programmation web en javascript

6 / 6

bonnes pratiques et patterns de programmation js

- éviter de polluer l'espace global
  - éviter les variables globales
- utiliser des variables privées dans les objets
- structurer le code

- utilisation intensive des closures et des fonctions immédiates

limiter les globales grace à une IIFE

- encapsuler tout le code dans une fonction immédiate

```
var compteur = 0 ;
var i = 1;

var setup=function(x) {
    …
}
function doAll( v ) {
    …
}
function close() {
    …
}

setup( 42 ) ;
doAll( 73 ) ;
close() ;
```

5 variables globales

```
(function () {
  var compteur = 0 ;
  var i = 1;
  var setup=function(x) {
      …
  }
  function doAll( v ) {
      …
  }
  function close() {
      …
  }
  setup( 42 ) ;
  doAll( 73 ) ;
  close() ;
}) () ;
```

0 variables globales

propriétés privées pour un objet

- les objets javascript ne peuvent pas déclarer de propriétés privées
- on se sert de closures

  et d'une IIFE
  - color : propriété **publique**
  - x, y : **privées**
  - origin() : méthode **privée**
  - move, getX, paint : méthodes publiques

```
let point = (function() {
 let x =0;
 let y =0;
 let origin = ()=>{...} ;

 return {
   color : 'blue',
   move(a,b) {
    x += a ; y += b ;
   },
   getX() {
    return x ;
   },
   paint(c) {
    origin() ;
    this.color= c ;
   }
 }
})() ;
```

le pattern *namespace*

- js n'a pas de mécanisme explicite pour structurer l'espace de nom
- On peut se servir des objets :

```
// 6 noms globaux

// constucteurs
function Personne(){}
function Etudiant(){}

//variables
var promo_num = 100 ;
var promo=[ ] ;

//objets
var module1 = {} ;
var module1.data = {
    a:1, b:2};
var module2 = {}
```

```
// NAMESPACE : 1 seul global

var MYAPP = {} ;
MYAPP.Personne = function (){}
MYAPP.Etudiant=function (){}

MYAPP.promo_num = 100 ;
MYAPP.promo=[ ] ;

MYAPP.modules = {}

MYAPP.modules.module1 = {} ;
MYAPP.modules.module1.data =
{a:1,b:2};
MYAPP.modules.module2 = {}
```

le pattern *module*

- consiste à combiner *namespace, IIFE,* et propriétés privées

```
let myLog = ( function(){
    let _logger = [ ] ;
    function _displayLog(log) {
      console.log( log.mod + " "+log.msg ) ;
    }
    function _log( modName, message ) {
      let log = { mod: modName, msg: message } ;
      _displayLog(log) ; logger.push(log) ;
    }
    function _showAllLogs() {
      logger.forEach((l) => displayLog(l))
    }
    // Révélation de l'API publique
    return {
      log : _log,
      showAll : _showAllLogs
    }
} )() ;
```

- Dans une application, on utilise souvent plusieurs modules définis pour l'application ou importés d'une librairie
- il est utile de structurer en modules/sous-modules

```
let labelApp = {
      config : { … },
      modules: { }
} ;

labelApp.modules.manager = mYLib.manager ;

labelApp.modules.logger = ( function(){

    // code module

    return { … }
} )() ;
labelApp.modules.dispatcher = ( function(){

    // code module
    return { … }
} )() ;
```