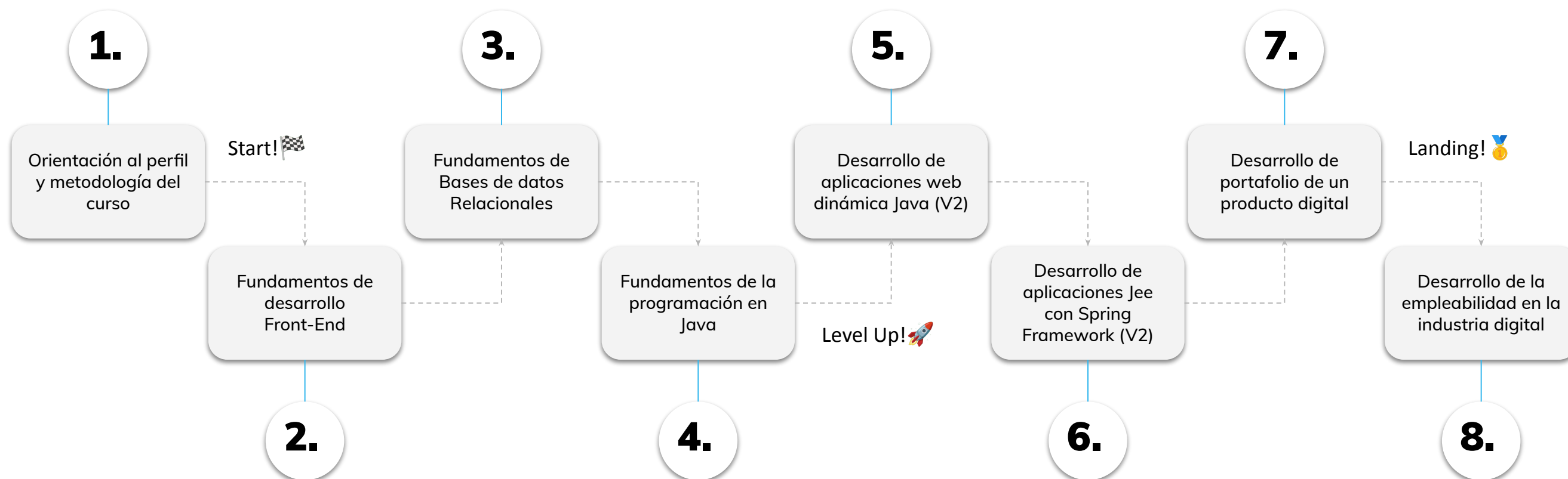


➤ Bases del lenguaje Javascript

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

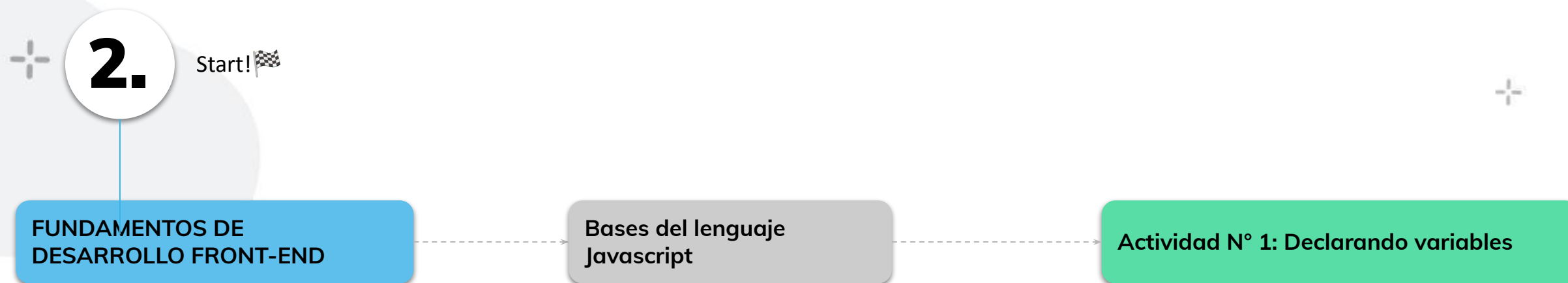
En la clase anterior trabajamos :

- ✓ Incorporar el componente form de bootstrap de manera más rápida para crear formularios
- ✓ Aprender a modificar y agregar menú a partir del componente navbar



LEARNING PATHWAY

¿Sobre qué temas trabajaremos?



En esta lección, exploraremos las bases del lenguaje JavaScript y su relevancia en el desarrollo web. Aprenderemos sobre la historia de JavaScript y cómo se ha convertido en un lenguaje esencial en el mundo de la programación web. También nos sumergiremos en el concepto de variables en JavaScript y cómo se utilizan para almacenar y manipular datos en nuestras aplicaciones.

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Comprender la historia y el origen de JavaScript como un lenguaje de programación esencial en el desarrollo web.



Reconocer la importancia de JavaScript en el contexto del desarrollo web y su papel en la creación de experiencias interactivas para los usuarios.



Aprender el concepto de variables en JavaScript y cómo se utilizan para almacenar y manipular datos en las aplicaciones web






Rompehielo

Consigna: 

Encuentra el animal que esconde cada palabra:

- TOGA
- NOTAR
- RUBOR
- GLORIA
- PRECIO
-  NEURALGICA

➤ Breve historia de JavaScript



Breve historia de JavaScript

JavaScript es un lenguaje de programación que fue creado en 1995 por Brendan Eich, quien trabajaba en Netscape Communications Corporation en ese momento. Inicialmente, el lenguaje se llamaba "LiveScript" y fue desarrollado como una forma de agregar interactividad y funcionalidad a las páginas web.





Breve historia de JavaScript

En ese momento, las páginas web se componían principalmente de contenido estático y no había una forma eficiente de agregar elementos dinámicos o interactuar con el usuario. Con la creación de JavaScript, se abrió un nuevo mundo de posibilidades para la web.

En 1996, Netscape presentó JavaScript junto con el lanzamiento de Netscape Navigator 2.0. Esto marcó el comienzo de la popularización de JavaScript, ya que se convirtió en un lenguaje de scripting incorporado en los navegadores web, lo que permitió a los desarrolladores crear páginas web más interactivas y dinámicas.

- 1997: ES 1. Primera versión.
- 1998: ES 2.
- 1999: ES 3.
- 2009: ES 5.
- 2015: ES 6.



Breve historia de JavaScript

En los años siguientes, JavaScript continuó evolucionando y ganando popularidad. En 1997, la especificación de JavaScript fue adoptada por la Organización Internacional de Normalización (ISO) y se convirtió en el estándar ECMA-262.

Con el tiempo, JavaScript se convirtió en un lenguaje de programación versátil que se utiliza tanto en el lado del cliente como en el lado del servidor. Junto con HTML y CSS, JavaScript se convirtió en uno de los pilares fundamentales del desarrollo web moderno.



1997: ES 1. Primera versión.



1998: ES 2.



1999: ES 3.



2009: ES 5.



2015: ES 6.





Breve historia de JavaScript

Javascript se utiliza tanto para construir aplicaciones de Frontend como de Backend. Por Frontend entendemos a la parte de la aplicación que corre en el navegador y con la cual interactúan los usuarios.

Nuestra aplicación de Frontend también consume datos y servicios ofrecidos por algún Backend. Javascript será la herramienta que nos permitirá comunicarnos e intercambiar información con APIs u otras aplicaciones.



1997: ES 1. Primera versión.



1998: ES 2.



1999: ES 3.



2009: ES 5.



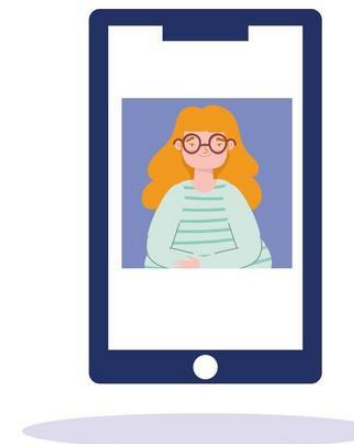
2015: ES 6.





Algoritmos

En programación, un algoritmo es un conjunto de procedimientos o funciones ordenados que se necesitan para realizar cierta operación o acción. Por ejemplo, en una suma el algoritmo implica tomar un dato, sumarlo a otro y obtener un resultado.





Algoritmos

Pensar en algoritmos es una práctica que debemos fortalecer como desarrolladores. Consiste en encarar un problema complejo y dividir su resolución en diversos pasos, pensar cómo resolver cada uno y luego secuenciarlos correctamente para llegar al resultado esperado.



➤ Relevancia de Javascript

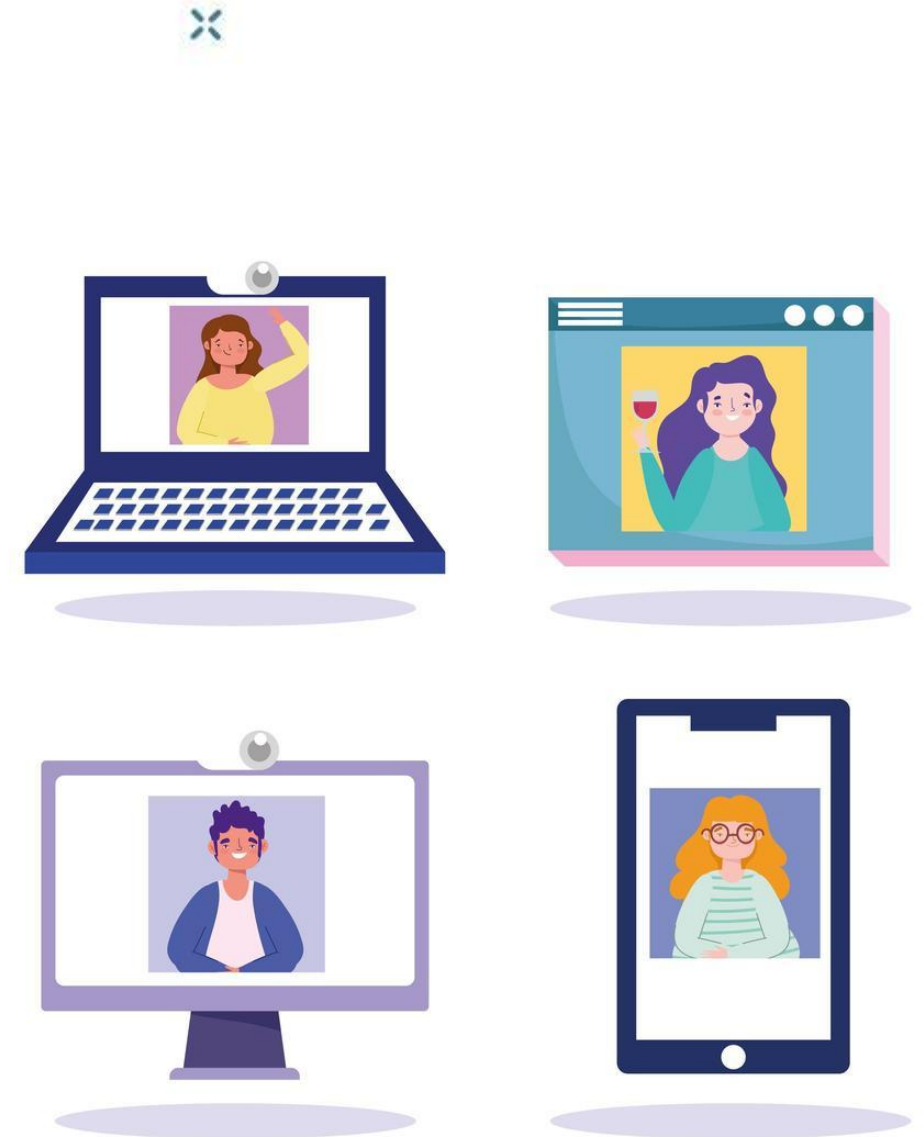


Relevancia de Javascript

JavaScript tiene sus propias reglas para la sintaxis, aunque respeta los estándares de muchos lenguajes de programación lógicos.

Nuestro código JavaScript se asocia al archivo HTML que se carga en el navegador. Tenemos dos maneras de escribir código JavaScript en nuestras aplicaciones web.

JavaScript es un lenguaje de programación de alto nivel que desempeña un papel fundamental en el desarrollo web y en la creación de aplicaciones interactivas en el navegador





Relevancia de Javascript

Interactividad en el navegador

JavaScript permite agregar interactividad y dinamismo a las páginas web. Gracias a este lenguaje, es posible crear efectos visuales, animaciones, juegos, validaciones de formularios y muchas otras características que mejoran la experiencia del usuario en el navegador.





Relevancia de Javascript



Desarrollo web front-end:

JavaScript es el principal lenguaje utilizado en el desarrollo de la capa de presentación o front-end de las aplicaciones web. Junto con HTML y CSS, JavaScript permite construir interfaces de usuario interactivas y receptivas, proporcionando una experiencia de usuario fluida e intuitiva.





Relevancia de Javascript



Frameworks y bibliotecas populares:

Existen numerosos frameworks y bibliotecas de JavaScript, como React, Angular y Vue.js, que facilitan y agilizan el desarrollo de aplicaciones web complejas. Estas herramientas proporcionan una arquitectura modular, componentes reutilizables y una forma eficiente de manejar la lógica del front-end.





Relevancia de Javascript



Desarrollo web back-end:

JavaScript no se limita solo al front-end, ya que también puede utilizarse para el desarrollo del lado del servidor o back-end. Node.js, una plataforma basada en JavaScript, permite ejecutar código JavaScript en el servidor, lo que brinda la posibilidad de crear aplicaciones web completas utilizando un solo lenguaje en todos los niveles.

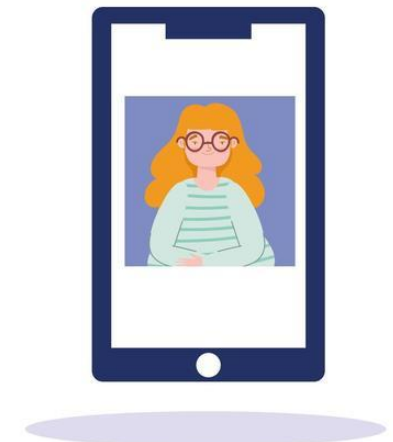




Relevancia de Javascript

Compatibilidad con múltiples navegadores:

JavaScript es compatible con la mayoría de los navegadores web modernos, lo que garantiza que las aplicaciones desarrolladas en JavaScript sean accesibles y se ejecuten correctamente en diferentes entornos.





Relevancia de Javascript

Integración con tecnologías emergentes:

JavaScript se ha adaptado y evolucionado para integrarse con tecnologías emergentes como la inteligencia artificial, la realidad virtual, la realidad aumentada y el Internet de las cosas. Esto permite desarrollar aplicaciones avanzadas y aprovechar las últimas tendencias tecnológicas.





¿Cómo escribir código JS?



Dentro de un archivo html, podemos escribir código de JavaScript entre medio de las etiquetas `<script>`

Al agregar comentarios en tu código JavaScript puedes hacerlo más legible y comprensible. Los comentarios son líneas de código que no se ejecutan y se utilizan para agregar notas o explicaciones. En JavaScript, puedes usar `//` para comentarios de una línea y `/* ... */` para comentarios de varias líneas.



```
<script>
    //aca va el código en JS
    // Esto es un comentario de una línea

    /*
        Esto es un comentario
        de varias líneas
    */
</script>
```



¿Cómo escribir código JS?

En un archivo individual con extensión .js podemos escribirlo de la siguiente manera:

Ejemplo: mi-archivo.js

Para ejecutar el código JavaScript en un navegador web, debes enlazar el archivo JavaScript al archivo HTML correspondiente. Para ello, puedes usar la etiqueta `<script>` en el encabezado o al final del archivo HTML y especificar la ruta del archivo JavaScript.

```
<script src="ruta/al/archivo/script.js">
</script>
```

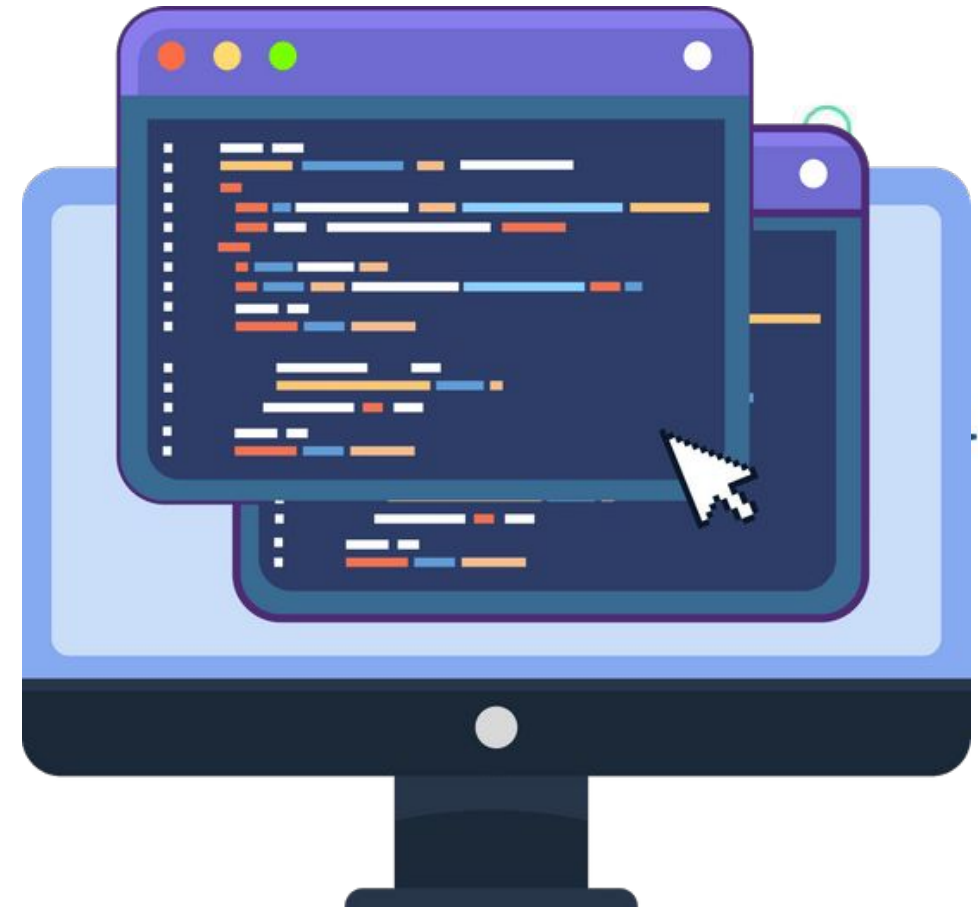


› Variables



Variables

Asegúrate de seguir la sintaxis correcta del lenguaje JavaScript. Presta atención a los paréntesis, llaves, puntos y comas. Un error de sintaxis puede provocar que el código no funcione correctamente.



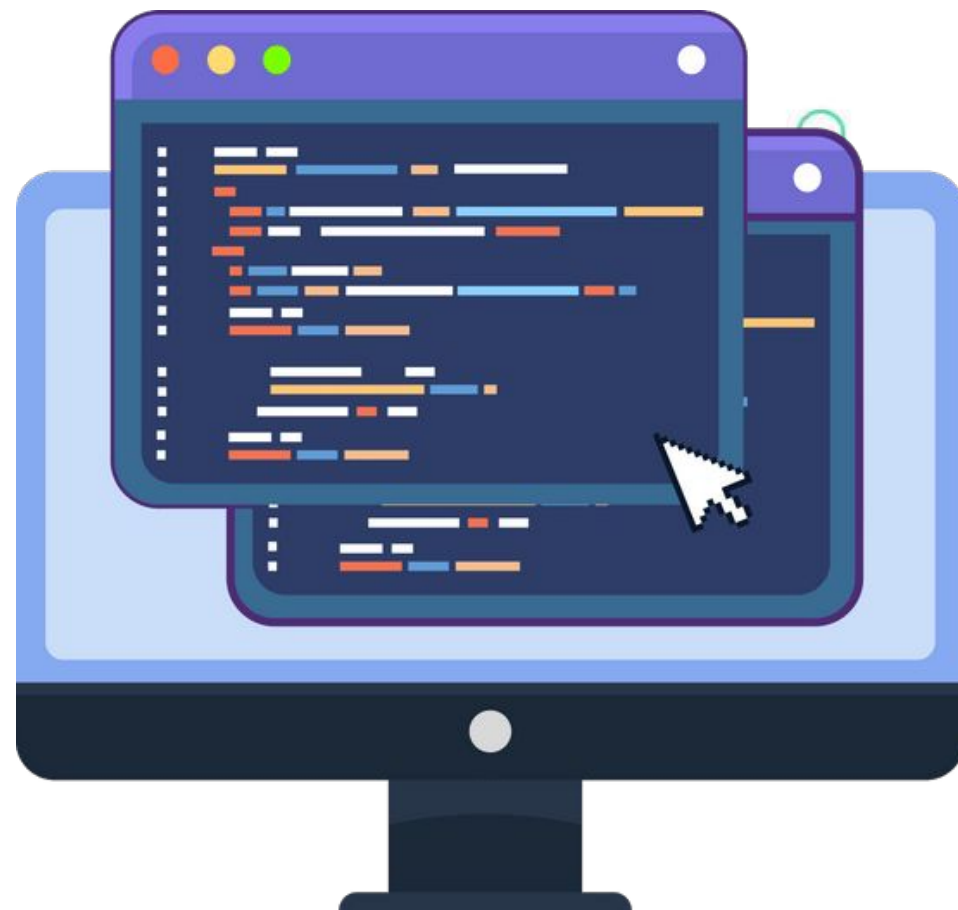


Variables

En JavaScript, las variables se utilizan para almacenar y manipular datos. Para declarar una variable en JavaScript, se utilizan las palabras clave `var`, `let` o `const`, seguidas del nombre de la variable. Acá tenes un resumen de cada una de estas palabras clave:

`var`: Era la forma tradicional de declarar variables en JavaScript, pero se ha vuelto menos común con la introducción de `let` y `const`.

```
var edad = 25;
```



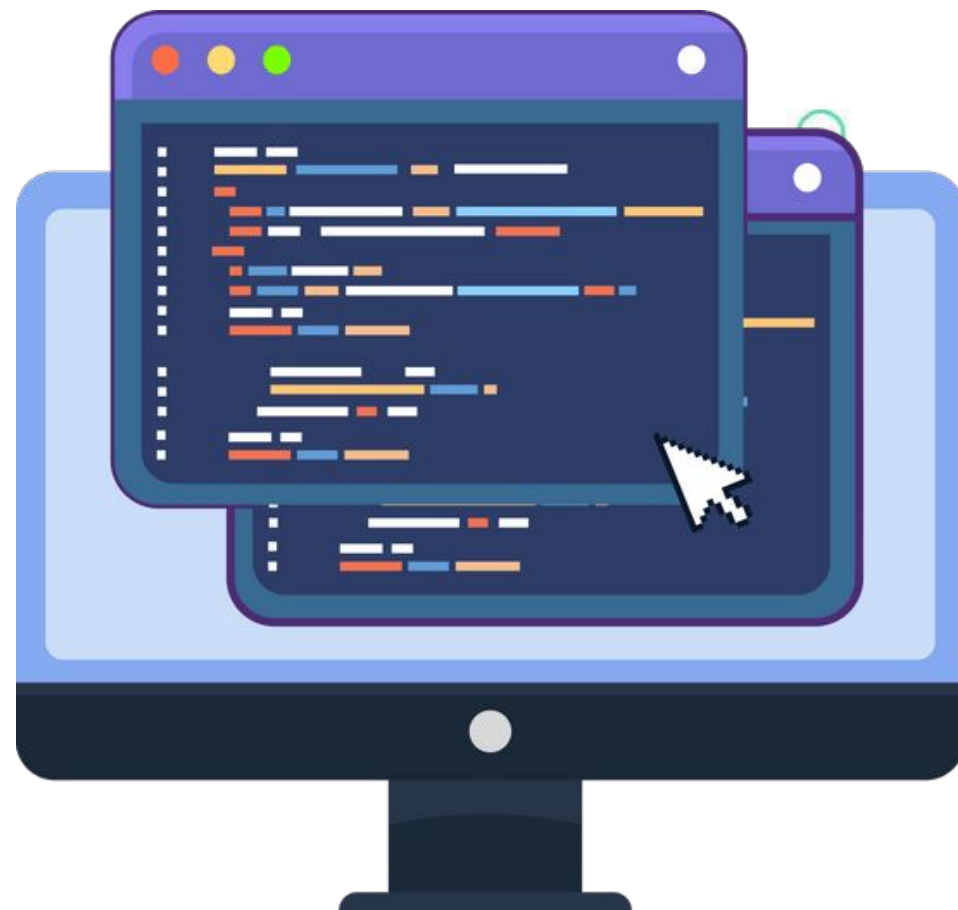


Variables

let:

Introducido en ECMAScript 6 (ES6), let permite declarar variables con un alcance de bloque. Las variables declaradas con let están limitadas al bloque en el que se declaran, como un bloque de código dentro de una función, un bucle for o un bloque if.

```
let nombre = 'Juan';  
const nacionalidad = 'Argentina';
```



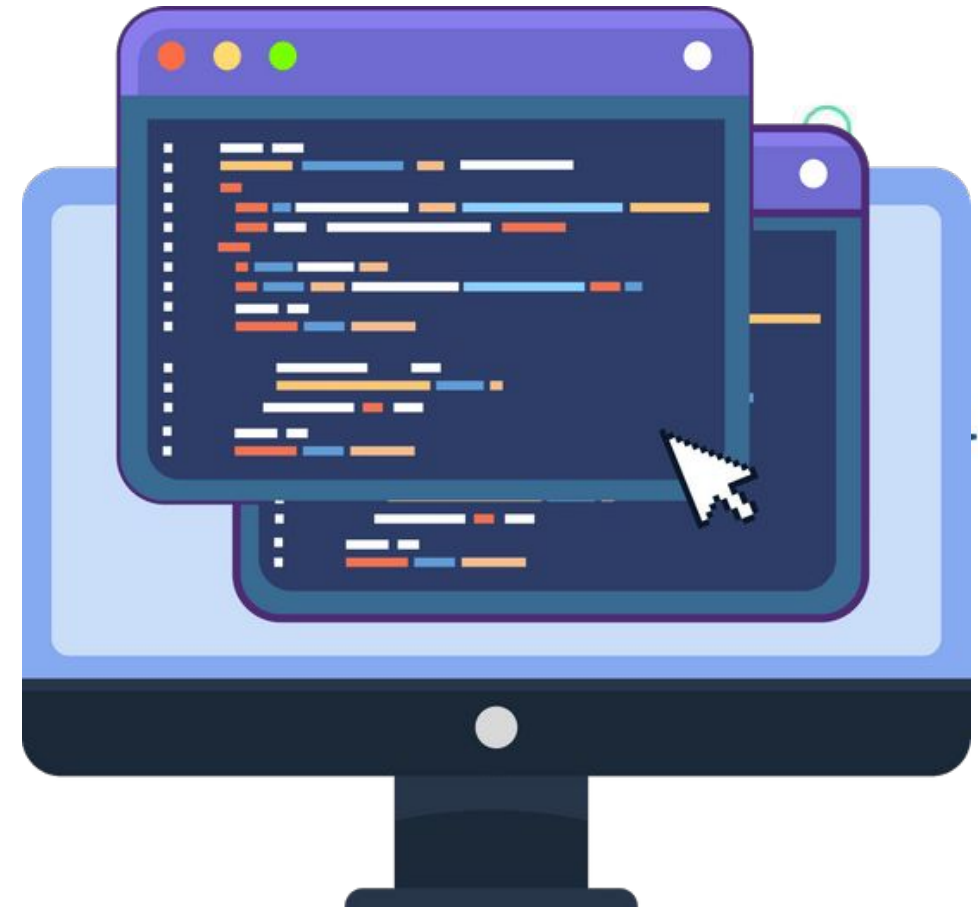


Variables

const:

También introducido en ES6, const se utiliza para declarar constantes, es decir, variables que no pueden cambiar su valor una vez asignado. Al igual que let, const tiene un alcance de bloque.

```
const pi = 3.1416;
```



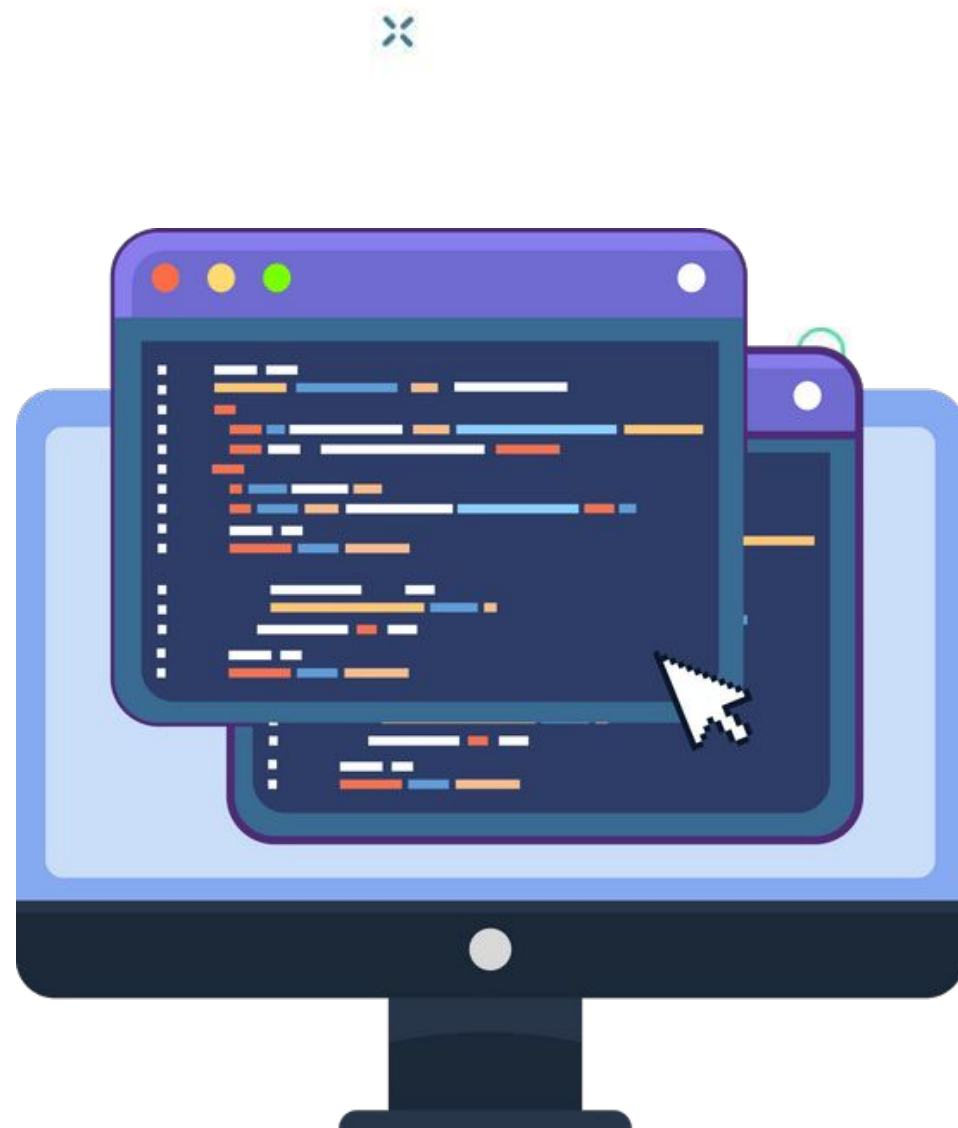


Variables

Es importante tener en cuenta que `let` y `const` siguen las mejores prácticas actuales para declarar variables en JavaScript. Se recomienda utilizar `const` siempre que sea posible, ya que ayuda a prevenir la reasignación accidental de valores. Si sabes que el valor de una variable cambiará, puedes utilizar `let`.

Además de declarar variables, puedes asignarles valores utilizando el operador de asignación `=`:

```
let mensaje = 'Hola, mundo!';
```

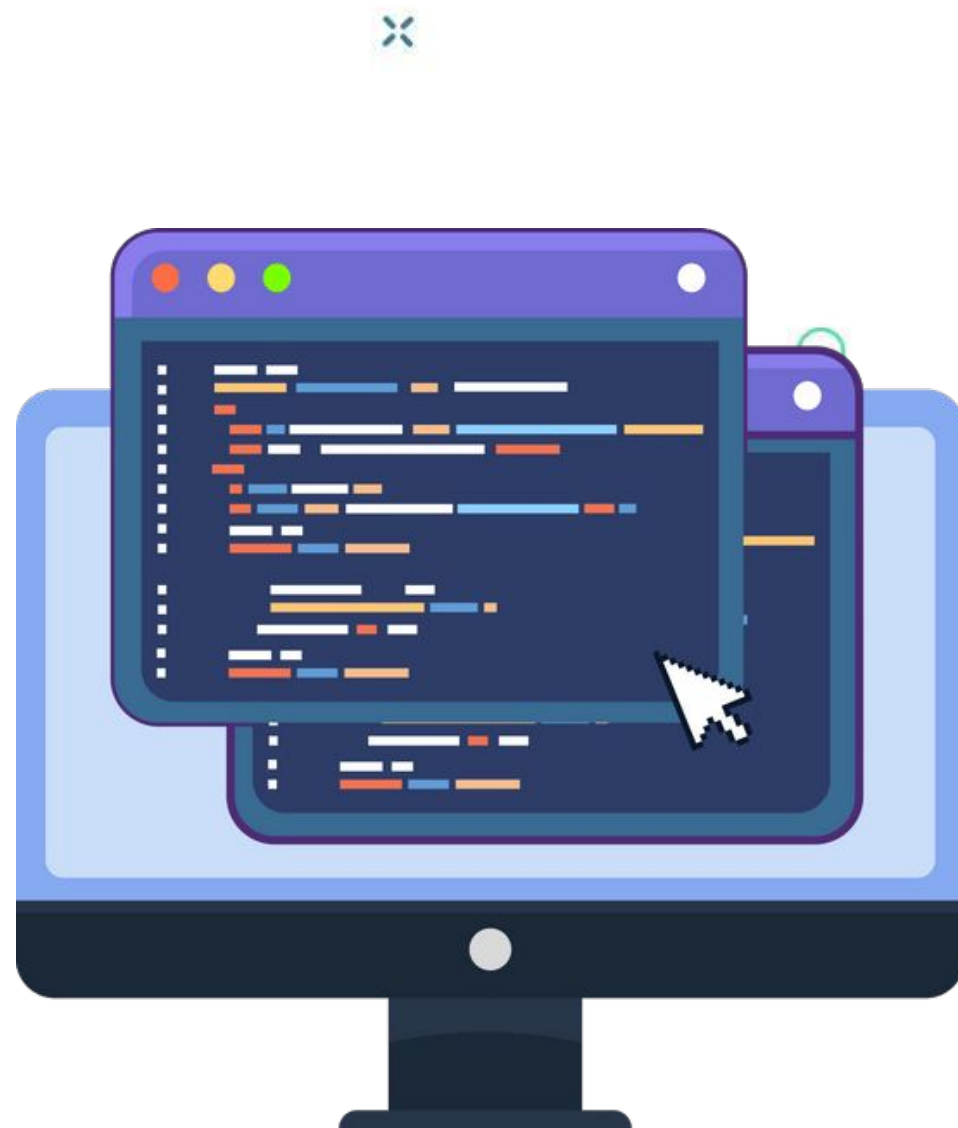




Variables

Una vez que una variable ha sido declarada, puedes utilizarla en tu código, ya sea para realizar operaciones, imprimir su valor en la consola o utilizarla en otras expresiones.

```
let x = 10;  
let y = 5;  
let suma = x + y;  
console.log(suma);  
// Imprime 15 en la consola
```





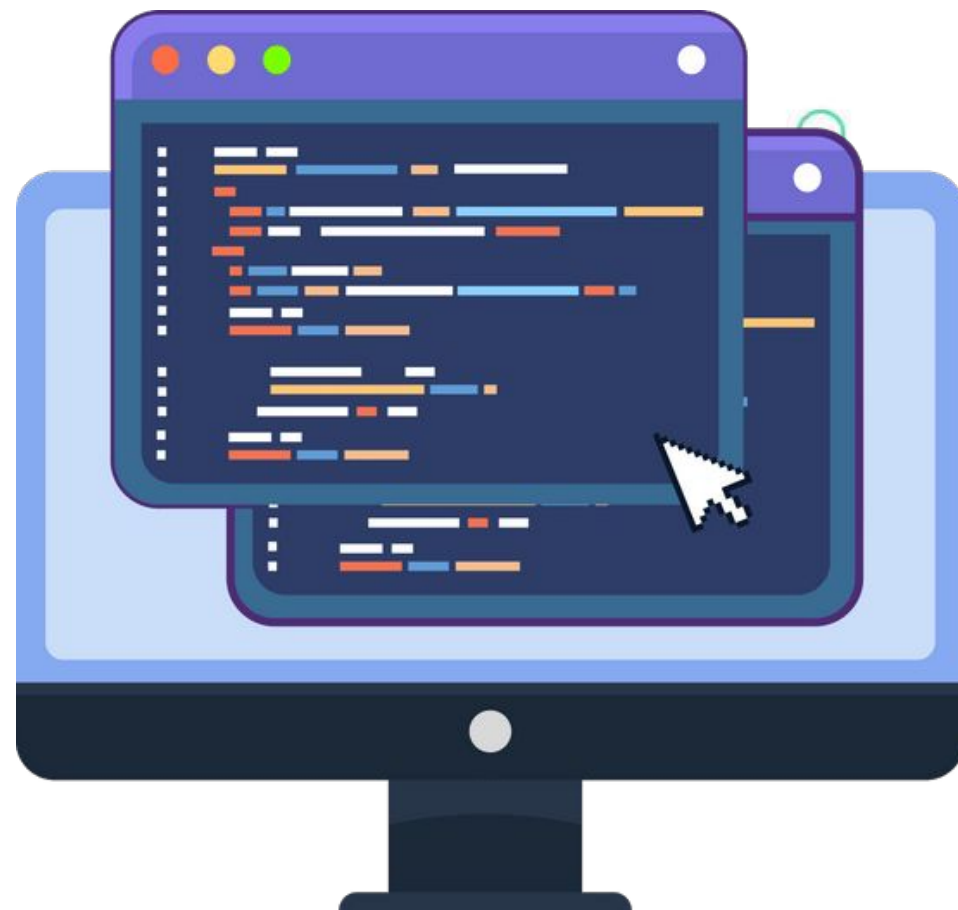
Variables

Declaración

La declaración de variables en JavaScript implica crear una variable utilizando las palabras clave `var`, `let` o `const`, seguidas del nombre deseado para la variable. Es importante seguir algunas reglas al nombrar las variables:

No se deben utilizar espacios en el nombre de la variable. En su lugar, se pueden utilizar guiones bajos (`_`) o notación `camelCase` (la primera letra de cada palabra, excepto la primera, se escribe en mayúscula).

```
var miVariable;  
let nombreCompleto;  
const numeroDeTelefono;
```





Variables



Declaración

Los nombres de las variables no pueden comenzar con un número. Deben comenzar con una letra, un guión bajo (_) o un signo de dólar (\$).

Evita el uso de caracteres especiales, como !, @, #, \$, %, etc. en los nombres de las variables. Solo se permiten letras, números, guiones bajos (_) y signos de dólar (\$).

```
var _precio;  
let $cantidad;  
  
var cantidad_total;  
let usuario2;  
const MAXIMO_VALOR;
```





Variables



Asignación

La asignación de valores a una variable, se realiza utilizando el operador de asignación, que es el signo igual (=). Este operador asigna el valor a la variable en el lado izquierdo de la expresión.

```
var nombre = "Juan"; // Asigna el valor
"Juan" a la variable nombre
let edad = 25;
// Asigna el valor 25 a la variable edad
const pi = 3.1416;
// Asigna el valor 3.1416 a la constante
pi
```





Variables



Asignación

Es importante tener en cuenta que en JavaScript, el tipo de dato de una variable no está predefinido y puede cambiar durante la ejecución del programa. Esto se conoce como tipado dinámico.

Es importante tener en cuenta que el operador de asignación (=) no debe confundirse con el operador de igualdad (== o ===) que se utiliza para comparar valores.



```
let numero = 10;  
// Asigna el valor 10 a la variable  
numero  
numero = "veinte";  
// Cambia el valor de la variable numero  
a la cadena de texto "veinte"
```



Variables



Inicializar variables

En JavaScript es posible declarar una variable y asignarle un valor inicial en el mismo proceso. Esto se conoce como inicialización de variable..

La inicialización de variables en el mismo proceso puede ser útil cuando conoces el valor inicial que deseas asignar a la variable al momento de declararla. Esto evita la necesidad de realizar una asignación por separado después de la declaración.



```
var edad = 25;  
// Declaración y asignación inicial de la  
variable edad  
let nombre = "Juan";  
// Declaración y asignación inicial de la  
variable nombre  
const pi = 3.1416;  
// Declaración y asignación inicial de la  
constante pi
```



Variables



Inicializar variables

Es importante tener en cuenta que al inicializar una variable, el valor inicial puede ser de cualquier tipo de dato válido en JavaScript, como números, cadenas de texto, booleanos, arreglos, objetos, etc.

Al declarar e inicializar variables, es recomendable seguir las mejores prácticas de nomenclatura y elegir nombres descriptivos que reflejen el propósito o la naturaleza de la variable.



```
let temperatura = 25.5;
// Declaración y asignación inicial de la
variable temperatura (tipo número)
const nombreCompleto = "María Pérez";
// Declaración y asignación inicial de la
constante nombreCompleto (tipo cadena de
texto)
var esActivo = true;
// Declaración y asignación inicial de la
variable esActivo (tipo booleano)
```

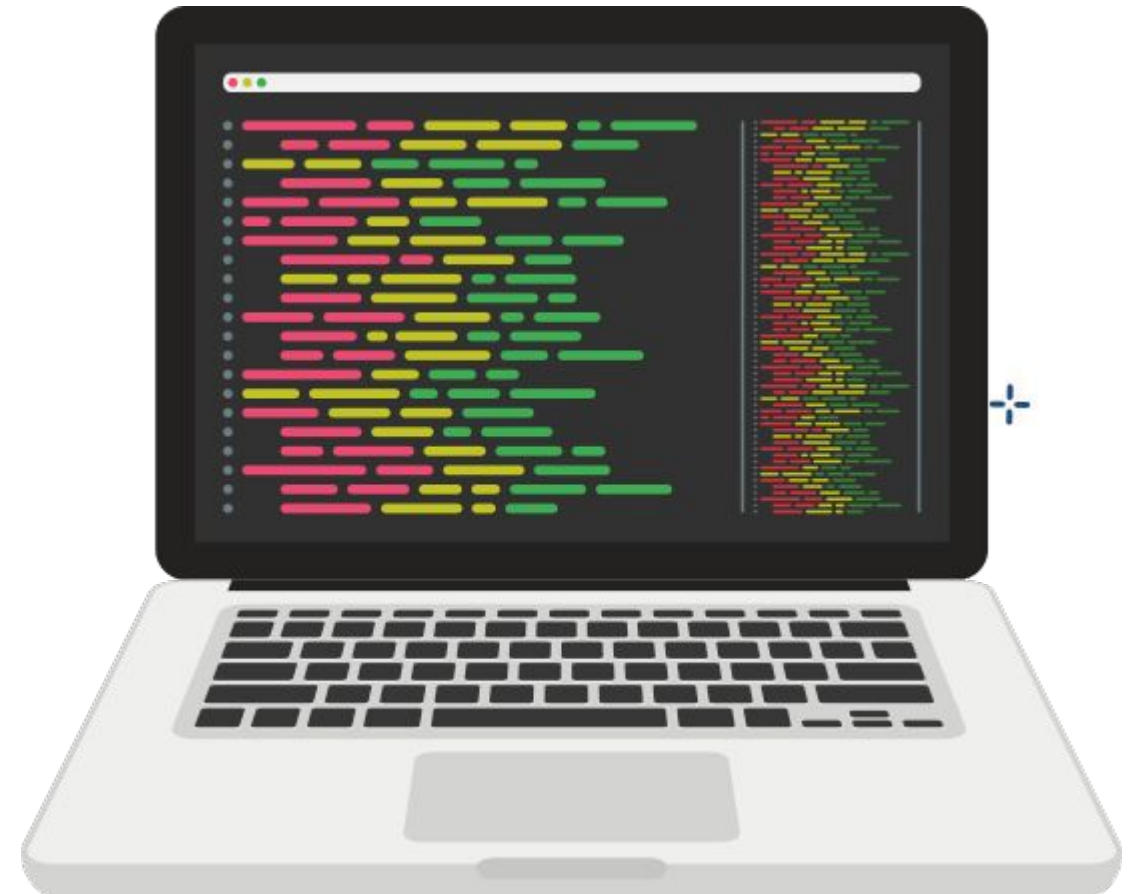
➤ Funciones Nativas



Funciones Nativas

Las funciones nativas, también conocidas como funciones incorporadas o funciones predefinidas, son funciones que vienen incluidas en JavaScript y están disponibles para su uso directo sin necesidad de definirlas previamente

Estas funciones proporcionan una amplia gama de funcionalidades y permiten realizar diversas tareas comunes de manera eficiente.





¿Cómo escribir código JS?

console.log():

Esta función se utiliza para imprimir mensajes en la consola del navegador o en la consola del entorno de desarrollo. Es útil para realizar pruebas y depurar el código, ya que puedes imprimir valores y mensajes para verificar el flujo de ejecución.

```
console.log("Mensaje de prueba");
```





¿Cómo escribir código JS?



alert():

La función alert() muestra una ventana emergente en el navegador con un mensaje para el usuario. Es útil para mostrar mensajes de advertencia, notificaciones o solicitar confirmación del usuario.

```
alert(";Hola, mundo!");
```





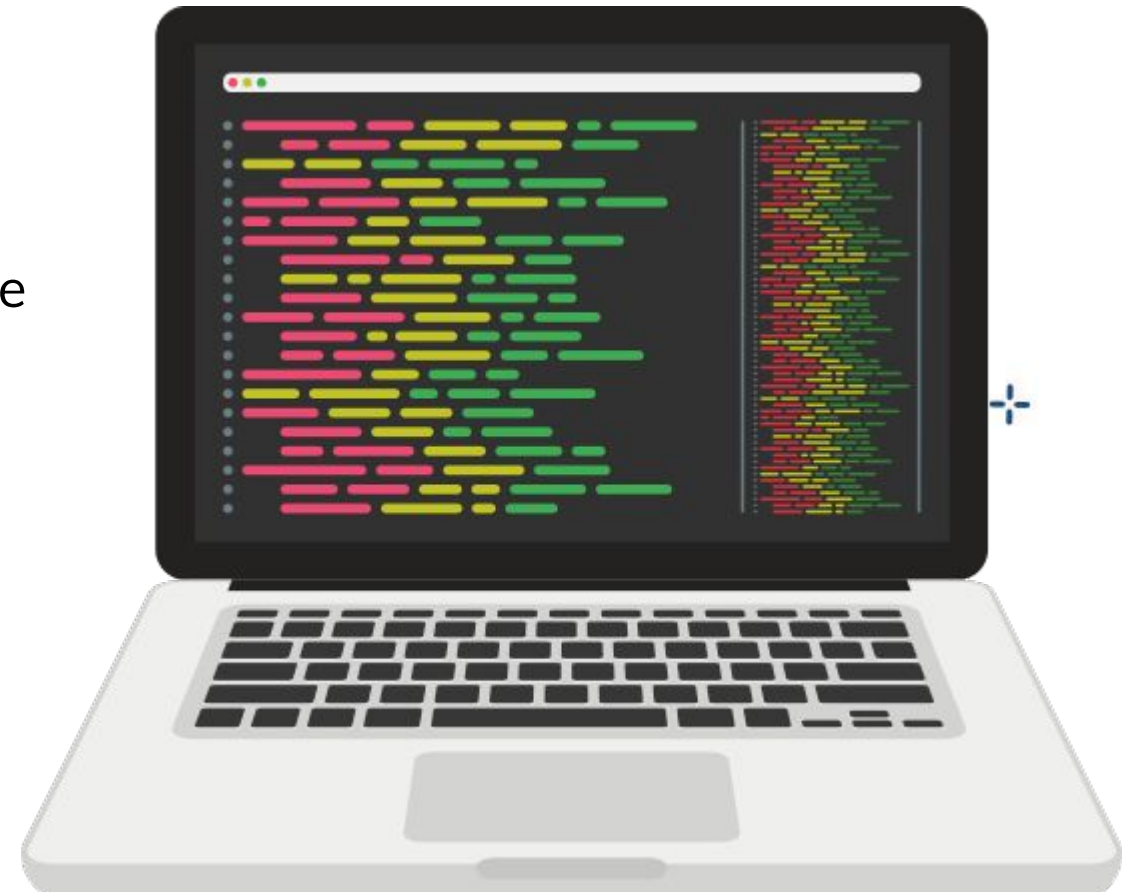
¿Cómo escribir código JS?



confirm():

La función `confirm()` muestra una ventana emergente en el navegador con un mensaje y botones de confirmación ("Aceptar" y "Cancelar"). Es útil para obtener la confirmación del usuario en situaciones específicas.

```
confirm("¿Está seguro de eliminar este  
elemento?");
```





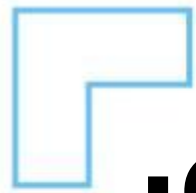
¿Cómo escribir código JS?

Prompt:

El prompt en JavaScript es una función que muestra un cuadro de diálogo al usuario para solicitar una entrada de datos. Proporciona una forma sencilla de obtener información del usuario a través de una ventana emergente en el navegador.

```
prompt("Por favor, ingrese su nombre:")
```





¿Cómo escribir código JS?



parseInt() y parseFloat():

Estas funciones se utilizan para convertir una cadena de texto en un número entero o de punto flotante, respectivamente. Son útiles cuando se necesita manipular datos numéricos ingresados como cadenas de texto.

```
parseInt("10");  
parseFloat("3.14");
```





¿Cómo escribir código JS?



Concatenación:

La concatenación en JavaScript se refiere a la unión de dos o más cadenas de texto en una sola. Es una operación común cuando necesitas combinar texto para formar mensajes, etiquetas, sentencias, entre otros.

En JavaScript, puedes realizar la concatenación de varias formas:

Utilizando el operador de concatenación +: El operador + se utiliza para unir dos cadenas de texto.

```
let nombre = "Juan";  
let apellido = "Pérez";  
let nombreCompleto = nombre + " " +  
    apellido;  
console.log(nombreCompleto);  
// Resultado: "Juan Pérez"
```



¿Cómo escribir código JS?



Utilizando plantillas de cadena (template literals):

Las plantillas de cadena son una forma más moderna y conveniente de realizar concatenación en JavaScript. Se utilizan utilizando comillas graves (``) y permiten incrustar variables o expresiones dentro del texto utilizando \${}.

La concatenación de cadenas es una operación fundamental en JavaScript y se utiliza ampliamente en el desarrollo de aplicaciones web para generar y manipular texto dinámicamente.

```
let nombre = "Juan";  
let mensaje = `Hola, ${nombre}!`;  
console.log(mensaje);  
// Resultado: "Hola, Juan!"
```

LIVE CODING

Ejemplo en vivo

Funcionalidades:

Eres un instructor que está enseñando los conceptos básicos de JavaScript a un grupo de estudiantes. Les mostrarás cómo declarar variables y asignarles valores en tiempo real.

Crear un algoritmo que solicita al usuario su nombre y edad, y luego muestra un mensaje de bienvenida en la consola.

 **Tiempo: 10 minutos**



Evaluación Integradora ✨

Trabajo Integrador del Módulo 💪

Iremos completándolo progresivamente clase a clase





Ejercicio

Declarando variables



Declarando variables

Breve descripción



Contexto: 🙌

Eres un instructor de programación que está enseñando cómo interactuar con el usuario utilizando JavaScript.



Consigna: 📝

Realizar un algoritmo para demostrar cómo utilizar las funciones prompt, alert y console en JavaScript para interactuar con el usuario.

El tema es a elección

Tiempo 🕒: 10 minutos

○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ Aprendimos que JavaScript se llamaba "LiveScript" pero se renombró a "JavaScript" para aprovechar la popularidad del lenguaje Java en ese momento.
- ✓ Comprender la importancia de JavaScript en el desarrollo web moderno.
- ✓ Explorar el concepto de variables en JavaScript y cómo se utilizan para almacenar y manipular datos.



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 🙌 🙌 🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. Material 1: Lección 5: Bases del lenguaje Javascript: 12-24
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

Momento: ✚

Time-out!

🕒 5 min.

