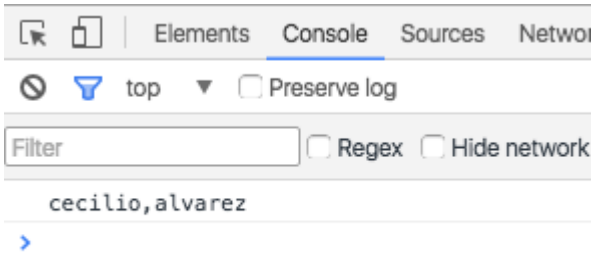
El uso de JavaScript apply vs call ,es una pregunta muy habitual cuando profundizamos en JavaScript . ¿Para que sirven estos dos métodos tan peculiares?. Vamos a construir un ejemplo que ayude a aclarar las dudas .

JavaScript Objetos

Imaginemonos que tenemos el siguiente objeto de JavaScript:

```
var objeto = {

nombre: "cecilio",
apellidos: "alvarez",
imprimir: function() {

console.log(this.nombre + "," + this.apellidos);
}
}

objeto.imprimir();
```

Invocamos al método imprimir y nos imprime por pantalla los datos de la persona.



Ahora bien , ¿Es la forma correcta de realizar la operación?. La realidad es que no , ya que se trata de un código poco flexible. Una solución mejor será construir una función de este estilo.

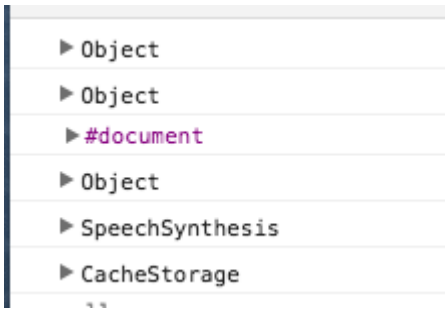
```
var objeto = {
nombre: "cecilio",
apellidos: "alvarez",
imprimir: function() {

for (var key in this) {

if (typeof(this[key]) != "function") {

console.log(this[key]);
}
}
}
}

objeto.imprimir();
```

Esta función es capaz de imprimir todas las propiedades del objeto (simplificación). Con lo cual nos vale para cualquier objeto. ¿Tiene sentido que se encuentre ubicada en este objeto en concreto? . Está claro que no por lo tanto podemos extraerla del objeto y convertirla en una función normal.

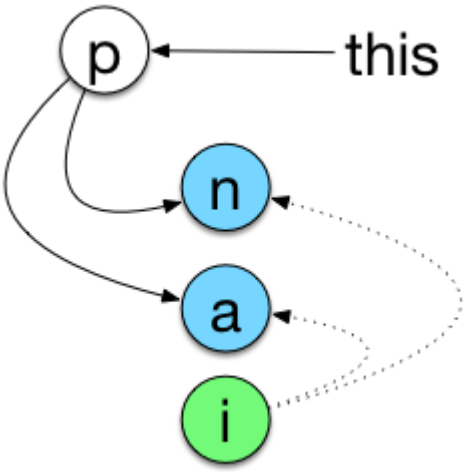
```
function imprimirPropiedades() {
```

```
for (var key in this) {

if (typeof(this[key]) != "function") {

console.log(this[key]);
}

}
}

var objeto = {

nombre: "cecilio",
apellidos: "alvarez",
imprimir:function() {

imprimirPropiedades();
}
}
objeto.imprimir();
```

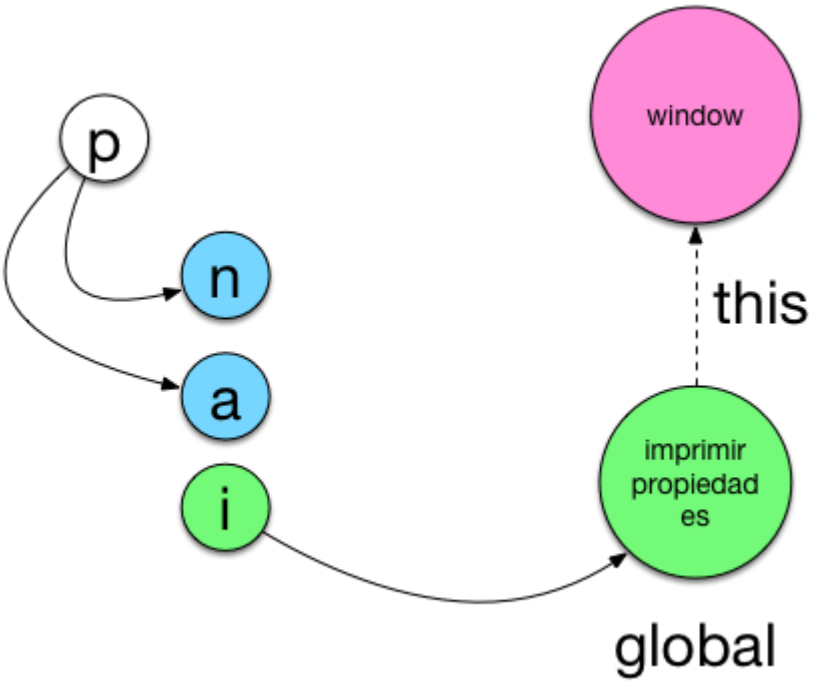
El resultado es para muchas personas sorprendente ya que no imprime los valores del nombre y apellidos sino un grupo enorme de valores que pertenecen al objeto window.



¿Qué ha pasado exactamente?. Para entenderlo es necesario entender como funciona el objeto this en JavaScript y que contexto usa como referencia. En el primer caso todo es muy sencillo se trata de un objeto y por lo tanto la variable this de la función apunta a sus propiedades.



En cambio en el segundo caso nos encontramos con una función que esta fuera del objeto y por lo tanto la variable this apunta al objeto windows ya que es el ámbito o scope general.



Es evidente que no queríamos ese comportamiento . ¿Como podemos solventarlo?.

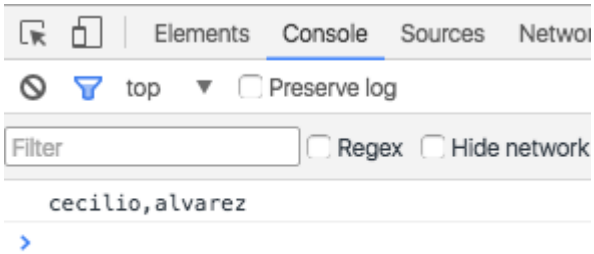
JavaScript apply vs call

Podemos obligar a JavaScript a que nos invoque esa función asignándole nosotros el contexto adecuado, para ello usaremos la función call de JavaScript.

```
function imprimirPropiedades() {

for (var key in this) {

if (typeof(this[key]) != "function") {

console.log(this[key]);
```

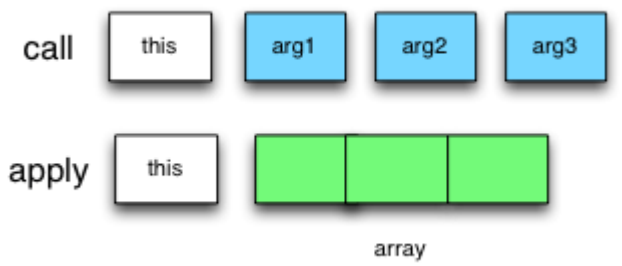
```
}

}
}

var objeto = {

nombre: "cecilio",
apellidos: "alvarez",
imprimir:function() {

imprimirPropiedades.call(this);
}
}
objeto.imprimir();
```

Como podemos observar delegamos en la función imprimirPropiedades usando javascript call y pasando la referencia a this que deseamos (la de nuestro propio objeto). El resultado volverá a ser coherente.



Esta casuística se produce en muchas situaciones. Acabamos de ver como usar call. ¿Qué diferencia existe entre JavaScript apply vs call?. Ambas funciones evidentemente soportan que se les pase parámetros adicionales. Estos parámetros son los argumentos que las

funciones que vamos a invocar soportan. Call pasa los argumentos de forma natural , en cambio apply los pasa en formato array.



Otros artículos relacionados: jQuery Promises , JavaScript Prototypes , JavaScript ES6 API