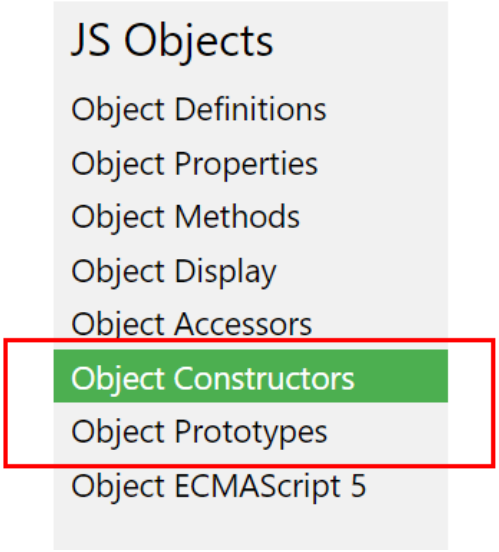
Realiza una **lectura comprensiva** de los apartados: “Constructores” y “Prototipos”:



Función constructora de objetos.

JavaScript es un lenguaje ***basado en prototipos*** que no contiene ninguna declaración de [1]clase, como ocurre en otros lenguajes como C++ o Java. Esto puede resultar confuso para programadores acostumbrados a los lenguajes con una declaración de clase. En su lugar, JavaScript utiliza **funciones** como clases. Definir una clase es tan fácil como definir una función. Para trabajar con propiedades dentro de la clase se utiliza la palabra reservada **this**, que se refiere al objeto actual. El acceso (lectura o escritura) a una propiedad desde fuera de la clase se hace con la sintaxis: ***NombreDeLaInstancia.Propiedad.***

Los ejemplos hasta ahora vistos únicamente han creado objetos individuales. En esta tarea se proponen las funciones constructoras como forma de crear muchos objetos del mismo tipo.

Nota: Se considera una buena práctica nombrar las funciones constructoras con una primera letra en mayúscula.

Ejer1. *Define una* ***función constructora*** *de objetos* ***Personaje*** *(propiedades: primerNombre, edadAños y estadoCivil) y crea los objetos personaje1 y personaje2 llamando a dicha función constructora. Muestra información de los mismos haciendo uso de la sintaxis adecuada.*

Ejer2. *Agrega a* ***personaje2*** *(sólo a este objeto) la propiedad nacionalidad. Comprobarlo visualizando valores en pantalla.*

---

[1] *Las* ***nuevas clases de EcmaScript6 sí que permiten una forma clara y expresiva de construir clases****. Son ampliamente usadas por muchos frameworks front basados en componentes (las veremos en typescript)*

Ejer3. *Agrega a* ***personaje2*** *(solo a él) el método "infoTotal()" que devuelve la concatenación de las propiedades con literales explicativos. Comprueba la invocación al método visualizando valores en pantalla.*

Ejer4. *¿Crees que puedes agregar a* ***Personaje*** *la propiedad colorOjos con el valor "verde" de la misma forma que has añadido la propiedad nacionalidad al objeto personaje2? Demuestra tu respuesta*

Recuerda:

- *No se puede agregar una nueva propiedad a un constructor de objetos de la misma manera que hemos agregado una nueva propiedad a un objeto existente en el Ejer2. Podremos hacerlo con la propiedad prototype como veremos en el siguiente apartado*
- *De igual manera, no se puede agregar un nuevo método a un constructor de objetos de la misma forma que hemos agregado un nuevo método a un objeto existente en el Ejer3. Podremos hacerlo con la propiedad prototype (la vemos en el siguiente apartado).*

Prototipos de objetos JavaScripts

Si has realizado la prueba propuesta en el ejercicio 4 habrás verificado que no es posible añadir una nueva propiedad al constructor con una simple operación de asignación.

Todos los objetos JavaScript heredan propiedades y métodos de un **prototipo**: los objetos de fecha heredan de Date.prototype, los objetos de matriz heredan de Array.prototype, los objetos de persona heredan de Person.prototype…

Así pues, cuando queremos agregar nuevas propiedades (o métodos) **a todos** los objetos existentes de un tipo determinado, debemos añadirlas **al prototipo** y esto lo conseguimos haciendo uso de la propiedad **prototype.** La propiedad prototype nos permite agregar nuevas propiedades (o métodos) a todos los objetos existentes de un tipo dado, o lo que es lo mismo, agregar nuevas propiedades (o métodos) al constructor de objetos. Dicho de otra forma, las propiedades deben establecerse a la propiedad prototype de la clase, para que la herencia funcione correctamente.

En el ejemplo que estás desarrollando, cualquier cosa asignada a Personaje.prototype quedará disponible para todas las instancias de ese constructor a través del objeto this.

Ejer5. *Modifica el prototipo del constructor definido en Ejer1 para que* ***todos*** *los objetos Personaje dispongan de la propiedad colorOjos ("azul") y el método infoTotal(). Crea un par de instancias de la clase y visualiza en pantalla la información. Pon en práctica para esta visualización, dos formas diferentes de visualizar los objetos.*