XML (eXtensible Markup Language)

1. **¿Qué es XML y para qué se utiliza?**
   XML (eXtensible Markup Language) es un lenguaje de marcado utilizado para almacenar y transportar datos de una forma estructurada y legible tanto por máquinas como por humanos. Es ampliamente usado para el intercambio de datos entre sistemas distintos.
2. **¿Cuáles son las principales diferencias entre XML y HTML?**
   - XML está diseñado para almacenar y estructurar datos, mientras que HTML se usa para representar y mostrar información en la web.
   - XML es extensible y permite definir etiquetas personalizadas; HTML tiene un conjunto fijo de etiquetas.
   - XML requiere documentos bien formados, mientras que HTML es más permisivo con los errores.
3. **Explica la estructura básica de un documento XML. ¿Cuáles son sus componentes principales?**
   Un documento XML comienza con una declaración (`<?xml version="1.0"?>`), seguido por un único elemento raíz que contiene elementos anidados. Los componentes principales son los elementos, atributos, texto, entidades y comentarios.
4. **¿Qué es un elemento en XML y cómo se diferencia de un atributo?**
   Un elemento en XML es una estructura de datos etiquetada (como `<nombre>Juan</nombre>`) que puede contener texto, otros elementos o ambos. Un atributo es información adicional dentro de un elemento (como `<persona edad="30">`), y no puede contener otros elementos.
5. **¿Qué es el XML Namespaces y por qué son importantes?**
   XML Namespaces se utiliza para evitar conflictos de nombres en documentos que integran varios conjuntos de etiquetas con prefijos (como `xmlns:ex="http://example.com/"`). Permite diferenciar elementos y atributos con el mismo nombre pero de diferentes esquemas.
6. **¿Cómo se representa un documento XML bien formado?**
   Un documento XML bien formado debe tener una estructura jerárquica correcta, con etiquetas de apertura y cierre en el orden adecuado y anidamiento adecuado, un solo elemento raíz y un uso correcto de comillas y caracteres especiales.
7. **¿Cuáles son las reglas para que un documento XML sea válido?**
   Un documento XML es válido si cumple con las reglas de un esquema o DTD asociado, lo cual asegura que el documento sigue una estructura y tipos de datos específicos.
8. **¿Qué son las DTD (Document Type Definition) y cómo se relacionan con XML?**
   DTD es un conjunto de reglas y definiciones que especifica la estructura permitida de un documento XML, asegurando la validez de los elementos, atributos y su organización dentro del documento.
9. **¿Cuáles son las ventajas de utilizar XML como formato de datos?**
   XML es independiente de la plataforma, extensible, legible por humanos, soportado por muchas aplicaciones y lenguajes de programación, y puede ser validado para asegurar la integridad de los datos.

XSD (XML Schema Definition)

1. **¿Qué es un esquema XSD y cuál es su propósito en XML?**
   XSD (XML Schema Definition) es un lenguaje de descripción de esquemas para definir la estructura y los tipos de datos de un documento XML. Su propósito es validar que el documento XML cumpla con una estructura específica.
2. **¿Cuáles son las principales diferencias entre XSD y DTD?**
   XSD permite tipos de datos más específicos, admite tipos complejos, es extensible y usa XML como formato de definición, mientras que DTD tiene un conjunto limitado de tipos y es menos flexible.
3. **¿Qué es un tipo de datos simple en XSD y qué tipos de datos soporta?**
   Un tipo de datos simple en XSD es aquel que no puede contener elementos o atributos. Incluye tipos como `xs:string`, `xs:integer`, `xs:boolean`, `xs:date`, entre otros.
4. **¿Qué es un tipo de datos complejo en XSD? Proporciona un ejemplo.**
   Un tipo de datos complejo en XSD puede contener otros elementos y atributos. Ejemplo:

   ```
   xml
   <xs:complexType name="Persona">
       <xs:sequence>
           <xs:element name="nombre" type="xs:string"/>
           <xs:element name="edad" type="xs:integer"/>
       </xs:sequence>
   </xs:complexType>
   ```

5. **¿Qué son los elementos `<xs:sequence>`, `<xs:choice>` y `<xs:all>` en XSD y cómo se usan?**
   - `<xs:sequence>`: Define un orden específico para los elementos hijos.
   - `<xs:choice>`: Permite que solo uno de los elementos especificados aparezca en el documento.
   - `<xs:all>`: Permite que todos los elementos hijos aparezcan en cualquier orden.
6. **¿Qué significa que un elemento en XSD sea opcional o repetitivo? ¿Cómo se configura esto en XSD?**
   Un elemento opcional tiene `minOccurs="0"`, y uno repetitivo tiene `maxOccurs` establecido a un valor mayor a 1 o a "unbounded" para permitir múltiples ocurrencias.
7. **¿Qué es el "targetNamespace" en XSD y cómo se usa?**
   `targetNamespace` especifica un espacio de nombres único para los elementos y tipos en un esquema, ayudando a evitar conflictos con otros esquemas.
8. **¿Cómo se definen restricciones de valor (como longitud o rango) en un elemento de XSD?**
   Las restricciones se establecen mediante elementos como `<xs:minLength>`, `<xs:maxLength>`, `<xs:minInclusive>`, y `<xs:maxInclusive>` para limitar el valor o la longitud.
9. **¿Qué son los atributos `minOccurs` y `maxOccurs` y cómo se usan en un esquema XML?**
   `minOccurs` define el número mínimo de veces que un elemento puede aparecer, mientras que `maxOccurs` define el número máximo.

XPath (XML Path Language)

1. **¿Qué es XPath y para qué se utiliza?**
   XPath es un lenguaje de consulta que se utiliza para navegar y seleccionar nodos en documentos XML.
2. **¿Cuál es la diferencia entre una expresión absoluta y una expresión relativa en XPath?**
   Una expresión absoluta empieza desde el nodo raíz (`/`), mientras que una relativa selecciona nodos desde la posición actual (`./`).
3. **Explica el funcionamiento del operador `/` y `//` en una expresión XPath.**
   `/` selecciona nodos directamente desde el raíz, mientras que `//` selecciona todos los nodos que coincidan desde cualquier lugar en el documento.
4. **¿Qué significa el símbolo `@` en una expresión XPath?**
   `@` se usa para seleccionar atributos, como en `@nombre` para seleccionar el atributo `nombre`.
5. **¿Cómo seleccionas todos los elementos de un tipo específico dentro de un documento XML usando XPath?**
   Usando una expresión como `//elemento`, que selecciona todos los elementos llamados `elemento`.
6. **¿Qué es una función de posición en XPath y para qué se usa? (ej. `position()`, `last()`)**
   Las funciones de posición (`position()`, `last()`) se usan para seleccionar elementos específicos en una posición o para filtrar resultados.
7. **¿Qué diferencia hay entre `text()` y `node()` en XPath?**
   `text()` selecciona el contenido de texto de un nodo, mientras que `node()` selecciona cualquier tipo de nodo (texto, elementos, comentarios).
8. **¿Qué es un predicado en XPath y cómo se utiliza en las expresiones?**
   Un predicado es una condición entre corchetes (`[ ]`) que filtra nodos según criterios específicos, como `//persona[edad>30]`.
9. **¿Cómo seleccionarías el primer elemento de un tipo en particular dentro de un grupo en XPath?**
   Con `//elemento[1]`.
10. **¿Qué son las funciones agregadas en XPath, como `count()` y `sum()`? Proporciona ejemplos de su uso.**
    - `count()`: cuenta el número de nodos seleccionados (`count(//persona)`).
    - `sum()`: suma valores de nodos (`sum(//precio)`).

XQuery (XML Query Language)

1. **¿Qué es XQuery y cuál es su relación con XPath?**
   XQuery es un lenguaje de consulta para XML que expande XPath para manipular y consultar datos de XML.
2. **¿Cuál es la principal diferencia entre XPath y XQuery?**
   XPath se usa principalmente para navegación y selección de nodos, mientras que

XQuery permite consultas avanzadas con filtros, operaciones y transformaciones.

3. **¿Qué significa FLWOR en XQuery y cuál es su propósito?**
   FLWOR (`For`, `Let`, `Where`, `Order by`, `Return`) es una estructura que permite iterar, filtrar, ordenar y transformar datos XML.
4. **¿Cómo se realiza un filtrado de elementos en XQuery utilizando la estructura FLWOR?**
   Con una expresión `where`, como en:

```
xquery
for $x in //persona
where $x/edad > 30
return $x
```

5. **¿Qué es el `let` en XQuery y cuándo es útil?**
   `let` asigna valores a variables y es útil para almacenar resultados intermedios.
6. **¿Cómo se puede ordenar un conjunto de resultados en XQuery?**
   Con `order by`, como en `order by $x/nombre`.
7. **¿Cuál es la diferencia entre `for` y `let` en una expresión FLWOR?**
   `for` itera sobre secuencias, mientras que `let` solo asigna un valor sin iteración.
8. **¿Qué es el operador `return` en XQuery y cómo funciona?**
   `return` define lo que se debe devolver en cada iteración de un `for`.
9. **¿Cómo se puede realizar una unión de datos en XQuery desde dos fuentes XML diferentes?**
   Iterando sobre ambos conjuntos y uniéndolos en una estructura `for` o `let`.
10. **¿Qué son las funciones en XQuery y cómo se definen? Proporciona un ejemplo.**
    Las funciones se definen con `declare function`, como:

```
xquery
declare function local:suma($a, $b) {
   $a + $b
};
```