



Navigation ☰

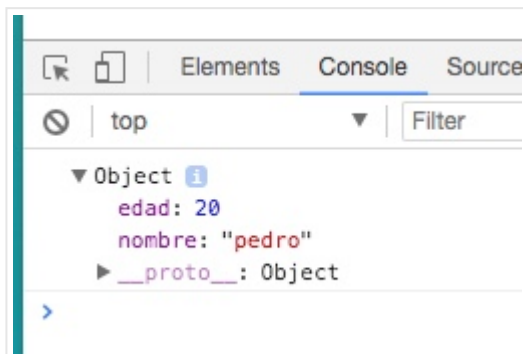
JavaScript Console log y el manejo de Objetos

por **Cecilio Álvarez Caules** sobre 5 octubre, 2017 en **Conceptos Importantes**

Todos usamos **JavaScript Console Log** para imprimir los típicos mensajes de consola en JavaScript. Sin embargo muchas veces **nos olvidamos de las posibilidades que la consola de JavaScript nos aporta**. Vamos a ver algunos ejemplos que nos ayudarán a clarificar. Para ello empezaremos con la operación más sencilla a realizar es imprimir un objeto por la consola.

```
1 | var objeto= {nombre:"pedro",edad:20};  
2 | console.log(objeto);
```

El resultado se muestra en la consola:



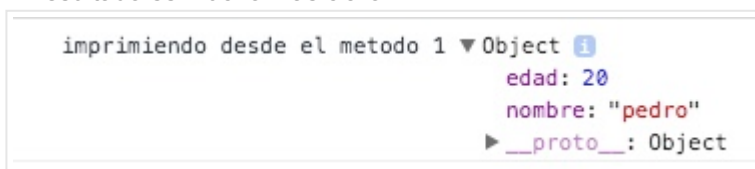
Aunque nos imprimir con claridad los datos del objeto, no es la mejor forma de imprimir los datos de un objeto en pantalla. ¿**Por qué?** porque únicamente imprime la información del objeto y no nos aporta información sobre el contexto. Vamos a ver como mejorarlo.

JavaScript Console Log

Podemos usar console.log y utilizar el comodín de sustitución de objeto, vamos con ello:

```
1 | var objeto= {nombre:"pedro",edad:20};  
2 | console.log("imprimiendo desde el metodo 1 %o",objeto);
```

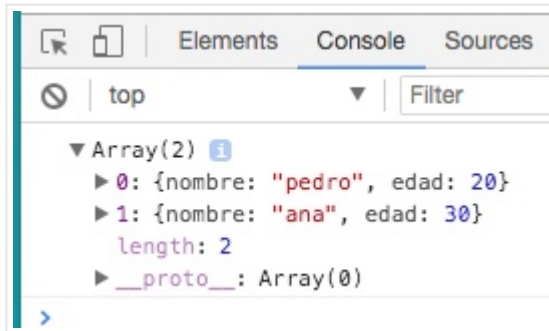
El resultado es mucho más claro:



JavaScript Console Log Arrays

De igual forma podemos trabajar con arrays de objetos , la forma que tiene de imprimir la información la consola es bastante clara:

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cooki](#)[ACEPTAR](#)

Podemos clarificarlos utilizando console.table():

```
1 var array= [{nombre:"pedro",edad:20},{nombre:"ana",edad:30}];
2 console.table(array);
```

(index)	nombre	edad
0	"pedro"	20
1	"ana"	30

► Array(2)

Como podemos ver la consola es una clase muy flexible . Otra opción interesante cuando trabajamos con objetos es la posibilidad de comprimir bloques de consoles.log con console.group().

```
1 var array= [{nombre:"pedro",edad:20},{nombre:"ana",edad:30}];
2 var array2= [{nombre:"gema",edad:50},{nombre:"ana",edad:40}];
3 var array3= [{nombre:"marco",edad:50},{nombre:"ana",edad:40}];
4 console.group();
5 console.table(array);
6 console.table(array2);
7 console.table(array3);
8 console.groupEnd();
9
10 console.group();
11 var objeto= {nombre:"pedro",edad:20};
12 var objeto2= {nombre:"pedro",edad:30};
13 console.log(objeto);
14 console.log(objeto2);
15 console.groupEnd();
```

El resultado como podemos observar se encarga de agrupar los bloques de información.

▼ console.group			004.js:4
(index)	nombre	edad	004.js:5
0	"pedro"	20	
1	"ana"	30	
▶ Array(2)			
▼ console.group			004.js:6
(index)	nombre	edad	
0	"pedro"	20	
1	"ana"	30	
▶ Array(2)			

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cooki](#)[ACEPTAR](#)

▼ console.group			004.js:10
▶ {nombre: "pedro", edad: 20}			004.js:13
▶ {nombre: "pedro", edad: 30}			004.js:14

Pudiendo comprimir el que deseemos:

🔊	top	▼	Filter
▶ console.group			
▼ console.group			
▶ {nombre: "pedro", edad: 20}			
▶ {nombre: "pedro", edad: 30}			
>			

Todos estos métodos nos serán muy útiles cuando trabajemos con la consola. La consola de JavaScript siempre nos sorprende con sus curiosidades.

Otros artículos relacionados:

1. [JavaScript console time y rendimiento](#)
2. [JavaScript setTimeout vs setInterval](#)
3. [¿Qué es un JavaScript Bundle?](#)
4. [La consola de Javascript](#)



0

COMPART

Subscribe

Síguenos en LinkedIn y Twitter o suscríbete al RSS.

Related Posts:

- [El concepto de Java Package Encapsulation](#)
- [Spring 5 Hello World](#)
- [Webpack y la gestión de dependencias en JavaScript](#)
- [JavaScript High Order Functions y su manejo](#)
- [Java Herencia vs Interfaces](#)

[◀ Spring Boot WAR sin Microservicios](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cooki](#)[ACEPTAR](#)

Deja un comentario

Name (required)

Email (will not be published) (required)

Website

[Submit Comment](#)

Buscar

[POPULAR](#)

**Mis Cursos de Java para desarrolladores**

27 JULIO, 2017

**Cursos de Java cupones de descuento ,packs y mini curso gratuito**

7 SEPTIEMBRE, 2017

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cooki](#)[ACEPTAR](#)

1 SEPTIEMBRE, 2017

**JPA DTO (Data Transfer Object) y JPQL**

14 JULIO, 2017

**Spring Boot WAR sin Microservicios**

29 SEPTIEMBRE, 2017

**Spring 5 Hello World**

22 SEPTIEMBRE, 2017

**El concepto de EJB Event y como desacoplar servicios**

25 JULIO, 2017

**Java 8 Factory Pattern y su implementación**

26 JUNIO, 2017

**Java Stream forEach y colecciones**

15 JUNIO, 2017

Contacto

contacto@arquitecturajava.com

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cooki](#)[ACEPTAR](#)

Redes Sociales

Síguenos en LinkedIn y Twitter o suscríbete al RSS.

© 2017 Arquitectura Java. All Rights Reserved.

Diseñado por [Clickea](#)