Unidad de Trabajo 1

Características de los lenguajes de marcas

RA1 - Reconoce las características de lenguajes de marcas analizando e interpretando fragmentos de código. ***(5% sobre nota final del módulo)***

¿Qué vamos a aprender?

01

Características generales

Identificaremos qué hace únicos a los lenguajes de marcas y cómo funcionan

02

Ventajas en el tratamiento de información

Descubriremos por qué son tan importantes para organizar datos

03

Clasificación y tipos

Aprenderemos a distinguir entre diferentes tipos de lenguajes de marcas

04

Ámbitos de aplicación

Exploraremos dónde y cómo se utilizan en el mundo real

05

Análisis práctico

Examinaremos ejemplos concretos de diferentes lenguajes

¿Cómo se va evaluar?

Criterios de evaluación

a) Se han identificado las características generales de los lenguajes de marcas.
b) Se han reconocido las ventajas que proporcionan en el tratamiento de la información.
c) Se han clasificado los lenguajes de marcas e identificado los más relevantes.
d) Se han diferenciado sus ámbitos de aplicación.
e) Se han reconocido la necesidad y los ámbitos específicos de aplicación de un lenguaje de marcas de propósito general.
f) Se han analizado las características propias de diferentes lenguajes de marcas.
g) Se ha identificado la estructura de un documento y sus reglas sintácticas.
h) Se ha contrastado la necesidad de crear documentos bien formados y la influencia en su procesamiento.
i) Se han identificado las ventajas que aportan los espacios de nombres.

Criterios de calificación

1) Actividad con ejercicios teórico/prácticos.
    a) Parejas o individual.
    b) Calificación 0-10.
    c) 40% nota RA.

2) Prueba tipo test escrito.
    a) Individual
    b) Calificación 0-10.
    c) 60% nota RA.
    d) Nota >=5 para superar RA.

Los Orígenes de los Lenguajes de Marcas

Del papel al mundo digital

Los lenguajes de marcas nacieron de una necesidad muy humana: **organizar y estructurar información**. Mucho antes de que existieran los ordenadores, los editores ya utilizaban marcas en manuscritos para indicar cómo debía presentarse el texto final.

En los años 1960, IBM desarrolló GML (Generalized Markup Language), el precursor de todos los lenguajes de marcas modernos. La idea era simple pero revolucionaria: separar el contenido de su presentación visual.

Esta evolución nos llevó a **SGML** (**Standard Generalized Markup Language**) en los años 80, y finalmente a la explosión de lenguajes que conocemos hoy: HTML, XML, y otros que veremos a lo largo del curso.

¿Qué es exactamente un Lenguaje de Marcas?

Un lenguaje de marcas es un sistema que utiliza **etiquetas** para definir la estructura, el significado y el formato de la información en un documento.

Separación de contenido y forma

El contenido (texto, datos) se separa de cómo debe presentarse visualmente

Estructura jerárquica

Los elementos se organizan en una estructura de árbol con relaciones padre-hijo

Legibilidad humana

Aunque los procesen las máquinas, las personas pueden leer y entender el código

Elementos Fundamentales: Etiquetas, Elementos y Atributos

🏷️ Las Etiquetas

Son las marcas que delimitan el contenido. Se escriben entre corchetes angulares: `<etiqueta>`

```
<titulo>Mi primer documento</titulo><parrafo>Este es un ejemplo</parrafo>
```

📦 Los Elementos

Un elemento incluye la etiqueta de apertura, el contenido y la etiqueta de cierre. Es la unidad básica de información.

```
<h1>Título Principal</h1><p>Contenido del párrafo</p>
```

⚙️ Los Atributos

Proporcionan información adicional sobre los elementos. Van dentro de la etiqueta de apertura.

```
<img src="foto.jpg" alt="Mi foto">
<a href="https://ejemplo.com">Enlace</a>
```

Características Comunes de los Lenguajes de Marcas

Estructura jerárquica

Los elementos se organizan en una estructura de árbol donde cada elemento puede contener otros elementos. Esta jerarquía permite representar relaciones complejas de información de manera ordenada y lógica, facilitando tanto el procesamiento automático como la comprensión humana.

Formato de texto plano

Se escriben en texto simple, sin formato binario, lo que los hace legibles por humanos y editables con cualquier editor de texto. Esta característica facilita la depuración, el mantenimiento y la colaboración entre desarrolladores.

Extensibilidad

Muchos lenguajes de marcas permiten definir nuevos elementos y atributos según las necesidades específicas del proyecto. Esta flexibilidad los convierte en herramientas poderosas para dominios especializados que requieren vocabularios particulares.

Sintaxis bien definida

Cada lenguaje tiene reglas estrictas sobre cómo escribir las etiquetas, qué elementos pueden contener otros elementos, y cómo se estructuran los atributos. Esta consistencia sintáctica es fundamental para que los procesadores puedan interpretar correctamente el documento.

Independencia de plataforma

Funcionan en cualquier sistema operativo y pueden ser procesados por diferentes aplicaciones. Esta universalidad es clave para la interoperabilidad en entornos heterogéneos y para el intercambio de información entre sistemas diversos.

Validación automática

Pueden verificarse automáticamente contra esquemas o definiciones que establecen qué elementos son válidos, sus atributos permitidos y la estructura correcta. Esta capacidad de validación es esencial para mantener la calidad y consistencia de los documentos.

Ventajas de los Lenguajes de Marcas

Los lenguajes de marcas ofrecen ventajas significativas que han revolucionado el tratamiento de la información digital:

Separación de contenido y presentación

Permite modificar el aspecto visual sin alterar el contenido, facilitando el mantenimiento y la reutilización. Un mismo documento puede tener múltiples presentaciones según el contexto: web, impresión, móvil, etc.

Intercambio de datos estructurados

Facilitan el intercambio de información entre sistemas diferentes, actuando como un lenguaje común. Esta interoperabilidad es fundamental en aplicaciones distribuidas y servicios web.

Búsqueda y procesamiento eficiente

La estructura definida permite búsquedas precisas y procesamientos automáticos. Los motores de búsqueda pueden entender mejor el contenido y los programas pueden extraer información específica fácilmente.

Clasificación de los Lenguajes de Marcas

Existen diferentes formas de clasificar los lenguajes de marcas según diversos criterios

Clasificación por Propósito

Presentación Web

HTML, XHTML

Diseñados específicamente para crear páginas web y definir su estructura y presentación en navegadores

Intercambio de Datos

XML

Enfocados en el intercambio estructurado de información entre sistemas y aplicaciones diferentes

Configuración

YAML, TOML

Utilizados para definir configuraciones de aplicaciones de manera legible y estructurada

Documentación

Markdown, LaTeX

Optimizados para crear documentos con formato, desde simples textos hasta publicaciones académicas complejas

Clasificación por Flexibilidad

Lenguajes Fijos

Tienen un conjunto **predefinido y limitado** de elementos y atributos:

- **HTML**: Vocabulario específico para páginas web
- **Latex:** Composición de documentos científicos, con comandos predefinidos.
- **RTF**: Desarrollado por Microsoft para representar documentos con formato (fuentes, colores, estilos, etc.).

Son más fáciles de aprender pero menos adaptables a necesidades específicas.

Lenguajes Extensibles

Permiten **definir elementos personalizados** según las necesidades:

- **XML**: Totalmente personalizable
- **SGML**: Base para crear otros lenguajes
- **Custom HTML**: Con web components

Ofrecen máxima flexibilidad pero requieren mayor planificación y conocimiento técnico.

Lenguajes de Marcas más Relevantes

HTML (HyperText Markup Language)

El lenguaje fundamental de la web. Define la estructura y el contenido de las páginas web. HTML5 es la versión actual, que incluye elementos semánticos y soporte multimedia avanzado. Es la base de toda aplicación web moderna.

XML (eXtensible Markup Language)

Lenguaje de propósito general para el intercambio de datos estructurados. Muy utilizado en servicios web, configuraciones y almacenamiento de datos. Su flexibilidad lo convierte en la base de muchos otros lenguajes especializados.

CSS (Cascading Style Sheets)

Aunque técnicamente es un lenguaje de hojas de estilo, utiliza sintaxis de marcas para definir la presentación visual de documentos HTML. Es fundamental para el diseño web responsive y la experiencia de usuario.

Ámbitos de Aplicación en el Desarrollo

Desarrollo Web Frontend

HTML para estructura, CSS para diseño, y XML para datos. La base de toda interfaz de usuario web, desde sitios simples hasta aplicaciones web complejas (SPAs).

APIs y Servicios Web

XML y JSON para intercambio de datos entre aplicaciones. Protocolo REST, SOAP, GraphQL - todos dependen de lenguajes de marcas para estructurar las comunicaciones.

Desarrollo de Apps Móviles

XML para interfaces en Android, plist en iOS, y React Native JSX. Los lenguajes de marcas definen tanto la UI como la configuración de aplicaciones multiplataforma.

Configuración y Despliegue

YAML para Docker Compose, XML para Maven/Gradle, JSON para package managers. Fundamentales en DevOps y automatización de despliegues.

XML Universal Language Integration Systems

¿Por qué necesitamos un lenguaje de propósito general?

El problema de los lenguajes específicos

Cuando cada aplicación o dominio tiene su propio lenguaje de marcas, surgen problemas de incompatibilidad:

- Dificultad para integrar sistemas
- Costes de aprendizaje multiplicados
- Herramientas específicas para cada lenguaje
- Barreras para el intercambio de datos

La solución: XML como estándar

XML se diseñó como meta-lenguaje que permite:

- Crear vocabularios específicos manteniendo la sintaxis común
- Reutilizar herramientas y conocimientos
- Facilitar la transformación entre formatos
- Garantizar la interoperabilidad

Ámbitos Específicos de XML

01

Intercambio de datos empresariales

EDI (Electronic Data Interchange), facturas electrónicas, órdenes de compra. XML permite que empresas con sistemas diferentes intercambien información comercial de forma automática y fiable.

02

Servicios web y APIs

SOAP, REST, microservicios. XML estructura las peticiones y respuestas entre aplicaciones distribuidas, facilitando la arquitectura de sistemas complejos y escalables.

03

Configuración de aplicaciones

Archivos de configuración de frameworks como Spring, Hibernate, Maven. Permite definir comportamientos complejos sin recompilar código, mejorando la flexibilidad del desarrollo.

04

Almacenamiento y transformación de datos

Bases de datos XML nativas, ETL processes, XSLT transformations. Facilita el procesamiento y migración de información entre sistemas heterogéneos.

Documentos XML Bien Formados

Un documento XML bien formado es aquel que respeta todas las reglas sintácticas de XML. Esta conformidad es esencial para que los procesadores puedan analizar el documento correctamente.

Verificación Sintáctica

El analizador XML verifica que se cumplan todas las reglas: etiquetas balanceadas, atributos entre comillas, caracteres especiales correctos.

1

2

Estructura Jerárquica

Comprueba que existe un único elemento raíz y que todos los elementos están correctamente anidados sin solapamientos.

3

Codificación de Caracteres

Verifica que la codificación declarada coincide con la codificación real del archivo, generalmente UTF-8.

❌ Documento MAL Formado

```
<libros>
      <libro id=001>
            <titulo>El Quijote
            <autor>Cervantes</AUTOR>
      </Libro>
      <libro>
            <titulo>La Celestina</titulo>
      </libro>
</libros>
```

Errores: Atributo sin comillas, etiqueta sin cerrar, diferente capitalización.

✅ Documento BIEN Formado

```
<libros>
      <libro id="001">
            <titulo>El Quijote</titulo>
            <autor>Cervantes</autor>
      </libro>
      <libro>
            <titulo>La Celestina</titulo>
      </libro>
</libros>
```

Correcto: Atributos entre comillas, todas las etiquetas cerradas, capitalización consistente.

Herramientas y Validación XML

Para garantizar que nuestros documentos XML son correctos y útiles, necesitamos herramientas que nos ayuden a validarlos, depurarlos y procesarlos eficientemente.

Validadores Online

XMLValidation.com, W3C Validator: Herramientas web gratuitas que permiten validar documentos XML rápidamente. Ideales para pruebas rápidas y aprendizaje.

Herramientas de Navegador

Chrome DevTools, Firefox Developer: Los navegadores modernos incluyen analizadores XML que muestran errores de formato de manera visual e intuitiva.

Herramientas de Línea de Comandos

xmllint, xmlstarlet: Potentes utilidades para validar, consultar y transformar documentos XML desde la terminal. Perfectas para automatización.

Librerías de Programación

Java JAXP, Python lxml, C# XmlDocument: APIs que permiten procesar XML programáticamente con validación automática y manejo de errores.

Espacios de Nombres en XML

Los espacios de nombres (namespaces) resuelven uno de los problemas fundamentales de XML: ¿qué hacer cuando diferentes vocabularios XML usan el mismo nombre de elemento pero con significados diferentes?

El Problema

Imagina que tienes un documento que mezcla información de libros y información HTML. Ambos podrían tener un elemento <title>, pero con significados completamente diferentes.

La Solución

Los espacios de nombres permiten cualificar elementos con un identificador único (URI), evitando conflictos: <libro:title> vs <html:title>

Implementación

Se declaran usando el atributo xmlns y se aplican mediante prefijos: xmlns:libro="http://ejemplo.com/biblioteca"

```
<?xml version="1.0" encoding="UTF-8"?>
<documento xmlns:libro="http://biblioteca.com/ns" xmlns:html="http://www.w3.org/1999/xhtml">
     <libro:catalogo>
          <libro:title>El Quijote</libro:title>
          <libro:autor>Cervantes</libro:autor>
     </libro:catalogo>
     <html:div>
          <html:title>Página de Biblioteca</html:title>
          <html:p>Bienvenido a nuestra biblioteca</html:p>
     </html:div>
</documento>
```

Ventajas de los Espacios de Nombres

Los espacios de nombres no son solo una característica técnica; son una herramienta fundamental que aporta beneficios significativos en el desarrollo y mantenimiento de aplicaciones XML complejas.

Modularidad y Reutilización

Permite combinar vocabularios XML diferentes en un mismo documento sin conflictos. Por ejemplo, puedes mezclar SVG para gráficos, MathML para ecuaciones y XHTML para texto en un único documento.

Mantenimiento Simplificado

Los cambios en un vocabulario específico no afectan a otros vocabularios en el mismo documento. Cada espacio de nombres evoluciona independientemente.

Interoperabilidad Mejorada

Facilita el intercambio de documentos entre diferentes sistemas y organizaciones, ya que cada parte puede procesar solo los espacios de nombres que entiende.

Versionado Controlado

Diferentes versiones de un vocabulario pueden coexistir usando URIs de espacio de nombres distintos, permitiendo transiciones graduales entre versiones.

Ejemplo práctico: En el desarrollo Android, el archivo layout.xml utiliza múltiples espacios de nombres: android: para atributos específicos de Android, app: para atributos de la aplicación, y tools: para herramientas de desarrollo.

Análisis de HTML: Características Propias

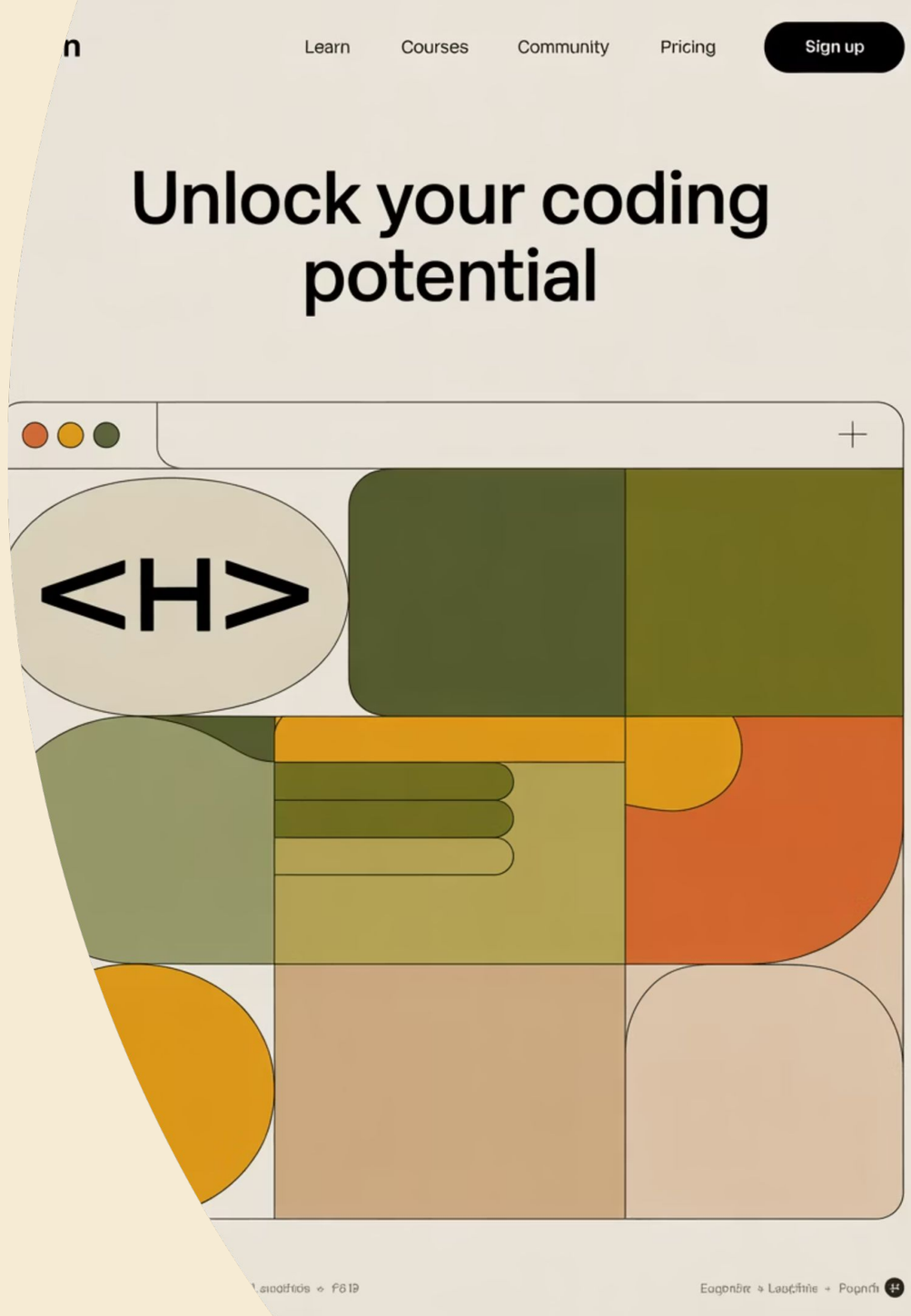
Características únicas de HTML

HTML es más que un lenguaje de marcas; es el lenguaje fundacional de la web:

Elementos semánticos: <article>, <section>, <nav> dan significado al contenido
Tolerancia a errores: Los navegadores corrigen automáticamente muchos errores de sintaxis

Integración multimedia: Soporte nativo para video, audio, gráficos (canvas, SVG)
Formularios interactivos: Elementos para entrada de datos con validación automática
Accesibilidad integrada: Atributos como 'alt'

```
<article> <header>
<h1>Mi artículo</h1>
<time
datetime="2024">2024</
time> </header>
<section>
<p>Contenido...</p>
<img src="imagen.jpg"
alt="Descripción">
</section></article>
```



Análisis de XML: El Meta-lenguaje Universal

—

Sintaxis estricta

A diferencia de HTML, XML requiere que todos los elementos estén bien formados: etiquetas cerradas correctamente, atributos entre comillas, case-sensitive. Esta rigidez garantiza procesamiento confiable.

Vocabulario personalizable

Puedes definir tus propios elementos y atributos según las necesidades del dominio. No hay elementos predefinidos, todo el vocabulario lo define el desarrollador o el estándar específico.

Espacios de nombres

Los namespaces permiten combinar vocabularios de diferentes orígenes sin conflictos. Esencial para la interoperabilidad en sistemas complejos con múltiples estándares.

Validación avanzada

Soporte para XSD (XML Schema), DTD y otras tecnologías de validación que permiten definir reglas complejas sobre la estructura y contenido permitido en los documentos.

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca xmlns:lib="http://ejemplo.com/biblioteca">
     <lib:libro id="001">
     <lib:titulo>Lenguajes de Marcas</lib:titulo>
     <lib:autor>Juan Pérez</lib:autor>
     </lib:libro>
</biblioteca>
```

Análisis de CSS: Más que Presentación

Características distintivas

Aunque CSS no es un lenguaje de marcas en sentido estricto, utiliza sintaxis similar y conceptos fundamentales:

Selectores: Patrones para identificar elementos HTML
Cascada: Reglas de precedencia para resolver conflictos. Cuando múltiples reglas de estilo se aplican al mismo elemento, el navegador utiliza la cascada como un algoritmo para decidir cuál regla CSS es la más importante y se aplica finalmente.
Herencia: Propiedades que se transmiten a elementos hijos
Responsive design: Permite aplicar estilos de manera condicional, basándose en las características del dispositivo o del navegador, como el ancho, alto o resolución de la pantalla.

```
/* Selector de clase */
.destacado {  color: #626C3B;  font-weight: bold;}
/* Media query responsive */
@media (max-width: 768px) {  .contenedor {    flex-direction:
column;  }}
```

La separación entre HTML (contenido) y CSS (presentación) ejemplifica perfectamente los **principios fundamentales** de los lenguajes de marcas modernos.

Análisis de JSON: Simplicidad y Eficiencia

Sintaxis minimalista

JSON utiliza una sintaxis más simple que XML, basada en la notación de objetos de JavaScript. Menos caracteres significa **menor tamaño** de archivo y **parsing más rápido**.

Tipos de datos nativos

Soporte directo para strings, numbers, booleans, arrays y objects. No necesita conversiones adicionales en la mayoría de lenguajes de programación modernos.

Ideal para APIs REST

Se ha convertido en el estándar de facto para el intercambio de datos en servicios web modernos. Su ligereza lo hace perfecto para aplicaciones móviles y IoT.

Ejemplo JSON:

```
{
"alumno":
     {
          "nombre": "María García",
          "edad": 20,
          "asignaturas": ["HTML", "CSS", "JavaScript"],
          "activo": true
     }
}
```

Equivalente XML:

```
<alumno>
     <nombre>María García</nombre>
     <edad>20</edad>
     <asignaturas>
          <asignatura>HTML</asignatura>
          <asignatura>CSS</asignatura>
          <asignatura>JavaScript</asignatura>
     </asignaturas>
     <activo>true</activo>
</alumno>
```