

Linguagens e Tecnologias Web

Faculdade de Engenharia da Universidade do Porto
1 de Fevereiro de 2016

Duração: 2h / Com Consulta

Nome: _____

Número: _____

1. Considere o seguinte código HTML:

```
1 <div id="profile" class="box">
2   <h2 class="name">John Doe</h2>
3   <ul>
4     <li><a href="www.johndoe.com">Website</a></li>
5     <li class="phone">111-222-333</li>
6     <li class="address">Saint Doe Street , 123</li>
7   </ul>
8 </div>
```

E o seguinte código CSS:

```
1 ul li:first-child {color: red;} /* R1 */
2 #profile.box .name {color: blue;} /* R2 */
3 #profile li ~ li ~ li {color: green;} /* R3 */
4
5 #profile .address {color: magenta;} /* R4 */
6 div ul li.address {color: yellow;} /* R5 */
7 a {color: cyan;} /* R6 */
```

1½ val.

(a) Indique a especificidade de cada uma das regras (e.g. 0,2,2,1):

R1	R2	R3	R4	R5	R6

1 val.

(b) Considerando apenas as regras de **R1 a R3**, indique a cor de cada um dos textos:

John Doe	Website	111-222-333	Saint Doe Street, 123

1 val.

(c) Considerando **todas as regras**, indique a cor de cada um dos textos:

John Doe	Website	111-222-333	Saint Doe Street, 123

2. Considere a seguinte *string*:

How many yaks could a yak pack pack if a yak pack could pack yaks

Para cada uma das expressões regulares apresentadas de seguida, sublinhe qual o primeiro *match*:

1/2 val.

(a) `/pack.*pack/`

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(b) `/[pack]{2}/`

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(c) `/(yak|pack).*\1/`

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(d) `/[^aeiou]{3}/`

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(e) `/(?<!ya)k/`

How many yaks could a yak pack pack if a yak pack could pack yaks

1/2 val.

(f) `/(\w{3,}?).*?\1/`

How many yaks could a yak pack pack if a yak pack could pack yaks

3. Considere o seguinte excerto HTML que representa um teclado virtual que pretende impedir ataques usando *keyboard loggers*:

```
1 <form id="pin" method="post">
2   <input type="text" name="username">
3   <input type="text" name="pin">
4   <input type="submit" value="Verify">
5 </form>
6 <div id="keypad">
7   <a href="#">1</a> <a href="#">2</a>   <a href="#">3</a><br>
8   <a href="#">4</a> <a href="#">5</a>   <a href="#">6</a><br>
9   <a href="#">7</a> <a href="#">8</a>   <a href="#">9</a><br>
10 </div>
```

Considere que pode haver outros elementos *a*, *input* e *submit* no documento. Escreva o código *jQuery* necessário para que:

1 val.

(a) Quando o utilizador *clique* num dos números do teclado virtual, o valor desse número seja acrescentado ao valor do *input* com o nome *pin*.

Nome: _____

Número: _____

2 val.

- (b) Quando o botão de *submit* seja *clicado*, os valores dos *inputs* *username* e *pin* sejam enviados, em duas variáveis com nome *username* e *pin*, num pedido *Ajax* do tipo *POST* para o endereço *verify-pin.php*. Caso a resposta indique que o *pin* não é válido, o *border* do input *pin* deve passar a vermelho e o seu valor deve ser apagado. Considere que o resultado, em JSON, vem no seguinte formato: `{"valid": "true"}` ou `{"valid": "false"}`.



(Continua do outro lado...)

2½ val.

4. Crie um documento XML que seja bem formado e válido segundo o seguinte XSD:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:complexType name="elementType">
4     <xs:sequence>
5       <xs:element name="title" type="xs:string"/>
6     </xs:sequence>
7     <xs:attribute name="num" type="xs:integer"/>
8   </xs:complexType>
9   <xs:complexType name="groupType">
10    <xs:sequence>
11      <xs:element name="element" type="xs:integer" maxOccurs="unbounded" minOccurs="2"/>
12    </xs:sequence>
13  </xs:complexType>
14  <xs:complexType name="descriptionType">
15    <xs:sequence>
16      <xs:element name="element" type="elementType" maxOccurs="unbounded"/>
17      <xs:element name="group" type="groupType" maxOccurs="unbounded" minOccurs="2"/>
18    </xs:sequence>
19  </xs:complexType>
20  <xs:element name="description" type="descriptionType">
21    <xs:key name="elementKey">
22      <xs:selector xpath="element"/>
23      <xs:field xpath="@num" />
24    </xs:key>
25    <xs:keyref name="elementRef" refer="elementKey">
26      <xs:selector xpath="group/element"/>
27      <xs:field xpath="."/>
28    </xs:keyref>
29  </xs:element>
30 </xs:schema>
```

