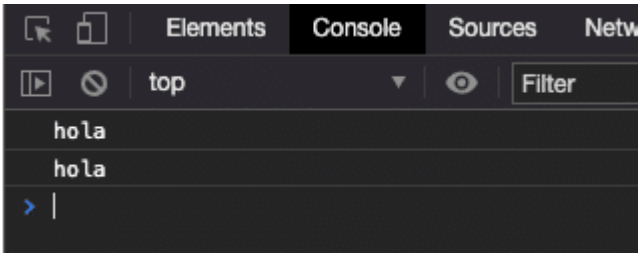
El uso de JavaScript var a la hora de declarar variables en JavaScript es lo más habitual del mundo . Ahora bien como todo lo que ocurre en el mundo de JavaScript … siempre tiene muchas peculiaridades .Vamos a hablar un poco a detalle de que implica declarar una variable con var en JavaScript para ello nos apoyaremos en varios ficheros HTML de prueba.

JavaScript var y ambitos

Normalmente cuando uno habla con desarrolladores de JavaScript y les pregunta que ámbito tiene una variable que se declara como var a; la mayoría suele responder que tiene un ámbito global. Algo que es muy muy razonable ya que si escribimos este código:

```
var a="hola"
console.log(a);
console.log(window.a)
```

El resultado por la consola será curioso . Primero se imprimirá el mensaje “hola” por la consola y segundo se volverá a imprimir un hola por la consola.



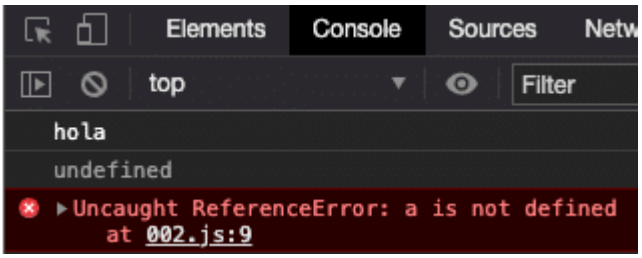
Esto se debe a que la variable adquiere un ámbito global y pasa a nivel del navegador a pertenecer al objeto Window.

Funciones y Ambitos

¿Es esto realmente así? . Vamos a ver un ejemplo complementario en este caso tendremos un código muy similar pero la variable la declaramos dentro de una función:

```
function f1() {
    var a="hola";

    console.log(a);
}
f1()
console.log(window.a)
console.log(a);
```

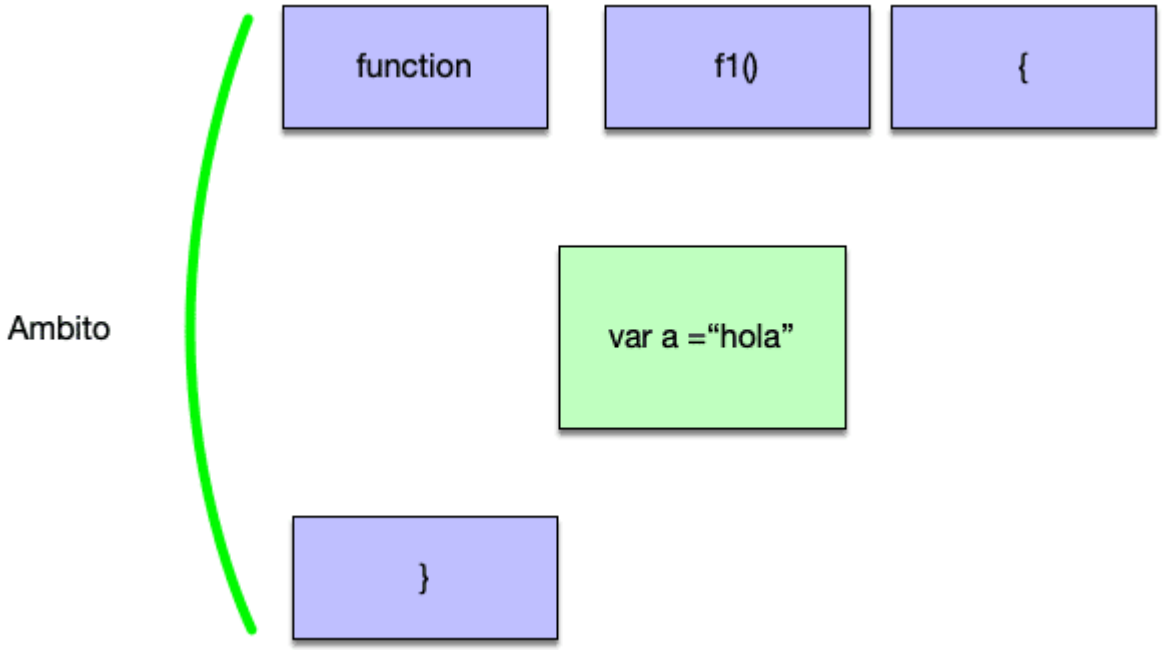
En este caso tenemos algunas sorpresas en la ejecución del código de JavaScript.



Nos imprime hola por la consola , si luego intentamos imprimir windows.a nos dice que es una variable undefined. Por lo tanto o sorpresa la variable no es “global” como pensábamos anteriormente. No solo eso sino que si intentamos acceder a ella de forma directa nos dirá que no esta definida ¿Qué esta pasando?

JavaScript var y sus lios

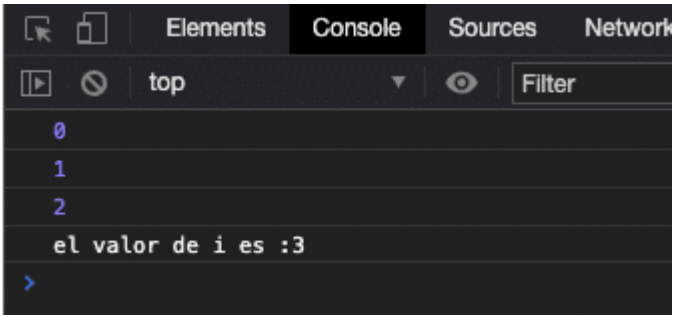
Cuando declaramos una variable con var la variable tiene un ámbito de función. Es decir existe dentro de la función que la declara , no es una variable global.



¿Entonces porque pensamos que es una variable global? .Bueno porque habitualmente nos encontramos con códigos de este estilo:

```
function f1() {
   for (var i=0;i<3;i++) {

    console.log(i);
   }
   console.log("el valor de i es :"+i);
}
f1()
```

El resultado en la consola es :

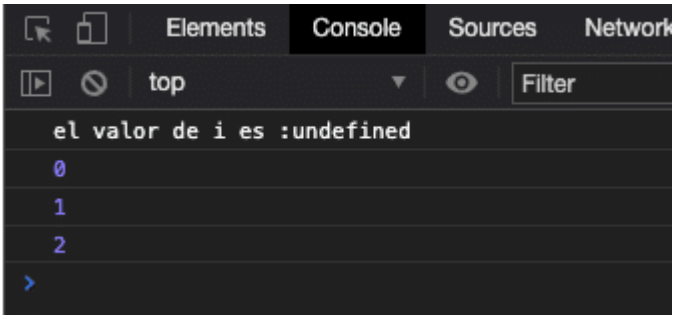


La variable la hemos declarado dentro de un bucle for y se puede acceder a ella desde fuera. Da la sensación de que tiene un ámbito global. Sin embargo no es cierto lo que tiene es un ámbito de función y por eso una vez declarada desde cualquier lugar de la función puede ser accedida .

No solo eso sino que si cambiamos el código por este :

```
function f1() {
    console.log("el valor de i es :"+i);
   for (var i=0;i<3;i++) {

    console.log(i);
   }

}
f1()
```

Al ejecutarla la consola nos mostrará :

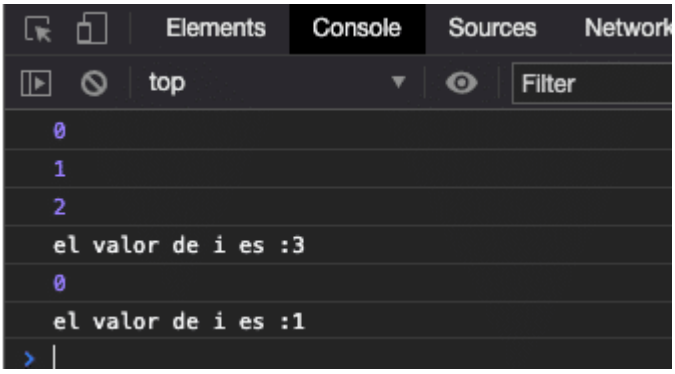


Es decir la variable no tiene valor pero existe . Esto se debe a las capacidades de hoisting de

Javascript. Que hace que las variables se suba su declaración al principio de la función. No solo eso sino que también puede suceder lo siguiente :

```
function f1() {
   for (var i=0;i<3;i++) {

    console.log(i);
   }

   console.log("el valor de i es :"+i);

   for (var i=0;i<1;i++) {

    console.log(i);
   }


   console.log("el valor de i es :"+i);

}
f1()
```

Estaríamos volviendo a declarar la variable i en un nuevo bucle e incrementando su valor a 1 . JavaScript no tiene problema con esto lo procesa , redeclara la variable en el ámbito de la función y todo se ejecuta correctamente.



Tengámoslo en cuenta cuando trabajemos con JavaScript y recordemos que a partir de JavaScript ES6 podemos declarar las variables como let.

Otros artículos relacionados

- JavaScript for loop y sus opciones
- Javascript Module Browser y modularidad
- JavaScript Map ES6 y el concepto de diccionario
- Curso de Introducción a Ajax