JavaScript a boli

¿Para poder utilizar cualquier nombre de variable con seguridad en nuestros ficheros .JS que estrategia podemos adoptar?

Para poder utilizar cualquier nombre de variable con Seguridad, cada programa debe definir sus variables dentro de un contenedor único. Este contenedor es el espacio de nombres (namespace).

¿Por qué razón funciona esa estrategia?

Porque JavaScript se programan aplicaciones muy complejas (enormes a veces) y se han desarrollado cientos de librerías que podemos importar en nuestro código. Esto hace necesario organizar de alguna manera nuestro código para asegurarnos de que nuestras variables, objetos y funciones sean únicas y no se estén sobrescribiendo (o siendo sobrescritas) por una variable del mismo nombre de alguna de las librerías externas que utilicemos. Nombres como *getElement*,*counter*, *checkValue*, *showMessage*, etc son muy comunes y podrían existir en casi cualquier *script*que importemos.

Da un ejemplo práctico en el que se explique el conflicto y su solución.

-Ejemplo conflicto: Si hacemos esto, no eres un buen programador que no esta hecho namespace y estarías modificando todo (que no son variables únicas).

```
var variable1 = “prova1”;
var variable2 = “prova2”;
var variable3 = function(){…};
var variable4 ={un:1,dos:2};
```

-Ejemplo solución: Para evitar ejemplo solución, es crear un objeto y guardar a dentro a todos variables (así es único contenedor para guardar todos sus variables).

```
var CrearProjecte = {};
CrearProjecte. variable1 = “prova1”;
CrearProjecte variable2 = “prova2”;
CrearProjecte .variable3 = function(){…};
CrearProjecte .variable4 ={un:1,dos:2};
```

O también se puede hacer esta manera:

```
var CrearProjecte = {
        variable1 = “prova1”,
        variable2 = “prova2”,
        variable3 = function(){…},
        variable4 ={un:1,dos:2}
};
```

que és el patró de [Christian Heilmann]?

Es una versión mejorada de patrón módulo tradicional, que proporciona algunas ventajas y soluciones algunas problemas como siguientes:

-Se pueden convertir a publicas las propiedades y funciones.

-No es necesario utilizar nombres largos para acceder a otro función pública a otro.
-Cuando ponemos this luego de un objeto creado, no es necesario volver poner nombre objecto que this ya va cogiendo y añadiendo.

Ejercicio módulo miCalculadora:

Archivo index.html:

```
<html>
<head>
 <title>miCalculadora</title>
 <script src="miCalculadora.js"></script>
</head>
<body>
 <h1>Calculadora</h1>
 <button onclick="miCalculadora.publicfuncio()" value="Saludar">Saludar</button>
</body>
</html>
```

Archivo miCalculadora.js

```
var miCalculadora = (function(){
   var operador1 = 8;
   var operador2 = 3;

   function sumar(){
      alert(operador1+operador2);
   }

   function restar(){
      alert(operador1-operador2);
   }

   function operacioRandom(){
      var x = Math.floor((Math.random() * 2));
      if(x==0){
         sumar();
      } else {
         restar();
      }
   }

   return {
      publicfuncio:operacioRandom
   }
}());
```