{JS}

Clase 14

JS

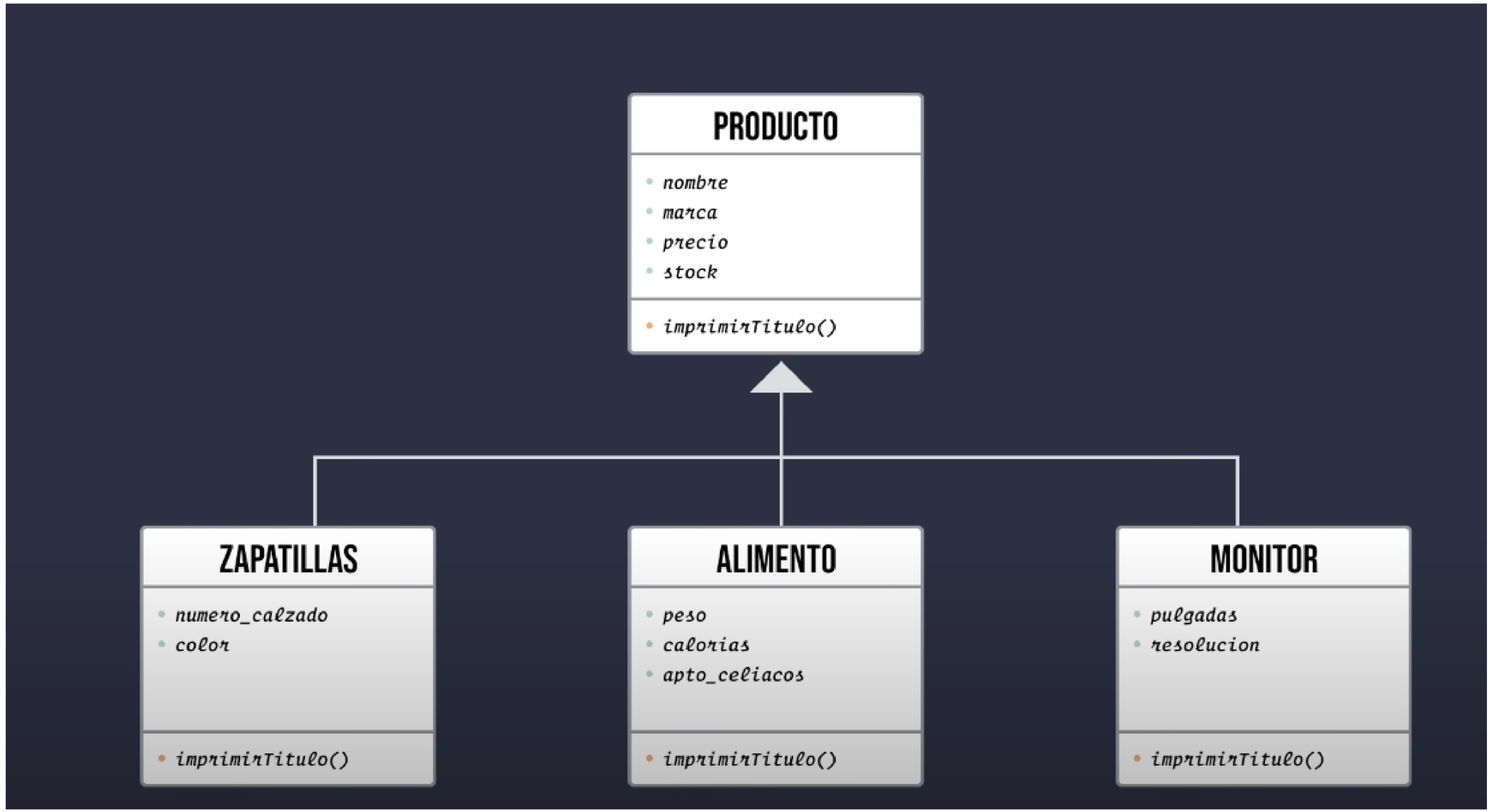
‹Índice›

Prototipos

Herencia

Herencia

- Herencia por clases típica



Herencia

Ventajas de la herencia

- Reutilización de código
- Mantenimiento más sencillo
- Jerarquía clara y estructurada
- Extensibilidad
- Polimorfismo
- Abstracción y encapsulamiento
- Modelado del mundo real

Herencia

¿Cómo funciona la herencia por prototipos en JS?

No funciona como tradicionalmente estamos acostumbrados. Veremos las clases de JS en otra clase

Prototipos

Prototipo

¿Qué es un prototipo?

Es como un delegado. Alguien a quien le damos una responsabilidad y la confiamos en él

Prototipo

Veámoslo en código

Prototipo

En JS no se denomina herencia normal, si no que se denomina herencia por prototipos.

O mejor todavía **delegación de objetos**

Prototipo

Los prototipos son dinámicos ya que son objetos normales en JS como los que usamos normalmente.

La búsqueda de métodos y propiedades se hace en tiempo de ejecución

Prototipo

Añadimos propiedades a un prototipo

Prototipo

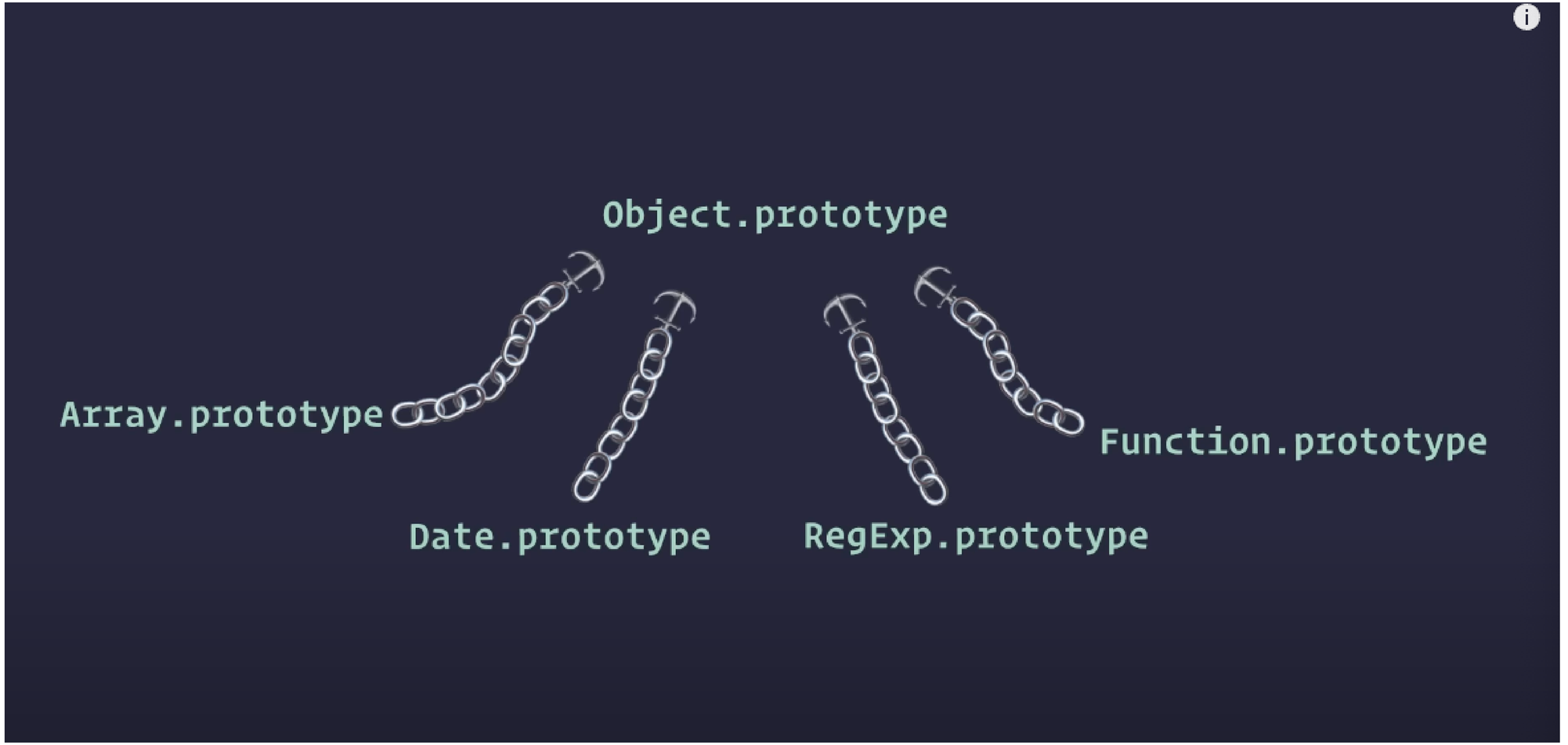
Añadimos propiedad a un objeto que comparte con su prototipo

Prototipo

Reescribir propiedades de un objeto

toString

Prototipo


Object.prototype
Array.prototype
Date.prototype
RegExp.prototype
Function.prototype

Prototipo

El final de la cadena: null

Prototipo

Accedemos a una propiedad de un objeto

¿Qué ocurre por detrás?

Prototipo

La cadena de prototipos solo ocurre cuando queremos acceder a una propiedad o método.

Si sobrescribo una propiedad solo se hace en el objeto en el que la hago

Prototipo

¿Podemos saber si un objeto es un prototipo de otro objeto?

Prototipo

¿Cómo definimos prototipos?

- Objetos literales: funciones, objetos o arrays: crear un objeto
- Object.create(proto)
- Funciones creadoras o clase (otra clase)
- setPrototypeOf y __proto__ (no usar)

‹Despedida›

Email

bienvenidosaez@gmail.com

Instagram

@bienvenidosaez

Youtube

youtube.com/bienvenidosaez