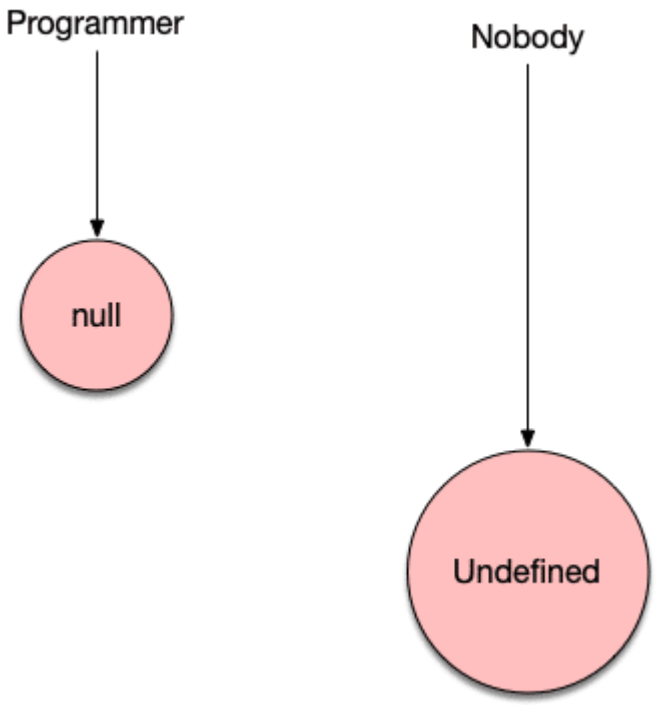
JavaScript null vs undefined ¿Cuál usar? . Normalmente cuando trabajamos con Javascript estamos cansados de ver variables que tienen un valor null y variables que tienen un valor undefined. ¿Qué diferencia existe entre los dos conceptos? . Ambos son conceptos que definen la ausencia de valor para una variable. Por ejemplo nosotros podemos tener algo tan sencillo como:

```
var a;
console.log(a);
```

Si ejecutamos el programa nos mostrará por la consola:

undefined

¿Qué quiere decir esto? . Nosotros no hemos inicializado la variable por lo tanto cabe pensar que devolvería null por la consola. Lamentablemente no es así la consola muestra undefined. Todos estamos acostumbrados a usar null en lenguajes de programación como Java o C# .Sin embargo undefined existe para diferenciar el hecho de no tener asignado un valor y el programador haya decidido no asignarlo.



De la situación en la que no tenemos un valor porque no hemos decidido asignarlo en ningún momento. Intentemos clarificar un poco más las cosas. Supongamos que el código es :

```
var a=null;
console.log(a);
```

En este caso el resultado será null ya que ha sido el programador el que lo ha inicializado

JavaScript null vs undefined

¿Qué sentido tiene undefined entonces? La realidad es que tiene mucho sentido , vamos a ver varios ejemplos . Empecemos por la sencilla definición de un array.

```
lista=[];
console.log(lista[2]);
```

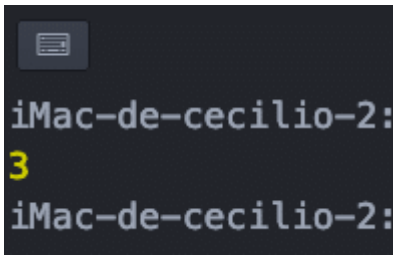
Estamos construyendo un array en estos momentos el array no tiene ningún elemento asociado que sucederá si accedemos a la posición 2 del array . Es evidente que Javascript no quiere que el programa de un error irrecuperable como pasaría en Java . Así pues como no es algo que el programador haya definido previamente nos devolvera :

undefined

Este mismo dato se puede corregir si antes asignamos un valor a esa posición :

```
lista=[];
lista[2]=3;
console.log(lista[2]);
```

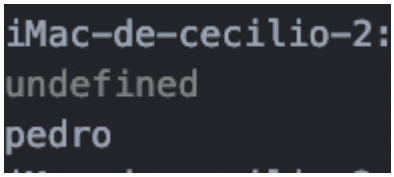
El resultado será :



Lo mismo sucede en casuísticas en las que definimos objetos. Recordemos que javascript permite el uso de propiedades dinámicas.

```
objeto={};
console.log(objeto.nombre);
objeto.nombre="pedro";
console.log(objeto.nombre);
```

Sino definimos la propiedad e imprimimos su valor por pantalla el resultado será undefined . Ahora bien podemos luego asignarle un valor y podremos acceder a él sin problemas.



Tengamos siempre en cuenta lo peculiar que es JavaScript al trabajar y como se comporta y recordemos que como programadores debemos usar null y esperar que algunas situaciones devuelvan undefined:

Otros artículos relacionados

1. Promise chaining en JavaScript
2. JavaScript Destructuring ,un concepto interesante
3. JavaScript reduce y su flexibilidad
4. JavaScript Arrow Function y scopes
5. JavaScript