**Proceso de selección Walmeric**

**Selección candidatos JAVASCRIPT**

Con el fin de conocer el nivel técnico de cada candidato IT para la plantilla de Walmeric, se remite este documento donde, tras varias pruebas de conocimientos técncios, podemos sacar una evaluación completa del nivel de cada persona.

Es muy importante intentar completarla lo máximo posible para tener una visión completa del nivel técnico.

Este documento incluye tres tipos de cuestionarios:

* **1. Preguntas de tests.** En las pruebas de tres se muestran 4 o 5 opciones a contestar (a, b, c, d y en algunos casos, e). Solo una opción es correcta. Si te has apuntado por Infojobs muchas ya te sonarán, pero igualmente es importante que contestes todas.
* **2. Pregunta de lógica.** Es una prueba de razonamiento lógico donde se pone un enunciado con una pregunta y se debe contestar a la pregunta razonándola.
* **3. Prueba de código.** Esta es la parte que más tiempo lleva, así que conviene que la hagas en un ambiente tranquilo donde puedas concentrarte. También puedes, además de realizar la prueba, enviarnos muestras de repositorio donde poder ver también tu trabajo, de forma que tengamos más información de cara a la evaluación de tu candidatura.

# 

# 

# Preguntas de test

*A continuación, encontrarás 10 preguntas de test con 4 o 5 opciones cada una (a, b, c, d y, en ocasiones, e); solo una de las opciones es correcta. Por favor, subraya la respuesta que consideres idónea para responder a cada pregunta en un color distintivo (como por ejemplo amarillo). Ejemplo:*

*Pregunta: Mi nombre es:*

1. *Pepito*
2. *Jaimito*
3. *Fulanito*
4. *Menganito*

**1. ¿Cuáles de estas son marcas para la inserción del código JavaScript en las páginas HTML?**

1. < javascript \_code > y < /javascript\_code >
2. < script > y < /script >
3. < ?script > y < script? >

**2. En JavaScript, para darle el nombre a una variable, objeto o función, debemos tener en cuenta que:**

1. No se pueden usar mayúsculas
2. JavaScript no distingue entre mayúsculas y minúsculas
3. JavaScript diferencia entre mayúsculas y minúsculas

**3. Para terminar las instrucciones en Javascript se utiliza:**

1. Un punto y coma
2. Un punto y coma o un salto de línea
3. La sentencia End.

**4. En JavaScript, ¿cómo se llama el método que comprueba una expresión regular y devuelve true si se cumple?**

a) check()

b) exec()

c) run()

d) test()

**5. En JavaScript, ¿cómo se obtiene la versión del navegador?**

a) browser.appVersion

b) browser.version

c) navigator.appVersion

d) navigator.version

**6. En JavaScript, ¿cómo se escribe un bucle "for" para contar desde 1 hasta 5?**

a) for(i = 1; i <= 5; i++)

b) for(i <= 5; i++)

c) for i = 1 to 5

d) Las anteriores respuestas no son correctas

**7.En JavaScript, ¿cómo se redondea el número 7.25 al entero más cercano?**

a) round(7.25)

b) Math.round(7.25)

c) Math.rnd(7.25)

d) rnd(7.25)

**8.Respecto a los navegadores web, el BOM es el:**

a) Byte Object Model

b) Browser Object Model

c) Binary Object Model

d) Las anteriores respuestas no son correctas

**9. Identifica la función de la etiqueta de HTML <noscript>**

a) La etiqueta <noscript> se emplea para definir un contenido que resume la funcionalidad de un script: este contenido se muestra sólo si el navegador no soporta scripting

b) La etiqueta <noscript> desactiva en el navegador la capacidad de ejecutar un script

c) La etiqueta <noscript> puede ser utilizada por los diseñadores para indicar que una página web emplea scripting: si se utiliza <noscript>, el navegador procesará la página web más eficientemente

d) Todas las respuestas son correctas

**10. La estandarización de JavaScript es realizada por**

a) ECMA

b) Mozilla

c) The Script Consortium

1. W3C

# Prueba de lógica

*Si 2 carpinteros pueden construir 2 cajones en 2 horas. ¿Cuántos carpinteros se necesitan para construir 18 cajones en 6 horas?*

Respuesta

Como cada carpintero puede hacer 1 cajón cada 2 horas en las seis horas finales pueden hacer 3 cajones.

Para conseguir los 18 cajones teniendo en cuanta que cada carpintero hace 3, necesitaríamos para cumplir el objetivo 6 carpinteros, pero como ya tenemos dos solo será necesario aumentar el numero en cuatro carpinteros mas

# 

# Prueba de código

*El objetivo de esta prueba es que demuestres tu capacidad para escribir código que sea mantenible. Entre otras cosas evaluaremos tu calidad escribiendo código, la limpieza, su estructuración, el diseño, la gestión de errores, el rendimiento, …*

***No deberías utilizar frameworks o librerías*** *que no sean nativas de JS. También nos gustaría que incluyeses, si es posible, tests* ***unitarios*** *de al menos alguna de las clases que desarrolles.*

*La prueba consiste en el* desarrollo de una librería sencilla en js*. Se ofrecen dos opciones. Sólo es necesario desarrollar una de ellas.*

***OPCIÓN A***

*Librería que gestione la carga asíncrona de módulos javascript.*

*Requisitos: ​​*

* *La programación debe realizarse en JS puro, sin librerías ni frameworks de terceros.*
* *La librería deberá ejecutarse a partir de un objeto/función en un Namespace propio en window.*

*Funcionalidad:*

* ***load (library, callback).*** *carga de una librería, con un callback (función) que se ejecutará una vez cargada.*
* ***Loadla ([lib1, lib2, ...], callback).*** *carga de varias librerías. dependientes una de otras por el orden que ocupen en el array: no se puede cargar una librería sin que las anteriores hayan sido cargadas. Una vez cargadas todas, se ejecutará el callback.*

*Se valorará:*

* *Las buenas prácticas en la codificación.*
* *El uso de patrones JS (elegir el/los adecuados).*
* *La creatividad a la hora de resolver esta problemática.*
* *Funcionalidades extra que aporten valor a esta librería demo.*

***OPCIÓN B***

*Librería que genera un formulario con campos dinámicos.*

*Requisitos:*

* *La programación debe realizarse en JS puro, sin librerías ni frameworks de terceros.*
* *La librería deberá ejecutarse a partir de un objeto/funcion en un Namespace propio en window.*

*Funcionalidad:*

* ***createForm([field1, field2, ...], callback).*** *genera un formulario con campos dinámicos.*
  + *fieldX. cada parámetro field debe ser un objeto con las siguientes opciones:*

*{*

*type : "text" // String. Tipos de input: text / number / email...*

*name : "phone" // String. Atributo "name" del campo*

*id : "campo\_telefono" // String. Id del elemento: #id*

*required : true // Bool. SI el campo es requerido: true | false*

*}*

* ***​callback​.*** *Funcion que se ejecutará tras crear el formulario*
* *Retorno. La función debe retornar un objeto con acceso al elemento formulario <form>*

*Se valorará:*

* *Las buenas prácticas en la codificación.*
* *El uso de patrones JS (elegir el/los adecuados).*
* *La creatividad a la hora de resolver esta problemática.*
* *Funcionalidades extra que aporten valor a esta librería demo. (por ejemplo: posibilidad de crear varios formularios y que los ids sean dinámicos incrementales)*