Disciplina: Linguagens e Tecnologias Web (edição 2013/14)
Data: Segunda-feira, 03 de Fevereiro de 2014
Hora: 09h00m
Duração: 120 minutos
Salas: B222, B227 e B229
Nota: Com consulta de apontamentos em papel
Época: Recurso
Docentes: JVV, AOR e TPF

1) Considerando o HTML representado abaixo responda às seguintes alíneas

a) (1v) Apresente um esboço da visualização no navegador assumindo que os ficheiros de CSS não existem.

```
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="1.css"/>
        <link rel="stylesheet" type="text/css" href="2.css"/>
        <link rel="stylesheet" type="text/css" href="3.css"/>
    </head>
    <body>
        <ul>
            <li><div>A</div><div>B</div></li>
            <li><div>C</div><div>D</div></li>
            <li><div></div><div></div></li>
        </ul>
    </body>
</html>
```

b) (2v) Repita a alínea anterior assumindo agora a existência do ficheiro 1.css com o seguinte conteúdo. Comece por indicar a especificidade de cada seletor.

```
*{margin:0px;padding:0px;}
ul{
    margin:0 auto;
    width:800px;
    height:200px;
}
li{
    display:block;
    float:right;
    width:200px;
    height:200px;
    position:relative;
    counter-reset: cnt;
}
div{
    width:198px;
    height:98px;
    border-width:1px;
    border-color:black;
    text-align:center;
}
li>div{border-style:solid;}
li div{border-style:dotted;}
div::after{
    content: " => " counter(cnt);
    counter-increment: cnt;
}
div:empty::after{content: counter(cnt);}
```

c) (1v) Assuma agora que o ficheiro 2.css também existe e tem o conteúdo representado abaixo. Apresente um novo esboço

```
li ~ li *:first-child{
    position:absolute;
    bottom:0px;
}
```

d) } (1v) Repita, considerando agora a existência dos 3 ficheiros de css. O conteúdo do ficheiro 3.css é o que se representa abaixo. Para clarificação do resultado indique a que elemento(s) está a aplicar as transformações.

```
:only-of-type:not(html):not(body){
    transform:rotate(-90deg);
    transform-origin:100% 100%;
}
```

2) (4v) Suponha que tem uma página HTML com uma única tabela, com um único elemento tbody, zero theads e zero tfoots. O tbody, inicialmente, tem pelo menos uma linha e, dinamicamente, podem ser adicionadas/removidas novas linhas, da seguinte forma:

- De cada vez que o rato entra em qualquer uma das últimas 10 linhas é feito um pedido ajax que retorna uma nova linha em formato json.
- De cada vez que o rato entra numa linha que não seja uma das 10 últimas é removida a última linha da tabela.

Utilizando jQuery implemente essa funcionalidade assumindo que o objeto json retornado é um array em que cada posição contém o valor a exibir numa célula da nova linha.
O pedido ajax deve ser feito para o url relativo ‘./getNextLine.php?currentLine=X’.
O valor X, a enviar, deve ser obtido da primeira célula da última linha e pode assumir que é um nó de texto. Isto é, a primeira célula de cada linha contém o número dessa linha. Pode também assumir que cada célula ocupa uma e uma só linha e coluna.
Sugestão tire partidos dos seletores CSS ou jQuery para obter as linhas da tabela desejadas e use os métodos jQuery apropriados para percorrer a árvore DOM (Traversing). Evite escrever código com muitas linhas, principalmente condições de teste que são facilmente evitáveis em jQuery e tornam o código difícil de interpretar.

3) (3v) Usando PHP escreva o script checkTimeFormat.php?time1=12:30:50&time2=02:10:00&…& timeN=23:59:59 que recebe uma lista variável de parâmetros timeI (I >= 1) e verifica, através de um expressão regular, se o valor de cada parâmetro segue o formato HH:MM:SS. Se todos os parâmetros contiverem valores válidos é devolvido uma mensagem com um array vazio. Caso contrário deve ser devolvido um array identificando cada um dos parâmetros que não validou, exemplo: ‘[“time3”,”time7”]’ As mensagens de resposta devem ser do tipo “application/json”. Apresente, no final, uma expressão regular mais elaborada que valide quer o formato HH:MM:SS quer o formato HH-MM-SS

4) (4v) Considerando a seguinte informação, relativa a um armazém de produtos hortícolas, represente-a num documento XML válido segundo o XSD apresentado na página seguinte. Use um atributo de nome ‘nota’, no espaço de nomes adequado, para assinalar arredondamentos, ou agregações de produtos, que eventualmente venha a fazer.

Encomendas			
Data	Produto	Kg	Preço
15/1/2014	Batata	700	1.1€/Kg
15/1/2014	Alho	200	
03/02/2014	Batata	400	0.9€/Kg

Stock		
Produto	Preço	Kg
Batata	0.9€/Kg	6000
Batata	1.11€/Kg	300
Batata	1.12€/kg	600
Cebola	0.92€/kg	2000
Alho	2€/kg	100

5) Para um documento com a estrutura exemplificada na última página, apresente:

a) (1v) Uma expressão Xpath para obter o número de produtos em Stock com existências superiores a 1000 unidades de medida (Kg no exemplo). (Ignore o espaço de nomes.)

b) (1v) Uma expressão Xpath para obter a soma das quantidades dos 3 produtos mais caros, assumindo que estão ordenados por ordem ascendente de preço (Ignore o espaço de nomes.)

c) (1v) O resultado da transformação do XML pelo XSL da última página. Para simplificar, assuma que o último template não existe

d) (1v) Repita a alínea anterior mas considerando agora os templates todos

Listagens de código

```xml
<?xml version="1.0" encoding="UTF-8"?>
<schema
    xmlns="http://www.w3.org/2001/XMLSchema"
    xmlns:t="urn:feup.2014"
    targetNamespace="urn:feup.2014"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified">
    <complexType name="pType">
        <attribute name="nome"/>
        <attribute name="preço" type="t:dType"/>
        <attribute name="disponibilidade" type="int" use="required" />
        <anyAttribute namespace="urn:feup.other" processContents="skip"/>
    </complexType>
    <element name="encomenda">
        <complexType>
            <simpleContent>
                <extension base="positiveInteger">
                    <attribute name="produto"/>
                    <attribute name="valor" type="t:dType"/>
                </extension>
            </simpleContent>
        </complexType>
    </element>
    <simpleType name="dType">
        <restriction base="decimal">
            <fractionDigits value="1"/>
        </restriction>
    </simpleType>
    <element name="Produtos">
        <complexType>
            <sequence>
                <element name="produto" type="t:pType" maxOccurs="unbounded"/>
                <sequence minOccurs="0" maxOccurs="unbounded">
                    <element name="data" type="date"/>
                    <element ref="t:encomenda" maxOccurs="unbounded"/>
                </sequence>
            </sequence>
        </complexType>
        <key name="k">
            <selector xpath="produto"/>
            <field xpath="@nome"/>
            <field xpath="@preço"/>
        </key>
        <keyref refer="t:k" name="kr">
            <selector xpath="t:encomenda"/>
            <field xpath="@produto"/>
            <field xpath="@valor"/>
        </keyref>
        <unique name="u">
            <selector xpath="data"/>
            <field xpath="."/>
        </unique>
    </element>
</schema>
```

```xml
<Produtos xmlns="urn:mercado.fruta">
<!-- exemplo da estrutura dos documentos XML tratados na pergunta 5 -->
    <Stock>
        <produto nome="banana" quantidade="2000" un="kg"/>
        <produto nome="pêra" quantidade="3000"  un="kg"/>
        <produto nome="maçã" quantidade="3500" un="kg"/>
        <produto nome="ananáz" quantidade="1000" un="kg"/>
    </Stock>
    <Preços>
        <produto nome="pêra" un="€">1.2</produto>
        <produto nome="maçã" un="€">1.5</produto>
        <produto nome="banana" un="€">2</produto>
    </Preços>
</Produtos>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet  xmlns:x="urn:mercado.fruta" version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="*">
        <xsl:element name="{local-name()}" namespace="urn:feup.other">
            <xsl:apply-templates select="node()|@*"/>
        </xsl:element>
    </xsl:template>
    <xsl:template match="@*">
        <xsl:attribute name="x:{local-name()}" namespace="urn:feup.other">
            <xsl:value-of select="."/>
        </xsl:attribute>
    </xsl:template>
    <xsl:template match="*[ancestor::x:Preços]">
        <xsl:copy/>
    </xsl:template>
    <xsl:template match="*[child::*/child::x:*]">
        <xsl:copy>
            <xsl:copy-of select="descendant::*[
              generate-id(parent::*) != generate-id(current())
            ]"/>
        </xsl:copy>
    </xsl:template>
</xsl:stylesheet>
```