1. ¿Cómo funciona “This” en JavaScript?
This es una palabra reservada que hace referencia al objeto en que estamos trabajando, para invocar las propiedades o métodos que tenga dicho objeto.

Particularmente en Javascript, This puede tener varias alternativas para asumir valores, su funcionalidad dependerá de las siguientes situaciones:

Existen 5 tipos de enlazamiento y JavaScript va a ejecutar el primero que se cumpla en el siguiente orden:

1) Lexical Binding (Arrow Functions): se produce cuando a los métodos las escribimos como funciones flecha “=>”. Las Arrow Functions se ejecutan en el mismo contexto en el que fueron creadas, creando un enlazamiento fuerte que después no se puede cambiar con otros métodos. Se recomienda no crear un arrow function en un contexto global, porque el This va a quedar enlazado a un contexto global en el navegador y windows no tiene las propiedades de mi objeto al cual yo quiero invocar.

2) New Binding (Instanciar objetos): Cuando instanciamos usando la palabra reservada new, estamos invocando al objeto, JavaScript crea un objeto vacío e invoca a la función constructora con este objeto con el valor de This.

3) Explicit Binding (Invocación Indirecta): Sirve para que nosotros elijamos que objeto queremos que sea This cuando ejecutamos una función, nos permite cambiar el contexto de manera explícita, por ejemplo cuando pasamos métodos como parámetros de otras funciones y nosotros queremos que This siga invocando a un método correcto. Hay dos funciones que nos ayudan y es .call que recibe el valor para This en el contexto que queremos y una lista n de parámetros; en segundo lugar tenemos .apply que recibe el valor de This en el contexto que queremos y un array. el más común es usar .call();

   call y apply invocan a la función en ese momento. aquí aparece .bind que nos retorna una nueva función con el valor que nosotros le digamos y JavaScript no lo invoca en ese momento.

4) Implicit Bindig (invocación implícita): Es cuando invocamos al método de un objeto. Si invocamos a la función de un objeto utilizando justo antes un punto “.”, y justo después del objeto que contiene esa misma función, entonces This asumirá el valor del objeto que figura a la izquierda de ese punto que tecleamos. Parece redundante, pero es específico, nos sirve para cuando tenemos un objeto con funciones propias y este se encuentra dentro de otro objeto con otras funciones. podríamos invocar varios métodos de objetos que estén anidados, y usaríamos varios *This* asumiendo valores distintos, según cada objeto.

5) Default Binding (invocación directa)
   Si tenemos una función que contiene un This que no está definido, y la invocamos

directamente, el This tendra el valor por defecto es el objeto global, en nuestro navegador será *windows*, no es recomendable usar *This*, si queremos referirnos al objeto global en vez de escribir This seria mejor opcion escribir *windows*

El objetivo que tenemos como desarrolladores es de enlazar el objeto a una función que contiene “This”, para que éste se ejecute en un contexto correcto.
La función del “This” dependerá el tipo de enlazamiento que hagamos con el objeto, o sea, saber que valor asumirá This a la hora de ejecutar la función.

2. ¿Cuál es la diferencia entre variables null, undefined y undeclared?

Null: es un valor primitivo de JavaScript, indica la intencionalidad de ausencia de un valor por parte del programador. la ausencia de valor en una variable declarada
Undefined: Es un valor primitivo de JavaScript y es una variable que ha sido definida pero no le asignamos un valor por el momento y que con seguridad tendrá un valor en algún otro momento. Undefined es un valor asignado por JavaScript cuando detecta que no tiene valor.
Undeclared: Ocurre cuando se intenta acceder a una variable que no ha sido declarada mediante var, let o const.

3. ¿Qué es un «clousure» y cuándo o por qué usaría uno?

Es una función autoejecutable que encapsula una serie de variables o funciones para trabajar con ellas. Estas definiciones locales únicamente serán accesibles si son devueltas con el operador return, esto nos permite tener variables cuasi-privadas, es decir, hacemos que sea difícil acceder desde afuera.

Se utilizaba mucho antes de que aparecieran las clases de javascript en ecmascript 6. Es un patrón muy parecido al de clases en POO con el fin de dificultar el acceso.

4. Escriba un programa en JavaScript que dada la entrada (por ejemplo):
El valor de la propiedad «nombre» de cada objeto debe extraerse, sólo en el caso que el valor de la propiedad «ciudad» sea «Córdoba» y almacenarse en un array ordenado por edad (de menor a mayor) y por orden alfabético.
Es decir, la salida debe ser:
["Juan", "María", "Pedro", "Ana", "Patricia"]

Respuesta: Actividad desarrollada en archivos adjuntos.

5. ¿Qué función cumple el doctype y cuántos tipos conoce?

Indica al navegador cómo debe interpretar el código html que contiene al documento. Es una etiqueta obligatoria, de no incluirse el navegador tratará de renderizar el contenido en modo a prueba de fallos.

Yo particularmente suelo usar el **<!DOCTYPE html>** en la primer línea del documento html, aunque existen otras versiones que casi no se usan como:

El doctype HTML 4.01 estricto.

El doctype HTML 4.01 transicional.
Los doctypes XHTML 1.0 estricto y transicional.
Los doctypes HTML 4.01 y XHTML 1.0 frameset.

6. ¿Cuál es la diferencia entre HTML y XHTML?

HTML y XHTML son las dos variantes que existen en el lenguaje HTML 5.

HTML significa Hyper Text Markup Language. XHTML significa Extensible HyperText Markup Language. XHTML, es estricta y extiende de HTML y XML, no permite a los desarrolladores mostrar sintaxis mal escritas invalidando las mismas y HTML. En XHTML los nombres y atributos van en minúsculas, las referencias a entidades también deben escribirse en minúsculas.

7. Generalmente, ¿por qué es buena idea agregar la etiqueta <link> dentro de la etiqueta <head> y la etiqueta <script> justo antes de cerrar la etiqueta <body>?

Por una cuestión de optimización, los enlaces que hacemos en el head con <link> serán leídos con prioridad por el código HTML, los <script> nos permiten insertar código JavaScript en nuestro archivo HTML, suelen escribirse antes de finalizar el body porque pueden generar algún tipo de demora si lo tenemos en el head en caso de que el código JavaScript sea muy pesado, al tenerlo al final del body la carga y recarga de la página se hace mucho más rápido.

8. Cree una página con dos preguntas propias. Su encuesta debe primero recopilar los datos personales de los participantes (nombre, dirección, sexo, fecha de nacimiento, una contraseña, etc.) seguida de cuatro preguntas para responder. Se valora la variedad de elementos usados en la página. Use CSS para mejorar el aspecto de su formulario.

Respuesta: Actividad desarrollada en archivos adjuntos.

9. ¿Cuál es la diferencia entre clases e identificadores en CSS?
Cuando queremos aplicar estilos a ciertas etiquetas de nuestro body, podemos utilizar las clases para que esos cambios de estilos impacten sobre varios elementos que pueda tener la clase que estamos invocando, es decir hacer cambios en forma general, pero si queremos cambiar el estilo de un solo elemento en particular dentro de una clase sin modificar los otros elementos entonces usamos un identificador en ese elemento específico, para poder hacer cambios de estilos de forma particular.

10. ¿Cuál es la diferencia entre las posiciones relative, fixed, absolute y static para un elemento dado?

Relative: (Posición relativa): Los elementos posicionados con posición relativa son desplazados una cantidad dada de su posición normal en el documento, pero sin que su desplazamiento afecte a otros elementos

Absolute: (Posición absoluta): Estos elementos posicionados de forma absoluta son removidos del flujo de esta manera, los demás elementos se posicionan como si el mismo no existiera. El elemento posicionado absolutamente se posiciona relativamente a su ancestro posicionado más cercano (es decir, el ancestro más cercano que no es static). Si no hay ningún ancestro posicionado se ubica relativo al bloque contenedor inicial.

Fixed: (Posición fija): El posicionamiento fijo es similar al posicionamiento absoluto, con la excepción de que el bloque contenedor del elemento es el viewport. Esto puede usarse para crear un elemento flotante que se mantiene en la misma posición independientemente del desplazamiento.

Static:(Posición estática):El elemento es posicionado de acuerdo al flujo normal del documento. Las propiedades top, right, bottom, left, and z-index no tienen efecto. Este es el valor por defecto.