UNIDAD 4

HOJAS DE ESTILO ESQUEMAS Y VOCABULARIOS XML

CSS/XML

Reglas de estilo o Unidades CSS

Un fichero CSS contiene una serie de **reglas de estilo** o, también llamadas, **unidades css**. Dichas unidades css se componen de un **selector** (normalmente el nombre de una etiqueta de nuestro documento XML) y una **lista o cadena de estilos**. Dicha lista consta de un conjunto de pares de elementos separados por el caracter ***;*** donde cada par, está formado por el nombre de una propiedad y un valor, separándose con el caracter ***:***

Sintaxis:

Selector {propiedad:valor; propiedad:valor; ... propiedad:valor}

Selectores

El selector más utilizado es simplemente el nombre de la etiqueta a la que se refiere la regla.

Ejemplo:

ejemplo {display: block; color: blue}

La agrupación de elementos se realiza mediante una lista de los nombres de las reglas separadas por comas. A esta práctica se le denomina ***Agrupación de selectores***.

Ejemplo documento xml con agrupación de selectores:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo1.css" type="text/css"?>
<TEXTOS>
        <POEMA>
           <VERSO>Por el molino del huerto </VERSO>
           <VERSO> Asciende una enredadera</VERSO>
           <VERSO>..... </VERSO>
        </POEMA>
        <AUTORA>Juana de Ibarbourou</AUTORA>

</TEXTOS>
```

Ejemplo documento CSS con agrupación de selectores:

POEMA,AUTORA{display: block; color: blue}

Existe un tipo de elementos especiales denominados **pseudo-elementos** que hacen referencia a dos elementos especiales que algunas veces necesitan estilos específicos como son los elementos *first-letter* y *first-line*. La forma de referenciarlos es : ELEMENTO:pseudo-elemento Lista_de_Estilos

Ejemplo documento css con pseudo-elementos:

```
POEMA {display: block; color: blue}
AUTORA {display: block; color: red}
AUTORA:first-letter {color: green}
```

Para los casos en que el mismo elemento necesite formatearse de diferente forma, se utiliza CLASS. La forma de utilizar CLASS es la siguiente. En el fichero CSS se especifica la regla general de formateo y la regla para los elementos especiales y en el fichero XML se identifican esos elementos con el atributo CLASS.

Ejemplo de documento XML,usando CLASS:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo3.css" type="text/css"?>
<POEMA>
      <VERSO CLASS="PRIMERO" > Por el molino del huerto</VERSO>
      <VERSO> Asciende una enredadera</VERSO>
      <VERSO CLASS="PRIMERO" >... </VERSO>
</POEMA>
```

Ejemplo de documento CSS, usando CLASS:

```
POEMA { display: block }
VERSO.PRIMERO{ color: blue }
VERSO{ display: block; color: black}
```

Nota: Esta opción no está disponible todavía en el navegador Mozilla, aunque sí funciona en el IE5. Esa práctica no está muy recomendada puesto que siempre es posible evitarla utilizando una etiqueta distinta para estos elementos.

En otras ocasiones es posible que se desee seleccionar un único elemento para asignarle un estilo especial, en este caso es posible seleccionarlo mediante el atributo ID. Para este caso la forma de utilizar el ID es la siguiente

Ejemplo de documento XML,usando ID:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo4.css" type="text/css"?>
<POEMA>
      <VERSO ID="PRIMERO" > Por el molino del huerto</VERSO>
      <VERSO> Asciende una enredadera</VERSO>
      <VERSO>... </VERSO>
</POEMA>
```

Ejemplo de documento CSS, usando ID:

```
POEMA { display: block }
VERSO#PRIMERO{ color: blue }
VERSO{ display: block; color: black}
```

Nota: Esta opción no está disponible todavía en el navegador Mozilla, aunque sí funciona en el IE5. Esa práctica no está muy recomendada puesto que siempre es posible evitarla utilizando una etiqueta distinta para estos elementos.

La última forma de seleccionar un elemento es mediante **contexto**. Es decir se desea aplicar estilo a algunos elementos dependiendo del contexto en el que se encuentren, para esta práctica se procede:

Ejemplo de XML, usando contexto:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo5.css" type="text/css"?>
<POEMA>
   <PRIMERO>
      <VERSO> Por el molino del huerto</VERSO>
    </PRIMERO>
```

```
    <VERSO> Asciende una enredadera</VERSO>
    <VERSO>... </VERSO>
</POEMA>
```

Ejemplo de CSS, usando contexto:

```
POEMA        {display:block;}
VERSO        {color: black}
PRIMERO      {display:block}
PRIMERO VERSO {color: blue}
```

Comentarios

Los comentarios en las hojas de estilo son similares a los comentarios de C. /*... */.

Ejemplo de CSS con comentarios:

```
POEMA{display:block;}  /* Estilos generales para un poema*/
VERSO{color: black; }  /* Estilos específicos para el verso*/
```

Valores

Los valores de las propiedades pueden ser de cuatro tipos:

- Longitud: Cuando estamos trabajando con el tamaño de algo, por ejemplo, de las fuentes
- URL: Cuando necesitamos un objeto externo, por ejemplo, una imagen
- Color
- Palabra clave: Hay propiedades que sólo pueden tomar determinados valores

Longitud

La longitud se utiliza para propiedades que expresan tamaño y se pueden utilizar tres tipos de longitud:

1. Longitud absoluta: Pueden ser:
    - Inch(pulgadas): in (1inch=2.53cm)
    - Centimetros: cm
    - Milimetros: mm
    - Puntos: pt (1pt=1/72inch)
    - Picas: pc (1pc=12pt)

    Ejemplo de CSS con longitudes absolutas:

    ```
    POEMA{ font-size: 1in } /* Ejemplo de in */
    VERSO{ margin: 1.5cm }  /* Ejemplo de cm */
    CITA { margin: 1000mm } /* Ejemplo de mm */
    ```

2. Longitud relativa: Pueden ser:
    - EME; em (tamaño de la fuente que se está usando)
    - Equis; ex (tamaño de la letra 'x' en la fuente que se está usando)
    - Pixel; px

    Ejemplo de CSS con longitudes relativas:

    ```
    POEMA{ font-size: 12px } /* Ejemplo de px */
    VERSO{ margin: 0.5em }   /* Ejemplo de em */
    CITA { margin: 1ex }     /* Ejemplo de ex */
    ```

3. Porcentajes: Se puede expresar un tamaño relativo como un porcentaje del valor actual de la propiedad.

   Ejemplo de CSS con porcentajes:

   POEMA{ font-size: 120% } /* Ejemplo de % */

URL

Este valor se puede aplicar a las propiedades que indican la imagen del fondo, y las imagenes de una lista, *background-image*, *list-style-image* y *list-style*. El formato para especificar una URL es mediante la funcion *url()*. Como argumento recibe una dirección relativa o absoluta que puede o no ir con comillas simples o dobles.

Ejemplo de XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo8.css" type="text/css"?>
<POEMA>
      <VERSO> Por el molino del huerto</VERSO>
      <VERSO> Asciende una <PLANTA>enredadera</PLANTA></VERSO>

      <VERSO>... </VERSO>
</POEMA>
```

Ejemplo de CSS usando url:

```
POEMA   {display: block; background-image: url(nombreImagen.ext) }
VERSO   {display: block; font-size: 20pt; font-family: sans-serif}
PLANTA  {font-style: italic; font-family: Helvetica }
```

Color

Existen muchas propiedades cuyo valor es un color y para indicar un color, CSS define cuatro formas diferentes, por nombre, por componentes hexadecimales, por componentes enteras y por porcentajes.

- Por nombre: Los valores son:

aqua	black	blue	fuchsia
gray	green	lime	maroon
navy	olive	purple	red
silver	teal	white	yellow

- Componentes Hexadecimales: Ejemplo : #FFFFFF, Blanco y el #000000, Negro.
- Componentes Enteras: Ejemplo : rgb(255,255,255), Blanco y el rgb(0,0,0), Negro.
- Porcentajes: Ejemplo : rgb(100%,100%,100%), Blanco y el rgb(0%,0%,0%), Negro.

Ejemplo de XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ejemplo8.css" type="text/css"?>
<POEMA>
      <VERSO> Por el molino del huerto</VERSO>
      <VERSO> Asciende una <PLANTA>enredadera</PLANTA></VERSO>

      <VERSO>... </VERSO>
</POEMA>
```

Ejemplo de CSS usando colores:

```
POEMA  {display: block; background-image: url(nombreImagen.ext) }
VERSO  {display: block; font-size: 20pt; font-family: sans-serif; color:rgb(10,50,40)}
PLANTA {font-style: italic; font-family: Helvetica; color:#33CCAA }
```

Palabras Clave

Estos son los valores más variables de los cuatro y serán introducidas detenidamente cuando se hable de cada una de las propiedades.

Propiedades

Display

La propiedad display determina como va a ser mostrado un elemento, qué espacio le va a ser asignado para su aparición en pantalla.

Existen dos posibles valores para esta propiedad: *block* y *none*.

- block: Estos elementos se separan unos de otros generalmente por un salto de línea.
- none: Esta propiedad será para los elementos invisibles, puesto que no se les asignará espacio para ser mostrados.

Propiedad white-space

Esta propiedad indica si los espacios, tabuladores o retornos de carro del fichero fuente son significativos o no. Los valores que puede tomar son *normal* y *pre*.

- normal: El funcionamiento es exactamente igual que en HTML y es la opción por defecto.
- pre: Actua como la etiqueta PRE en HTML.

Propiedades de Fuentes

Existen cinco propiedades para los tipos de letra que son:

- font-family: El valor que puede tomar es una lista separadas por comas de nombres de familias de fuentes que son (por ejemplo): *Serif*, *sans-serif*, *Monospace*, *Cursive* ,*Fantasy*, etc.. La lista proporciona simplemente un orden de preferencia a la hora de mostrar el texto. También es posible indicar el nombre concreto de la fuente detras del nombre de su familia. El valor de esta propiedad se hereda por los elementos anidados o hijos.
- font-style: Los valores disponibles son *normal*, *italic* y *oblique*(igual que italic).
- font-variant: Puede tomar dos valores *normal* y *small-caps*(todas mayúsculas y las mayúsculas más grandes) .
- font-weight: Los valores que puede tomar son *normal*, *bold*, *bolder*, *lighter*, *100* ... *900*.
- font-size: El tamaño puede ser especificado como un tamaño normal (14pt por ejemplo) o mediante una palabra clave.

  Con palabras clave:

  - xx-small
  - x-small
  - small
  - medium (opción por defecto)
  - large
  - x-large
  - xx-large

  Todas estas medidas son relativas a la fuente activa en la página.

Otra forma de expresar tamaño con palabras clave de forma relativa es teniendo en cuenta el tamaño de fuente del padre, esto se puede hacer expresando el tamaño mediante un porcentaje o

mediante *larger* o *smaller*

Existe una propiedad que engloba todas las propiedades relacionadas con la fuente denominada *font*. Esta propiedad toma como valor una lista de valores separados por espacios cuyo orden debe ser: estilo, variante y ancho de fuente que deben aparecer al principio en cualquier orden y además cualquiera de ellos puede ser omitido. El siguiente elemento debe ser el tamaño que no puede ser omitido y justo despues y de forma opcional puede aparecer el caracter */* y el ancho de la línea de escritura. El último valor es la familia de la fuente y es un valor que no se puede omitir.

Ejemplos de CSS para expresar características de una fuente.

```
POEMA {font: bold x-large Helvetica, sans-serif }
POEMA {font: italic smaller serif }
POEMA {font: italic 14pt/100pt serif }
```

Color

Esta propiedad se puede expresar mediante una palabra clave, números decimales, porcentajes o un número hexadecimal. Esta propiedad es heredada por los elementos hijos.

Propiedades background

Esta propiedad expresa el color del fondo y puede tomar valores de color o valores de la dirección de una imagen y engloba a varias propiedades individuales.

- background-color: Los valores posibles son cualquier expresión de color. Tambien admite el valor *transparent* que hace que el elemento tome el color del elemento superior, es el valor por defecto.
- background-image: Como valor puede tomar cualquier dirección relativa o absoluta de una imagen con la función *url()*. Si para un elemento se especifica esta propiedad y la anterior, el orden de precedencia hace que aparezca como fondo la imagen.

  Si esta propiedad se aplica al elemento principal, la imagen se toma como la propiedad *background* del elemento *BODY* de HTML. en cualquier otro caso se aplica sólo al espacio del elemento.

- background-repeat: Esta propiedad indica cómo se utiliza la imagen de fondo asignada para rellenar el fondo. Los valores que puede tomar son: *repeat*, *repeat-x*, *repeat-y* y *no-repeat*.
- background-attachment: Los posibles valores son *scroll* y *fixed*. *Nota: Esta propiedad no la soporta ningún navegador*.
- background-position: Indica la posición de la imagen con respecto a la ventana del navegador. Los valores que puede tomar son:
  - Palabras clave: top, center, bottom, left, rigth.
  - Porcentajes: Se puede indicar la posición indicando un porcentaje con respecto al elemento padre. Se deben dar dos porcentajes, x e y respectivamente.
  - Posiciones absolutas: utilizando dos medidas para la x y la y respectivamente.

  Ejemplos de cada una de las anteriores son:.

```
POEMA {background-image: url(fondo-azul.jpg);
    background-repeat:no-repeat;
```

```
    background-position: bottom}
POEMA {background-image: url(fondo-azul.jpg);
    background-repeat:no-repeat;
    background-position: 25% 50%}
POEMA {background-image: url(fondo-azul.jpg);
    background-repeat:no-repeat;
    background-position: 1cm 3cm}
POEMA {background-image: url(fondo-azul.jpg);
    background-repeat:no-repeat;
    background-position: top right}
```

Al igual que con las propiedades de las fuentes la propiedad *background* engloba las 5 anteriores. El valor de esta propiedad es una lista de elementos separados por espacios en el orden background-image, background-color, background-repeat, background-attachment y background-position.

Propiedades Text

Este conjunto de propiedades resumen la apariencia del texto.

En este grupo hay 8 propiedades:

- word-spacing: Permite expander o contraer el espacio entre palabras. Puede tomar cualquien valor de longitud y el valor normal es 1em.
- letter-spacing: Permite expander o contraer el espacio entre letras. Puede tomar cualquien valor de longitud y el valor normal es 0.3em.
- text-decoration: Puede tomar uno de estos cinco valores: *none*, *underline*, *overline*, *line-through* y *blink*. Y ninguno son autoexcluyentes excepto *blink*.
- vertical-align: Especifica la posición relativa de un elemento respecto a la línea de escritura. Los valores que puede tomar son *baseline*, *sub*, *super*, *top*, *text-top*, *middle*, *bottom* y *text-bottom*.
- text-transform: Permite indicar que tipo de letra utilizar. Los valores permitidos son *capitalize*, *uppercase*, *lowercase* y *none*.
- text-align: Indica la alineación del texto *left*, *right*, *center* y *justify*. Se aplica a elementos de bloque.
- text-indent: Indica la indentación de la primera línea de los elementos de bloque. Los posibles valores es cualquier medida de longitud.
- line-height: Indica la distancia entre una línea de escritura y la siguiente. Puede tomar cualquier valor absoluto de longitud o un porcentaje que indica la altura con respecto al alto de la letra.

Propiedades Box

Este conjunto de propiedades permite especificar las propiedades de ancho, alto, margenes, bordes de los espacios para mostrar los elementos.

En este grupo se incluyen:

1. **Propiedades de márgenes**: Especifican la cantidad de espacio que se añade a un bloque fuera de su borde. Se especifican mediante:
    - **margin-top**
    - **margin-bottom**
    - **margin-left**
    - **margin-right**

- La propiedad **margin** incluye a las otras cuatro recibiendo cuatro valores top, bottom right y left.

2. **Propiedades de bordes**: Los bordes se pueden especificar con:
   - **border-style**: que puede tomar los valores
     - **dotted .....**
     - **dashed - - - - -**
     - **solid ----------**
     - **double ==========**
     - **ridge** (con volumen)
     - **inset** (borde 3D hacia dentro)
     - **outset**(borde 3D hacia fuera)

   La forma de especificar un borde es por ejemplo:

   ```
   VERSO {display: block; border-style: dotted dashed solid double } /* -> top, left, bottom, right*/
   VERSO {display: block; border-style: dotted dashed solid } /* -> top, dashed(left, right), bottom*/
   VERSO {display: block; border-style: dotted dashed } /* ->dotted(top, bottom), dashed (left, right)*/
   VERSO {display: block; border-style: dotted  } /* -> dotted (top, left, bottom, right)*/
   ```

   - **border-width**: Que engloba *border-bottom-width*, *border-top-width*, *border-right-width* y *border-left-width*. Cada uno toma un valor de longitud.
   - border-color: Especifica los colores de los bordes, y admite, uno, dos, tres o cuatro valores de colores.

   Las propiedades de los bordes se agrupan tambien como:

   - **border-top**
   - **border-right**
   - **border-bottom**
   - **border-left**
   - **border**

   Cada una de las cuatro primeras puede admitir valores de ancho, estilo y color para cada uno de los bordes.

   La ultima propiedad *border* admite valores de ancho, estilo y color para los cuatro bordes.

3. **Propiedades de ajuste**: Especifican el espacio entre el bloque y el borde del bloque. Se especifica mediante:
   - padding-bottom
   - padding-top
   - padding-left
   - padding-right

   Todas admiten un valor de longitud. Y se agrupan las cuatro como *padding*.

4. **Propiedades de tamaño**: Es posible obligar a una caja a tener unas dimensiones determinadas. Las propiedades que te permiten esto son ***width* y *height*.** Los valores por defecto de estas propiedades es *auto*. Se aplican a elementos de bloque.

5. **Propiedades de posicionamiento:** Su utilidad es para cambiar la posición por defecto de cada caja. float: Puede ser inicializada a *none*, *left* o *right*.

Listas

- **display:list-item**: Los elementos aparecen como ítems de una lista, y se les puede aplicar las propiedades descritas a continuación:

  - **List-style-type:** Indica el tipo de viñeta o numeración a aplicar en los items de lista.

  Valores posibles:

  - disc: La viñeta es un círculo opaco.
  - circle: La viñeta es un círculo transparente y sólo se marca su circunferencia.
  - square: La viñeta es un cuadrado.
  - decimal: La viñeta se sustituye por una numeración que comienza en 1.
  - decimal-leading-zero: La viñeta se sustituye por una numeración que comienza en 01.
  - lower-roman: La viñeta se sustituye por la numeración romana, en minúsculas.
  - upper-roman: La viñeta se sustituye por la numeración romana, en mayúsculas.
  - lower-greek: La viñeta se sustituye por las letras del alfabeto griego, en minúsculas.
  - lower-latin: La viñeta se sustituye por las letras del alfabeto latino, en minúsculas.
  - upper-latin: La viñeta se sustituye por las letras del alfabeto latino, en mayúsculas.
  - armenian: La viñeta se sustituye por la numeración tradicional armenia.
  - georgian: La viñeta se sustituye por la numeración tradicional georgiana.
  - lower-alpha: La viñeta se sustituye por las letras del alfabeto occidental, en minúsculas. Es igual a lower-latin.
  - upper-alpha: La viñeta se sustituye por las letras del alfabeto occidental, en minúsculas. Es igual a upper-latin.
  - none: No se muestra viñeta alguna.
  - inherit: El elemento hereda el valor del elemento padre para esta propiedad.

  Valor por defecto:disc.

  - **List-style-position:** Indica la posición de la viñeta con respecto al ítem de lista, para el caso de que éste ocupe más de una línea.

  Valores posibles

  - inside: Indica que la viñeta debe situarse dentro de la caja de texto que componen las líneas del ítem de lista.
  - outside: Indica que la viñeta debe situarse fuera de la caja de texto que componen las líneas del ítem de lista.
  - inherit: El elemento hereda el valor del elemento padre para esta propiedad.

  Valor por defecto: outside.

Tablas

display:table. Aplica a un elemento estilos que hacen que se represente como diversos elementos de una tabla.

Valores posibles

- table-row: El elemento se representa como una fila de tabla.
- table-cell: El elemento se representa como una celda de tabla.
- table-caption: El elemento se representa como el título de una tabla.
- table-row-group: El elemento se representa como un grupo de filas.
- table-header-group: El elemento se representa como una cabecera de tabla.
- table-footer-group: El elemento se representa como un pie de tabla.
- table-column-group: El elemento se representa como un grupo de columnas.

border-collapse: Indica el modelo de borde de celdas.

Valores posibles

- collapse: Las celdas adyacentes pierden la separación entre ellas, por lo que comparten sus bordes.
- separate: Aplica a las celdas una separación, por lo que los bordes de estas son independientes.
- inherit: El elemento hereda el valor del elemento padre para esta propiedad.

border-spacing: Indica el espacio de relleno entre celdas de una tabla cuando se ha escogido el modelo de bordes separados.

Valores posibles

- Una medida absoluta o relativa.
- inherit: El elemento hereda el valor del elemento padre para esta propiedad.

ESQUEMAS XML

El XML Schema Definition Language es una recomendación del W3C que mejora en muchos aspectos la validación de documentos utilizando DTD.

Un Schema XML describe la estructura de un documento XML. El Schema XML es una alternativa al uso del DTD.

NOTA: Muchas veces el lenguaje empleado en el Schema XML se denomina XSD (XML Schema Definition).

El objetivo de un Schema XML es definir la estructura base de un documento XML, tal y como lo hace un DTD.

Un Schema XML:

- define los elementos y atributos que poden aparecer en un documento
- define que elementos son elementos hijo
- define el orden de los elementos hijo
- define o número de elementos hijo
- define cuando un elemento está vacio cuando puede incluír texto
- define tipos de datos para elementos y atributos

Los Schemas XML son los sucesores de los DTD

Las razones por las que los Schemas están reemplazando a los DTD son:

- Los Schemas XML se pueden ampliar con futuras adiciones
- Los Schemas XML son más potentes que los DTD
- Los Schemas XML están escritos en XML
- Los Schemas XML soportan tipos de datos
- Los Schemas XML soportan espacios de nombres

EL ELEMENTO SCHEMA

Veamos un ejemplo de un documento bien formado, pero que no está siendo validado ni con DTD ni con XSchema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<vehiculos>
      <nombre>coche</nombre>
      <nombre>moto</nombre>
      <nombre>carro</nombre>
</vehiculos>
```

Para validar ese documento, necesitaremos dos cosas:

- un esquema (que debe estar en un fichero .xsd diferente)
- referenciar a ese esquema dentro del documento XML

El documento XML para el ejemplo de los vehiculos:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<vehiculos
   xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation = "vehiculos.xsd"   >
      <nombre>coche</nombre>
      <nombre>moto</nombre>
      <nombre>carro</nombre>
</vehiculos>
```

El esquema para el ejemplo de los vehiculos:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
   <xs:element name = "vehiculos">
     <xs:complexType>
      <xs:sequence>
        <xs:element name = "nombre"
              type = "xsd:string"
              maxOccurs = "unbounded"/>
      </xs:sequence>
```

```
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Aclaremos algunas cosas que podemos ver en ese ejemplo:

En cuanto a la declaración del espacio de nombres en el fichero XML:

La línea xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance" indica que queremos utilizar los elementos definidos en http://www.w3.org/2001/XMLSchema-instance.
La línea xsi:noNamespaceSchemaLocation = "vehiculos.xsd" indica que vamos a usar ese fichero que contiene el XSChema, pero sin asociar un espacio de nombres a esas definiciones. Sin esta línea, no tendremos esquema de validación.

En cuanto a la definición del XSchema:

La línea xmlns:xs = "http://www.w3.org/2001/XMLSchema" indica que todos los elementos del XSchema se nombrarán con el prefijo xs:.

Elementos XSD

Un elemento simple es un elemento XML que contiene solamente texto. No puede contener ni otros elementos ni atributos.

Elementos simples

Sintaxis

```
<xs:element name="xxx" type="yyy"/>
```

Donde xxx es el nombre del elemento e yyy es el tipo de datos del elemento.

Los tipos de datos más comunes son:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Los elementos simples pueden tener un valor por defecto o un valor fixo.

El valor por defecto es asignado automaticamente al elemento cuando no se especifica otro valor. Ej.

```
<xs:element name="color" type="xs:string" default="azul"/>
```

Un valor fixo es asignado automaticamente al elemento y no se pode especificar otro valor diferente.

Ej:

```
<xs:element name="color" type="xs:string" fixed="azul"/>
```

Atributos XSD

Todos los atributos son declarados como tipos simples. Los elementos simples no pueden tener atributos. Si un elemento tiene atributos entonces será considerado un elemento complejo.

La **sintaxis** para definir un atributo es:

<xs:attribute name="xxx" type="yyy"/>

donde xxx es el nombre del atributo e yyy especifica el tipo de datos del atributo.

Tipos de datos más comunes:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

Ej:

<apellido idioma="ES">García</apellido>

Definición del atributo:

<xs:attribute name="idioma" type="xs:string"/>

Atributos opcionais e requiridos

Para especificar que un atributo es opcional o obligatorio, tenemos que emplear el atributo "use":

<xs:attribute name="idioma" type="xs:string" use="optional"/>

Restricciones XSD

Restricciones en valores

Ejemplo: El valor de la edad no puede ser menor que 0 o mayor que 120:

```
<xs:element name="edad">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restriciones en un conjunto de valores

Para limitar el contenido de un elemento XML a un conjunto aceptable de valores, emplearemos la restricción enumeration.

Ejemplo, en este código definimos un elemento llamado "coche" con una restricción. Los únicos valores aceptables son: BMW, Audi, Golf:

```
<xs:element name="coche">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="BMW"/>
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
    </xs:restriction>
  </xs:simpleType>
```

```
</xs:element>
```

Restricción en una serie de valores

Para limitar el contenido de un elemento XML a una serie de números o letras, emplearemos la restricción pattern.

Ejemplo: Se define el elemento letra con una restricción. Los únicos valores aceptables son UNA das letras del abecedario entre a-z (en minúsculas):

```
<xs:element name="letra">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El siguiente ejemplo define un elemento llamado "iniciales" con una restricción. El único valor aceptado son 3 letras en mayúsculas desde a hasta z:

```
<xs:element name="iniciales">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Ejemplo: Se define de nuevo otro elemento llamado "iniciales" con una restricción. Solamente se aceptan 3 letras en mayúsculas o minúsculas desde a hasta z:

```
<xs:element name="iniciales">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Ejemplo: Se define un elemento llamado "opcion" con una restricción. Solamente se acepta UNA de las siguientes letras: x, y o z:

```
<xs:element name="opcion">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[xyz]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Ejemplo: Se define un elemento llamado identificador con una restricción. Solamente se aceptan 5 dígitos en secuencia, y cada dígito estará enel rango de 0 hasta 9:

```
<xs:element name="identificador">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
    </xs:restriction>
  </xs:simpleType>
```

```
</xs:element>
```

Otras restriccións en una serie de valores

El siguiente ejemplo define un elemento llamado "letras" con una restricción. El valor aceptado puede ser 0 ou máis ocurrencias de las letras del abecedario en minúsculas:

```
<xs:element name="letras">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="([a-z])*"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El siguiente ejemplo define un elemento llamado sexo con una restricción. Los únicos valores aceptados son hombre o mujer:

```
<xs:element name="sexo">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="hombre|mujer"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El siguiente ejemplo define un elemento llamado "contrasena" con una limitación. Debe tener exactamente 8 caracteres, en minúsculas o maiúsculas con letras de la-z o números 0-9:

```
<xs:element name="contrasena">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Limitaciones de longitud

Para limitar la longitud de un valor en un elemento, podemos emplear las limitaciones maxLength, minLength y length

El siguiente ejemplo muestra un elemento llamado "contrasena" con una limitación. El valor debe tener exactamente 8 caracteres:

```
<xs:element name="contrasena">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

El siguiente ejemplo define otro elemento "contrasena" con una restricción. El valor debe tener como mínimo 5 caracteres y como máximo 8 caracteres:

```
<xs:element name="contrasena">
```

```
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="5"/>
      <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restriciones para tipos de datos

Restricción	Descripción
enumeration	Define los valores permitidos en una lista
fractionDigits	Especifica el número máximo de decimales permitidos. Debe ser mayor o igual a 0.
length	Especifica el número exacto de caracteres o elementos permitidos. Debe ser mayor o igual a 0.
maxExclusive	Especifica el límite superior para valores numéricos (el valor debe ser menor que este límite)
maxInclusive	Especifica el límite superior para valores numéricos (el valor debe ser menor o igual que este límite)
maxLength	Especifica o número máximo de caracteres ou valores permitidos. Debe ser maior ou igual que cero.
minExclusive	Especifica o límite inferior para valores numéricos (o valor debe ser maior que este límite)
minInclusive	Especifica o límite inferior para valores numéricos (o valor debe ser menor ou igual a este límite)
minLength	Especifica o número mínimo de caracteres ou valores permitidos. Debe ser maior ou igual que cero.
pattern	Define a secuencia exacta de caracteres permitidos.
totalDigits	Define o número exacto de díxitos permitidos. Debe ser maior que cero.
whiteSpace	Especifica como se xestionan os espazos en branco (tabs, saltos de liña, etc.)

Tipos Complejos (Complex Types)

En los antiguos DTDs, el modelo de contenido de un elemento está definido dentro de una declaración de ELEMENT, como se muestra aquí:

```
<!ELEMENT empleado (nombre, fechaingreso, salario)>
```

Esta declaración de elemento dice que un elemento empleado contiene un elemento nombre, seguido de un elemento fechaingreso, y luego por un elemento salario –en ese orden.

XML Schema permite definir en forma similar el modelo de contenido de un elemento, al anidar un elemento xsd:complexType dentro dela declaración del xsd:element, como se muestra a continuación:

```
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema”” targetNamespace=”http://ejemplo.org/empleado” >
  <xsd:element  name=”empleado”>
    <xsd:complexType>
      <!-- modelo de contenido del empleado va aquí -->
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Elementos complejos

Son elementos que contienen a otros elementos hijos, o que tienen atributos

Se suelen dividir en 4 tipos:

Elementos vacíos

Elementos no vacíos con atributos

Elementos con elementos hijos

Elementos con elementos hijos y con “texto” o valor propio (como el contenido mixto de las DTD)

Ejemplos:

```
<product pid="1345"/>
<food type="dessert">Ice cream</food>

<description>Sucedió el <date>03.03.99</date> .... </description>

<employee>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
</employee>
```

Para definir elementos complejos se utiliza la siguiente sintáxis:

```
<xsd:element name="employee">
    <xsd:complexType>
        <xsd:sequence>
        <xsd:element name="firstname" type="xsd:string"/>
        <xsd:element name="lastname" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

Para declarar un elemento no vacío con atributos, y sin elementos hijos, se utilizará la siguiente sintáxis:

```
<xsd:element name="shoesize">
    <xsd:complexType>
        <xsd:simpleContent>
            <xsd:extension base="xsd:integer">
                <xsd:attribute name="country" type="xsd:string" />
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
</xsd:element>
```

Para declarar un elemento con contenido “mixto”, basta con añadir un atributo “mixed” al elemento xsd:complexType:

```
<xsd:element name="letter">
    <xsd:complexType mixed="true">
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="orderid" type="xsd:positiveInteger"/>
            <xsd:element name="shipdate" type="xsd:date"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

La declaración anterior permitiría un texto como el siguiente:

```
<letter>Estimado cliente: <name>Juan Perez</name>. Su pedido número
<orderid>1032</orderid> se enviará el día <shipdate>2001-07-13</shipdate>. </letter>
```

En los ejemplos anteriores hemos utilizado el elemento xsd:sequence como elemento hijo del elemento xsd:complexType

Xsd:sequence indica que los elementos anidados en él deben aparecer en un orden determinado

Los esquemas XML nos ofrecen otras alternativas, además de xsd:sequence, para indicar cómo se deben tratar los elementos que aparecen anidados en un elemento complejo.

Las opciones o “indicadores” son: **xsd:all** y **xsd:choice**

El indicador **xsd:all** indica que los elementos que contiene pueden aparecer en cualquier orden, pero como máximo sólo una vez

```
<xsd:element name="person">
    <xsd:complexType>
        <xsd:all>
```

```
                <xsd:element name="firstname" type="xsd:string"/>
                <xsd:element name="lastname" type="xsd:string"/>
            </xsd:all>
        </xsd:complexType>
</xsd:element>
```

El indicador xsd:choice indica que puede aparecer sólo uno de los elementos que contiene

```
<xsd:element name="person">
        <xsd:complexType>
            <xsd:choice>
                <xsd:element name="firstname" type="xsd:string"/>
                <xsd:element name="lastname" type="xsd:string"/>
            </xsd:choice>
        </xsd:complexType>
</xsd:element>
```

En esquemas XML también contamos con un modelo de contenido ANY, que permite incluir elementos no declarados inicialmente en el esquema

```
<xsd:element name="person">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="firstname" type="xsd:string"/>
                <xsd:element name="lastname" type="xsd:string"/>
                <xsd:any minOccurs="0"/>
            </xsd:sequence>
        </xsd:complexType>
</xsd:element>
```

También contamos con un elemento que permite extender el número de atributos de un elemento:

```
<xsd:element name="person">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="firstname" type="xsd:string"/>
                <xsd:element name="lastname" type="xsd:string"/>
            </xsd:sequence>
            <xsd:anyAttribute/>
        </xsd:complexType>
</xsd:element>
```