Programación orientada a objetos

con JavaScript

Francisco Arce
www.pacoarce.com

La palabra reservada this

Con this indicamos que estamos trabajando “con este” objeto, sin necesidad de nombrarlo.

Francisco Arce
www.pacoarce.com

Constructores

OOP con JavaScript

Francisco Arce
www.pacoarce.com

Constructores

En JavaScript todo el código que se encuentre dentro de una función se ejecuta inmediatemente y se puede considerar un “constructor”. O sea, no hay constructor propiamente dicho.

Francisco Arce
www.pacoarce.com

UML

OOP con JavaScript

Francisco Arce
www.pacoarce.com

UML

El lenguaje unificado de modelado o Unified Modeling Language (UML) permite visualizar las clases y las relaciones entre ellas.

Francisco Arce
www.pacoarce.com

UML

En JavaScript, el constructor y el nombre de l clase son los mismos.

Luego se enumeran las propiedades y posteriormente los métodos.

UML

Aunque en JS no se definen los tipos de dato, en el UML se indica el tipo de mismo.

Si el método no regresa ningún valor, se indica con :void.

Francisco Arce
www.pacoarce.com

Modificadores de acceso

OOP con JavaScript

Francisco Arce
www.pacoarce.com

Modificadores de acceso

En JavaScript, en específico en ECMAScript 5 y anteriores, no hay modificadores de acceso, por lo que todas las propiedades y métodos son públicos.

Francisco Arce
www.pacoarce.com

Modificadores de acceso

En JavaScript las funciones creadas dentro de la “clase” o mejor llamado métodos, son consideradas como otra variable.

Francisco Arce
www.pacoarce.com

Referencias a clases externas

OOP con JavaScript

Francisco Arce
www.pacoarce.com

Referencias externas

En JavaScript, podemos definir un método dentro de una “clase” y tener el código fuera de la misma, es decir, escribir el código en una función externa.

Francisco Arce
www.pacoarce.com

Prototyping

OOP con JavaScript

Francisco Arce
www.pacoarce.com

Prototyping

En JavaScript podemos añadir propiedades y métodos a una “clase” por medio del comando ***prototype***.

A la acción de añadir propiedades o métodos a una clase se le conoce como “prototyping”.

Francisco Arce
www.pacoarce.com

Prototyping

Al usar ***prototype***, no se duplica el código en las diferentes instancias creadas, pero todos pueden utilizar estos recursos añadidos.

Francisco Arce
www.pacoarce.com

Prototyping

En JavaScript, todas las funciones tienen un propiedad llamada ***prototype***, que es a su vez un objeto.

Francisco Arce
www.pacoarce.com

Prototyping

Se puede utilizar prototype solo después de haber creado la función de la clase, no antes.

Francisco Arce
www.pacoarce.com

Prototyping

Es una práctica común crear las propiedades dentro de la función de la clase, y añadir los métodos con ***prototype***.

Francisco Arce
www.pacoarce.com

Prototyping

SI nosotros modificamos un método haciendo referencia a una instancia, sólo será modificado o sobreescrito (overwriting) esa instancia, y no todas las demás instancias.

Francisco Arce
www.pacoarce.com

Propiedades y métodos estáticos

OOP con JavaScript

Francisco Arce
www.pacoarce.com

Propiedades y métodos estáticos

Una propiedad estática no es otra cosa que una variable añadida a la función de la clase (sin prototype), pero por lo general se escriben con mayúsculas.

Francisco Arce
www.pacoarce.com

Propiedades y métodos estáticos

Un método estático no es otra cosa que una función añadida a la función de la clase (sin prototype).

Francisco Arce
www.pacoarce.com

Propiedades y métodos estáticos

En ambos casos pueden ser llamados sin necesidad de crear una instancia, sólo referenciarla con el nombre de la función clase.

Francisco Arce
www.pacoarce.com

Propiedades privadas

OOP con JavaScript

Francisco Arce
www.pacoarce.com

Propiedades privadas

Aunque JavaScript no tiene modificadores de acceso propiamente dichos como los demás lenguajes orientados a objetos, podemos hacer propiedades privadas si las definimos dentro de la función de clase con la palabra reservada var.

Propiedades privadas

Es una buena práctica diferencias a estas propiedades de alguna manera. Por lo general se le antepone un guión bajo en el nombre.

Francisco Arce
www.pacoarce.com

Espacio de nombres o name space

OOP con JavaScript

Francisco Arce
www.pacoarce.com

OOP con JavaScript

Un espacio de nombres (name space) es un contenedor que permite asociar toda la funcionalidad de un determinado objeto con un nombre único.

Francisco Arce
www.pacoarce.com

OOP con JavaScript

En JavaScript un espacio de nombres es un objeto que permite asociarse a métodos, propiedades y objetos.

Francisco Arce
www.pacoarce.com

OOP con JavaScript

La idea de crear namespace en JavaScript es simple: Crear un único objeto global para las variables, métodos y funciones, convirtiéndolos en propiedades de ese objeto.

Francisco Arce
www.pacoarce.com

OOP con JavaScript

El uso de los namespace permite minimizar el conflicto de nombres con otros objetos haciéndolos únicos dentro de nuestra aplicación.

Francisco Arce
www.pacoarce.com

OOP con JavaScript

JavaScript es un lenguaje basado en prototipos que no contiene ninguna declaración de clase, como se encuentra, por ejemplo, en C++ o Java.

Francisco Arce
www.pacoarce.com

OOP con JavaScript

En su lugar, JavaScript utiliza funciones como clases. Definir una clase es tan fácil como definir una función.

Francisco Arce
www.pacoarce.com

OOP con JavaScript

Como una buena práctica, los nombres de espacio los escribiremos con mayúsculas.

Francisco Arce
www.pacoarce.com