UD1. Javascript

OBJETOS Y CLASES

Objectos

- JavaScript siempre ha soportado objetos (clases no)
- Un objeto es como un **array indexado** por nombre (array asociativo en PHP o diccionario en Python)
- Se puede acceder a los atributos con lo **.** y los **[ ]**

```
var persona = {
 nombre: ['Bob', 'Smith'], edad: 32, genero: 'masculino', intereses:
['música', 'esquí'],

bio: function () {
   alert(this.nombre[0] + '' + this.nombre[1] + ' tiene ' + this.edad + '
años. Le gusta ' + this.intereses[0] + ' y ' + this.intereses[1] + '.');
 }};
```

for .. in

```
let user = { name: "John", age: 30 };
alert( "age" in user ); // true, user.age exists
alert( "blabla" in user ); // false, user.blabla doesn't exist

let user = { name: "John", age: 30, isAdmin: true
};
for (let key in user) {
// keys
alert( key );  // name, age, isAdmin
// values for the keys
alert( user[key] ); // John, 30, true
}
```

https://www.youtube.com/watch?v=a7b4S9Zk65w

Objeto predefinido: String

- **Funciones importantes**:

  toLowerCase(), **concat**(), **charAt**(), **indexOf**(text,[index]), **lastIndexOf**(text,[index]), **replace**(text1,text2).

- **split**(caracter, [trozos]): Separa la cadena por un carácter separador, devuelve un array.

- **subString**(inici, [fin]): Saca la subcadena dado un principio y un posible fin

Objeto predefinido: Date

```
// Crea una fecha con la fecha y hora del sistema
var d=new Date();
// Crea una fecha como una cadena
d=new Date("October 13, 2014 11:13:00");
// Crea una fecha con año, mes, dia, hora, minutos, segundos,
milisegundos
d=new Date(99,5,24,11,33,30,0);
// Crea una fecha con any, mes y dia
d=new Date(99,5,24);
```

Objeto predefinido: Date

- Funciones importantes:
  - **setMonth**(mes), **getMonth**(); **setDate**(dia), **getDate**(), **setHours**(hora, minuto, segundo), **getHours**(), etc.
  - **getDay**(): devuelve del 0 al 6 el dia de la semana.
  - **toDateString**(): De fecha a cadena
  - **toGMTString**(): de fecha a cadena en formato GMT
  - **toUTCString**(): de fecha a cadena en formato UTC

Objeto predefinido: Array

- Métodos interesantes:
    - **Join**([separador]): Crea una cadena con los elementos de un array con un separador.
    - **push** (element,element2...): Mete elementos al final del array.
    - **pop**(): Extrae el último elemento y lo quita del array.
    - **reverse**(): invierte el array.
    - **sort**(): Ordena alfabéticamente el array
    - **slice**(inici,[final]): Saca los elementos entre un inicio y un final.

Objeto predefinido: Math

- Contiene funciones que nos ayudan en **operaciones matemátiques**.
- Constantes importants: E, PI, LN2 (logaritmo neperiano de 2), LN10, LOG2E (logaritmo en base 2 de E), LOG10E.
- Funciones de redondeo: **floor**(), **ceil**(), **round**()
- Funcions matemátiques: **abs**(), **max**(x,y), **min**(x,y), **pow**(x,y), **random**(), **sqrt**()

Otros objetos predefinidos

- Nativos:
  - String
  - Number
  - Boolean
  - Data
  - RegExp
  - Array
  - Funcion
  - Object
- Alto nivel (dependen del navegador):
  - Window
  - Screen
  - Navigator
  - Location
  - History
  - Document

Clases

- Antes de ES6 las **clases se hacían en funciones** y todavía se utiliza mucho esta técnica.
- ES6 incorporó el término **class** y se parece más a otros lenguajes.
- JavaScript **no funciona por clases**, sino por **prototipos**.
- JavaScript siempre ha tenido **objetos**, pero **no clases** de la forma tradicional.
- Las **clases** en ES6 **no tienen atributos** y **funciones privados**. Esto se hace en **scopes**.
- **Todo son objetos** y las clases son objetos función.

Objetos con funciones (Métodos internos)

```
function Apple (type) {
 this.type = type;
 this.color = "red";
 this.getInfo = function() {
     return this.color + ' ' + this.type + ' apple';
 };
}
```

- Lo que se hace es declarar la función “constructor” de los objetos.
- Es como una “**plantilla**” para crear nuevos objectos.
- Es invocada en **new** para ser constructor. **this.** representa al objecto creado.
- El objeto creado es una **instancia** de la función.

Objectos con funciones (Métodos con prototype)

```
function Apple (type) {
 this.type = type;
 this.color = "red";
}
Apple.prototype.getInfo = function() {
 return this.color + ' ' + this.type + ' apple';
};
```

- Más eficiente al no recrear la función cada vez que hagamos un objecto.

Objectes literals

```
var apple = {
    type: "macintosh",
    color: "red",
    getInfo: function () {
        return this.color + ' ' + this.type + ' apple';
    }
  }
```

- Se hacen nuevas instancias con Object.create()
- No recomanable para hacer más de una instancia.

Que es Prototype

- Todos los objetos tienen un prototype.
- El prototype es un objecte que a su vez tiene un prototype.
- Varios objetos unidos por prototypes se les llama prototype chain.
- Con prototype, un objecto puede delegar con otros objetos hijos.
- Los objetos tienen un prototype común Object

```
var homework = {
   topic: "JS"
  };
  homework.toString();     // [object Object]
```

La función toString() está en Object.prototype

https://www.youtube.com/watch?v=a7b4S9Zk65w