99 preguntas de entrevista para Desarrolladores Web

Mariano Álvarez
Alfredo Bonilla
Freddy Montes

Introducción

Acerca de los autores
- Mariano Álvarez
- Freddy Montes
- Alfredo Bonilla

HTML
- ¿Qué es HTML5?
- ¿Cómo puedo incluir CSS en un HTML?
- ¿Dónde puede ser incluida la etiqueta de script en HTML?
- ¿Cuál es la diferencia entre los atributos async y defer en la etiqueta de script?
- ¿Cuál es la diferencia entre la etiqueta de <b> y ?
- ¿Qué etiqueta es necesaria para que el sitio sea responsivo?
- ¿Qué es un SVG?
- ¿Qué son atributos de una etiqueta HTML?
- ¿Cuál es la diferencia entre class y id?
- ¿Qué es HTML semántico?
- ¿Cómo se establece el idioma de un documento?
- ¿Qué es ARIA? ¿Para qué sirve?
- ¿Cómo se puede hacer una página accesible?
- ¿Para qué sirve la etiqueta <base>?
- ¿Qué es un iframe?
- ¿Cómo funciona la técnica de inlining-fonts?

CSS
- ¿En qué consiste el modelo de caja (box model)?
- ¿Qué es box-sizing?
- ¿Cuáles son las diferencias entre flexbox y grid?
- ¿Cómo centrar un elemento en la pantalla?
- ¿Para qué sirve un preprocesador de CSS?
- ¿Sabes la diferencia entre unidades relativas y absolutas?
- ¿Cuáles son las unidades relativas de CSS que conoces?
- ¿Cuáles son las unidades absolutas de CSS que conoces?
- ¿Puedes mencionar 3 maneras de ocultar un div?
- ¿Qué es !important?
- ¿Cuál es la diferencia entre normalize y reset?
- ¿Cómo optimizarías el uso de imágenes para dispositivos móviles?
- ¿Para qué sirve el selector *?
- ¿Qué son los pseudo selectores? ¿Puedes explicar algunos?
- ¿Qué hace el selector div + p?
- ¿Qué hace el pseudo selector :is?

- ¿Qué hace position: absolute?
- ¿Qué son media queries?
- ¿Qué es @support?
- ¿Para qué sirve z-index?

JavaScript
- ¿Qué es el scope?
- ¿Qué es context?
- ¿Qué es hoisting?
- ¿Cuál es la diferencia entre var, const y let?
- ¿Cómo funciona .map?
- ¿Cómo funciona .reduce?
- ¿Cómo funciona .filter?
- ¿Cuáles son las diferencias entre == y ===?
- ¿Es JavaScript un lenguaje orientado a objetos?
- ¿Qué es Prototype?
- ¿Qué es coerción de tipo implícito?
- ¿Cuál es la diferencia entre parámetros y argumentos?
- ¿Qué funciones bloquean el event loop de JavaScript?
- ¿Cuáles maneras existen para declarar una función?
- ¿Cómo es JavaScript tan eficiente si no es multi-thread?
- ¿Para qué sirve el "strict mode"?
- ¿Qué pasa si declaro una variable sin let, var o const?
- ¿Qué es un IIFE?
- ¿Qué es un closure?
- ¿Qué son template string?
- ¿Qué es un callback?
- ¿Qué es un callback hell?
- Explique la diferencia entre while y do while
- ¿Qué es una promesa?
- ¿Cuándo NO es recomendable usar arrow functions?
- ¿Para qué sirve la palabra reservada delete?
- ¿Cómo se puede crear una propiedad en un objeto de manera dinámica?
- ¿Qué es isNaN()?
- ¿Qué es null?
- ¿Qué es event delegation?
- ¿Cuáles son los estados de document?
- ¿Qué es un polyfill?
- ¿Qué es un package manager y cuáles son los más utilizados en JavaScript?
- ¿Qué es ECMAscript?

Otras preguntas

- ¿Cómo funciona AJAX?
- ¿Qué es un HTTP request?
- ¿Qué es CORS?
- ¿Qué métodos de HTTP requests conoces y para qué se usan?
- ¿Qué es control de versiones?
- ¿Para qué sirve un CDN?
- ¿Qué es webpack? ¿Conoces alguna otra alternativa?
- ¿Cuál es la diferencia entre sessionStorage, localStorage y cookie?
- ¿Cómo mejoraría el rendimiento de una página web en general?
- ¿Cuáles son los formatos de imágenes que soporta el navegador? ¿Cuál es más eficiente?
- ¿Cómo investigas si algo que estás haciendo es soportado por los navegadores?
- ¿Cuál es la diferencia entre un framework y una librería?
- ¿Qué estás estudiando en este momento?

Ejercicios
- Ejercicio 1
- Ejercicio 2
- Ejercicio 3
- Ejercicio 4
- Ejercicio 5

La última pregunta:
¿Qué preguntas tienes?
- ¿Cuál es el stack tecnológico del proyecto?
- ¿Cuáles son los principales problemas que tienen en este momento a nivel de proyecto?
- ¿Qué metodología de desarrollo utilizan en el proyecto (waterfall, agile, shape up)?
- ¿Cómo es el día a día de un desarrollador en su proyecto?
- ¿Cuál es el camino profesional para los desarrolladores? ¿Cómo se puede crecer dentro de la empresa?
- ¿Cuál sería el desarrollador ideal para su empresa / proyecto?
- ¿Cómo se asegura la organización de mantener una cultura sana dentro de la empresa?
- ¿Cuál sería mi rol dentro del proyecto/empresa?
- ¿Cuál es la visión de la empresa a corto y largo plazo?
- ¿Con quién estaría trabajando?
- ¿Qué es lo que sigue después de esta entrevista?

Conclusión

Introducción

Después de más de **30 años de experiencia** combinada en la industria y haber realizado **más de 300 entrevistas**, hemos logrado determinar los principales puntos donde los candidatos fallan y por ello, consolidamos 99 preguntas y respuestas que solventan este problema.

Sabemos que las entrevistas de trabajo son un reto importante para todos, por ello el objetivo de este libro es brindarte una sólida compilación de ejemplos, preguntas y respuestas para que puedas tener una mejor idea de cómo enfrentarte a una entrevista laboral para un puesto como web developer.

El libro está dividido en los siguientes bloques: HTML, CSS, JavaScript, los cuáles engloban las tres grandes áreas más importantes de la web junto con otros dos bloques de preguntas y de ejercicios extra que pone a prueba la lógica de programación y el conocimiento de las particularidades del proceso desarrollo de aplicaciones web.

Definitivamente existen más de 99 posibles preguntas de entrevista, cada entrevistador tiene su estilo por lo tanto no es de extrañar que las preguntas que recibas en las entrevistas sean distintas a los ejemplos aquí planteados, sin embargo, consideramos que si practicas lo suficiente y entiendes los conceptos básicos del desarrollo web no tendrás problemas para responder adecuadamente.

Recomendamos utilizar este libro como un manual de consulta, si quieres aprovecharlo al máximo, escribe y ejecuta los ejemplos en tu editor de código favorito, realiza cambios y con mucha curiosidad investiga en Internet los conceptos que no queden claros.

Esperamos que disfrutes de "99 preguntas de entrevista para Desarrolladores Web".

Mariano, Freddy y Alfredo.

Acerca de los autores

Mariano Álvarez

Google Developer Expert y Full-Stack developer, con más de 8 años de experiencia, apasionado por las tecnologías web, speaker. Le encanta aportar y colaborar con la comunidad web.

Twitter | LinkedIn

Freddy Montes

Front-End Developer y Diseñador UX/UI con más de 10 años de experiencia desarrollando productos digitales, creador de contenido y profesor.

Twitter | LinkedIn

Alfredo Bonilla

Desarrollador web independiente enfocado en tecnologías web 3.0 y con más de 10 años de experiencia en desarrollo de aplicaciones web, liderazgo de equipos y organización de comunidades.

Twitter | LinkedIn

Créditos

Un gran agradecimiento a Oscar Barajas(@gndx), Bezael Pérez(@domini_code), Leonardo Picado(@leopyc), Dany Paredes(@danywalls), por su colaborar en la revision, retroalimentación y consejos para la creación de este libro.

HTML

HTML o *HyperText Markup Language* es el lenguaje de marcado estándar de la web. Las preguntas relacionadas a HTML pretenden evaluar tu nivel de entendimiento sobre cómo funciona y lo que implica escribir un documento web de la manera correcta.

¿Qué es HTML5?

Es un acrónimo comúnmente utilizado para referirse a la última versión de HTML, que introdujo nuevos elementos y capacidades como los etiquetas de audio, nav, footer, etc.

¿Cómo puedo incluir CSS en un HTML?

Existen 4 maneras en el que podría ser incluido:

1. En línea (inline), directamente en el etiqueta con el atributo style

```
<p style="color:#000;">This is a paragraph.</p>
```

2. Utilizando el etiqueta style

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p {
        color: #000;
      }
    </style>
  </head>
  <body>
    <p>This is a paragraph.</p>
  </body>
</html>
```

3. Con el etiqueta link dentro del etiqueta head

```
<head>
  <link href="styles.css" rel="stylesheet">
</head>
```

4. Utilizando JavaScript

```
<script>
  const note = document.querySelector('.note');
  note.style.backgroundColor = 'yellow';
  note.style.color = 'red';
</script>
```

¿Dónde puede ser incluida la etiqueta de script en HTML?

Técnicamente puedes incluirlo en cualquier parte dentro del <head> o del <body> pero lo más recomendado es agregarlo en el cierre del etiqueta body(</body>) esto evita bloqueos en el navegador permitiendo que se muestre todo el HTML antes de descarga el archivo del script.

¿Cuál es la diferencia entre los atributos async y defer en la etiqueta de script?

Con `async` el archivo se descarga de manera asíncrona y el código se ejecuta tan pronto termine de descargarse.

```
<script async src="utils.js"></script>
```

Con `defer` el archivo se descarga también de manera asíncrona, pero el código se ejecuta cuando el resto de la página ha terminado de cargar.

```
<script defer src="utils.js"></script>
```

¿Cuál es la diferencia entre la etiqueta de `<b>` y `<strong>`?

En un navegador ambas etiquetas se van a ver igual, el texto se marcará como "negrita" (bold), pero si el sitio estuviera siendo leído a través de un screen reader no tendría manera de saber.

que es un texto resaltado, el mismo problema aplica para <i>. Con y se resuelve este problema.

¿Qué etiqueta es necesaria para que el sitio sea responsivo?

La etiqueta es <meta> y recibe dos atributos "name" y "content".

Para hacer el sitio web responsivo hay que agregar los atributos name="viewport" y content="width=device-width, initial-scale=1" .

La propiedad width controla el ancho del viewport. El viewport es una ventana gráfica constituye el área del sitio web que se está viendo en ese momento. En términos de navegador web, suele ser lo mismo que la ventana del navegador, excluyendo la interfaz de usuario, la barra de menús, etc.

Puedes definir el valor de un número en píxeles, por ejemplo width=1000 o el valor "especial" device-width qué es el ancho de la pantalla en píxeles CSS a una escala del 100%.

¿Qué es un SVG?

Un Gráfico Vectorial Escalable (o SVG por sus siglas en inglés) es un tipo de imagen definido en XML, que usa vectores y no píxeles, para poder acomodarse a cualquier tamaño sin perder fidelidad.

¿Qué son atributos de una etiqueta HTML?

Los atributos nos permiten asignar valores a ciertas propiedades de los elementos, por ejemplo: style, id, class. Algunos elementos tienen atributos únicos tales como el atributo alt para la etiqueta <img>.

¿Cuál es la diferencia entre class y id?

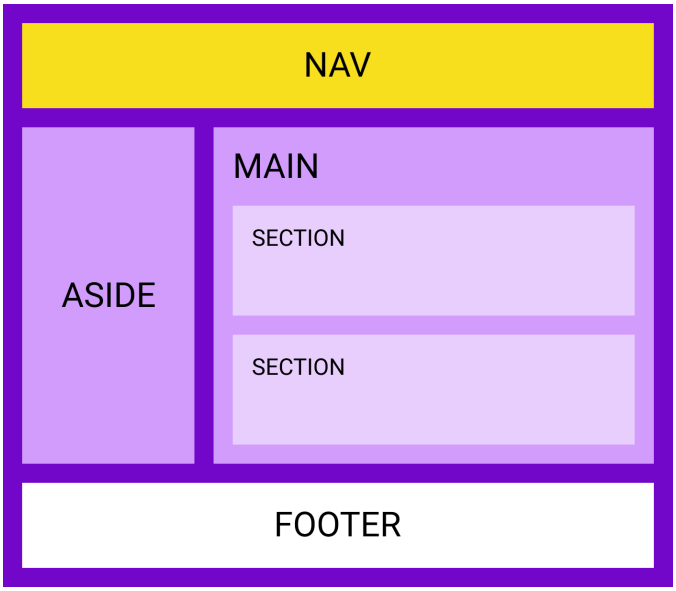
id es un identificador único que solo debe estar asociado a un elemento dentro la misma página de HTML.

class puede ser utilizado en más de un elemento.

Ambos pueden ser utilizados como selectores para aplicar estilos de CSS pero se debe de considerar que usar id como selector tiene mayor precedencia que class por sí solo.

¿Qué es HTML semántico?

Es escribir HTML de una manera en que la información es organizada junto a etiquetas que representen su contenido. Un ejemplo clásico es el siguiente:



Este layout se podría construir de la siguiente manera

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div class="nav"></div>
    <div class="sidebar"></div>
    <div class="main">
      <div class="section1"></div>
      <div class="section2"></div>
    </div>
    <div class="footer"></div>
  </body>
</html>
```

Este código no es semántico, usamos puras etiquetas para todas las secciones del sitio web.

Ahora un ejemplo con código de HTML semántico:

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <nav>...</nav>
    <aside>...</aside>
    <main>
      <section>...</section>
      <section>...</section>
    </main>
    <footer>...</footer>
  </body>
</html>
```

¿Cómo se establece el idioma de un documento?

Utilizando el atributo lang en la etiqueta `<html>`. Poniendo el parámetro en la etiqueta `<html lang="en">` hace que se herede a todos los demás elementos del documento.

¿Qué es ARIA? ¿Para qué sirve?

ARIA significa "*Accessible Rich Internet Applications*". Son una serie de atributos que permite hacer un documento web más accesible para personas con discapacidades.

Complementa el HTML para que las interacciones en las aplicaciones puedan pasar a las tecnologías de asistencia cuando no exista otro mecanismo.

Por ejemplo, ARIA permite que haya sugerencias para formularios y mensajes de error, actualizaciones de contenido en directo, etc.Ejemplo:

```
<button>send</button>
// accessible name: send

<button aria-label="send email">send</button>
// accessible name: send email
```

¿Cómo se puede hacer una página accesible?

Algunas recomendaciones son:

1. Agregar `alt` para todas las imágenes con un texto descriptivo.
2. `<button>` para acciones y `<a>` para links.
3. Usa suficiente contraste entre el fondo y el color del texto.
4. Usa textos descriptivos en los links, no "click here".
5. Usa correctamente las etiquetas de encabezado (h1, h2, etc.) y con su respectiva jerarquía dentro del documento.
6. Agrega un título correcto a la página usando la etiqueta `<title>`.
7. Usa `<label>` en los campos del formulario.
8. Implementa el ARIA standard.

¿Para qué sirve la etiqueta `<base>`?

Permite definir el URL que va a ser utilizado en todas las direcciones relativas dentro del HTML.

Por ejemplo

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <base href="/custom/path/">
</head>
<body>
    <a href="about-me.html">About Me</a>
</body>
</html>
```

Agregar la etiqueta <base> hace que el enlace “about-me.html” del <a> vaya a “/custom/path/about-me.html”.

¿Qué es un iframe?

Es un elemento en línea que permite insertar una página de HTML dentro de otra.

¿Cómo funciona la técnica de inlining-fonts?

Es una técnica en la que el código de CSS de los fonts se pone en el <head> del HTML, de esta manera el navegador encuentra las declaraciones más rápido y no tiene la necesidad hacer una petición y esperar a que se cargue un archivo externo de CSS con el mismo código.

Ejemplo:

```html
<head>
  <style>
    @font-face {
        font-family: "Open Sans";
        src: url("/fonts/OpenSans-Regular-webfont.woff2") format("woff2");
    }

    body {
        font-family: "Open Sans";
    }
  </style>
</head>
```

CSS

CSS o “Cascading Style Sheet” es un lenguaje para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado y en este caso el lenguaje de marcado del que hablamos es HTML..

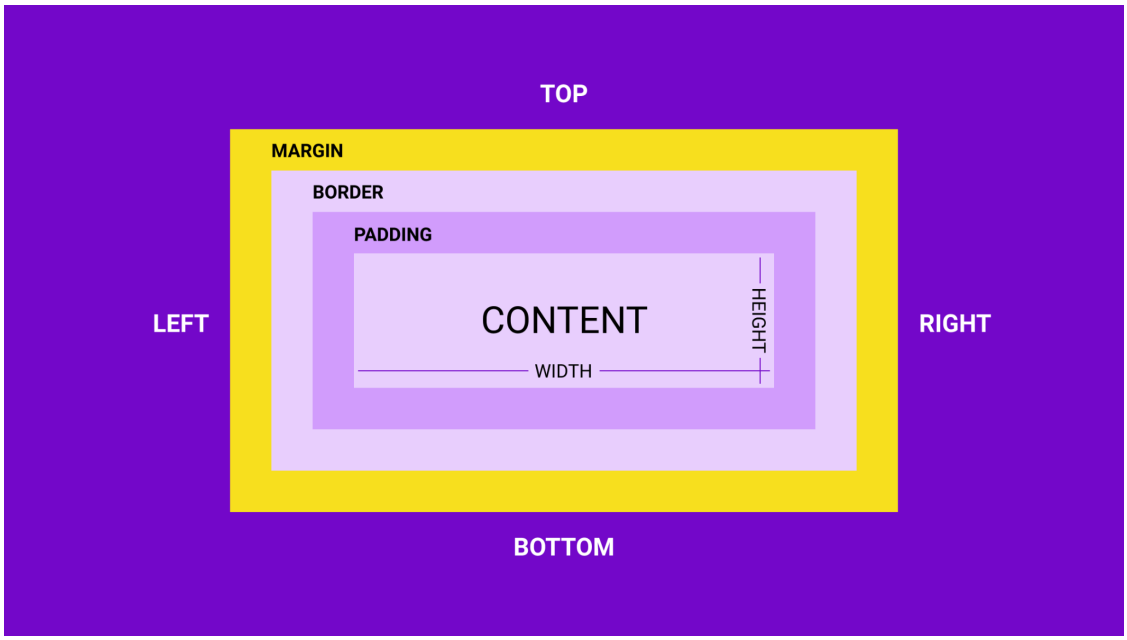
Al desarrollar para la web te vas a enfrentar a muchos escenarios que requieren que los elementos se vean de cierta manera. Las preguntas de CSS buscan evaluar tus conocimientos sobre cómo utilizar el lenguaje para hacer que un documento HTML responda a un diseño específico y además cómo hacer que se vea bien en distintos dispositivos.

¿En qué consiste el modelo de caja (box model)?

Todos los elementos de HTML para CSS tienen una caja alrededor. Existen dos tipos de cajas "inline" y "block". El modelo de caja aplica principalmente al de bloque.

Partes de la caja

- Context box: Donde se muestra su contenido, puedes ponerle dimensiones especificando un ancho (width) y un alto (height)
- Padding box: Es el espacio interno de la caja, se establece con la propiedad padding
- Border box: Es la primera capa externa y envuelve el contenido y el padding, se establece con la propiedad border
- Margin box: Es la capa más externa, envuelve todas las demás capas y se establece con la propiedad margin



¿Qué es box-sizing?

Es una propiedad que permite definir como el total de width y el height de una caja es calculado.

Su valor predeterminado es content-box que suma border y padding al valor de width y height.

Su otro valor es border-box que no suma nada extra al valor de width y height.

¿Cuáles son las diferencias entre flexbox y grid?

Ambos flexbox y grid son métodos para la colocación de elementos en la página web.

Flexbox permite distribuir elementos en una dimensión, es decir en filas o columnas, mientras que grid permite hacerlo en dos dimensiones: filas y columnas al mismo tiempo.

Típicamente, se recomienda flex para componentes y grids para layouts.

¿Cómo centrar un elemento en la pantalla?

Existen diferentes formas, una fácil es hacerlo con flexbox:

```
.parent {
    display: flex; /* Pone el elemento en modo flexbox */
    height: 100vh; /* Pone el alto al máximo del viewport */
    width: 100vw; /* Pone el ancho al máximo del viewport */
    justify-content: center; /* Centra elementos hijos horizontalmente */
    align-content: center; /* Centra elementos hijos verticalmente */
}
```

¿Para qué sirve un preprocesador de CSS?

Son lenguajes que permiten extender la funcionalidad del CSS. Cada preprocesador tiene sus particularidades y sintaxis, pero generalmente escribir CSS válido es aceptado por cualquier preprocesador. Además trae beneficios como crear funciones, mixins, variables entre otros. Algunos ejemplos son: Saas, Less

¿Sabes la diferencia entre unidades relativas y absolutas?

Una unidad relativa obtiene su tamaño con respecto a otro elemento de la página, por ejemplo:

- **em,** toma el valor basado en el fontsize de su elemento padre
- **rem,** toma el valor del font size del root element.

Mientras que las unidades absolutas son medidas exactas como px o pt.

¿Cuáles son las unidades relativas de CSS que conoces?

- em
- rem
- %
- vh

- vw

¿Cuáles son las unidades absolutas de CSS que conoces?

- px
- pc
- cm
- mm
- pt

¿Puedes mencionar 3 maneras de ocultar un div?

En HTML se puede usar el atributo hidden.

En CSS se pueden usar las siguientes reglas pero tienen diferentes comportamientos:

Propiedad	Ocupa espacio	Es clickable
opacity: 0	✓	✓
visibility: hidden	✓	✗
display: none	✗	✗

¿Qué es !important?

Al agregar !important a una propiedad de CSS va a sobreescribir cualquier regla anterior rompiendo la cascada de estilos. Por dicha razón su uso es considerado una mala práctica.

Si sientes la necesidad de usarla es mejor revisar bien la especificidad del elemento e intentar crear un selector más específico para aplicar la regla que quieres.

¿Cuál es la diferencia entre normalize y reset?

De manera predeterminada los documentos HTML no tienen ningún estilo y todos los elementos se ven igual. El problema se encuentra en que los navegadores crearon sus propios estilos básicos (a esto se le llama el user-agent stylesheet) para cada elemento, haciendo que un mismo elemento se vea diferente en cada navegador.

Normalize ayuda a estandarizar con una serie de estilos (definidos a gusto del creador de dicha librería) que son aplicados a los elementos para que se vean de la misma manera a través de navegadores.

A diferencia de Reset, el cual elimina los estilos básicos impuestos por el navegador para asegurar la uniformidad.

¿Cómo optimizarías el uso de imágenes para dispositivos móviles?

- Usando la etiqueta <picture> ya que permite pasar diferentes imágenes basado en un media query.
- Utilizando el atributo lazy y comprimiendo las imágenes.
- También implementando .webp, que es un formato moderno que permite utilizar imágenes de buena calidad con un peso menor.

¿Para qué sirve el selector *?

Es el selector universal, que si se usa solo * {} (sin complementarlo con otro selector) aplicará los estilos definidos a **TODOS** los elementos en el HTML. También puedes usarlo dentro de otro selector por ejemplo header * {} y va a aplicar a todos los elementos dentro del header.

¿Qué son los pseudo selectores? ¿Puedes explicar algunos?

Es una palabra clave añadida a un selector que especifica un estado especial de los elementos seleccionados.

- :hover aplica los estilos a un elemento cuando está en su estado "hover", es decir, cuando el cursor está encima del elemento.
- :visited aplica estilos a un <a> cuyo link ya ha sido visitado por el usuario.
- :playing aplica los estilos a un elemento media (audio o video por ejemplo) cuando el usuario inicia la reproducción.

¿Qué hace el pseudo selector :is?

Es un pseudo selector de tipo function que toma como argumento un listado de selectores y los utiliza como condicionales para aplicar el estilo definido. Por ejemplo:

```
/*
  Pone red a cualquier <p> dentro de un header o footer
  Es el equivalente de escribir header p, footer p {}
*/
:is(header, footer) p {
  color: red;
}

/*
  Pone blue a cualquier <div> que tenga la clase "some" o "blue"
  Es el equivalente de hacer div.some, div.blue {}
*/
div:is(.some, .blue) {
  color: blue;
}
```

¿Qué hace el selector div + p?

Selecciona el <p> que están inmediatamente después de un .

Ejemplo:

```
<div></div>
<p>Selecciona este</p>
<p>No selecciona este</p>
```

¿Qué hace position: absolute?

La posición absoluta saca el elemento del flujo del documento, es decir ya no ocupa espacio como lo hacen static y relative. Cuando se utiliza en un elemento, se posiciona absolutamente con referencia al padre más cercano que tenga una posición relativa.

Un elemento con posición absoluta ocupa al menos dos valores para su posición, uno en X y otro en Y y para configurarlos se usan las propiedades: top, bottom (para Y) y left, right (para X).

¿Qué son media queries?

Es una funcionalidad de HTML/CSS que permite modificar como se ve y funciona un sitio Web dependiendo del tipo general de dispositivo (como print o screen) o de características y parámetros específicos (como la resolución de la pantalla o la anchura del viewport del navegador).

¿Qué es @support?

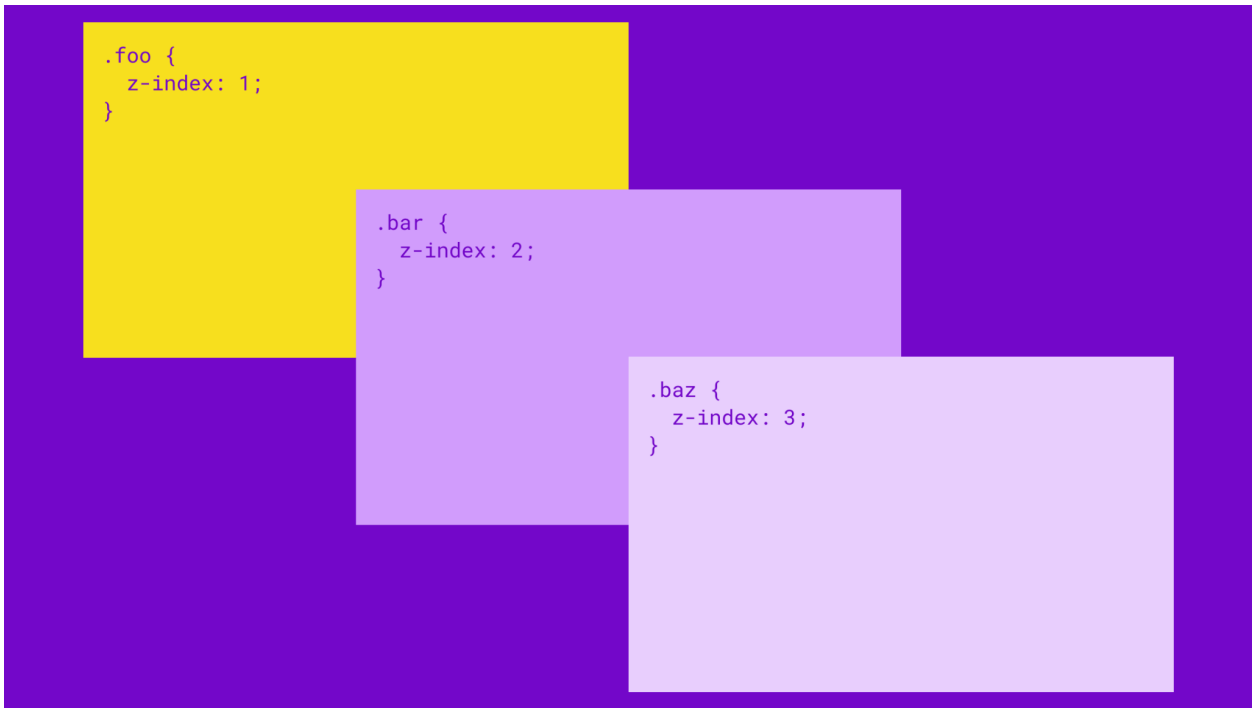
Es una regla que permite aplicar estilos de CSS si el navegador soporta o no una funcionalidad específica de la condicional. Ejemplo:

```
@supports (display: grid) {
  div {
    display: grid;
  }
}

@supports not (display: grid) {
  div {
    float: right;
  }
}
```

¿Para qué sirve z-index?

Controla el orden de apilamiento vertical de los elementos que se superponen y solo afecta a los elementos que tienen un valor de posición distinto al estático (el predeterminado).



JavaScript

JavaScript es un lenguaje de programación ligero, interpretado con funciones de primera clase. Entender el funcionamiento de JavaScript es crucial en el desarrollo web, dado que es un lenguaje con muchas particularidades. Las preguntas de esta sección buscan medir tu nivel de experiencia y entendimiento, sus características y los patrones ya existentes en el lenguaje.

¿Qué es el scope?

Es el alcance que determina la accesibilidad de las variables a JavaScript.

Existent dos tipos de scopes:

- Global: las variables pueden ser accedidas en cualquier parte del código.
- Local: solo pueden ser utilizadas y accedidas en una parte del código

¿Cuál es la diferencia entre var, const y let?

`let` son variables que pueden ser redefinidas

`const` es un constante que no puede ser redefinida o reasignada, esto no quiere decir que su valor no pueda cambiar o sea inmutable. Por ejemplo:

```
const persona = { nombre: "Mariano" }
persona.nombre = "Juan"
console.log(persona.nombre) // Resultado: Juan
```

`let` y `const` son block-scope, esto significa que sólo lo vivirán en el bloque (entre {}) dónde fueron declaradas.

`var` son variables que son accesibles en el scope local, sin importar si están fuera o dentro de un bloque, cuando es declarada en una función. Si es declarada fuera de una función, es una variable global.

¿Qué pasa si declaro una variable sin let, var o const?

Si no se está utilizando el modo estricto, la variable se convierte en global.

¿Qué es context?

Es el valor del objeto al que hace referencia this dentro de una función. Ejemplo:

```
function Animal(....) {
   this.nombre = 'Perro'
}
```

En este ejemplo this hace referencia al objeto que fue creado utilizando la función Animal.

¿Qué es hoisting?

Conocido en español como "elevación", es un comportamiento predeterminado de JavaScript que lo que hace es elevar las declaraciones al principio de la ejecución del código. Esta es tarea del intérprete para asignar memoria a las declaraciones de variables y funciones antes de la ejecución del código.

Conceptualmente, hoisting se presenta a menudo como que el intérprete "divide la declaración y la inicialización de las variables, y mueve (sólo) las declaraciones a la parte superior del código".

Es por este comportamiento que no tenemos acceso a una variable antes de su declaración y posterior inicialización.

```
// Ejemplo

console.log(name) // undefined
const name = "Jon"

// Lo que hace el intérprete es

const name
console.log(name) // undefined
name = "Jon"
console.log(name) // “Jon”
```

¿Qué es un callback?

Es una función que se pasa como argumento a otra función para ser ejecutada "en el futuro".

```
function saludar(name) {
  console.log(`¡Hola! ${name}`);
}

function obtenerNombre(callback) {
  const name = prompt('¿Cuál es tu nombre>');
  callback(name);
}

obtenerNombre(saludar);
```

¿Cómo funciona el método .map?

Es un método del array que retorna un array nuevo y "transformado" usando la función que es pasada como argumento (comúnmente llamada callback). La misma ejecutará con cada ítem que tenga el array.

¿Cómo funciona el metodo .reduce?

Es un método del arreglo que retorna un valor "reducido" o calculado al ejecutar una función reductora (el callback que se le pasa) en cada elemento del array, dando como resultado un único valor de salida.

La función reductora recibe como parámetros el valor "acumulado" de la anterior iteración y el valor actual del array.

¿Cómo funciona el método .filter?

Esta función retorna un nuevo arreglo solo con los elementos que cumplan con la condición que es definida en función callback.

¿Cuáles son las diferencias entre == y ===?

== comparación de valores, por ejemplo 2 == '2' es true

=== comparación estricta de valores y tipos, por ejemplo 2 === '2' es false

¿Es JavaScript un lenguaje orientado a objetos?

A pesar de que es posible crear objetos y utilizar clases en el JavaScript moderno en realidad es un lenguaje orientado a prototipos, se puede demostrar fácilmente:

- En lenguajes orientados a objetos, al definir una clase, todas sus instancias van a tener las mismas propiedades y funciones.
- En JavaScript los objetos usan otros objetos como base para definir sus propiedades pero a su vez pueden definir las propias, sea al momento de la creación o en tiempo de ejecución. Esto demuestra que es un lenguaje basado en prototipos.

¿Qué es Prototype?

Es una propiedad de Object , el "padre" del que derivan todos los demás objetos.

Cuando se intenta acceder a una propiedad de un objeto que no la tiene, la busca en su "prototype".

Prototype contiene métodos y propiedades que comparten todos los objetos.

¿Qué es coerción de tipo implícito?

Término complicado, para explicar que JavaScript convierte un variable de un tipo a otro automáticamente.

```
console.log("1" + 1). //resultado: "11"
```

El 1 se convierte a "1" y se concatena con el string.

¿Cuál es la diferencia entre parámetros y argumentos?

Los argumentos son los valores que se pasan a la función cuando esta se ejecuta, por ejemplo:

`sumar(2,3)` donde 2 y 3 son los argumentos.

Los parámetros son los valores que esperan ser llamados por la función, por ejemplo:

```
function sumar(num1, num2) {
      ......
}
```

Donde num1 y num2 son los parámetros.

¿Qué funciones bloquean el event loop de JavaScript?

`window.alert`, `window.prompt` y `window.confirm` bloquean el event loop y solo va a continuar una vez que el usuario cierre estas ventanas. Son considerados una mala práctica, y eventualmente podrían traer problemas en la aplicación.

¿Cuáles maneras existen para declarar una función?

```
const getName = function() {}

const getName = () => {}

function getName() {}
```

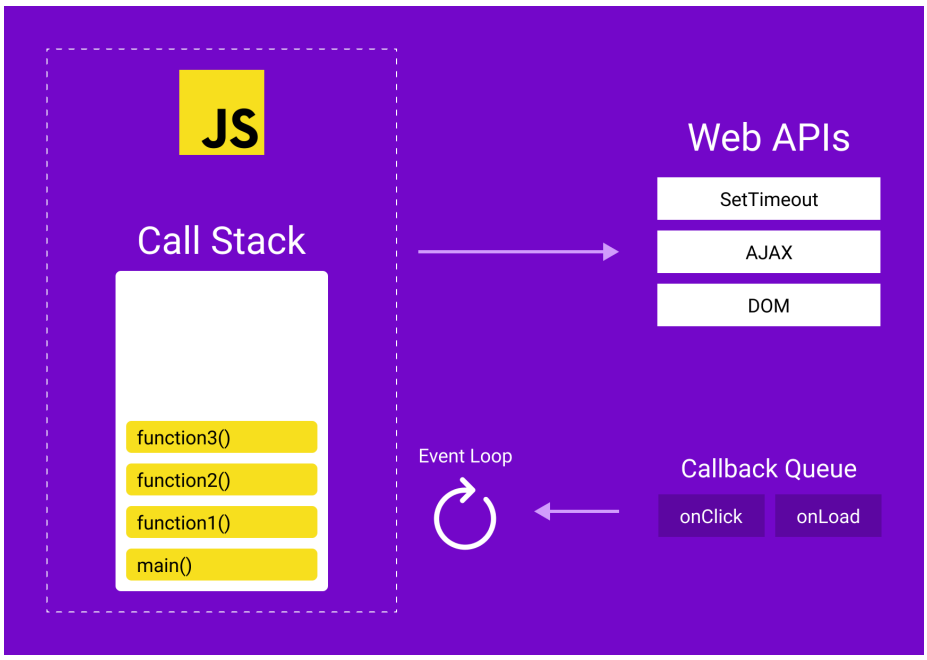
¿Cómo es JavaScript tan eficiente si no es multi-thread?

JavaScript maneja la concurrencia basada en modelo del event-loop, funciona de esta manera:

Los llamados a funciones son agregados al call stack, en el que la primera función que entra es la última en resolverse dado que va ejecutar primero lo que la función contenga en sí. Las funciones asíncronas no son parte del runtime de JavaScript en sí (como el setTimeout, AJAX),

son agregadas por el navegador, estas conforman el Web APIs, y son ejecutadas en otro hilo de esta manera no bloquean al hilo principal. El callback que es pasado a la función asíncrona es tomado y agregado al callback queue (por ejemplo: una función encargada del click de un botón), al momento que esta función debe de correr, el evento loop la ejecutará una vez que el call stack esté vacío.

Es importante que sepas que ninguna de las funciones de Web API son ejecutadas inmediatamente, si el call stack está muy lleno tendrá que esperar a que se vacíe para poder ejecutar los llamados del callback queue.



Esta es una explicación muy simple y breve, te sugerimos investigar más a fondo para entenderlo mejor.

¿Para qué sirve el "strict mode"?

Es una funcionalidad introducida en ECMAScript5 en la que se agregan algunas reglas como:

- Corrige errores que dificultan a los motores de JavaScript para que realicen optimizaciones.
- Prohíbe el uso de sintaxis que pueda ser utilizada en versiones futuras.
- Evita que this apunte al scope global si es undefined.

- Además elimina algunos errores silenciosos que no pueden ser capturados en modo normal.

¿Qué es un IIFE?

Immediately Invoked Function Expression - Es una función que se ejecuta inmediatamente apenas se define.

```
(function () {
  ....
})();
```

¿Qué es un closure?

Una clausura o closure es una función que guarda referencias del estado adyacente, es decir una función en el interior que puede acceder al ámbito de una función padre.

```
function crearPersonaje(tipo) {
  const poderBase = 50;

  // Esta función puede acceder al poder base del personaje y su tipo
  // además retorna otra función que puede ser utilizada para generar un
  // objeto de personaje.
      return function(nombre) {
          const personaje = {
            nombre,
            poderBase,
            tipo
          };
          return personaje;
      }
}


......

.........
```

```
// Esta línea declara una posición de memoria para la función "interna"
// retornada por nuestra función "exterior" crearPersonaje
const crearMago = crearPersonaje("mago")

// Ahora con la función crearMago podemos crear todos los magos
// que necesitemos utilizando el mismo poder base
// referenciado en la función crearPersonaje
const miMago = crearMago("Gandalf")
const miOtroMago = crearMago("Merlín")
```

Una de las razones por la que se utiliza esta técnica es para poder crear variables privadas, en el ejemplo de arriba poderBase es una variable privada, su valor no es expuesto y solo puede ser accedido dentro del closure.

¿Qué son template strings?

Son strings que se delimitan por los backticks (``) esto permite que se puedan declarar strings multi línea y además interpolar variables de esta manera:

```
const titulo = '99 preguntas de'
const ejemplo = `${titulo} entrevista`
```

¿Qué es un callback hell?

Es un anti-patrón que se crea cuando tenemos una serie de funciones que reciben más funciones, creando callbacks anidados. Una posible solución a esto es usar Promise.

```
getUser(id, function(user) {
    getOrders(user.id, function(orders) {
        updateOrderOwner(orders, function(result) {
            console.log('Esto es un callback hell!')
        });
    });
});
```

Explique la diferencia entre while y do while

- `while` ejecuta el código si se cumple la condición
- `do while` lo ejecuta hasta que se cumpla, entonces se puede decir que el código se corre una vez "más" porque va a chequear la condición luego de ejecutarse.

¿Qué es una Promise?

Es un objeto asincrónico que eventualmente se completará y retornará un valor o fallará. Tiene 3 estados:

- *pending*: estado inicial, ni completada o rechazada.
- *fulfilled*: se resolvió de manera satisfactoria.
- *rejected*: la operación falló.

Puedes crear una promesa de esta manera:

```
const miPromesa = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('foo');
  }, 300);
});
```

`Resolve` y `reject` son las funciones que permiten completar la promesa.

Con `.then()` se obtiene el resultado de la promesa y con `.catch()` se capturan los errores.

```
miPromesa
  .then(resultado => { console.log(resultado) })
  .catch(error => { console.log(error) });
```

Uno de los objetivos de las promesas es eliminar los "callback hells" pero debes de tener cuidado, puedes terminar con el mismo problema, por ejemplo:

```
miPromesa
  .then(res1 => {
      miPromesa2(res2)
        .then(res3 => {
            miPromesa3(res4)
              .then(res4 => {
                  console.log(res4)
              })
        })
  })
  .catch(error => { console.log(error) });
```

Afortunadamente, las promesas pueden ser encadenadas, resolviendo el problema:

```
miPromesa
    .then(res1 => {
     return miPromesa(res1)
  })
  .then(res2 => {
    return miPromesa2(res2)
  })
  .then(res3 => {
    return miPromesa3(res3)
  })
  .then(res3 => {
    console.log(res4)
  })
```

Si quisieras escribir un código aún más limpio puedes utilizar async/await:

```
const resultado = async () => {
 const res1 = await miPromesa();
 const res2 = await miPromesa(res1);
 const res3 = await miPromesa(res2);
 console.log(res3);
}
```

¿Cuándo NO es recomendable usar arrow functions?

Una de las grandes diferencias entre un function y un arrow function es el comportamiento de this dentro de ellas.

En un function el this hace referencia a un contexto dinámico, dependiendo de dónde es invocada. Mientras que un arrow function siempre va a apuntar al contexto externo.

Por lo tanto, si necesitamos que el contexto referenciado por medio de this sea dinámico, entonces deberíamos utilizar un function en lugar de una función de flecha.

```
const gato = {
  sonido: "miau",
  saludar: function () { return this.saludo; }
}
// Este llamado va a retornar "miau" pues function utiliza
// this como una referencia al objeto que contiene la función.
gato.saludar()

const perro = {
  sonido: "woof",
  saludar:  () => this.saludo
}
// Este llamado va a retornar *undefined* dado que
// this es una referencia al contexto externo,
// en este caso el contexto global donde la variable
// *saludo* no esta definida.
perro.saludar()
```

¿Para qué sirve la palabra reservada delete?

Se utiliza para eliminar una propiedad directamente en objeto. Ejemplo:

```
const producto = {
      nombre: "manzana",
      price: "30",
      descripcion: "Es de color rojo"
};
```

```
delete producto.descripcion;

console.log(producto); // {nombre: 'manzana', price: '30'}
```

¿Cómo se puede crear una propiedad en un objeto de manera dinámica?

Con computed properties se puede crear dinámicamente una propiedad en un objeto. Ejemplo:

```
const propName = "profesion";

const persona = {
    nombre: "Daniela",
    edad: 27,
    [propName]: "Administradora",
};

console.log(persona.profesion); // Administradora
```

¿Qué es isNaN()?

Es una función que retorna true si el valor de una variable no es número.

```
function isNumber(num) {
  if (isNaN(num)) {
    return "NO es un número";
  }
  return "Es un número";
}
console.log(isNumber('1000F'));
// expected output: "NO es un número"

console.log(isNumber(1000));
// expected output: "Es un número"
```

¿Qué es null?

Es un valor que representa la inexistencia de un objeto o valor, es decir, no es ni un objeto, string, boolean o number. **Importante**: no es un tipo.

¿Qué es event delegation?

Es una práctica en la cuál se le asigna un evento al padre/contenedor de otros elementos del mismo tipo.

Por ejemplo si tienes un menú como este:

```
<nav>
  <a>One</a>
  <a>Two</a>
  <a>Three</a>
</nav>
```

En lugar de iterar y asignar el evento a cada <a>, se le asigna al <nav> y con la propiedad currentTarget puedes saber sobre cuál de los <a> fue quién ejecutó el evento.

¿Cuáles son los estados de document?

document.readystate es una propiedad que permite determinar el estado de carga de un documento.

Puede contener los siguientes valores:

- loading: el documento está cargando
- interactive: el documento terminó de cargar y se puede interactuar con los elementos del DOM pero los recursos externos como scripts, imágenes, estilos y frames están cargando todavía.
- complete: el documento y todos sus recursos están cargados.

¿Qué es un polyfill?

Es código agregado para brindar soporte, brindado por otros navegadores, a uno que no lo tiene de manera nativa.

¿Qué es un package manager y cuáles son los más utilizados en JavaScript?

Es un software que automatiza el proceso de instalación, actualización, configuración y eliminación de paquetes.

En el mundo de JavaScript el más conocido es npm (Node Package Module) y este permite manejar las dependencias de un proyecto pero también crear y publicar los propios paquetes.

Otro muy utilizado es yarn que al igual que npm permite manejar paquetes pero además ofrece capacidades de manejo de proyectos y monorepos.

¿Qué es ECMAscript?

ECMAscript es el estándar que define las reglas y directrices que permiten la interoperabilidad entre navegadores.

Otras preguntas

El desarrollo web es un universo enorme así que muchas preguntas pueden ir más allá de los lenguajes utilizados.

Estas preguntas pretenden analizar el conocimiento que tienes en general sobre la web como plataforma, el navegador y tus propios procesos para resolver problemas relacionados a estos.

¿Cómo funciona AJAX?

AJAX significa JavaScript asíncrono y XML (Asynchronous JavaScript and XML) y es un conjunto de técnicas de desarrollo web que comprenden:

1. HTML.
2. Manipulación del DOM (JavaScript).
3. XML, aunque ahora se usa JSON para el intercambio de datos.
4. Y XMLHttpRequest para la comunicación con el servidor aunque ahora se usa el fetch API.

Antes de explicar cómo funciona analicemos el siguiente ejemplo: Imagina que estás en tu red social favorita, y dejas un comentario en un post, el UI se actualiza, agregando el texto, sin ninguna necesidad de recargar la página, esto se logra con AJAX, pero ¿cómo funciona?:

1. El navegador hace una llamada a un servidor con XMLHttpRequest (o fetch), esto lo hace en segundo plano.
2. El servidor recibe datos del navegador, procesa y devuelve nuevos datos.
3. El navegador recibe estos datos y actualiza el DOM con la información recibida.

¿Qué es un HTTP request?

Es un "mensaje" enviado desde el cliente (por ej: navegador) hacia el servidor. Se utiliza para realizar intercambio de información entre ambos.

Hay dos tipos de mensajes: las solicitudes enviadas (request) por el cliente para desencadenar una acción en el servidor, y la respuesta del servidor (response).

¿Qué métodos de HTTP requests conoces y para qué se usan?

- GET: para solicitar información
- PUT: para actualizar una entidad
- POST: para crear una entidad

- DELETE: para eliminar una entidad
- OPTIONS: permite al cliente identificar qué métodos soporta el servicio

¿Qué es CORS?

CORS (Cross-origin resource sharing) es un mecanismo para poder restringir el acceso de recursos almacenados en un dominio diferente.

¿Qué es control de versiones?

Es una herramienta responsable de manejar y controlar los cambios hechos en el código, de esta manera se crea un historial dónde se puede hacer seguimiento de cada modificación. El más utilizado se llama Git pero antiguamente se utilizaban sistemas como SVN o TFS.

¿Para qué sirve un CDN?

Content Delivery Network - es una red de servidores que está distribuida en diferentes partes del mundo con copias del contenido, con el objetivo de minimizar la latencia al momento en el que se solicitan los archivos estáticos , esto no solo ayuda la latencia si no también el SEO.

¿Qué es webpack? ¿Conoces alguna otra alternativa?

Es un "module bundler". Básicamente toma todo el código de la aplicación y lo "procesa" para que sea usable por el navegador.

Alternativas:

1. Parcel
2. Vite
3. Esbuild
4. Snopack

¿Cuál es la diferencia entre sessionStorage, localStorage y cookie?

- Session Storage, es el almacenamiento de información en el navegador que solo va a persistir en la sesión del usuario, si el navegador o el tab se cierra, la información va ser borrada.
- LocalStorage, la información persiste independientemente de la sesión.
- Cookies, es un archivo pequeño que es creado por el servidor y enviado al navegador.

Nota:

- Local y session storage sólo pueden ser leídos por el cliente (navegador) además son features de HTML5.
- Todos pueden ser borrados por el usuario en cualquier momento, además sólo persiste en el dispositivo en el que fue usado. Generalmente, son usados para almacenar tokens, o información básica para la sesión del usuario.

¿Cómo mejoraría el rendimiento de una página web en general?

Lo primero que se debe de hacer es evaluar la situación e identificar dónde están los problemas de rendimiento. Generalmente se encuentran relacionados a:

- Una carga lenta de los archivos - imágenes, archivos de JS, fuentes etc. que son muy pesados, causan que el sitio cargue lento. Se puede mejorar el rendimiento utilizando estrategias como la de lazy loading, code-splitting, tree-shaking, inlining-fonts. Otra manera técnica para mejorar la latencia es poner los archivos en un CDN.
- Problemas en tiempo de ejecución - llamados a APIs lentos, procesamiento de mucha información, rendereo de muchos nodos entre otros pueden causar que el hilo de JavaScript se bloquee. Se debe determinar si el problema se encuentra en el front-end o el back-end y de ahí en adelante implementar las mejoras.

¿Cuáles son los formatos de imágenes que soporta el navegador? ¿Cuál es más eficiente?

Los formatos de imagen generalmente soportados por la mayoría de los navegadores son:

1. JPG: es el formato más popular, puede manejar millones de colores y diferentes niveles de compresión. En general es una excelente opción para mostrar fotografías complejas con muchos colores.
2. PNG: Al igual que JPG puede soportar millones de colores además de transparencias. Hay que tener mucho cuidado porque mientras más colores más pesado se pone. El uso principal de PNG en la web es para imágenes con fondos transparentes, como logos, íconos e ilustraciones.
3. GIF: Soporta solo 256 colores y transparencias pero la fortaleza es que puedes hacer imágenes animadas aunque recientemente ha caído en desuso porque los formatos de videos en algunos casos proporcionan mejor performance.
4. WebP: es un formato desarrollado por Google en 2010 y ya es soportado por todos los navegadores, su principal ventaja sobre JPG es que es entre 25% y 35% más pequeño pero manteniendo la misma calidad.
5. AVIF: es el más nuevo (2019) sin embargo no es soportado todavía por todos los navegadores, compañías grandes como Facebook y Netflix están apostando para hacerlo el formato por defecto en la web. Puede alcanzar hasta 50% menos de tamaño con la misma calidad comparado con un JPG.

Con respecto a la eficiencia de cada formato va a depender del tipo de imagen que necesites mostrar:

1. Para fotografías full color es más eficiente WebP y pronto va a ser mejor AVIF porque tiene más compresión.
2. Para imágenes que necesitas que tenga fondo transparente lo mejor es usar PNG.
3. Antes se usaban GIF para imágenes animadas, pero ahora los formatos de video han mejorado tanto que es mejor usar video.

¿Cómo investigas si algo que estás haciendo es soportado por los navegadores?

Existen sitios web en los que puedes verificar si una capacidad está soportada o no. Puedes utilizar canluse.com o developer.mozilla.org.

¿Cuál es la diferencia entre un framework y una librería?

Una librería normalmente solo resuelve un problema en sí, por ejemplo, librerías como lodash, scrollreveal, etc.

Un framework provee todas las herramientas para construir una aplicación e indica estándares que deben de seguirse. Además, es común que un framework establezca una arquitectura base para la aplicación.

Algunos ejemplos de frameworks web son Angular, NextJS, Vue.js

¿Qué estás estudiando en este momento?

El entrevistador busca saber si proactivamente te gusta aprender y estudiar y en caso de que así sea, querrá saber más sobre qué estudias y cómo lo haces. Para las empresas es de mucho valor trabajar con personas que puedan aprender por cuenta propia y que actualicen sus conocimientos constantemente.

Ejercicios

Es común que en las entrevistas te soliciten resolver algún problema utilizando código. El objetivo de estas preguntas es evaluar tú lógica para resolver problemas y tus habilidades de programación.

Estos problemas varían desde construir un algoritmo hasta optimizar un código existente.

Ejercicio 1

Implementa el código de la siguiente función donde el signature sea éste sumar(2)(5);

La manera de resolver este ejercicio es usando el patrón del closure, el que nos permite utilizar los parámetros de la función padre, pero también devolver una función que reciba argumentos, de tal manera que el código se vería así:

```
const sumar = (num1) => {
  return (num2) => {
    return num1 + num2
  }
}
```

Ejercicio 2

¿Cuál es el resultado del console.log del siguiente código?

```
for(var i=1; i < 5; i++) {  }
console.log(i)
```

El resultado es **5.** ¿Pero por qué? Como se menciona en la pregunta ¿cuáles son las diferencias entre var, let y const?, var hace que la variable no sea block-scope así que su valor es definido cuando se recorre el array, dando así como resultado el valor de 5.

Ejercicio 3

¿Cuál es el resultado del siguiente ejemplo?

```
let promise = new Promise(function(resolve, reject) {
  resolve("done");

  reject(new Error("..."));

  setTimeout(() => resolve("..."));
});
```

El resultado es done porque una vez que la promesa se complete ignora cualquiera otro llamado de resolve o reject .

Ejercicio 4

Optimiza el siguiente código:

```
var items = ['car', 'tire', 'street' ];

for(var i=0; i < items.length; i++) {
    var listEL = document.querySelector('ul')
    listEL.append(`<li>${items[i]}</li>`);
}
```

Este ejercicio busca entender el conocimiento de la persona con respecto a la manipulación de elementos y su impacto en el rendimiento. La solución es la siguiente:

```
const items = ['car', 'tire', 'street' ];

const listEL = document.querySelector('ul')

let nodes = ''

for(let i=0; i < items.length; i++) {
    nodes += `<li>${items[i]}</li>`
}
listEL.append(nodes);
```

La selección de elementos y el uso de append es muy lento y costoso, así que se debe evitar en la medida de lo posible. El extraerlos fuera del loop permite crear un string que al final se agregará al nodo. De esta manera solo se selecciona y se agrega una vez al elemento.

El ejemplo tiene algunas distracciones, como que hace falta un ; o se utiliza var en vez de let o const , es valioso que lo reemplaces pero el enfoque real es el rendimiento.

Ejercicio 5

¿Cuál es el resultado de este código?

```
console.log("1");

setTimeout(function() {

 console.log("2");

}, 0);

console.log("3")
```

El resultado es: 1, 3, 2. La razón es porque setTimeout es una función asincrónica implementada por el WebAPI, y a pesar de que el delay es de 0, el código dentro la función va ser ejecutado hasta que el callback stack se encuentre vacío.

La última pregunta:

¿Qué preguntas tienes?

Las entrevistas no son solamente un espacio para evaluar tus habilidades, también son un excelente momento para que conozcas más sobre la cultura de la empresa y puedas decidir si es para ti o no.

Además, a los entrevistadores les encanta que les hagas preguntas, ya que esto demuestra tu interés en la empresa y las intenciones que tienes para aportar a ella.

Por último, siempre recuerda que las entrevistas son tanto para la empresa como para el candidato, estas pueden permitirte saber si realmente quieres trabajar ahí o no y si te va a dar un valor agregado en tu carrera profesional.

El siguiente grupo de preguntas permite entender mejor qué clase de empresa es, su apertura a adopción de nuevas tecnologías y además te dará información valiosa para las siguientes entrevistas.

¿Cuál es el stack tecnológico del proyecto?

Es importante que te quede claro con qué tecnologías vas a trabajar, de esta manera puedes prepararte antes de entrar al trabajo y poder disminuir la curva de aprendizaje. Recuerda que una vez que te unes a la empresa debes demostrar porqué estás ahí, así que entre más preparado mejor.

¿Cuáles son los principales problemas que tienen en este momento a nivel de proyecto?

Tienes que entender a lo que vas, si aceptas el reto, además te va dar visibilidad sobre el nivel de complejidad del proyecto y como tu rol va a estar asociado al proyecto, las responsabilidades que vas a tener a cargo y saber si vas a tener que trabajar horas extra si el trabajo lo requiere.

¿Qué metodología de desarrollo utilizan en el proyecto (waterfall, agile, shape up)?

Siendo honesto, esto te puede dar una mejor perspectiva de qué tipo de empresa es y su apertura a optar nuevas tendencias. Desafortunadamente metodologías como waterfall son "anticuadas" y poco beneficiosas, hacen el desarrollo doloroso y no hay oportunidad de feedback, esto es un claro red flag.

¿Cómo es el día a día de un desarrollador en su proyecto?

Esta pregunta puede ayudarte a identificar la manera en la que funciona la empresa, si es organizada o no, que tanto orden o caos existe, y si lo vas a poder manejar del lado tuyo.

¿Cuál es el camino profesional para los desarrolladores? ¿Cómo se puede crecer dentro de la empresa?

Es importante que sepas cómo puedes crecer y escalar dentro la empresa, el dinero no lo es todo, eventualmente te vas a llegar a aburrir si siempre vas a hacer el mismo trabajo. Así que es importante entender cuál es la estructura con la que la empresa cuenta para hacer crecer el talento y se pueda optar por mejores puestos en el futuro.

¿Cuál sería el desarrollador ideal para su empresa / proyecto?

Esto te va a dar una mejor idea de lo que la empresa anda buscando, así puedes prepararte mejor para las siguientes entrevistas o bien decidir si continuas con el proceso o no.

¿Cómo se asegura la organización de mantener una cultura sana dentro de la empresa?

Dado que vas a interactuar durante bastantes horas de tu día en una empresa, es saludable entender que esfuerzos realiza esta organización para que la interacción de sus miembros ocurra de la mejor manera.

Respuestas positivas a esta pregunta podrían mencionar cosas como códigos de conducta, inclusión, incentivos, contratos bien definidos y roles definidos o al menos personas que puedas contactar en caso de problemas.

¿Cuál sería mi rol dentro del proyecto/empresa?

Es importante saber cuáles van a ser tus responsabilidades, de esta manera te aseguras que estás de acuerdo con el trabajo esto además te da más información para poder hacer una mejor negociación.

¿Cuál es la visión de la empresa a corto y largo plazo?

Esto te ayudará a tener tus expectativas claras, saber a qué empresa te estarás uniendo y las oportunidades que la compañía podría eventualmente ofrecerte. Aunque estés iniciando como un jr. developer, tienes que tener oportunidades de crecimiento. Además te ayudará a determinar si la empresa sabe exactamente hacia dónde va y si tiene sus objetivos claros.

¿Con quién estaría trabajando?

No es lo mismo trabajar con otro desarrollador que con el CEO de la empresa. Esto te dará más información sobre el grado de responsabilidad y exposición que el rol tiene.

¿Qué es lo que sigue después de esta entrevista?

Esta pregunta te permitirá:

- Prepararte para lo que viene. Entre más información tengas, mejor. No es lo mismo tener una entrevista con un ejecutivo o líder, que una prueba técnica.
- Intuir si lograse pasar a la siguiente fase. Siendo honestos, todos sabemos cuando no nos va bien en una entrevista.

Conclusión

Has llegado al final del libro, ¡felicitaciones! Ahora tienes más herramientas para prepararte para obtener el trabajo que tanto deseas.

En definitiva la preparación para una entrevista de trabajo técnica es bastante extensa y es probable que tome algunas experiencias para masterizarla.

Es posible que participes en entrevistas que no sean exitosas y esto se puede deber a muchos factores, incluso ajenos a tu preparación técnica. Si esto llegara a pasar no te desanimes, siempre habrá más y mejores oportunidades.

Te recomendamos además no solo quedarte con los ejercicios y ejemplos planteados en este libro, sino que también practiques implementar casos de uso, la verdadera experiencia es lo más valioso a la hora de responder una pregunta de entrevista.

Esperamos que hayas disfrutado el libro tanto como nosotros disfrutamos escribiéndolo.

¡Te deseamos muchos éxitos en tu carrera!

Mariano, Freddy y Alfredo.