

¡Felicidades!

Aprobaste el curso. Ya puedes acceder a tu [diploma digital](#).

9

Calificación

18 / 20

Aciertos

1. Aunque JS nunca dejará de ser "extraño", entendiendo cómo funciona por dentro tendremos menos confusión.

Verdadero



2. ¿Para qué creamos objetos en JavaScript?

Todas las respuestas son correctas.



3. ¿Qué es __proto__ en JavaScript?

El mecanismo para que las instancias hereden los métodos y atributos de sus prototipos.



4. ¿Cuál de las siguientes afirmaciones sobre los objetos en JavaScript es INCORRECTA?

Los objetos literales NO son instancias de ningún prototipo.



5. ¿Qué son los objetos literales en JavaScript?

Prototipos/moldes para crear instancias.

Repasar

6. ¿Qué son las clases en JavaScript?

Una sintaxis más amigable para los prototipos de JavaScript.



7. ¿Cuáles son las palabras clave que diferencian a una función "normal" de un prototipo en JavaScript?

this y prototype.

Repasar

8. ¿Qué es programación orientada a objetos?

Un paradigma de programación (imperativo).



9. ¿Cuáles son ventajas de utilizar programación orientada a objetos?

Orden, reusabilidad y menos líneas de código a largo plazo.



10. ¿Qué es un paradigma de programación?

Modelos para resolver problemas comunes.



11. Completa la frase. "Todos los paradigmas de programación..."

Intentan resolver los problemas, falencias o incomodidades de los paradigmas históricamente anteriores.



12. ¿Cuál de las siguientes es una mejor forma de calificar nuestro código JavaScript?

Legibilidad.



13. ¿Cuáles son los pilares de la POO?

Abstracción, encapsulamiento, herencia y polimorfismo.



14. Los atributos __proto__ en los objetos de JS pueden contener otro atributo __proto__ por dentro. Esto es:

Verdadero



15. Los módulos de ECMAScript 6 nos permiten:

Aplicar encapsulamiento a nivel de archivos.



16. ¿Cómo podrías crear un getter que exponga al atributo "privado" `_name` en JavaScript?

```
get name() { return this._name; }
```



17. ¿Cuál de las siguientes líneas de código para acceder al atributo de un objeto en JavaScript es INCORRECTA?

```
objeto->atributo
```



18. ¿Qué son los prototipos en JavaScript?

Moldes para crear objetos.



19. ¿Qué podemos guardar dentro de los objetos?

Todas las respuestas son correctas.



20. ¿Si todos los arrays tienen el método `.push` dentro de su atributo `__proto__`, cuál es la forma más cómoda de ejecutarlo desde el array asignaturas en JavaScript?

```
asignaturas.push()
```




[Ver menos](#)

Cursos que podrían interesarte

Curso Intermedio de

Programación orientada a objetos con JavaScript


Platzi

 Curso Intermedio de Programación Orientada a Objetos en JavaScript
Por Juan David Castro Gallego

Platzi

Curso de **ECMAScript**

Historia y Versiones de JavaScript

 Curso de ECMAScript: Historia y Versiones de JavaScript
Por Oscar Barajas Tavares

Curso de Manipulación de Arrays en JavaScript

Por Nicolás Molina

Platzi

 Curso de Manipulación de Arrays en JavaScript
Por Nicolás Molina

¡Felicidades!

Aprobaste el curso. Ya puedes acceder a tu [diploma digital](#).

9

Calificación

18 / 20

Aciertos

1. ¿Qué son las propiedades estáticas?

Métodos y atributos que podemos llamar sin necesidad de crear una instancia del prototipo.



2. ¿Cuál de los siguientes métodos estáticos de Object nos permite listar los nombres de las propiedades en "objetito"?

Object.keys(objetito)



3. ¿Cuál de los siguientes métodos estáticos de Object nos permite listar los nombres y valores de las propiedades de "objetito" en forma de arrays?

Object.entries(objetito)



4. ¿Cuál de los siguientes métodos estáticos de Object nos permite listar los atributos "ocultos" en las propiedades de "objetito"?

Object.getOwnPropertyDescriptors(objetito)



5. ¿Cuál de los siguientes métodos estáticos de Object nos permite definir o editar los atributos "ocultos" en la propiedad "patito" de "objetito"?

Object.defineProperty(objetito, "patito", { /* ... */ })



6. ¿En qué memoria se guardan los objetos de JavaScript?

Heap



7. ¿En qué memoria se guardan los nombres de las variables en JavaScript?

Stack



8. ¿Si `const patito = "Donald"`, en qué memoria se guarda "Donald"?

Heap

Repasar

9. ¿Qué es recursividad?

Cuando una función se llama a sí misma.



10. ¿Cuál de los siguientes es un problema importante que debemos evitar con ciclos y funciones recursivas?

No programar una validación o condición de salida.



11. ¿Cómo creamos getters y setters en un prototipo?

Dentro del prototipo usamos `Object.defineProperty(this, "nombrePropiedad", { /* ... */ })`, fuera del prototipo con `Object.defineProperty(NombrePrototipo.prototype, "nombrePropiedad", { /* ... */ })`.



12. ¿Cómo creamos getters y setters en una fábrica de objetos literales?

Retornando un objeto con `get nombrePropiedad() { /* ... */ }` y `set nombrePropiedad() { /* ... */ }`.



13. ¿Qué es una fábrica de objetos?

Una función creadora de objetos.



14. ¿Cuál de las siguientes respuestas NO FUNCIONA para crear propiedades privadas con fábricas de objetos en JavaScript?

Usar la palabra reservada `private` antes del nombre de la propiedad "privada".



15. ¿Cuál de las siguientes afirmaciones es VERDADERA con respecto a "ser" vs. "tener" cuando evaluamos la identidad de nuestros objetos en JavaScript?

"Ser" requiere validaciones mucho más específicas y complejas para evitar problemas de seguridad.

Repasar

16. ¿Cómo identificamos objetos con Duck Typing en JavaScript?

Validando propiedad por propiedad si el objeto cumple ciertas condiciones mínimas.



17. ¿Para qué sirve `instanceof` en JavaScript?

Para validar si un objeto es una instancia de algún prototipo en específico.



18. ¿Cuál es el problema de crear copias de objetos con `JSON.parse` y `JSON.stringify`?

Que no copian los métodos.



19. ¿Cuál es el problema de copiar objetos en JavaScript con el operador de asignación?

Que copiamos la referencia en memoria del objeto original, no las propiedades del objeto como tal.



20. ¿Qué pasa cuando usamos `Object.entries` para listar un objeto con métodos que llaman a `this`?

Esto hará referencia al array donde se encuentran la `key` y el `value` del método, no al objeto original.



[Ver menos](#)