valeur)  
getItem(cle)  
removeItem(cle)  
clear()
```

Développer des librairies en JavaScript

Démonstration



Développer des librairies en JavaScript

TP



Développer des librairies en JavaScript

Les traitements asynchrones

- Application réactive
- Gestion des traitements longs

Les callbacks

Les promesses

Les callbacks

```
function enregistrer(data, callback)
{
    console.log("j'enregistre les informations '%s'", data); 2
    setTimeout(function() { callback("succès de la sauvegarde") }, 1000);
} 3 5
```

```
function afficherResultat(message) 6
{
    if(message) console.log(Message : %s", message);
    else console.log("Aucun message");
}
```

```
enregistrer("mes datas", afficherResultat); 1
console.log("pendant ce temps, la vie continue..."); 4
```

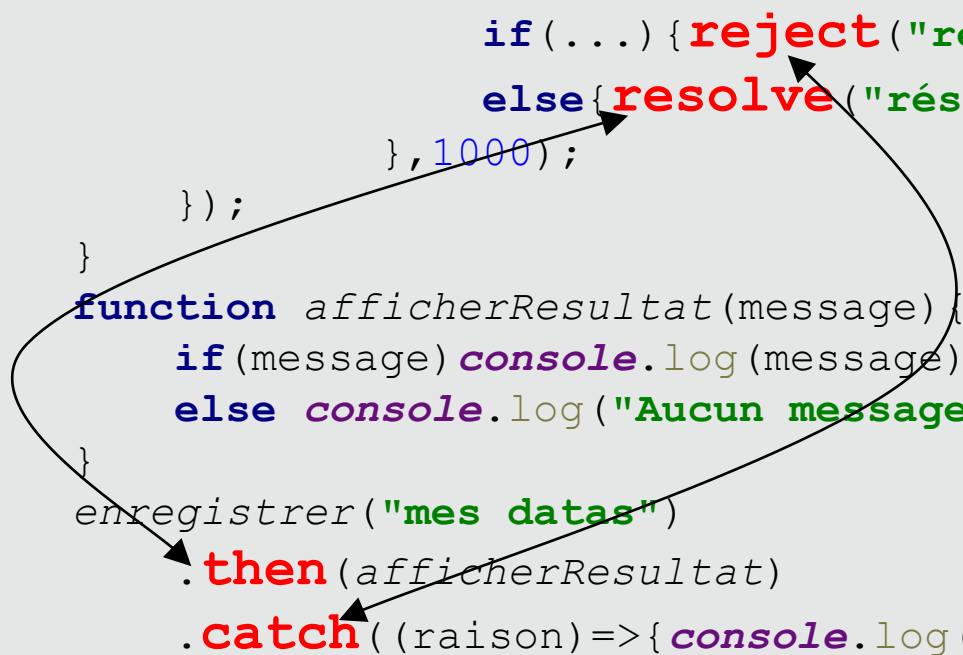

Développer des librairies en JavaScript

Démonstration



Les promesses

```
function enregistrer(data) {  
  return new Promise(function(resolve, reject)  
  {  
    console.log("j'enregistre les informations '%s'", data);  
    setTimeout(function()  
    {  
      if(...) {reject("rejeté");}  
      else {resolve("résolu");}  
    }, 1000);  
  });  
}  
  
function afficherResultat(message) {  
  if(message) console.log(message);  
  else console.log("Aucun message");  
}  
  
enregistrer("mes datas")  
  .then(afficherResultat)  
  .catch((raison) => { console.log(raison) });
```



Développer des librairies en JavaScript

Démonstration



Développer des librairies en JavaScript

TP



Conclusion

- Vous connaissez les principaux mécanismes entrant en jeu dans l'écriture de librairies en JavaScript