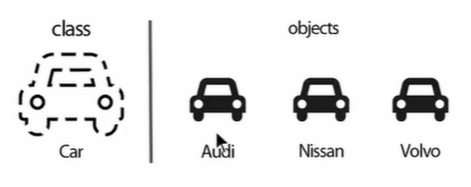
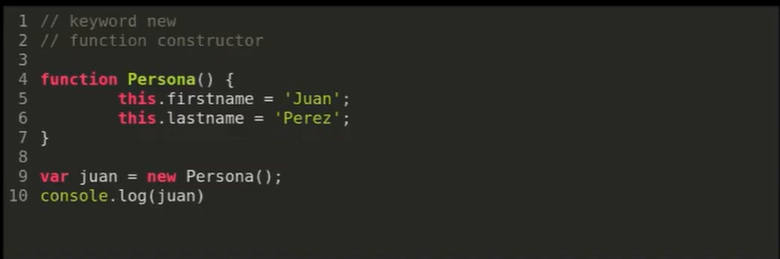
**Clases en JavaScript**

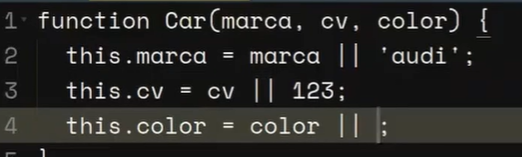


Muchas veces cuando creamos un objeto, estamos creando una plantilla. En lugar de copiar esa plantilla una y otra vez, JavaScript nos da acceso a lo un constructor o class. Las clases comparten gran parte de la misma funcionalidad que los objetos normales, pero también se expande mucho más en esa funcionalidad. Las clases son útiles para crear muchos objetos que comparten algunas de las mismas propiedades y métodos (como os usuarios en un sitio web).

Sintaxis



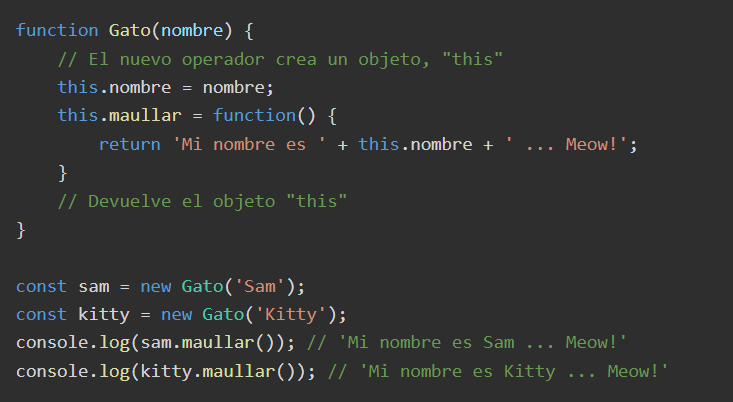
Podemos también pasarle parámetros, cuando no conocemos el valor que se le quiere asignar, si no se le pasa nada por parámetros, podemos utilizar el operador or || para asignar un valor por defecto si no recibimos uno por argumento.



**Class e instanciación pseudo-clasica.**

JavaScript no proporciona un “verdadero” sistema de objetos, pero hay algo muy familiar, en aras de la discusión, llamaremos a nuestros objetos clases. Se instancian de la misma manera pseudo-clasica, usando la palabra clave new, y puede tomar argumentos

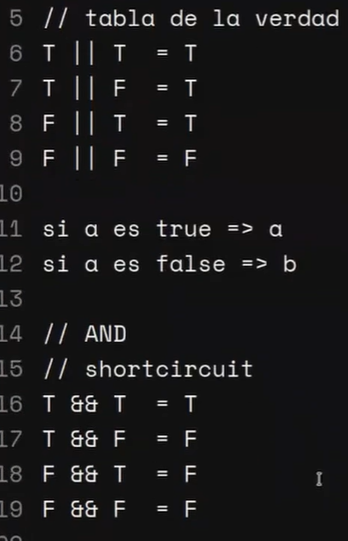
En este ejemplo crearemos una clase Gato. La convención para las clases consiste en dar un nombre en mayúscula al nombre de todo lo que se puede instanciar con la palabra new, cuando usamos la palabra clave new, JavaScript hace un gran trabajo detrás de escena para nosotros y crea y devuelve un objeto automáticamente



**This en las clases**

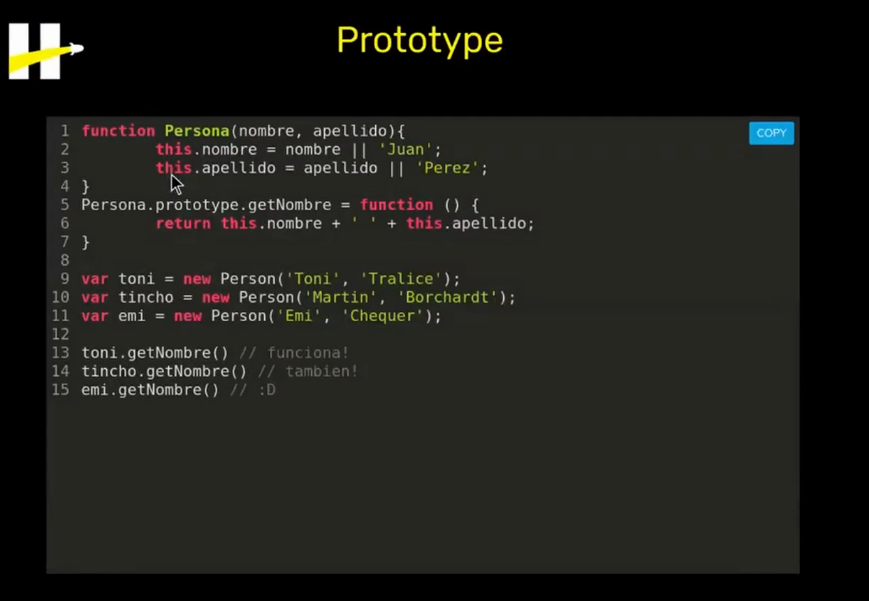
La palabra clave this puede comenzar a volverse muy confusa cuando comenzamos a usar las clases. En el último ejemplo lo usaremos en el método de los maullidos. Una buena regla general, si no está seguro a qué se refiere this, es observar dónde se llama el método y el objeto ala izquierda del “punto”, ese es el objeto al que se refiere this.

**Tablas de la verdad or and**



**Prototipo**

Las clases tienen una forma única de establecer un método una vez y dar acceso a cada objeto de esa clase a esos métodos. Esto se llama prototype. Cada clase tiene una propiedad prototype, que luego podemos establecer en métodos.

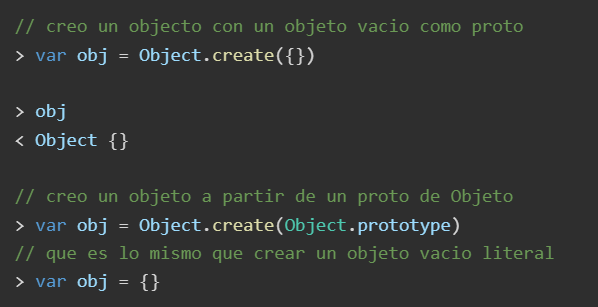


Los métodos de prototype tienen acceso a la palabra clave this y al igual que antes, siempre apuntará al objeto, (a la izquierda del punto) que los está llamando.

Hasta ahora siempre que teníamos que crear un objeto nuevo declarábamos un object literal, pero vamos a ver que hay otros métodos que nos da el prototype de object para cumplir esa tarea.

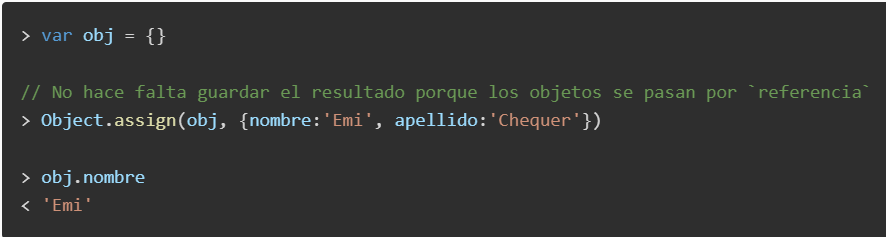
**Object.create**

El método créate de los objetos nos permite crear un nuevo abjetoa partir de un prototype especificado.



**Object.assing**

El método assing de los objetos te permite agregar propiedades a un objeto por parámetro



**Herencia clásica**

En el paradigma de programación orientada a objetos un tema muy importante es la herencia y polimorfismo y de las clases (Los vamos a llamar constructores).

Cuando hacemos referencia a Herencia nos referimos a la capacidad de un constructor de heredar propiedades y métodos de otro constructor, así como un gato es mamífero antes que gato, y hereda sus propiedades.

Cuando hablamos de polimorfismo nos referimos a la capacidad de que los objetos puedan responder a un llamado igual de acuerdo a su propia naturaleza.

**Herencia en JavaScript**

En JS a diferencia de la herencia clásica nos manejamos con prototipos, que van a tomar los métodos pasados por sus padres mediante la prototype chain