***LENGUAJE DE MARCAS- UD1***

***→1.1-***

Un lenguaje de marcas o lenguaje de marcado es un lenguaje que mediante marcas incorporadas al texto de un documento permite informar de la estructura del propio texto, su significado o su presentación visual.

Dentro de los lenguajes de marcas existen:

* ***Lenguajes de marcas ligeros***
* ***Lenguajes de serialización de datos***

En el marcado se pueden distinguir cinco tipos distintos:

* ***Marcado de puntuación:***  
  Es el sistema de marcas que proporciona, fundamentalmente, información sintáctica sobre el texto.
* ***Marcado de presentación:***  
  Es el marcado que se realiza sobre entidades de nivel superior para que la presentación sea más clara
* ***Marcado de procedimiento:***  
  Es el marcado que expresa las órdenes para formatear el texto
* ***Marcado descriptivo:***  
  El marcado descriptivo identifica qué significa cada elemento de texto, pero sin expresar cómo ha de procesarse
* ***Marcado referencial:***  
  Son marcas que refieren a entidades externas y que durante el procesamiento deben ser reemplazadas por estas.
* ***Metamarcado:***Es marcado que permiten controlar la interpretación del propio marcado o ampliar el vocabulario con nuevas marcas.

Atendiendo a la estrategia que siga un lenguaje al marcar, podemos distinguir:

* ***Marcado de presentación:*** Es aquel en que se informa del formato del texto. Se usan para maquetar el texto
* ***Marcado procedimental:*** Son aquellas marcas que proveen instrucciones para que el sistema informático procese el texto, por lo general, para su presentación.
* ***Marcado descriptivo (o semántico):*** Son marcas que definen qué significa cada fragmento del texto, pero por lo general, sin proporcionar información de cómo debe ser procesado.
* ***Marcado descriptivo orientado al documento:*** Las marcas describen cuál es la función semántica de cada parte del documento. Hay dos maneras de obtener, posteriormente, un texto cómodamente legible:  
  + 1- Definir cómo debe representarse cada componente de la estructura del texto mediante hojas de estilo
  + 2- Transformarlo en otro documento que sí tenga representación. Para obrar la transformación es preciso utilizar un procesador.
* ***Marcado descriptivo orientado al dato:*** En archivos que contienen datos, las marcas identifican el significado de cada dato y los estructuran.

Como iremos descubriendo más adelante, todos estos lenguajes no son lenguajes independientes, sino que por lo general existen definidas unas reglas generales que deben cumplir todos los lenguajes basados en un determinado tipo; y, luego, cada lenguaje en particular tiene un vocabulario propio que deben recoger todos los documentos que se escriban con él.

* + Se dice que un documento está bien formado cuando tal documento cumple con las reglas generales del tipo al que pertenece.
  + Se dice que un documento es válido cuando, además de bien formado, cumple con las reglas gramaticales particulares que se han definido para el lenguaje.

Los lenguajes que más deben atraer nuestro interés:

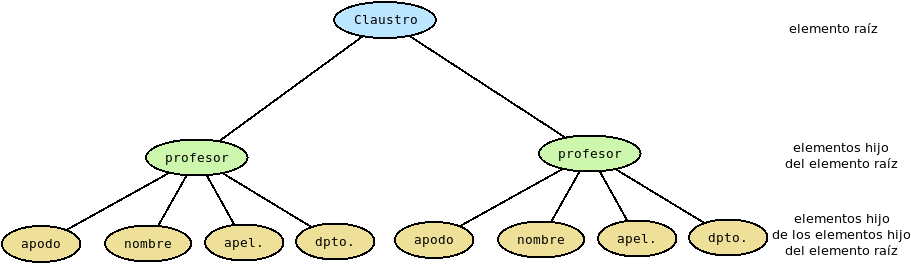
* ***Lenguajes de serialización de datos:***
  + CSV
  + TOML
  + JSON
  + YAML
* ***Lenguajes de marcas:*** Hay dos tipos:
  + ***Procedimentales:***
    - **TeX**
    - **RTF**
    - **Postscript**
    - **PDF**
    - **troff y nroff**
  + ***Descriptivos:***
    - **reStructuredText**
    - **Markdown**
    - **Wiki Text**
    - **LaTeX, AMS-TeX o ConTeXt**
    - **SGML, XML y derivados**

***→1.2-***

Trataremos la sintaxis de tres lenguajes: CSV, JSON y YAML.

* ***CSV:*** Es un formato centrado en la representación de datos en columnas y está descrito en el RFC 4180. Es muy simple y sus principales reglas son las siguientes:  
  1. Es un formato de texto, aunque no hay forma de definir el sistema de codificación.
  2. Los datos están estructurados en registros, cada uno de los cuales ocupa una línea.
  3. Cada registro, a su vez, se divide en campos separados entre sí por un delimitador.
  4. Todos los registros deben contener el mismo número de campos.
  5. Opcionalmente, el primer registro, puede ser un registro de cabecera que contenga los nombres de los campos.
  6. Los campos, en general, no necesitan entrecomillarse (con comillas dobles) a menos que contenga un carácter que distorsione la lectura
  7. En un campo entrecomillado, si el texto contiene un carácter de comilla doble este debe representarse con dos caracteres seguidos de comilla doble.
* ***JSON:*** Este lenguaje nació como forma de representar objetos en Javascript, por lo que su aspecto guarda bastante parecido con ellos. Muy probablemente por esta filiación con el lenguaje de programación se usa habitualmente en la transmisión de datos entre servidor y cliente web, tarea en la que ha desplazado casi por completo a XML.  
  1. ***Validadores***:
     + [Validador JSON online](https://codebeautify.org/jsonvalidator).
     + P[ython3-demjson](https://packages.debian.org/stable/python3-demjson)
     + [Visual Studio Code](https://code.visualstudio.com/)
  2. ***Estructura***:
     + Escalar, que es un dato constituido por un único valor.
     + Secuencia (o array), que es un dato constituido por una colección ordenada de nodos.
     + Mapa (u objeto), que es una colección ordenada de parejas clave-valor.
  3. ***Reglas básicas:***
     + Un archivo o flujo de datos sólo puede contener un documento JSON.
     + Cada documento JSON esta constituido por un único nodo mapa o un único nodo secuencia (al que, por supuesto, pueden estar anidados más nodos).
     + No pueden escribirse comentarios.
     + No hay reglas estrictas para el uso del espaciado.
     + Por carácter de espaciado se entiende el propio espacio, el cambio de línea (\n), el retorno de carro (\r) o la tabulación.
  4. ***Tipos de nodos:***
     + Cadena: Es un escalar que debe estar encerrado entre comillas dobles (las simples no valen).
     + Número: Es un escalar que representa:
       - Enteros
       - Coma flotante, que usan como separador el punto.
       - Coma flotante en notación científica
     + Lógico (o booleano): Como en Javascript este tipo escalar se expresa con true (verdadero) o false.
     + Nulo: Este tipo escalar sólo tiene un posible valor que es null y representa la ausencia de valor.
     + Array: Un array es una colección ordenada de datos de cualquiera de los tipos posibles.
     + Objeto: Un objeto es una colección desordenada de parejas clave-valor
* ***YAML:*** YAML es otro lenguaje de serialización que, como JSON, construye su estructura de datos basándose en el concepto de nodo (escalar, de secuencia o de mapa), lo que permite que en principio puedan compartir nichos de uso  
  1. ***Validadores:*** 
     + YAMLLint online
     + Linux yamllint
     + Visual Studio Code
  2. ***Estructura:*** 
     + El contenido del documento es el único indispensable.
     + La marca inicial ---, que señala el inicio del contenido.
     + La marca final ... (tres puntos), que señala explícitamente el final del contenido y es opcional.
     + Las directivas que afectan al documento.
  3. ***Contenido:*** El contenido está constituido por un único nodo, como en el caso de JSON, pero a diferencia de éste, también es válido un nodo escalar.
  4. ***Esquema:*** Un esquema en YAML es un conjunto de etiquetas y un mecanismo para resolverlas. Los procesadores deben implementar un esquema, aunque la especificación define tres:
     + **Failsafe schema (o sea esquema a prueba de fallos)**: Es un esquema que sólo reconoce tres tipos: secuencias, mapas y cadenas.
     + **JSON schema**: Es un esquema que implementa al menos los tipos existentes en JSON. Por tanto, añade a los anteriores los tipos null, entero, flotante y lógico.
     + **Core schema (o sea, esquema básico)**: Es un esquema que no añade tipos al anterior, sino que, simplemente, amplía el modo en que se pueden escribir algunos tipos de datos.
  5. ***Tipos:***
     + **Enteros (int)**: Pueden escribirse:
     + Decimales cómo -123.
     + Octales como 0o10
     + Hexadecimales como 0x1a.
     + **Numeros en coma flotante (float)**: Tiene también varias expresiones:  
       - La habitual con punto separador
       - En notación científica
       - Valores infinitos: .inf y -.inf.
       - No es un número: .nan.
     + **Lógicos o booleanos** (bool): Como en Javascript (y JSON) se representan con true y false.
     + **Nulo (null)**: Es la ausencia de valor y se escribe null.
     + **Cadenas (str)**: Las cadenas. a diferencia de JSON, no necesitan escribirse entrecomilladas
     + **Array (seq)**: Las colecciones ordenadas (reléase lo expuesto para los arrays en JSON) se notan con guiones
     + **Mapa u objeto (map)**: Las colecciones ordenadas de parejas clave-valor se notan no colocando guiones y utilizando como separador la secuencia de dos caracteres
     + **Conjunto**: Es una secuencia desordenada, que se nota usando el carácter de interrogación (?) en vez del guión
     + **Binario**: Aunque YAML es una representación de datos en texto plano, este tipo de dato permite incluir valores binarios.
     + **Fecha (date)**: El tipo permite definir fechas de calendario.
     + **Marcas de tiempo (timestamp)**: Permite definir un instante de tiempo concreto en distintos formatos.
  6. ***Referencias:*** Las referencias (o anclas) permiten referir el valor de un nodo dentro de otro.

Entre los lenguajes de marcas explicaremos solo: HTML y la dupla SGML/XML

* ***SGML:*** SGML, pues, no es un lenguaje concreto, sino un conjunto de normas para crear lenguajes de marcas descriptivos. Dicho de otra forma, es un metalenguaje. En consecuencia, todo documento SGML debe ir acompañado de un documento que defina sobre sus marcas:
  + Cómo deben ser (en principio, tienen la forma <marca>, pero puede definirse otra)
  + Cuáles son las marcas posibles y qué atributos pueden tener.
  + Si pueden omitirse.
  + Cuáles son las reglas para utilizar estas marcas dentro del texto.  
    - ***Características*:** 
      * Los documentos deben ser de texto plano.
      * El marcado debe ser descriptivo y absolutamente independiente de cómo vaya a ser procesado.
      * El marcado debe ser riguroso e inequívoco a fin de permitir su procesamiento posterior.
    - ***Derivados:***
      * DocBook
      * TEI
      * HTML
      * XML
* ***XML:***Es una simplificación de SGML que elimina muchas de sus características más complejas, con el objeto de facilitar la creación de procesadores capaces de tratar los documentos escritos en alguno de sus dialectos.XML se aceptó muy rápidamente. Las simplificaciones son:
  + Toda etiqueta debe ser expresamente abierta y cerrada.
  + Los atributos de las marcas siempre requieren un valor y que éste se entrecomille.
  + Se define que las marcas sólo pueden expresarse del modo predeterminado.
  + Se define también una única notación para definir entidades.
  + Los nombres son sensibles al uso de mayúsculas y minúsculas.  
      
      
      
    - ***Lenguajes:***
      * ***XHTML***
      * ***XSLT***
      * ***RSS***
      * ***MathML***
      * ***SVG***
      * ***OpenDocument***
    - ***Aplicaciones:***
      * La creación de documentos semánticos estructurados
      * Como formato para el intercambio de datos.
      * Para el almacenamiento de cantidades modestas de datos estructurados.
    - ***Componentes:***
      * **Etiqueta (o marca):** Declara cuál es la función o el significado de una parte concreta del contenido.
      * **Texto:** Es la información en sí del documento, desprovista de cualquier marca.
      * **Elemento**: Toda parte del contenido que constituye una parte diferencia del resto.
      * **Atributo**: Pares nombre/valor que sirven para caracterizar al elemento.
      * **Instrucción de procesamiento**: Instruyen al procesador y no forman parte de la información del documento.
      * **Comentario**: Permiten hacer alguna anotación al marcado. Se abren con la secuencia <!-- y se cierran con -->.
      * **Entidad**: Sirven para referenciar bien un carácter (que no puede escribirse directamente), bien un conjunto de carácteres a los que se ha preferido dar nombre.
      * **Nodo**: Término genérico pasa designar al resto de componentes.
    - ***Partes:*** 
      * **Preámbulo**: Se encuentran en él, habitualmente, algunas instrucciones de procesamiento y la declaración del tipo de documento (equivalente a la que se hace en SGML)
      * **Contenido**: Es la información que contiene propiamente el documento junto a las marcas que lo caracterizan descriptiva o procedimentalmente.
    - ***Estructura Jerárquica:***
    - ***Reglas Generales:***
      * **Etiquetas**:  
        + Tienen la forma <etiqueta> en su apertura y </etiqueta> en su cierre.
        + En caso de que la marca abarque un elemento vacío, se puede usar la notación: <vacío />
        + Toda etiqueta de apertura debe tener su correspondiente de cierre explícita.
        + El nombre de la etiqueta debe ser una sola palabra.
        + Los nombres de etiquetas que empiecen por xml se reservan para un uso específico.
        + XML es sensible a mayúsculas y minúsculas (esta regla, en realidad, es aplicable a cualquier componente).
      * **Elementos**:  
        + Todo el contenido del documento debe estar incluido dentro de un único elemento raíz.
        + Los elementos deben estar correctamente anidados dentro de otros, lo cual implica que el último elemento que se abrió deba ser el primero que se cierre.
        + Estas dos reglas determinan que los elementos se organicen en una [estructura jerárquica](https://sio2sio2.github.io/LM/01.intro/02b.marcas.html#xml-jerarq).
      * **Atributos**:  
        + Tienen la forma nombre="valor".
        + Pueden obviarse atributos bien cuando tienen valor predeterminado, bien cuando es lícito que su valor quede indefinido.
        + En caso de que un atributo aparezca, debe forzosamente tener valor.
        + Si un elemento se caracteriza con varios atributos, estos pueden escribirse en cualquier orden.
        + Para un mismo elemento no puede repetirse el atributo.
      * **Entidades**:
        + Se representan con un símbolo «&» antes del nombre de la entidad y un «;» después.
        + Las entidades pueden ser de dos tipos:
        + Predefinidas
        + Definidas por el usuario
    - ***Espacios de nombres:*** Un espacio de nombres es un contenedor de nombres dentro del cual cada nombre es único. Es un concepto común a muchas ramas de la programación y también se adaptó a XML

[Página de Jose Miguel](https://sio2sio2.github.io/LM/index.html)