Estructura del Taller:

1. **Introducción a ECMAScript**
   - Breve historia y origen de ECMAScript.
   - Relación con JavaScript y su importancia en el desarrollo web.

2. **Evolución de ECMAScript**
   - Versiones principales de ECMAScript y sus características distintivas.
   - Impacto de cada versión en el desarrollo web y en la comunidad de programadores.

3. **Finalidad y Aplicaciones de ECMAScript**
   - Usos comunes de ECMAScript en el desarrollo web.
   - Ventajas y desventajas de utilizar ECMAScript en comparación con otros lenguajes.

4. **Características Clave de ECMAScript**
   - Principales características y funcionalidades que hacen a ECMAScript único.
   - Ejemplos prácticos de cómo estas características pueden mejorar el desarrollo de aplicaciones web.

5. **Objetivos y Beneficios de ECMAScript**
   - Objetivos que se pueden lograr al utilizar ECMAScript en proyectos de desarrollo.
   - Beneficios tangibles de implementar ECMAScript en aplicaciones web modernas.

Historia de ECMAScript.

Breve historia y origen:

Se crea ECMAScript en 1997, y esta basado en el leguaje de JavaScript por puesto por Netscape, a partir de su creación, se genera una estándar para que los navegadores trabajen bajo ese sistema y así se pueda interactuar con cualquier lenguaje de programación. ECMAScript es un lenguaje de tipo dinámico

Relacion con Javascript y su importancia en el desarrollo web

Su creación, fue con base a l lenguaje de JavaScrip cuando Netscape envío el borrador de JavaScript a ECMA para que creara el estándar.

Evolucion de ECMAScript

Versiones principales y sus características distintivas.

Edición	Fecha de publicación	Cambios desde la edición anterior	Editor

1	Junio de 1997	Primera edición	Guy L. Steele, Jr.
2	Junio de 1998	Cambios editoriales para mantener la especificación completa alineada con el estándar internacional ISO/IEC 16262	Mike Cowlishaw
3	Diciembre de 1999	Se agregaron expresiones regulares, mejor manejo de strings, nuevo control de declaraciones, manejo de excepciones con try/catch, definición más estricta de errores, formato para la salida numérica y otras mejoras.	Mike Cowlishaw
4	Abandonado	La cuarta edición fue abandonada debido a diferencias políticas respecto a la complejidad del lenguaje. Muchas características propuestas para la cuarta edición fueron completamente abandonadas; algunas fueron propuestas para la edición ECMAScript Harmony.	
5	Diciembre de 2009	Agrega el modo estricto ("strict mode"), un subconjunto destinado a proporcionar una mejor comprobación de errores y evitar constructores propensos a errores. Aclara varias ambigüedades de la tercera edición, y afina el comportamiento de las implementaciones del "mundo real" que difieren consistentemente desde esa especificación. Agrega algunas nuevas características, como getters y setters, librería para el soporte de JSON, y una más completa reflexión sobre las propiedades de los objetos.[11]	Pratap Lakshman, Allen Wirfs-Brock
5.1	Junio de 2011	Esta edición 5.1 de la ECMAScript Standard está completamente alineada con la tercera edición del estándar internacional ISO/IEC 16262:2011.	Pratap Lakshman, Allen Wirfs-Brock

6	Junio de 2015[12]	La sexta edición agrega cambios significativos en la sintaxis para escribir aplicaciones complejas, incluyendo clases y módulos, definiéndolos sémanticamente en los mismos términos del modo estricto de la edición ECMAScript 5. Otras nuevas características incluyen iteradores for/of loops, generadores y generador de expresiones estilo Python, funciones de dirección, datos binarios, colecciones (mapas, sets, mapas débiles), y proxies (metaprogramación para objetos virtuales y wrappers). Al ser la primera especificación "ECMAScript Harmony", es también conocida como "ES6 Harmony".	Allen Wirfs-Brock
7	Junio de 2016	La séptima edición fue una mera actualización de la versión 6. Incorpora el método Array.prototype.includes() y el operador exponencial (**).	Brian Terlson
8	Junio de 2017	La 8.ª edición, oficialmente conocida como ECMAScript 2017, fue finalizada en junio de 2017.[11] Incluye constructores `async/await`, los cuales funcionan usando generadores y promesas.	Brian Terlson
9	Junio de 2018	La 9.ª edición, oficialmente conocida como ECMAScript 2018, incluye operadores rest/spread para variables (tres puntos: `...identificador`), iteracion asincronica, `Promise.prototype.finally()`	Brian Terlson
10	Enero de 2019	La 10.ª edición, oficialmente conocida como ECMAScript 2019, incorporó `Array.flat()`, `Array.flatMap()`, `String.trimStart()`, `String.trimEnd()`, errores opcionales en el bloque catch, `Object.fromEntries()`, `Symbol.description`	Mathías Bynens
11	Junio 2020	Ver 11.ª edición – ECMAScript 2020	Jordan Harband, Kevin Smith

12	Junio 2021	Ver 12.ª ECMAScript® 2021	
13	Junio 2022	Estas son las nuevas características de esta versión de ECMAScript: 1. Top-level await: Permite el uso del operador `await` fuera de las funciones asíncronas, en el nivel superior de los módulos. 2. Private instance fields, methods, and accessors: Permite declarar campos de instancia, métodos y accesorios (getters y setters) como privados en las clases, lo que mejora el encapsulamiento. 3. Static class fields and methods: Introduce campos y métodos estáticos en las clases. 4. Static class initialization blocks: Introduce bloques de inicialización estáticos en las clases, que se ejecutan una vez cuando se crea una clase. 5. Error: .cause: Permite proporcionar una causa para los errores lanzados, lo que puede ayudar a rastrear la causa original de un error. 6. Array, String, and TypedArray: .at() Method: Introduce un nuevo método `.at()` para los objetos Array, String, y TypedArray, que permite acceder a los elementos de estos objetos desde el final si se pasan índices negativos. 7. Object: .hasOwn(): Introduce un nuevo método `.hasOwn()` para los objetos, que permite comprobar si un objeto tiene una propiedad propia (no heredada). 8. RegExp: match .índices ('d' flag): Permite obtener los índices de inicio y fin de las coincidencias al realizar operaciones de búsqueda con expresiones regulares[13].	Shu-yu Guo, Michael Ficarra y Kevin Gibbons[14]

Impacto de cada versión en el desarrollo web y en la comunidad de programadores.

Cada versión genera nuevas características, que ayudan a la sintaxis del lenguaje, también a corregir errores, además gracias a el grupo TC39, la comunidad de programadores tiene la opción de participar en las actualizaciones futuras de ECMASCript que se conoce como EX.NEXT. todos lo años en el mes de Junio se está generando la actualización de ECMASCript

Finalidad y Aplicaciones de ECMAScript.

Su objetivo principal es proporcionar un estándar para el desarrollo de aplicaciones web. Al tener una especificación clara y precisa, los desarrolladores pueden escribir
código JavaScript compatible con diferentes navegadores y entornos de ejecución.
En otras palabras, ECMAScript define las reglas y normas que deben seguirse para implementar el lenguaje JavaScript. A través de esta especificación, se establece cómo JavaScript debe funcionar y cómo se deben escribir los programas en este lenguaje.

Usos comunes en el desarrollo web

Ventajas y desventajas vs otros lenguajes

ECMAScript es un estándar de lenguaje de programación que define las especificaciones para JavaScript. A continuación, te presento algunas ventajas y desventajas de ECMAScript y su relación con JavaScript:

1. Ventajas de ECMAScript (JavaScript):
    - Compatibilidad con navegadores: ECMAScript es ampliamente compatible con la mayoría de los navegadores modernos. Esto significa que puedes escribir código en ECMAScript y esperar que funcione en diferentes entornos de navegación.
    - Orientación a objetos: ECMAScript sigue un paradigma orientado a objetos, lo que facilita la creación de programas estructurados y reutilizables.
    - Soporte para características avanzadas: ECMAScript ha evolucionado con el tiempo, agregando nuevas características y funcionalidades. Esto permite a los desarrolladores escribir código más eficiente y elegante.
    - Manejo de errores: ECMAScript proporciona mecanismos para manejar errores de manera más efectiva, lo que ayuda a depurar y mantener el código.
2. Desventajas de ECMAScript (JavaScript):
    - Dependencia de frameworks y bibliotecas: Al utilizar ECMAScript, a menudo se heredan las ventajas y desventajas de los frameworks o bibliotecas que se utilizan junto con él. Esto puede afectar el rendimiento y el control sobre el código.
    - Posible pérdida de rendimiento: En algunos casos, el uso de frameworks o bibliotecas puede afectar ligeramente el rendimiento del código ECMAScript.
    - Complejidad: A medida que ECMAScript ha evolucionado, también se ha vuelto más complejo. Algunas características avanzadas pueden ser difíciles de entender para los principiantes.

Caracteristicas Claves de ECMAScript

A lo largo de los años, ha evolucionado y ha introducido numerosas características y mejoras. Aquí están algunas de las características clave de ECMAScript:

1. Funciones de flecha: Las funciones de flecha (=>) son una forma concisa de definir funciones en JavaScript. Son especialmente útiles para funciones anónimas y para mantener el contexto de this.
2. Declaración de variables con let y const: En ES6 (también conocido como ECMAScript 2015), se introdujeron las palabras clave let y const para declarar variables. let permite declarar variables con alcance de bloque, mientras que const declara constantes inmutables.
3. Clases y herencia de prototipos: ES6 introdujo la sintaxis de clases, que facilita la creación de objetos y la herencia. Antes de ES6, la herencia se basaba en prototipos, pero las clases proporcionan una abstracción más familiar para los desarrolladores.
4. Desestructuración de objetos y arrays: La desestructuración permite extraer valores de objetos y arrays de manera más concisa. Por ejemplo, puedes asignar múltiples variables a la vez a partir de un objeto o array.
5. Expresiones regulares: Las expresiones regulares son patrones utilizados para buscar y manipular texto. ES6 mejoró la sintaxis y funcionalidad de las expresiones regulares en JavaScript.

Objetivos y beneficios de EMCAScript

ECMAScript específicamente es el estándar que a partir del año 2015 a la actualidad se encarga de regir como debe ser interpretado y funcionar el lenguaje JavaScript, siendo este (JS – JavaScript) interpretado y procesado por multitud de plataformas, entre las que se encuentran los navegadores web, NodeJS u otros ambientes como el desarrollo de aplicaciones para los distintos sistemas operativos que actualmente existen en el mercado. Los responsables de dichos navegadores y JavaScript deben encargarse de interpretar el lenguaje tal como lo fija ECMAScript.

En la actualidad ECMAScript, viene siendo el estándar que abarca la web en general, todo esto gracias a las mejoras continuas y a las optimizaciones que se van desarrollando con el objetivo de tener siempre un mejor rendimiento, entre otros aspectos. Hoy en día cuenta con gran importancia en el mercado, por su versatilidad JavaScript se ha convertido en un lenguaje universal y puede ser encontrado tanto a nivel móvil como de hardware, servidor, web, entre otros. Adicionalmente, cuenta con una variedad de librerías y frameworks a fin de facilitar el desarrollo de nuevos proyectos.