Coding Bootcamp

Sprint 1

Programación Orientada a Objetos

(en JavaScript)

Programación Orientada a Objetos (POO) (1/9)

- Es un paradigma de programación que se organiza alrededor de **objetos**, en lugar de funciones (como hace la programación funcional).
- Históricamente, un programa se veía como una serie de instrucciones que la computadora debe ejecutar. Con POO el programa se ve como una colección de objetos que interactúan entre sí.
- Un objeto es una entidad que tiene una responsabilidad definida.
- Los objetos contienen:
  - Propiedades (o atributos).
  - Métodos.
- POO se hizo popular porque permite generar un código más mantenible y flexible.

Programación Orientada a Objetos (POO) (2/9)

Ejemplo: una película se puede pensar (modelar) como un objeto que tiene:

- Propiedades:
  - Título.
  - Año.
  - Director.
  - Cantidad de veces que fue vista.
- Métodos:
  - Comprar.
  - Alquilar.
  - Ver.

Además, en general, se tienen varias **instancias** de un objeto:

- Película: Cinema Paradiso – 1988 – Giuseppe Tornatore.
- Película: Braveheart – 1995 – Mel Gibson.
- Película: The Godfather – 1972 – Francis Ford Coppola.

Programación Orientada a Objetos (POO) (3/9)

¿Cómo podríamos generar esas tres películas en nuestro código?

Una opción podría ser lo siguiente, crear tres objetos por separado:

```
var m1 = {
  title: "Cinema Paradiso",
  year: 1988,
  director: "Giuseppe Tornatore",
  timesWatched: 3500000,
};
var m2 = {
  title: "Braveheart",
  year: 1995,
  director: "Mel Gibson",
  timesWatched: 60500111,
};
var m3 = {
  title: "The Godfather",
  year: 1972,
  director: "Francis Ford Coppola",
  timesWatched: 700000222,
};
```

¿Qué problemas tiene este código?

Programación Orientada a Objetos (POO) (4/9)

- JavaScript es un lenguaje basado en prototipos y no contiene (en su versión ES5) “clases” como sí tienen otros lenguajes como C++, Java, C#, PHP, etc.
  Esto suele ser confuso para programadores acostumbrados a estos lenguajes.
- JavaScript utiliza funciones para construir objetos:

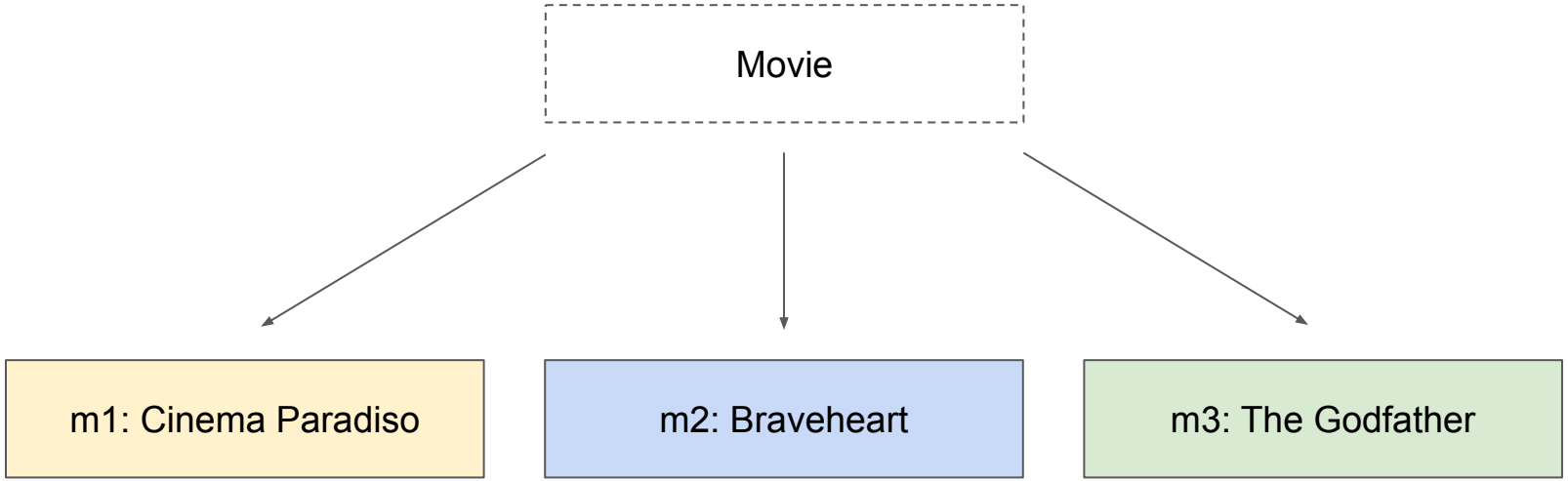
```
var Movie = function () {
    // Código...
};

var m1 = new Movie();
var m2 = new Movie();
var m3 = new Movie();
```

A esta función se le llama constructor de la "clase" Movie, pero no es más que una función común y corriente de JavaScript.

Al usar la palabra "new" se crean objetos nuevos a los cuales se les llama "instancias" de la clase Movie.

Nota: Recordar que en JavaScript, las funciones también son objetos.

Programación Orientada a Objetos (POO) (5/9)



`Movie` es como un molde que nos permite construir objetos a los que se les llama instancias de `Movie`.

“Clase”

Instancias
(Objetos)

Programación Orientada a Objetos (POO) (6/9)

A la “clase” `Movie` se le pueden definir **propiedades**:

```
var Movie = function (title, year, director) {
    this.title          = title;
    this.year           = year;
    this.director       = director;
    this.timesWatched   = 0;
};
```

Probarlo

Programación Orientada a Objetos (POO) (7/9)

A la “clase” `Movie` también se le pueden definir métodos.
A diferencia de las propiedades anteriores, los métodos se deben compartir entre todas las películas. Para ello se usa la propiedad `prototype`.

```
Movie.prototype.buy = function () {
  console.log("Se compró la película.");
  // Código de comprar una película...
};

Movie.prototype.rent = function () {
  console.log("Se alquiló la película.");
  // Código de alquilar una película...
};

Movie.prototype.watch = function () {
  console.log("Se miró la película.");
  this.timesWatched++;
};
```

La propiedad `prototype` permite definir las propiedades o métodos que queremos que se compartan entre todos los objetos que se creen a partir de la “clase”.

Probarlo

Programación Orientada a Objetos (POO) (8/9)

La slide anterior es equivalente a hacer esto:

```
Movie.prototype = {
  constructor: Movie,
  buy: function () {
    console.log("Se compró la película.");
  },
  rent: function () {
    console.log("Se alquiló la película.");
  },
  watch: function () {
    console.log("Se miró la película.");
    this.timesWatched++;
  },
};
```

En este caso es necesario definir el constructor (porque el `prototype` se sustituye completamente).

Programación Orientada a Objetos (POO) (9/9)

Finalmente, se crean las instancias de `Movie`.

```
var m1 = new Movie("Cinema Paradiso", 1988, "Giuseppe Tornatore");
var m2 = new Movie("Braveheart", 1995, "Mel Gibson");
var m3 = new Movie("The Godfather", 1972, "Francis Ford Coppola");
```

Ahora es posible hacer cosas como:

```
m1.buy();
m2.buy();
m3.watch();
m3.watch();
console.log(m1.timesWatched);
console.log(m3.timesWatched);
```

Probarlo