

Bootcamp Java Developer

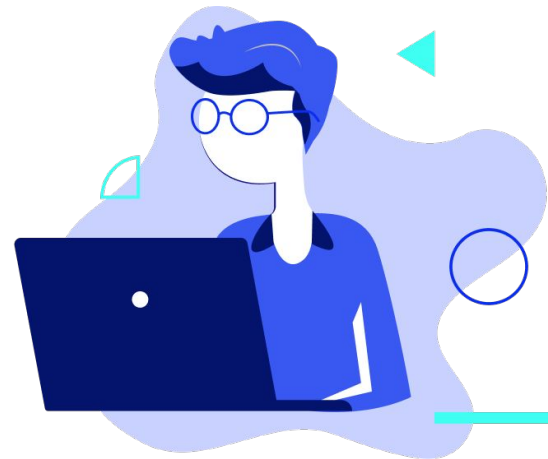
Fase 2 - Java Web Developer
Módulo 17

Introducción

Aprender a programar

Un programa es un **conjunto de instrucciones** que ejecuta un procesador de computadora. Todo programa tendrá un conjunto finito de instrucciones, que **se van ejecutando 1 a 1 en cadena, hasta finalizar la ejecución.**

Para empezar a trabajar con un programa es importante entender que **es un plan** que se va a llevar a cabo.



Pseudocódigo

El **pseudocódigo omite detalles** que quizás se vayan a trabajar con el lenguaje de programación elegido, porque no son esenciales para comprender en sí de qué se trata el programa y cuál es su fin. Sirve para planificar correctamente **aquello que se llevará a cabo en el lenguaje de programación que se decida implementar**.

Se basa en convenciones de un lenguaje de programación cualquiera pero lo hace entendible para las personas, y aparte lo realiza **independientemente de cualquier lenguaje específico de programación**.

```
algoritmo Sumar

variables
    entero a, b, c

inicio
    escribir( "Introduzca el primer número (entero): ")
    leer(a)
    escribir( "Introduzca el segundo número (entero): ")
    leer(b)
    c ← a + b
    escribir( "La suma es: ", c)
fin
```

Aplicación: ¿Cómo reconocerla?

Una aplicación es un **programa específico que resuelve un problema concreto**. A menudo hablamos de aplicaciones contables, aplicaciones de gestión de RR.HH, aplicaciones de liquidación de sueldos, y otras. Una de las características principales de las aplicaciones es la **interacción directa con el usuario**.

Un sistema está formado por un **conjunto de programas, involucra también herramientas de hardware** (partes físicas: monitores, teclados, impresoras, etc.), **redes de comunicación, bases de datos, servidores**.




Tipos de aplicaciones

Aplicaciones de escritorio

Son aquellas que típicamente corren en un **sistema Windows** y pueden ser abiertas yendo a la lista de programas instalados en el sistema operativo. **Trabajan con ventanas, tienen un menú** en la parte superior (con opciones tales como: *archivos, herramientas, configuración*, y otros). Estas aplicaciones permiten ingresar datos, obtener reportes de datos, etc. Existe mucha **interacción con el teclado y el mouse** de la computadora.

El botón **secundario del mouse** suele generar el conocido menú contextual, muy útil ya que representa un atajo para la ejecución de una funcionalidad específica.

Podemos decir que las aplicaciones de escritorio son las “**aplicaciones tradicionales**” de **interfaz gráfica**.



Aplicaciones de consola

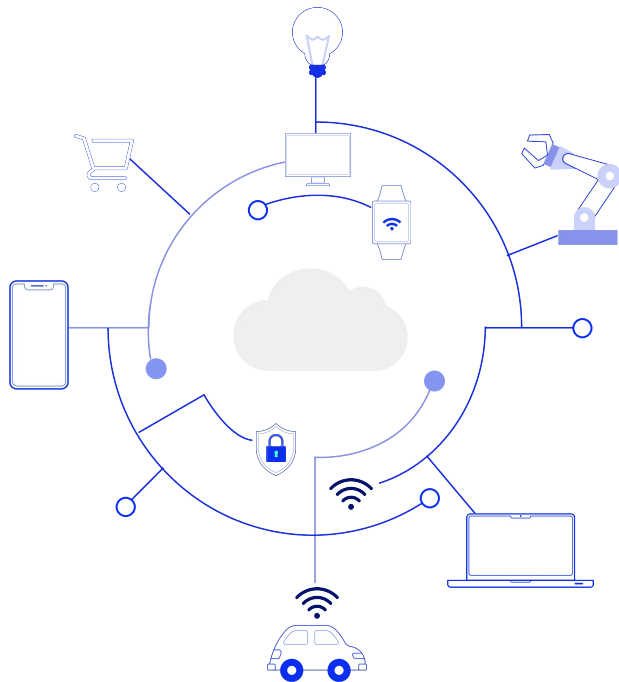
Son las que utilizan una ventana de **MS-DOS** como **salida**. Quizás, esta definición no te ayude demasiado a entender de qué se trata, pero básicamente **Visual Basic.Net y C#** utilizan este tipo de consola para poder **programar y crear aplicaciones de escritorio**.



Aplicaciones web

Se accede a ellas desde un browser (Edge, Firefox, Chrome, y otros), a través de alguna dirección **web** o **url**.

Los lenguajes de programación propios de la web han invadido diferentes espacios, y no necesariamente se utilizan para crear páginas o sitios web. Las empresas pueden requerir aplicaciones web para manejar cuestiones internas porque brindan la posibilidad de **acceder desde cualquier dispositivo mientras haya conexión a internet**.



Aplicaciones Mobile

Se trata nada más y nada menos de las famosas "apps". Funcionan sobre dispositivos mobile (tablets, celulares, etc.). Corren en sistemas operativos móviles como **Android, iOS, BlackBerry OS, Windows Phone, y otros.**

Resuelven de forma más simple gestiones y operaciones del usuario. Por ejemplo, las apps de **Mercado Libre, Instagram, Uber, Rappi, Cabify** seguramente son las que más utilizas en tu teléfono o dispositivo móvil.



Lenguajes de programación

Se trata de un lenguaje formal con **reglas estrictas de escritura**, que permite comunicarle a una computadora qué es lo que debe hacer con absoluto detalle.

Todo lenguaje de programación se conforma por un conjunto de símbolos, signos de puntuación, operadores, valores, palabras clave e identificadores que permiten escribir las instrucciones a ejecutar.

A través de los lenguajes de programación es posible **crear programas**.

Existen docenas y docenas de lenguajes de programación, muchos son similares entre sí, también tienen algunas diferencias. Lo más importante es comprender cómo JavaScript y todo lo que deriva de este lenguaje de programación es, hoy, el centro del universo tecnológico.

Es impensado, casi en todos los ámbitos, que un programador aunque conozca y maneje otro lenguaje, no sepa JavaScript. Vamos a contarte por qué.

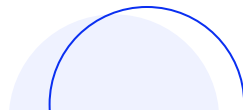
JavaScript como lenguaje de programación

JavaScript es el centro de todo porque es un lenguaje de programación **interpretado por el propio navegador** (Chrome, Firefox, Opera, IE, y otros), sin necesidad de absolutamente nada más.

La web domina el mundo de la tecnología, desde la creación de interfaces para fábricas de autos, cajeros automáticos, o simplemente aplicaciones para lograr que los empleados puedan desde cualquier lugar donde hay conexión a Internet resolver cualquier problema laboral o trabajar sin necesidad de movilizarse, . Esto reduce costos y mejora el rendimiento.

Entre las tantas cosas que podemos hacer con JavaScript se pueden mencionar:

- **Abrir ventanas.**
- **Mostrar mensajes.**
- **Validar datos en un formulario.**
- **Hacer una galería de imágenes.**



Implementar JS: manera interna

Se ubica la **etiqueta script** en el head y el body de cualquier documento **HTML** y se comienza a trabajar con el código.

Los elementos HTML con los que trabajaremos siempre deberán ser ubicados debajo de la etiqueta **<script>** que escribimos en el **body**.

```
<script>  
alert('hola soy js')  
</script>
```

Implementar JS: en línea

La idea es hacerlo dentro de las propias etiquetas de HTML. En el ejemplo siguiente, la misma alerta anterior se dispara al momento de levantar la página en el navegador.

Lo malo de trabajar de esta forma es que solo afecta al HTML en el que se encuentra y es muy engorroso si tenemos sitio web al momento de hacer cualquier cambio o en el usual mantenimiento.

```
<body onload="alert('hola soy js')">
```

Esta página dice

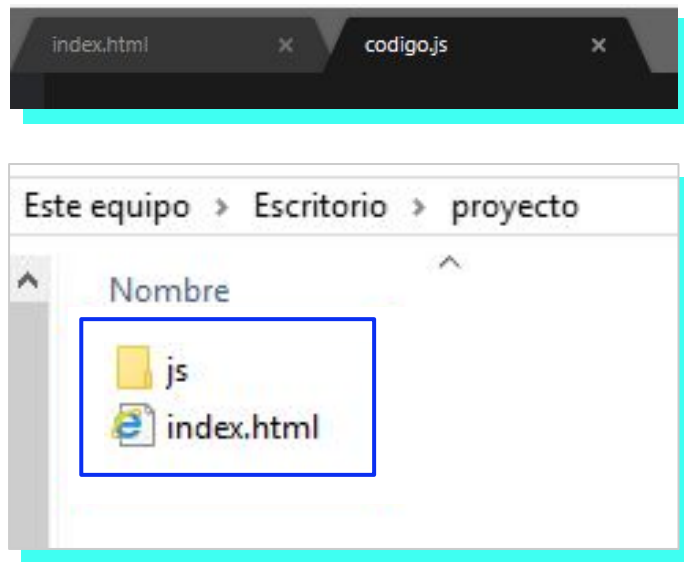
hola soy js

Aceptar

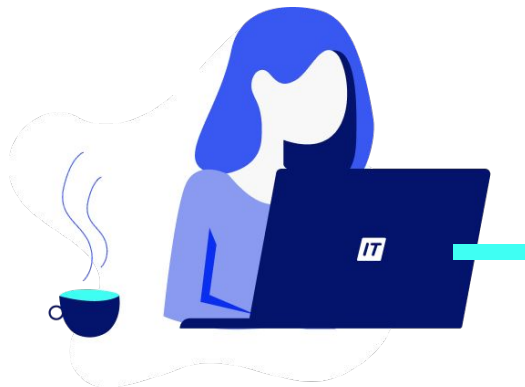
Implementar JS: manera externa

Esta forma es la ideal porque se trabajará con un archivo **externo de .js**. Por ejemplo, se guardarán los siguientes elementos para trabajar (imagen superior):

Para lograr un proceso **más ordenado te recomendamos generar una carpeta *js***, como en la imagen de abajo.



Dentro del **archivo.js**, se comenzará a escribir el código JavaScript necesario, pero este archivo **debe vincularse con el HTML** para que las funciones programadas se ejecuten sobre los tags correspondientes.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Desarrollo</title>
  <script src="js/codigo.js"> </script>
</head>

<body>

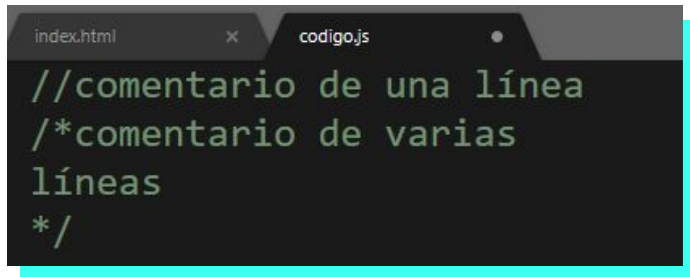
</body>
</html>
```

Sintaxis básica de JS

JavaScript se maneja a través de sentencias que le dicen al navegador (Chrome, Firefox, y otros) qué hacer.

En base a eso, es interesante también saber que tenemos la posibilidad de hacer **comentarios**.

Éstos no afectan a nuestro programa y nos permiten hacer que un código deje de ejecutarse o dejar mensajes que faciliten el desarrollo o la comunicación entre quienes trabajan en una determinada aplicación.

A screenshot of a code editor with two tabs: 'index.html' and 'codigo.js'. The 'codigo.js' tab is active and shows three lines of JavaScript code in a dark theme. The first line is a single-line comment starting with '//' and the second line is a multi-line comment starting with '/*' and ending with '*/'.

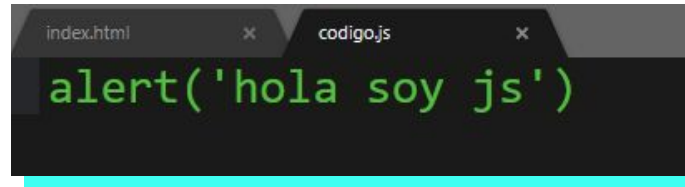
```
//comentario de una línea
/*comentario de varias
líneas
*/
```


Primer script: Ventana de alerta

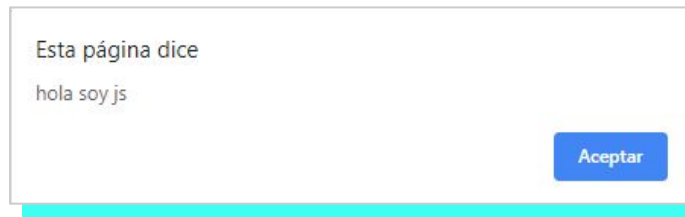
Esta ventana alerta al usuario sobre diferentes situaciones. Aunque, actualmente, las ventanas de alerta fueron reemplazadas, en su gran mayoría, por **ventanas modales más vistosas**.

La realidad es que no ayudarán a aprender la **sintaxis de JS** y son la base de todo lo que aprenderás más adelante.

En tu **codigo.js** escribe lo siguiente:

A screenshot of a code editor with two tabs: 'index.html' and 'codigo.js'. The 'codigo.js' tab is active, showing the code `alert('hola soy js')` in green text on a dark background.

El resultado será el siguiente:

A screenshot of a JavaScript alert dialog box. It has a white background and a thin border. The text inside reads 'Esta página dice' followed by 'hola soy js' on the next line. In the bottom right corner, there is a blue button with the text 'Aceptar' in white.

Si bien no es obligatorio se recomienda el uso de **;** (**punto y coma**) **al final de cada sentencia** para poder así **separarlas y evitar errores**.

En otros lenguajes de programación, como **PHP**, estos separadores son obligatorios, pero en este caso no es necesario. De todas formas, para realizar un código **más ordenado y prolijo** **terminaremos nuestra sentencia con punto y coma**.



```
index.html x  codigo.js x  
alert('hola soy js');
```

A screenshot of a code editor interface. At the top, there are two tabs: 'index.html' and 'codigo.js', both with a close button (x). The 'codigo.js' tab is active, showing a single line of JavaScript code: `alert('hola soy js');`. The code is written in a light green font on a dark background. The entire editor window is highlighted with a thick cyan border.

Revisión

- Repasar los conceptos básicos de un **lenguaje de programación**.
- Trabajar con **pseudocódigos** para empezar a adaptarse a esta lógica.
- Implementar una **ventana de alerta en línea, de forma interna y externa**.
- Aplicar todas las propiedades en el **proyecto integrador**.
- Realizar todas las preguntas necesarias antes de continuar.



**¡Sigamos
trabajando!**