Manejo de clases en ES6 JavaScript

Francisco Javier Arce Anguiano

ES6-0901-Clases

Dentro de ECMAScript vamos a tener definición de clases. Nos alejamos de lo que son los prototipos y ahora nos acercamos más a la programación orientada a objetos más apegada a Java o C++ y ese tipo de lenguajes.

Si nos vamos a nuestro editor de código vamos a guardar nuestro archivo con el nombre JSOOP-Clases y vamos a crear nuestras etiquetas HTML. Pulsamos la tecla tabulador vamos a poner aquí llave script clases y creamos nuestro script y aquí pues vamos a crear nuestra clase vamos a tener la palabra reservada ***class***. Vamos a definir generalmente pues ya habíamos visto que las clases la primera letra se ponen mayúscula.

Nombremos a nuestra clase como ***Gato*** y abrimos lo que son nuestros llaves y con eso nosotros ya vamos a tener una clase. Ya no necesitamos hacerla como función. Para crear una instancia podemos utilizar la palabra reservada ***var*** o también podemos utilizar la nueva palabra reservada ***let***, la cual define un objeto o variable dentro de dentro de llaves, es decir, un bloque. En este ejemplo sería indistinto y vamos a utilizar lo que es el var y nombramos a nuestra instancia como **Benito** es igual a new Gato.

En este caso como no tenemos ningún parámetro, ninguna función pues podemos utilizarla sin necesidad de los paréntesis característicos. Una clase, opcionalmente, no precisa argumentos en su definición, por lo tanto no le acompañan los paréntesis habituales.

No es necesario tampoco lo que es el punto y coma al final, pero es una buena práctica y como habíamos dicho es una buena práctica poner el nombre de clase en mayúsculas para diferenciarlo de otros tipos de objetos como funciones o como variables.

Esta clase no hay que confundir no es una función el navegador no lo va a permitir utilizar como navegador sino que se reserva con un constructor. El contenido de la clase se ejecuta automáticamente en modo estricto.

Las declaraciones de las clases no siguen la regla de las funciones de lo que es el ***hoisting*** y de forma implícita una clase se comporta como una constante. No es posible declararla más adelante en el código en el mismo programa, que es el ambiente el alcance o también en inglés se conoce como el ***scope***.

Como ya habíamos comentado, realmente no es un nuevo modelo de programación, sino este lo estamos agrupando de manera diferente internamente y funciona exactamente igual qué es lo que hemos visto en los clases anteriores. Más adelante vamos a seguir analizando lo que es esta poderosa forma de programar.

Vamos a guardar nuestro archivo y vamos ejecutarla en un navegador. Obviamente aquí no tenemos mayor salida.

Pero bueno, aquí no es más que por el encoding perdón que está puesto pero si le damos digamos Benito pues vamos a ver que lo estamos manejando exactamente como un objeto que de momento está vacío y si bien ahí tenemos nuestro viejo amigo el prototipo.

Así que en los próximas clases vamos a ver cómo podemos utilizar esta nueva estructura de las clases dentro de ECMA6.

Es muy importante que su navegador debe de ser compatible con ECMA6. Si entre por ejemplo a la página caniuse.com y buscamos Class vamos a ver por ejemplo que dentro de lo que es ECMA 6, dentro de las clases Internet Explorer ya está prácticamente eliminado. Lo que es el Opera Mini también prácticamente ya van de salida y todos los demás navegadores pues funcionan bastante bien.

Entonces hay que tener cuidado porque esta estructura no es compatible con Internet Explorer prácticamente en ninguna de sus versiones, pero con Edge funciona sin mayor problema.

ES6-0902-Instancias

Vamos a ver poco a poco lo que son las partes de nuestra programación orientada a objetos con lo que es el ES6 que temen le llaman ECMA 2015.

Si tomamos el ejemplo anterior vamos a poner aquí nuestro charset. Estamos viendo que estamos creando lo que es nuestra clase y estamos haciendo una instancia de Gato.

Por ejemplo, vamos a hacer otro con var que se llame Cucho y que sea igual a la instancia de un Gato… punto y coma. En este caso como no estamos pasando parámetros, a lo que es no hay una función constructora lo vamos a hacer más adelante.

Pues entonces no es necesario los paréntesis aquí. Sí también le ponemos los paréntesis no pasa absolutamente nada.

Podemos mandar, digamos lo que sería un typeof Benito y un ***typeof()*** de Cucho y ambos nos deben decir de qué son objetos.

También podemos utilizar digamos en Benito podemos utilizar el ***instanceOf*** de gato y un instanceof igual de Perro. Guardamos

Nos vamos a nuestro navegador y obviamente nos va a decir que es una referencia que no existe, la de perro. Vamos a mandarlo a la consola aquí a poner console.log() y aquí console.log() igual Consol punto.

Obviamente nos dice que [la clase] perro no existe. No podemos creer vamos a creer rápidamente lo que sería la instancia perro... perdón la clase de Perro, entonces simplemente le vamos a decir aquí Perro. Nos vamos a nuestro navegador, y refrescamos.

Obviamente nos va a decir que es objeto objeto las luces de los tipos con el typeof() nos van a decir que son objetos. No nos dice que son clase sino objetos y después nos dice true (verdadero) si es instancia de Gato y nos dice false (falso) cuando preguntamos si es instancia de Perro. Así que vamos a utilizar lo que es el new... vamos a utilizar lo que es el new para crear nuestros objetos o sus estancias. En este caso si no lleva parámetros podemos omitir los paréntesis y podemos utilizar lo que es el ***typeof()*** y lo que es instanceof para poder reconocer lo que son nuestras instancias de clases dentro de ECMA 6.

ES6-0903-Constructor

Vamos a ver poco a poco lo que son las partes de nuestra programación orientada a objetos con lo que es el ES6 que también le llaman ECMA 2015.

Así que lo vamos a ver cómo hacer una clase. Si tomamos el ejemplo anterior vamos a poner aquí nuestra etiqueta charset. Estamos viendo que estamos creando lo que es nuestra clase y estamos haciendo una instancia de ***Gato***.

Por ejemplo, vamos a hacer otra instancia con la palabra reservada ***var*** que se llame Cucho y que sea igual a la instancia de un ***Gato***. En este caso, como no estamos pasando parámetros, a lo que es no hay una función constructora, la cual la vamos a hacer más adelante. Pues entonces no es necesario los paréntesis aquí, pero sí también le ponemos los paréntesis no pasa absolutamente nada.

Podemos mandar, digamos lo que sería un ***typeof*** a la instancia Benito y un typeof() a la instancia de Cucho y ambos nos deben decir de qué son ***objetos***.

También podemos utilizar, digamos, en Benito podemos utilizar el ***instanceOf*** de Gato y un ***instanceof*** igual de Perro. Guardamos el archivo y nos vamos a nuestro navegador. Obviamente nos va a decir que es una referencia que no existe, la de Perro. Vamos a mandarlo a la consola aquí a poner ***console.log()*** y aquí ***console.log()***. Obviamente nos dice que [la clase] Perro no existe.
Vamos a crear rápidamente lo que sería la instancia perro... perdón la clase de Perro,. Entonces simplemente le vamos a decir aquí ***Perro***. Nos vamos a nuestro navegador, y lo refrescamos.

Obviamente nos va a decir que es objeto los tipos con el ***typeof()*** nos van a decir que son objetos. No nos dice que son ***clase*** sino objetos y después nos dice ***true*** (verdadero) si es instancia de ***Gato*** y nos dice ***false*** (falso) cuando preguntamos si es instancia de ***Perro***. Así que vamos a utilizar lo que es la palabra reservada ***new***... vamos a utilizar lo que es ***new*** para crear nuestros objetos o sus estancias. En este caso, si no lleva parámetros podemos omitir los paréntesis y podemos utilizar lo que es el ***typeof()*** y lo que es ***instanceof*** para poder reconocer lo que son nuestras instancias de clases dentro de ECMA6.

ES6-0904-GettersSetters

Otro punto de la programación orientada a objetos que nos va a simplificar mucho la nueva especificación ES6 van a ser los setter y los objetos que se utilizan para obtener y asignar valores a los atributos de nuestros objetos respectivamente.

En JavaScript ambos métodos se corresponden con los atajos sintácticos, que son funciones finalmente que no añaden funcionalidad adicional adicional alguna simplemente pues hay que tener cuidado con las referencias circulares así que si nos vamos a nuestro editor de código, vamos a partir de un archivo nuevo y vamos a guardarlo.

En este caso como JavaScript Getters y Setters, punto HTML y creamos lo que son nuestras etiquetas HTML. Vamos a poner aquí como Java Script setter y getter y ponemos nuestro charset con el meta charset ponemos nuestras etiquetas de script y vamos a hacer nuestra nuestra clase de Gato y vamos a poner lo que es el nombre es igual a Minino por omisión si viene vacío, pues va a tomar minino y lo que es el pelo, pues, vamos a poner que es por omisión negro.

Llave que abre y llave que cierra... para evitar referencias circulares. Vamos a utilizar el this, punto, con un guion bajo y vamos a escribir el nombre. Hay personas que le gusta ponerlo al inicio. Hay personas que les gusta ponerlo al final incluso llegan a poner doble guion bajo pues para diferenciar lo que va a hacer después las diferencias con el getter y el setter y vamos a decir que esto va a ser igual a nombre, punto y coma.

Y después el this, punto, guión bajo pelo vendría siendo igual a con la misma lógica que este que estamos siguiendo. Vamos a crear lo que es el setter que es cuando nosotros podemos modificar. El setter finalmente por lo que va a ser es una equivalente a la función set y después le decimos nombre y se observa pues ahí no ponemos el guión bajo ahí lo vamos a tomar y después recibimos el valor o la cadena y aquí entonces lo vamos a designar a this, punto.

Y aquí si ponemos el guion bajo el nombre va a ser igual al valor. Por lo general en esas funciones pues nosotros podemos hacer validaciones, podemos hacer muchas cosas, después vamos a utilizar lo que es el get y vamos a poner lo que es el set igual de pelo, vamos a ponerlo que es el valor llave que abre y que cierra, vamos a poner aquí this punto que un bajo pelo en ese caso si va con el guion bajo y pasabamos lo que es el valor.. punto y coma. Vamos a utilizar lo que es el toString(), que ya lo habíamos hecho entonces toString(), y va a recibir cuando pasamos a los dos llaves y vamos a decir return this, punto, guión bajo nombre... concatenamos con.. "tiene el pelo color"... concatenamos con this, punto y guión bajo, pelo… con punto y coma… así qué bueno fuera de lo que es nuestra clase, vamos a crear las dos instancias, vamos a decir let Benito es igual a new Gato y vamos a ponerlo.

Así, sin valores, después let Cucho es igual a new Gato y vamos a llamarlo como mucho Cucho. Y después de color rosa y su punto y coma, y vamos a poner lo que es el console.log() de Benito punto toString(), con los valores con punto y coma, y vamos a copiar esto mismo para Cucho.

Entonces pesamos aquí la cadena y escribimos Cucho. Bueno, como deberá de suponer, pues este vamos a abrirlo que es la consola y tenemos minino tiene el pelo color negro y Cucho tiene el color rosa como que hubiéramos pensado.

Como lo habíamos pensado ahora vamos a cambiar de Benito, punto pelo, es igual azul, punto y coma y Benito, punto, nombre, es Benito Bodoque y su punto y coma.

Y volvemos a utilizar, pues, la información de Benito. Aquí, repetimos, podemos hacer este validaciones de todo tipo aquí en el setters, guardamos nos vamos a nuestra página, refrescamos y entonces pues Benito Bodoque tiene el pelo azul. De acuerdo, entonces esa es una manera mucho más limpia de trabajar con las variables, simplemente hay que tratar de evitar las referencias circulares. Si ponemos nombres, nombres, nombres nuevos por todos lados pues a veces caemos aquí en ciertos problemas entonces eso lo solucionamos, pues poniendo a lo que es la variable interna de la clase con algún pequeño diferenciador, por ejemplo un guion bajo podemos hacer también una función que se llame get() que en este caso no es exactamente necesario lo que es el get().

Pero podemos decir color pelo ponemos llave que abre y llaves que cierra vamos a copiar esto mismo y en vez de lo que es el get() vamos a utilizar lo que es ahora color pelo y pero no lo vamos a utilizar con un paréntesis vamos a poner color pelo como si fuera una variable.

Aquí lo estamos recuperando por medio del get() guardamos nos vamos a nuestro navegador y refrescarnos y el resultado pues es lo mismo. Benito Bodoque tiene el pelo color azul. Esto quiere decir que pudimos manejarlo como si fuera una propiedad sin necesidad de pasarle parámetros y sin necesidad de pasarle lo que son los paréntesis porque lo estamos haciendo por medio de un get(), este color de pelo.

Como ve pues es una manera muy limpia de trabajar lo que son los gettes y los setters conocidos pues en el ambiente del desarrollo orientado a objetos de la programación orientada a objetos simplemente hay que tener precaución del problema de no caer en referencias circulares entonces si va a utilizar setter y getters, pues lo más sano es que haga una pequeña diferenciación en sus variables internas o viceversa una pequeña variación en lo que son los nombres de sus propiedades settlers y getters siempre nos van a regresar, siempre nos van a regresar como propiedades, aunque bueno sabemos que son métodos.

ES6-0906-MetodoEstatico

Otra característica que vamos a tener en la programación orientada a objetos dentro de lo que es el ES6 va a ser los métodos estáticos, los métodos estáticos. nosotros podemos llamarlos sin necesidad de crear una instancia. En los ejercicios anteriores creamos una instancia con el neón y después utilizamos propiedades y utilizamos métodos.

En este caso nosotros podemos sin necesidad de crear una instancia llamar alguno de los métodos por ejemplo la clase Math pues no necesitamos crear un objeto de tipo ni un math para utilizarlo.

Para poder utilizar el coseno el seno etcetc y Math viene dentro de lo que es el paquete estándar de JavaScript.

Nosotros también podemos hacer entonces métodos estáticos. Así que si nos vamos a nuestro editor de código vamos a crear un archivo nuevo y vamos a guardarlo como JavaScript orientado a objetos métodos estáticos punto HTML.

Guardamos creamos nuestras etiquetas HTML. Vamos a escribir como título JavaScript métodos estáticos y creamos nuestro meta charset, vamos a crear nuestro script nuestras etiquetas de script y vamos a crear nuestra clase de datos Class Gato, llave que abre llave que cierra y vamos a crear un método estático por medio de Static y el nombre del método.

Por ejemplo vamos a poner this y podemos pasar lo que es un valor y vamos a escribir aquí.

Vamos a suponer como dice y vamos a poner un retorno que diga los gatos dicen "miau", de acuerdo, para poder llamar a nuestra clase y entonces aquí vamos a escribir, sin necesidad de crear el objeto, vamos a decir gato, punto, dice y es un método estático, de acuerdo.

Y aquí vamos a mandarlo a la consola. console.log(), punto y coma y lo ponemos antes del punto y coma. Guardamos y nos vamos a nuestro navegador.

Abrimos nuestra consola. En este caso vamos a Web Developer y web Console y entonces ahí están "los gatos dicen miau". Como puede ver pues es una de las muchísimas aplicaciones que podemos nosotros definir lo que es un método de forma estática y lo podemos llamar sin necesidad de crear la instancia dentro de JavaScript ECMA 6.

ES6-0907-Herencia

Vamos a ver ahora lo que es el proceso de la herencia en lo que es la programación orientada a objetos con el ECMA 6 y en este caso vamos a utilizar la sentencia extends, que se utiliza pues en programación orientada a objetos como PHP o en otros lenguajes.

Así que si nos vamos a nuestro editor de código. Vamos a trabajar con un archivo en blanco y vamos a guardarlo como JavaScript orientado a objetos herencia punto HTML y vamos a hacer nuestras etiquetas HTML.

Vamos a poner aquí como título JavaScript herencia y ponemos también lo que es nuestro charset con meta charset escribimos nuestras este etiquetarse script y ebrias aquí vamos a crear una clase class que sea "mascota" y dentro de "mascota" vamos a crear en el constructor constructor lo que hemos visto del constructor que es parte del ES6.

Vamos a poner los parámetros para hacer cualquier nombre pero bueno para el común como un objeto vacío en el caso de que no nos mande ninguna información pues subjetivación y aquí vamos a utilizar el constructor entonces vamos a poner el paréntesis y las llaves es igual a parms, digamos este es el constructor y vamos a poner los datos.

Vamos a poner por ejemplo el nombre del animalito nombre dos puntos va a ser igual a this y vamos a utilizar un bajo nombre o lo que vemos visto perder y íbamos a ponerle sin nombre por omisión, coma, género.

Vamos a poner aquí un punto y un bajo género y ponemos sin género y después especie, dos puntos de un bajo especie es igual sin especies sin especial y no lleva la última como aquí lo que estamos haciendo entonces es pasándolo por el constructor vamos a hacer también lo que son los getter y los setters. Vamos a utilizar por ejemplo get nombre y vamos a poner un return de this, punto, de un bajo nombre que también le habíamos visto lo que son setters y getters,

vamos a copiar esto es getter y vamos a pegarlo aquí y ponemos género y aquí abajo sería un bajo género.

Pegamos, también lo que sería especial especie y aquí bajo especia... especie y especie especie, especie especie, muy bien.

Entonces ahí tenemos nuestros Getter aquí vas a ponerle comentario que son los getters y vamos a copiar tú mismo para ser nuestros Settlers. Ese va a ser digamos que nuestra función pues base y aquí vamos a poner el valor y nombre sería igual valor, valor igual aquí ponemos este setter y también setter, género vamos a copiar esto mismo… entonces aquí vamos a borrar el return y vamos a poner aquí valor y pesamos valor y entonces que hoy tenemos nuestra es nuestra clase base o madre y vamos a hacer entonces al menos una de las [clases], no llevamos mucho tiempo Class en Gato y decimos extends Mascota, llave que abre y cierra y vamos a poner aquí un constructor, constructor que tenga igual vamos a poner los parámetros es igual a vacío, llave que abre que cierra y vamos a utilizar lo que super().

Más adelante lo que hace superes de que la clase madre y recupera los parámetros que tengamos en la función correspondiente en este caso del constructor, entonces decimos súper pasamos como parámetro, género dos puntos y vamos a poner que es un Gato() y su ahora que ya tenemos nuestras nuestras clases y la clase heredada pues vamos a generar un objeto vamos a ponerle Benito y este va a ser igual a new Gato y ya por omisión pues debe de ser un gato porque aquí lo estamos pasando como género, vamos a pasar consola.log() y vamos a imprimir primero que sea Benito, punto, nombre nos lo hemos modificado. después Benito, punto, género y después Benito punto, especie y su punto y coma.

Y ahora vamos a modificarlo vamos a poner Benito en tu nombre. Ahí estamos utilizando los getters y los setters de este de la clase madre y vamos a poner que se llama Benito Bodoque como Benito punto género no sé decir que es un macho, punto y coma. Podemos copiar esta misma línea pero a ver antes y después el género no lo modificamos porque sigue siendo un gato. Guardamos.

Vamos a revisarlo y bueno aquí me faltó...l perdón. Igual aquí le puse género Gato y el y no en el género pues macho y hembra no gato aquí es especial que bueno gato creo que son especies una subespecie pero bueno especie un gato o felino y entonces ahora si generó macho.

Guardamos, nos vamos a nuestro navegador, refrescamos y ahora se nos dice sin nombre, sin género. Gato, Benito Bodoque, macho, gato.

Vamos a crear pues también lo que es una leyenda lo podemos hacer desde desde la clase madre. Vamos a poner aquí enunciado y vamos a modificar el link Trunk llave que abre y cierra y vamos a poner algo similar a lo que tenemos aquí y aquí podemos por ejemplo return. "Mi nombre es", espacio concatenamos con lo que sería this, punto, guión bajo, nombre más,

coma, "soy un ", contactemos con el más, ***this***, punto, guión bajo especie, más "y soy" con él más que un bajo this perdón, this punto que un bajo género, punto y coma, así que vamos a revisar muy bien, y aquí entonces vamos a poner Benito toString() y aquí también Benito toString(), guardamos nos vamos a nuestro navegador refrescamos y "mi nombre es sin nombre Soy un gato y soy sin género y después nos dice Mi nombre es Benito Bodoque Soy un gato y soy macho.

Así que bueno con eso es una de las muchas muchas posibilidades que vamos a tener por medio del extends de heredarse una clase que generalmente se llama clase padre o clase madre con lo que serían diferentes subclases que es una de las facilidades que vamos a tener en ES6 con JavaScript.

ES6-0908-Super

Vamos a ver ahora a más detalle lo que es la sentencia súper cuando utilizamos la herencia. Entonces super va a llamar tanto propiedades o métodos de la clase padre a la clase hijo.

Hemos comentado de que en español la clase debería de ser hija y madre pero tomando como es en inglés pues dicen "Son and Father" pero bueno se puede ser como se quiera llamarlo la clase superior y la clase inferior heredada. En este caso nosotros podemos llamar de desde el método constructor de la clase padre con super() y después los argumentos, que fue lo que hicimos en el ejercicio anterior pero también podemos llamar a cualquier otro tipo de método que se encuentre en la clase padre hacia la clase hijo.

Aquí lo importante es que cuando estamos utilizando super() desde el constructor debe de utilizarse antes de la palabra clave this. Esto es muy importante porque muchas veces nos da error y lo primero que tenemos que hacer es llamar a super() y bueno también como hemos visto puede ser utilizado para llamar a otras funciones, es decir métodos desde el objeto padre hacia la función hacia la clase heredada.

Así que si nos vamos a nuestro editor de código, vamos a crear lo que es un un ejemplo muy sencillo. De hecho viene en lo que es la documentación de Mozilla que creo que es la mejor documentación para JavaScript y vamos a guardarlo como script orientado a objetos super() junto a ello lo guardamos creamos nuestras etiquetas HTML.

Vamos a poner aquí este script super y creamos nuestro charset con tabulador y ahora aquí vamos a escribir el script y abrimos aquí pues vamos a hacer una clase. Como ya hemos dicho la programación orientada a objetos explica con gatos o con geometría, entonces vamos a hacer algo de geometría y vamos a decir clásico de rectángulo, rectángulo y bueno hay hasta una teoría bien fundamentada de que este no un rectángulo no puede quedar a un cuadrado pero es simplemente un ejemplo constructor.

Abrimos paréntesis y vamos a tomar altura y ancho o anchura y abrimos las llaves y vamos a poner con el this.

Aquí sí podemos utilizar this. nombre es igual a rectángulo rectángulo y su punto y coma. Después vamos a poner this, punto, altura es igual altura y después this, punto, ancho es igual a ancho punto y coma. Entonces ahí tenemos nuestra función constructora vamos a mandar el clásico saludo, cerramos llave que abre y cierra y vamos a poner para el console, punto, los, vamos escribir "Hola soy un" y aquí podemos poner coma, this, punto, nombre rectángulo cuadrado, lo que sea. Cerramos. Vamos a tener también por medio del set un área y en este caso pues vamos a regresar return, this, punto, altura por this, punto, ancho o anchura y ahí lo tenemos y también vamos a poner con el set, área el valor value o es escribir y vamos a decir que entonces la altura, this, punto, altura siendo igual a lo que es this, punto, ancho menos lo que es el cuadrado del valor Math junto a sqrt(). Ya tenemos nuestra clase.

Ahora vamos a utilizar lo que es la clase Cuadrado Cuadrado y extends y heredamos de rectángulo. Abrimos y vamos a hacer lo que es el constructor y vamos a tomar solamente que en ese caso es un cuadrado pues de tener los dos lados iguales [obviamente] y vamos a creer lo que es este clásico.

Vamos a llamar a un this simplemente como altura, punto y coma, no necesitamos modificarlo y después vamos a tomar un super. Que en este caso pues le vamos a pasar a altura y el ancho pues que ser lo mismo entonces altura, coma, altura con super, punto y coma, y después le vamos a decir de this, punto, name, cuadrado y punto y coma que es el. Este es el ejemplo clásico que tenemos y crear una instancia vamos a decir este podemos utilizar let o var como quiera usted y puede poner mi cuadrado y cuadrado es igual a new cuadrado y estamos pasando digamos 10. Lo guardamos y nos vamos a nuestro navegador refrescamos y bueno esta mi consola y entonces este es muy muy común que lo hagamos que es el “reference error” o que debe de llamarse el constructo rantes de cualquier this.

Entonces regresamos y vamos a comentar la primera instrucción debe de ser súper buena en ese caso pues ya no tenemos el error que es algo muy muy común que tengamos vamos a decir mi cuadrado punto saludo, que como se ve pues lo tenemos aquí en el en el rectángulo en la clase padre. “Hola soy un rectángulo” o porque no lo modificamos si bien aunque tengamos aquí name cuadrado en realidad no lo estamos modificando porque lo estamos pasando después, entonces como lo estamos llamando aquí pues entonces no este no lo está modificando

Si por ejemplo utilizamos área en vez de saludo vamos a utilizar aquí un console punto log y vamos a llamar a área como lo estamos haciendo como set y get pues no lo hacemos como una función entonces aquí lo aplicamos y nos dice efectivamente que es 100 si se está modificando lo que dices de lo que es la altura y lo que es el ancho, la anchura pero no el no las las propiedades que tengamos aquí, por ejemplo vamos a modificar ahora por medio de mi

cuadrado, punto, altura, área, perdón, área y le voy a poner y es igual a 50 recordemos que lo estamos haciendo con set y get entonces no es su función sino lo manejamos como una propiedad vamos a revisarlo y aquí tengo un error pues aquí value o valor value, guardamos y lo ejecutamos en nuestro navegador a esta calculando el área entonces
es hay que simplemente tener cuidado con el súper, el súper nos sirve para llamar lo de lo que tengamos en la clase padre ya sea propiedades podemos llamar también métodos.

Así que ese sí de las nuevas características que vamos a tener en ES6 para la programación orientada a objetos en JavaScript.