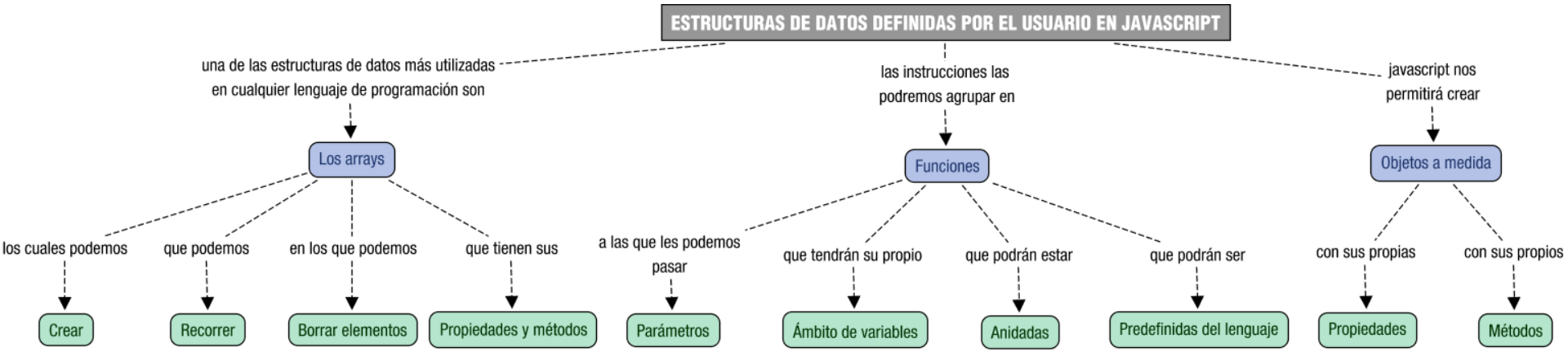
ESTRUCTURAS DEFINIDAS POR EL USUARIO EN JAVASCRIPT

Unidad didáctica 3 - Parte 4

Creación de Objetos de Usuario

Mapa conceptual



Creación de objetos

- JavaScript NO es POO tal y como lo conocemos.
- JavaScript es un lenguaje basado en objetos.
- JavaScript SI es un lenguaje basado en prototipos.

- Un objeto en JavaScript es una colección de propiedades.
- Las propiedades pueden tener forma de datos, tipos, funciones (métodos) o incluso otros objetos.
- Es más fácil de entender un objeto de JavaScript como un array de valores, cada uno de los cuales está asociado a una propiedad (un tipo de datos, método u objeto).

¿Un método puede ser una propiedad de un objeto?

Creación de objetos

- Una función contenida en un objeto se conoce como un método.
- Los métodos no son diferentes de las funciones que has visto anteriormente.
- Los métodos han sido diseñados para ser utilizados en el contexto de un objeto, y por lo tanto, tendrán acceso a las propiedades de ese objeto.
- En JavaScript se pueden añadir y quitar propiedades a un objeto en tiempo de ejecución.
  - Si se añade una propiedad a un objeto que está siendo utilizado como el prototipo de otros objetos, los objetos para los que es un prototipo también tienen la nueva propiedad añadida.

Creación de objetos. Constructor

- Los objetos se crean empleando una función especial denominada **constructor**, determinada por el nombre del objeto.
- El constructor es el **molde o plantilla** utilizada para crear objetos.
- Convención:
  - nombre del constructor con inicial de cada palabra en mayúscula.
  - nombre de instancia con primera inicial en minúscula.
- La palabra reservada **new** se emplea para crear objetos en JavaScript.

Creación de objetos. Constructor

- Ejemplo constructor:

```
function MiObjeto ( [ lista-argumentos ] )  {
        // propiedades   p=valor;
        // métodos       m=function …{};
};
```

Ejemplo de instancia u objeto del tipo MiObjeto:

```
var unObjeto = new MiObjeto ( [ lista-argumentos ] );
var otroObjeto = new MiObjeto ( [ lista-argumentos ] );
```

Propiedades

```
function Coche(){
      // Propiedades valores iniciales fijos
      this.marca =  "Audi A6";
      this.combustible =  "diesel";
      this.cantidad = 0;                    // Cantidad de combustible en el depósito.
};
```

this hace referencia al objeto actual

```
//Ejemplos de diferentes instancias
var cocheDeMartin = new Coche();
var cocheDeSilvia = new Coche();
var cochesAlumnos = new Array();
cochesAlumnos[0] = new Coche();
/*
cochesAlumnos = {
      "daw201": new Coche(),
      "daw202": new Coche()    //.....
};
alert (cochesAlumnos["daw201"].marca);     //¡¡¡ VIOLACIÓN DEL ENCAPSULAMIENTO en una POO!!!
cochesAlumnos["daw201"].color = "azul";  //añade una propiedad color con valor azul
*/                    // en JavaScript se hace continuamente con sus propios objetos
```

Al llamar al constructor this hace referencia a cocheDeMartin

Al llamar al constructor this hace referencia a cochesAlumnos[0]

Creación de objetos. Constructor

Ejemplo constructor con argumentos opcionales:

```
function Coche(marca,modelo,combustible,deposito,cilindrada,caballos,color ) {
    // Propiedades
    this.marca =  marca || "Audi";
    this.modelo = modelo || "";
    this.combustible =  combustible || "diesel";
    this.cantidad = deposito || 0;  // Cantidad de combustible en el depósito.
    this.cc = cilindrada || 0;
    this.potencia = caballos || 0;
    this.color = color || "";
};
```

Definición de métodos

Supón que queremos tener una forma de rellenar el depósito de cualquier coche.

Podemos definir una propiedad rellenarDeposito que permita modificar la cantidad de litros del depósito.

Podemos hacerlo de dos formas:

1. Usando una función anónima (sin nombre) declarada dentro del constructor.
2. Asignando como valor a una de las propiedades del constructor una función externa.

```
//Forma 1
function Coche(propietario,marca,combustible){
    this.propietario = propietario;
    this.marca = marca;
    this.combustible = combustible;
    this.cantidad=0;  //todos los coches se inicializan con 0 litros en el depósito
    this.rellenarDeposito = function (litros){              //función anónima
                                this.cantidad=litros;
                            };
}

//Forma 2
function rellenarDeposito(litros){
    this.cantidad = litros;
}
function Coche(propietario,marca,combustible){
    this.propietario = propietario;
    this.marca = marca;
    this.combustible = combustible;
    this.cantidad=0;
    this.rellenarDeposito = rellenarDeposito;       //función externa
}
//En ambos casos el método o función se usa de la misma manera
var coche1 = new Coche("Martín","Volkswagen Golf","gasolina");
coche1.rellenardeposito(45);  //rellenamos el depósito con 45 litros
```

Fíjate que las funciones anónimas llevan un punto y coma después de la llave de cierre.
Esto se debe a que terminamos la definición del valor asignado a una propiedad.

Acceso a las propiedades de un objeto

- A través de los métodos definidos en el propio objeto.
- Encapsulamiento.

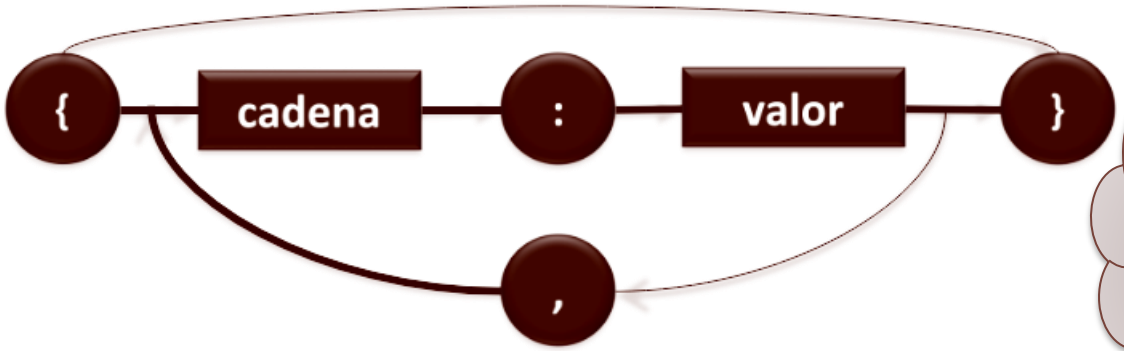
```
function Coche(propietario,marca,combustible){
   this.propietario = propietario;
   this.marca = marca;
   this.combustible = combustible;
   this.cantidad=0;  //todos los coches se inicializan con 0 litros en el depósito
   this.imprimeMarca = function(){
                        return this.marca;
                       };
   this.imprimePropietario = function(){
                              return this.propietario;
                             };
   this.cambiarCoche = function(marca,combustible) {
                         this.marca = marca;
                         this.combustible = combustible;
                       };
   this.venderCoche =  function(nuevoPropietario) {
                         this.propietario = nuevoPropietario;
                       };

}
var c = new Coche("Martín","Volkswagen Golf","gasolina");
```

```
alert (c.imprimePropietario());
c.cambiarCoche("Toyota","Diesel");
```

Definición de Objetos Literales

- Los literales de objeto se utilizan para almacenar información en parejas nombre-valor.
- Un literal de objeto se define mediante llaves ({ y })
- Dentro las llaves podemos colocar cualquier número de parejas nombre-valor.
- Una pareja nombre-valor se define mediante:
  - una cadena,
  - un símbolo de dos puntos y
  - un valor.
- Cada pareja nombre-valor deben estar separada por coma de la anterior.

Sintaxis Objeto Literal



Definición

```
//Forma 1
  avion = { "marca":"Boeing","modelo":"747","pasajeros":450 };

//Forma 2
  var avion = new Object();
  avion.marca = "Boeing";
  avion.modelo = "747";
  avion.pasajeros = 450;
```

Acceso

```
//Forma 1 (a través de su propiedad)
  document.write(avion.marca);

//Forma 2
  document.write(avion["modelo"]); //usando el nombre de la propiedad como índice
```

Ejemplo

```
var datos=[
  {"id":"2","nombrecentro":"IES A Piringalla" ,"localidad":"Lugo","provincia":"Lugo"},
  {"id":"10","nombrecentro":"IES As Fontiñas","localidad":"Santiago","provincia":"A Coruña"},
  {"id":"9","nombrecentro":"IES A Carballeira","localidad":"Ourense","provincia":"Ourense"},
  {"id":"4","nombrecentro":"IES de Teis","localidad":"Vigo","provincia":"Pontevedra"}];

for (var i=0; i< datos.length; i++){
  document.write("El centro "+datos[i].nombrecentro+" con ID nº "+datos[i].id + " está en ");
  document.write(datos[i].localidad+" en la provincia de " +datos[i].provincia+"<br/>");
}
```