CONTENIDO:

JavaScript

Unidad 9 - Objetos en Javascript

Vamos a introducirnos en un tema muy importante de Javascript como son los objetos. Es un tema que aun no hemos visto y sobre el que en adelante vamos a tratar constantemente pues todas las cosas en Javascript, incluso las más sencillas, las vamos a realizar a través del manejo de objetos. De hecho, en los ejemplos realizados hasta ahora hemos hecho grandes esfuerzos para no utilizar objetos y aun así los hemos utilizado en alguna ocasión, pues es muy difícil encontrar ejemplos en Javascript que, aunque sean simples, no hagan uso de ellos.

La programación orientada a objetos (POO) representa una nueva manera de pensar a la hora de hacer un programa. Javascript no es un lenguaje de programación orientado a objetos, aunque los utiliza en muchas ocasiones: podemos crear nuevos objetos y utilizar muchos que están creados desde un principio. Sin embargo la manera de programar no va a cambiar mucho y lo que hemos visto hasta aquí relativo a sintaxis, funciones, etc. puede ser utilizado igual que se ha indicado. Solo vamos a aprender una especie de estructura nueva.

1.– Introducción

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Con la POO tenemos que aprender a pensar las cosas de una manera distinta, para escribir nuestros programas en términos de objetos, propiedades, métodos y otras cosas que veremos rápidamente para aclarar conceptos y dar una pequeña base que permita soltarnos un poco con este tipo de programación.

Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otras personas se creó la POO. Que es una serie de normas de realizar las cosas de manera que otras personas puedan utilizarlas y adelantar su trabajo, de manera que consigamos que el código se pueda reutilizar.

La POO no es difícil, pero es una manera especial de pensar, a veces subjetiva de quien la programa, de manera que la forma de hacer las cosas puede ser diferente según el programador. Aunque podamos hacer los programas de formas distintas, no todas ellas son correctas, lo difícil no es programar orientado a objetos sino programar bien. Programar bien es importante porque así nos podemos aprovechar de todas las ventajas de la POO.

La POO no es difícil, pero es una manera especial de pensar, a veces subjetiva de quien la programa, de manera que la forma de hacer las cosas puede ser diferente según el programador.

Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real. Por ejemplo vamos a pensar en un coche para tratar de modelizarlo en un esquema de POO. Diríamos que el coche es el elemento principal que tiene una serie de características, como podrían ser el color, el modelo o la marca. Además tiene una serie de funcionalidades asociadas, como pueden ser ponerse en marcha, parar o aparcar.

Pues en un esquema POO el coche sería el objeto, las propiedades serían las características como el color o el modelo y los métodos serían las funcionalidades asociadas como ponerse en marcha o parar.

Por poner otro ejemplo vamos a ver cómo modelizaríamos en un esquema POO una fracción, es decir, esa estructura matemática que tiene un numerador y un denominador que divide al numerador, por ejemplo 3/2.

La fracción será el objeto y tendrá dos propiedades, el numerador y el denominador. Luego podría tener varios métodos como simplificarse, sumarse con otra fracción o número, restarse con otra fracción, etc.

Estos objetos se podrán utilizar en los programas, por ejemplo en un programa de matemáticas harás uso de objetos fracción y en un programa que gestione un taller de coches utilizarás objetos coche. Los programas Orientados a objetos utilizan muchos objetos para realizar las acciones que se desean realizar y ellos mismos también son objetos. Es decir, el taller de coches será un objeto que utilizará objetos coche, herramienta, mecánico, recambios, etc.

2.– Objetos y Clases

Objetos en POO

Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar (que viene de una mala traducción de la palabra instace que en inglés significa ejemplar). Por ejemplo, un objeto de la clase fracción es por ejemplo 3/5. El concepto o definición de fracción sería la clase, pero cuando ya estamos hablando de una fracción en concreto 4/7, 8/1000 o cualquier otra, la llamamos objeto.

Para crear un objeto se tiene que escribir una instrucción especial que puede ser distinta dependiendo el lenguaje de programación que se emplee, pero será algo parecido a esto.

miCoche = new Coche()

Con la palabra new especificamos que se tiene que crear una instancia de la clase que sigue a continuación. Dentro de los paréntesis podríamos colocar parámetros con los que inicializar el objeto de la clase coche.

Estados en objetos

Cuando tenemos un objeto sus propiedades toman valores. Por ejemplo, cuando tenemos un coche la propiedad color tomará un valor en concreto, como por ejemplo rojo o gris metalizado. El valor concreto de una propiedad de un objeto se llama estado.

Para acceder a un estado de un objeto para ver su valor o cambiarlo se utiliza el operador punto.

miCoche.color = rojo

El objeto es miCoche, luego colocamos el operador punto y por último el nombre e la propiedad a la que deseamos acceder. En este ejemplo estamos cambiando el valor del estado de la propiedad del objeto a rojo con una simple asignación.

Mensajes en objetos

Un mensaje en un objeto es la acción de efectuar una llamada a un método. Por ejemplo, cuando le decimos a un objeto coche que se ponga en marcha estamos pasándole el mensaje ponte en marcha.

Para mandar mensajes a los objetos utilizamos el operador punto, seguido del método que deseamos invocar.

miCoche.ponerseEnMarcha()

En este ejemplo pasamos el mensaje ponerseEnMarcha (). Hay que colocar paréntesis igual que cualquier llamada a una función, dentro irían los parámetros.

Clases en POO

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase. En los ejemplos anteriores en realidad hablábamos de las clases coche o fracción porque sólo estuvimos definiendo, aunque por encima, sus formas.

Propiedades en clases

Las propiedades o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

Métodos en las clases

Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

3.– Instanciar un objeto

Instanciar un objeto es la acción de crear un ejemplar de una clase para poder trabajar con él luego. Recordamos que un objeto se crea a partir de una clase y la clase es la definición de las características y funcionalidades de un objeto. Con las clases no se trabaja, estas sólo son definiciones, para trabajar con una clase debemos tener un objeto instanciado de esa clase.

En javascript para crear un objeto a partir de una clase se utiliza la instrucción new, de esta manera.

var miObjeto = new miClase()

En una variable que llamamos miObjeto asigno un nuevo (new) ejemplar de la clase miClase. Los paréntesis se rellenan con los datos que necesite la clase para inicializar el objeto, si no hay que meter ningún parámetro los paréntesis se colocan vacíos. En realidad lo que se hace cuando se crea un objeto es llamar al constructor de esa clase y el constructor es el encargado de crearlo e inicializarlo. Hablaremos sobre esto más adelante.

En Javascript podemos acceder a las propiedades y métodos de objetos de forma similar a como se hace en otros lenguajes de programación, con el operador punto (".").

Las propiedades se acceden colocando el nombre del objeto seguido de un punto y el nombre de la propiedad que se desea acceder. De esta manera:

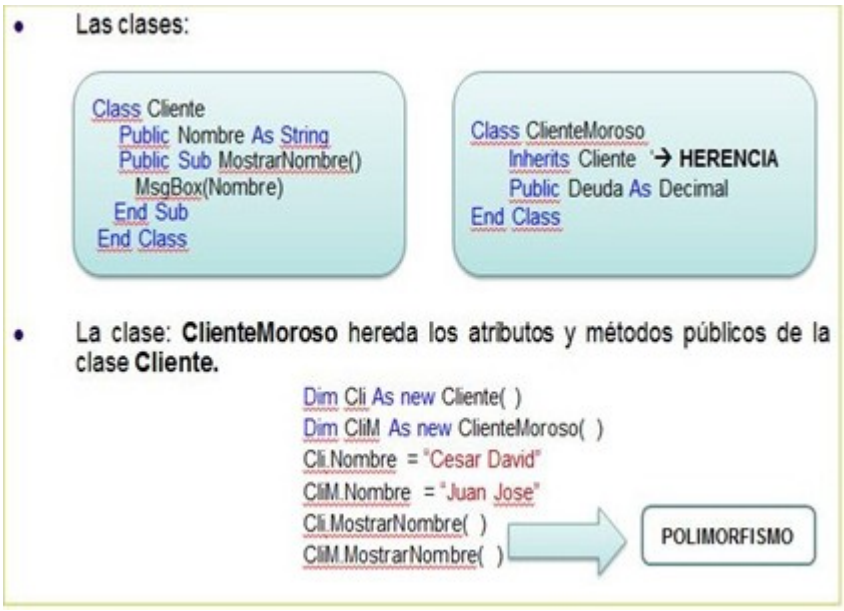
miObjeto.miPropiedad

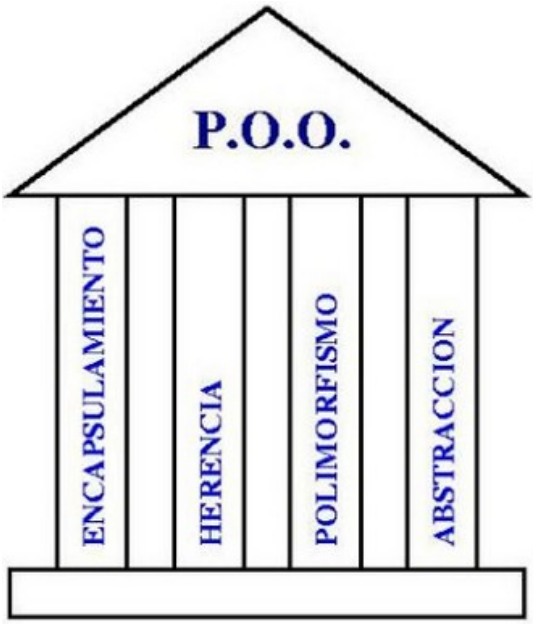
Para llamar a los métodos utilizamos una sintaxis similar pero poniendo al final entre paréntesis los parámetros que pasamos a los métodos. Del siguiente modo:

miObjeto.miMetodo(parametro1,parametro2)

Si el método no recibe parámetros colocamos los paréntesis también, pero sin nada dentro.

miObjeto.miMetodo()





4.– El objeto Date

JavaScript dispone de varias clases predefinidas para acceder a muchas de las funciones normales de cualquier lenguaje, como puede ser el manejo de vectores o el de fechas.

El objeto date nos va a permitir hacer manipulaciones con fechas: poner fechas, consultarlas... para ello, debemos saber lo siguiente: JS maneja fechas en milisegundos. Los meses de Enero a Diciembre vienen dados por un entero cuyo rango varía entre el 0 y el 11 (es decir, el mes 0 es Enero, el mes 1 es Febrero, y así sucesivamente), los días de la semana de Domingo a Sábado vienen dados por un entero cuyo rango varía entre 0 y 6 (el día 0 es el Domingo, el día 1 es el Lunes, ...), los años se ponen tal cual, y las horas se especifican con el formato HH:MM:SS.

Podemos instanciar un objeto Date vacío, o podemos crearlo dándole una fecha concreta. Si no le damos una fecha concreta, se creará con la fecha correspondiente al momento actual en el que se crea.

Se invoca así:

fecha = new Date();//creación de un objeto de la clase Date

Para instanciarlo dándole un valor, tenemos estas posibilidades:

var Mi_Fecha = new Date(año, mes);
var Mi_Fecha = new Date(año, mes, día);
var Mi_Fecha = new Date(año, mes, día, horas);
var Mi_Fecha = new Date(año, mes, día, horas, minutos);
var Mi_Fecha = new Date(año, mes, día, horas, minutos, segundos);

Si no utilizamos parámetros, el objeto fecha contendrá la fecha y hora actuales, obtenidas del reloj de nuestro equipo. En caso contrario hay que tener en cuenta que los meses comienzan por cero. Así, por ejemplo:

navidad06 = new Date(2014, 11, 25)

Métodos

El objeto Date dispone de varios métodos que nos permiten agilizar el utilización de fechas. Entre ellos vamos a destacar los siguientes:

Obtener la fecha actual.

getDate(). Devuelve el día del mes actual como un entero entre 1 y 31.

getDay(). Devuelve el día de la semana actual como un entero entre 0 y 6 .

getMonth(). Devuelve el mes del año actual como un entero entre 0 y 11.

getFullYear(). Devuelve el año actual como un número de cuatro dígitos.

Obtener la hora actual.

getHours(). Devuelve la hora del día actual como un entero entre 0 y 23.

getMinutes(). Devuelve los minutos de la hora actual como un entero entre 0 y 59.

getSeconds(). Devuelve los segundos del minuto actual como un entero entre 0 y 59 .

Asignar la fecha.

setDate(día_mes). Pone el día del mes actual en el objeto Date que estemos usando.

setDay(día_semana). Pone el día de la semana actual en el objeto Date que estemos usando.

setFullYear(año). Pone el año actual en el objeto Date que estemos usando .

setMonth(mes). Pone el mes del año actual en el objeto Date que estemos usando.

Asignar la hora.

setSeconds(segundos). Pone los segundos del minuto actual en el objeto Date que estemos usando.

setMinutes(minutos). Pone los minutos de la hora actual en el objeto Date que estemos usando.

setHours(horas). Pone la hora del día actual en el objeto Date que estemos usando.

5.– El objeto Math

Este objeto se utiliza para poder realizar cálculos en nuestros scripts. Tiene la peculiaridad de que sus propiedades no pueden modificarse, sólo consultarse. Estas propiedades son constantes matemáticas de uso frecuente en algunas tareas, por ello es lógico que sólo pueda consultarse su valor pero no modificarlo.

Propiedades

E. Número 'e', base de los logaritmos naturales (neperianos).

LN2. Logaritmo neperiano de 2.

LN10. Logaritmo neperiano de 10.

LOG2E. Logaritmo en base 2 de e.

LOG10E. Logaritmo en base 10 de e.

PI. Número PI.

SQRT1_2. Raíz cuadrada de 1/2.

SQRT2. Raíz cuadrada de 2.

Métodos

abs(numero). Función valor absoluto.

acos(numero). Función arcocoseno. Devuelve un valor cuyas unidades son radianes o NaN. 'numero' debe pertenecer al rango [-1,1], en otro caso devuelve NaN.

asin(numero). Función arcoseno. Devuelve un valor cuyas unidades son radianes o NaN. 'numero' debe pertenecer al rango [-1,1], en otro caso devuelve NaN.

atan(numero). Función arcotangente. Devuelve un valor cuyas unidades son radianes o NaN.

atan2(x,y). Devuelve el ángulo formado por el vector de coordenadas (x,y) con respecto al eje OX.

ceil(numero). Devuelve el entero obtenido de redondear 'numero' "por arriba".

cos(numero). Devuelve el coseno de 'numero' (que debe estar en radianes) o NaN.

exp(numero). Devuelve el valor enumero.

floor(numero). Devuelve el entero obtenido de redondear 'numero' "por abajo".

log(numero). Devuelve el logaritmo neperiano de 'numero'.

max(x,y). Devuelve el máximo de 'x' e 'y'.

min(x,y). Devuelve el mínimo de 'x' e 'y'.

pow(base,exp). Devuelve el valor baseexp.

random(). Devuelve un número pseudoaleatorio entre 0 y 1.

round(numero). Redondea 'numero' al entero más cercano.

sin(numero). Devuelve el seno de 'numero' (que debe estar en radianes) o NaN.

sqrt(numero). Devuelve la raíz cuadrada de número.

tan(numero). Devuelve la tangente de 'numero' (que debe estar en radianes) o NaN.

6.– El objeto Array

Este objeto nos va a dar la facilidad de construir arrays cuyos elementos pueden contener cualquier tipo básico, y cuya longitud se modificará de forma dinámica siempre que añadamos un nuevo elemento (y, por tanto, no tendremos que preocuparnos de esa tarea). Para poder tener un objeto array, tendremos que crearlo con su constructor, por ejemplo, si escribimos:

a=new Array(15);

tendremos creada una variable a que contendrá 15 elementos, enumerados del 0 al 14. Para acceder a cada elemento individual usaremos la notación a[i], donde i variará entre 0 y N-1, siendo N el número de elementos que le pasamos al constructor.

También podemos inicializar el array a la vez que lo declaramos, pasando los valores que queramos directamente al constructor, por ejemplo:

a=new Array(21,"cadena",true);

que nos muestra, además, que los elementos del array no tienen por qué ser del mismo tipo.

Por tanto: si ponemos un argumento al llamar al constructor, este será el número de elementos del array (y habrá que asignarles valores posteriormente), y si ponemos más de uno, será la forma de inicializar el array con tantos elementos como argumentos reciba el constructor.

Podríamos poner como mención especial de esto lo siguiente. Las inicializaciones que vemos a continuación:

a=new Array("cadena");
a=new Array(false);

Inicializan el array a, en el primer caso, con un elemento cuyo contenido es la cadena cadena, y en el segundo caso con un elemento cuyo contenido es false.

Lo comentado anteriormente sobre inicialización de arrays con varios valores, significa que si escribimos

a=new Array(2,3);

NO vamos a tener un array con 2 filas y 3 columnas, sino un array cuyo primer elemento será el 2 y cuyo segundo elemento será el 3. Entonces, ¿cómo creamos un array bidimensional? (un array bidimensional es una construcción bastante frecuente). Creando un array con las filas deseadas y, después, cada elemento del array se inicializará con un array con las columnas deseadas. Por ejemplo, si queremos crear un array con 4 filas y 7 columnas, bastará escribir:

a=new Array(4);
for(i=0;i<4;i++) a[i]=new Array(7);

y para referenciar al elemento que ocupa la posición (i,j), escribiremos a[i][j];

Propiedades

length. Esta propiedad nos dice en cada momento la longitud del array, es decir, cuántos elementos tiene.

prototype. Nos permite asignar nuevas propiedades al objeto String.

Métodos

join(separador). Une los elementos de las cadenas de caracteres de cada elemento de un array en un string, separando cada cadena por el separador especificado.

reverse(). Invierte el orden de los elementos del array.

sort(). Ordena los elementos del array siguiendo el orden lexicográfico.

IES MIGUEL
ROMERO ESTEO

Este material forma parte del currículo del módulo Implantación de Aplicaciones WEB correspondiente al ciclo formativo de grado superior "Administración de sistemas informáticos en red".

La información contenida en el magazine ha sido extraída de la Web y, en parte, de la experiencia profesional acumulada durante mis años de docencia.

Por tanto, es de uso público para tareas de docencia y aprendizaje.