Lenguajes web

Lenguajes cliente *(se ejecutan en el navegador o cliente)*

- HTML
- CSS
- JAVASCRIPT

HTML (HyperText Markup Language)

- **Lenguaje de marcado estándar para crear páginas web**
- Describe la **estructura de la página web.**
- **Consta de una serie de ELEMENTOS**
- **Los elementos HTML** indican al **navegador LA ESTRUCTURA y EL TIPO DE CONTENIDO** y el navegador decidé cómo mostrarlos **por defecto**.

  - El **diseño y aspecto visual** se controla con **CSS**, no con HTML.

Elementos HTML

- Se definen mediante **etiqueta de inicio o apertura**, **contenido del elemento** y **etiqueta de cierre**.

```
<tagname> Content goes here... </tagname>
```

✓ El elemento HTML **es todo** *(desde la etiqueta de apertura a la de cierre)*

✓ ***Un elemento puede tener otro elemento en su interior (HIJO)*** → ***ESTRUCTURA ÁRBOL***

✓ Algunos elementos HTML **no tienen contenido:**

- Se llaman **elementos vacíos**
- **No tienen etiqueta de cierre**
- **Se usan para modificar el contenido que los rodea o incluir recursoS externos.**
- **Ejemplos:** **
, <hr>, <base>, <img>, <input>, <link>, <meta>, <wbr>**

Ejemplos de elementos:

<h1> El origen de la vida </h1>

* Esto es un párrafo *

✓ Las **etiquetas de inicio, y las individuales** (*elementos vacíos)* pueden llevar **ATRIBUTOS**, que:

- Ofrecen **información adicional** acerca de sus contenidos
- **Ej:** <html **lang = "es"**>

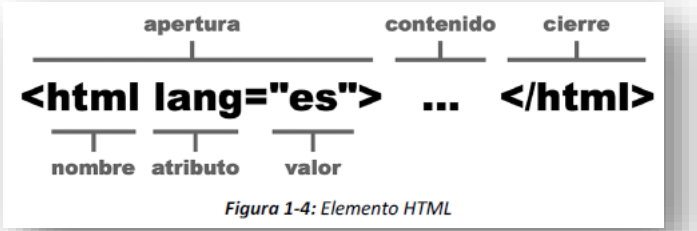

Figura 1-4: Elemento HTML

Cómo es un documento HTML

- Tiene **estructura de árbol**, **con UNA SOLA RAÍZ** que es el elemento **<html>** **// elemento ROOT**
- Estructura básica de un documento HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
</head>
<body>

</body>
</html>
```

CSS (*Cascading Style Sheets*)

- Sirve para dar **estilo y diseño** a las páginas web
- Permite definir para cada elemento HTML:
  - ✓ Colores y tipografías
  - ✓ Crear distribuciones de página (columnas, grids, menús, cabeceras,...)
  - ✓ Bordes, márgenes, sombras, etc.
  - ✓ Animaciones, ...

Regla CSS de nombre “body” con 3 propiedades: width, margin, background-color

```
body {
  width: 100%;
  margin: 0px;
  background-color: #FF0000;
}
```

JavaScript

- Es un **lenguaje de programación.**

- Dentro de un documento HTML se escribe dentro del elemento **<script>**

```
<script>
  function cambiarColor() {
    document.body.style.backgroundColor = "#0000FF";
  }
  document.addEventListener("click", cambiarColor);
</script>
```

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

LENGUAJES SERVIDOR

- HTML, CSS y JavaScript se ejecutan por el **navegador** en el ordenador del usuario(cliente)**.**
- Hay situaciones que se requiere procesar la información en el **servidor web** antes de enviarla al usuario.
  - ✓ Por ej: cuando nos logueamos y nos devuelven una página personalizada.
- **¿Qué son?**
  - ✓ Son lenguajes de programación que **se ejecutan en el servidor web** (antes de que la página llegue al navegador del usuario).
  - ✓ El **navegador no ve el código del servidor**, solo recibe el resultado procesado (HTML, CSS, JS, JSON, etc.).
- **¿Para qué sirven?**
  1. **Generar páginas dinámicas**
     - El servidor puede crear HTML personalizado. Ej: un portal que muestra “Hola, Ana” solo cuando te identificas.
  2. **Conectar con bases de datos**
     - Los lenguajes de servidor permiten guardar y recuperar datos.
  3. **Procesar lógica de negocio**
     - Validar usuarios, calcular precios, controlar permisos, etc.
  4. **Seguridad**
     - Datos sensibles se procesan en el servidor, sin exponer claves ni procesos internos al cliente.
  5. **Comunicación con otros servicios**
     - APIs, envío de correos automáticos, integración con sistemas externos.

LENGUAJES SERVIDOR en 2025 más usados

Lenguaje	Características	Usos comunes	Mini-ejemplo servidor

JavaScript (Node.js)	Asincronía, mismo lenguaje en frontend y backend.	APIs, apps en tiempo real (chats, juegos online).	const http = require('http');
			http.createServer((req,res)=>{
			res.end('Hola desde Node.js');
			}).listen(3000);

Python (Flask / FastAPI / Django)	Sencillo de aprender, muy usado en IA y ciencia de datos.	APIs, apps web, integraciones con ML.	from flask import Flask app = Flask(__name__) @app.route('/') def home(): return \"Hola desde Flask\“ app.run()

Java (Spring Boot)	Robusto, multiplataforma, muy usado en empresas.	Bancos, seguros, grandes sistemas.	import org.springframework.web.bind.annotation.*;
			@RestController
			class Hola {
			@GetMapping(\"/\")
			public String home() {
			return \"Hola desde Spring\";
			}
			}

PHP (Laravel / WordPress)	Fácil de desplegar, base enorme en la web.	Blogs, e-commerce, webs dinámicas.	<?php echo \"Hola desde PHP\"; ?>

Go (Golang)	Muy rápido y concurrente, ideal para microservicios.	Sistemas distribuidos, APIs de alto rendimiento.	package main
			import (\"fmt\"; \"net/http\")
			func main(){
			http.HandleFunc(\"/\", func(w http.ResponseWriter, r *http.Request){
			fmt.Fprint(w, \"Hola desde Go\")
			})
			http.ListenAndServe(\":8080\", nil)
			}

C# (.NET)	Integra muy bien con Windows y Azure.	Empresas, aplicaciones de escritorio y web.	using Microsoft.AspNetCore.Builder;
			var app = WebApplication.Create();
			app.MapGet(\"/\", () => \"Hola desde .NET\");
			app.Run();

Kotlin (Ktor / Spring)	Moderno, menos verboso que Java.	Backend de apps móviles Android, APIs.	import io.ktor.application.*
			import io.ktor.response.*
			import io.ktor.routing.*
			fun Application.module(){
			routing {
			get(\"/\") { call.respondText(\"Hola desde Kotlin\") }
			}
			}