Universidad de La Laguna. Escuela Técnica Superior de Ingeniería Informática
Tercero del Grado de Informática
PROCESADORES DE LENGUAJES. REPASO 2ª PARTE
2014/2015 11 páginas

Índice

1. jQuery

1. jQuery uses CSS selectors to select elements? True or false?
2. Which sign does jQuery use as a shortcut for jQuery?
3. With jQuery, look at the following selector: `$("div")`. What does it select?
4. The jQuery `html()` method works for both HTML and XML documents. True or false?
5. What is the correct jQuery code to set the background color of all p elements to red?
6. With jQuery, look at the following selector: `$("div.intro")`. What does it select?
7. Which jQuery method is used to hide selected elements?

8. Which jQuery method is used to set one or more style properties for selected elements?
9. Which jQuery method is used to perform an asynchronous HTTP request?
10. What is the correct jQuery code for making all div elements 100 pixels high?
11. Which jQuery function is used to prevent code from running, before the document is finished loading?
12. Which jQuery method should be used to deal with name conflicts?
13. Which jQuery method is used to switch between adding/removing one or more classes (for CSS) from selected elements?
14. Look at the following jQuery selector: `$("div#intro .head")`. What does it select?
15. Is jQuery a W3C standard?

2. Underscore

1. ¿Que argumentos espera el método `template` de Underscore? ¿Cual es la función de cada uno de ellos?
2. ¿Que diferencia hay entre `<% ... %>`, `<%= ... %>` y `<%- ... %>`?
3. El atributo `templateSettings` de Underscore puede ser usado para configurar los templates. Rellene las expresiones regulares que faltan para que se usen delimitadores con llaves como `{{ ... }}` para evaluar, `{{= ... }}` para interpolar y `{{- ... }}` para interpolar y escapar el HTML:

```
_.templateSettings = {
       interpolate: /____________________/gim,
       evaluate:    /____________________/gim
       escape:      /____________________/gim
 }
```

4. Queremos mostrar una lista de `items` en una tabla con dos columnas. En la primera columna va el número de orden y en la segunda el nombre del item. El template Underscore se carga desde el elemento `#usageList` del DOM y el resultado del template se vuelca en el elemento `#target` como sigue:

```
var items = [ {name:"Alejandro"}, {name:"......."}, {name:"Zacarias"} ];
var template = usageList.innerHTML;
target.innerHTML = _.template(template,{items:items});
```

Complete el identificador del `<script>`, el identificador del `<div>` usado para la salida y la parte que falta entre las etiquetas `<tbody>`:

```
<script type="text/html" id='_________'>
    <table>
      <thead>
         <tr>
            <th> Id </th> <th>Name</th>
         </tr>
      </thead>
      <tbody>
```

```
    </tbody >
   </table>
</script>
<!-- Create your target -->
<div id="______"></div>
```

3. localStorage

1. ¿Que diferencias y que similitudes hay entre los cookies y localStorage?
2. ¿Que diferencias hay entre localStorage y sessionStorage?
3. ¿En que consiste las reglas de segregación conocidas como *same origin policy*? ¿Como se relacionan con localStorage?
4. ¿Como guardo en localStorage el contenido de una variable `chuchu`? ¿Como leo el contenido almacenado en localStorage con clave `chuchu`?
5. ¿Como borro una clave almacenada en localStorage?
6. *a*) ¿Que es Local Storage? ¿Que hace la siguiente línea?

   ```
   if (window.localStorage) localStorage.original  = temp;
   ```

   b) ¿Cuando se ejecutará esta callback? ¿Que hace?

   ```
   window.onload = function() {
     // If the browser supports localStorage and we have some stored data
     if (window.localStorage && localStorage.original) {
       document.getElementById("original").value = localStorage.original;
     }
   };
   ```

4. Heroku

1. Una vez instalado el Heroku toolbelt nos debemos autenticar en heroku con el cliente. ¿Cual es el comando para autenticarnos?
2. ¿Cual es el comando para crear nuestra aplicación en Heroku (suponemos que ya esta bajo el control de `git`? ¿Qué remoto tendremos definido después de crear nuestra aplicación en Heroku?
3. ¿Cual es el comando para desplegar nuestra aplicación en Heroku?
4. Si la versión que queremos publicar en heroku no está en la rama `master` sino que está en la rama `tutu` ¿Como debemos modificar el comando anterior?
5. ¿Con que comando puedo abrir una ventana en el navegador que visite la aplicación desplegada? ¿Que formato tiene la URL para nuestra aplicación?
6. ¿Con que comando puedo ver los logs de la aplicación desplegada?

7. ¿Como se debe llamar el fichero en el que explicito que comando debe usarse para arrancar nuestra aplicación en Heroku?
8. Heroku se percata que nuestra aplicación es una aplicación desarrollada con `Node.js` por la presencia de un cierto fichero. ¿De que fichero estamos hablando?
9. ¿Cual es la mejor forma de ejecutar en local una aplicación express.js que va a ser desplegada en Heroku?

5. EJS

1. Véase Getting Started with EJS
2. Mejore este template

```
<li>
    <a href='supplies/<%= supplies[i] %>'>
        <%= supplies[i] %>
    </a>
</li>
```

usando un helper proveído por EJS

6. Jade

1. Véase http://jade-lang.com/demo/
2. Traduzca a Jade el siguiente código HTML:

```
<div id="content">
  <div class="block">
    <input id="bar" class="foo1 foo2"/>
  </div>
</div>
```

3. ¿Que hace el punto después de un elemento? como en:

```
p.
  foo bar
  hello world
```

4. Supongamos que el objeto interpolado es:

```
{"name": "Hello <em>World</em>"}
```

¿Que diferencia hay entre estas dos interpolaciones?

   a) `li= name`
   b) `li!= name`

5. ¿Que diferencia hay entre estas dos interpolaciones?

   a) `li Say #{name}`
   b) `li Say !{name}`

6. Traduzca a Jade el siguiente fragmento HTML:

```
<input type="text" placeholder="your name"/>
```

7. SASS

1. ¿Como se inicializa una constante en SASS?

2. Reescriba este fragmento CSS en SASS:

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}

nav li {
  display: inline-block;
}

nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

8. Expresiones Regulares

1. Rellene las partes que faltan:

```
> re = /d(b+)(d)/ig
/d(b+)(d)/gi
> z = "dBdxdbbdzdbd"
'dBdxdbbdzdbd'
> result = re.exec(z)
[ ______, _____, ______, index: __, input: 'dBdxdbbdzdbd' ]
> re.lastIndex
------
> result = re.exec(z)
[ ______, _____, ______, index: __, input: 'dBdxdbbdzdbd' ]
> re.lastIndex
------
> result = re.exec(z)
[ ______, _____, ______, index: __, input: 'dBdxdbbdzdbd' ]
> re.lastIndex
------
> result = re.exec(z)
-----
```

2. Escriba la expresión regular `r` para que produzca el resultado final:

```
> x = "hello"
> r = /l(___)/
> z = r.exec(x)
[ 'l', index: 3, input: 'hello' ]
```

3. Rellene el valor que falta:

```
> z = "dBdDBBD"
> re = /d(b+)(d)/ig
> re.lastIndex = ________
> result = re.exec(z)
[ 'DBBD',
  'BB',
  'D',
  index: 3,
  input: 'dBdDBBD' ]
```

4. Conteste:

   a) Explique que hace el siguiente fragmento de código:

   ```
   > RegExp.prototype.bexec = function(str) {
   ...   var i = this.lastIndex;
   ...   var m = this.exec(str);
   ...   if (m && m.index == i) return m;
   ...   return null;
   ... }
   [Function]
   ```

   b) Rellene las salidas que faltan:

   ```
   > re = /d(b+)(d)/ig
   /d(b+)(d)/gi
   > z = "dBdXXXXDBBD"
   'dBdXXXXDBBD'
   > re.lastIndex = 3
   > re.bexec(z)
   ------------------------------------------------------
   > re.lastIndex = 7
   > re.bexec(z)
   ------------------------------------------------------
   ```

5. Escriba una expresión JavaScript que permita reemplazar todas las apariciones de palabras consecutivas repetidas (como `hello hello`) por una sóla aparición de la misma

6. ¿Cual es la salida?

   ```
   > "bb".match(/b|bb/)

   > "bb".match(/bb|b/)
   ```

   Justifique su respuesta.

7. El siguiente fragmento de código tiene por objetivo escapar las entidades HTML para que no sean intérpretadas como código HTML. Rellene las partes que faltan.

   ```
   var entityMap = {
       "&": "&___;",
       "<": "&__;",
       ">": "&__;",
       '"': '&quot;',
       "'": '&#39;',
   ```

```
    "/": '&#x2F;'
  };

function escapeHtml(string) {
  return String(string).replace(/_________/g, function (s) {
    return ____________;
  });
```

9. JSON

1. ¿Como se llama el método que permite obtener una representación como cadena de un objeto? ¿Que parámetros espera? ¿Como afectan dichos parámetros?

10. OOP

1. Escriba un código JavaScript que defina una clase `Persona` con atributos `nombre` y `apellidos` y que disponga de un método `saluda`.

11. Code Smelling

1. Defina el término *code smelling*
2. ¿Que diferencia hay entre un *code smell* y un *bug*?
3. Explique el code smell *Duplicated Code*
4. Explique el Switch smell

12. HTML

1. ¿Cual debe ser el valor del atributo `rel` para usar la imagen como favicon?

```
<link rel="_____________" href="etsiiull.png" type="image/x-icon">
```

13. CSS

1. ¿Que hacen las siguientes pseudo-clases estructurales CSS3?

```
tr:nth-child(odd)    { background-color:#eee; }
tr:nth-child(even)    { background-color:#00FF66; }
```

2. ¿Que contiene el objeto `window` en un programa JavaScript que se ejecuta en un navegador?
3. Enumere tres elementos HTML que sean de la categoría `inline`
4. Enumere tres elementos HTML que sean de la categoría `block`
5. ¿Cómo se hace para que elementos de la página web permanezcan ocultos para posteriormente mostrarlos? ¿Que hay que hacer en el HTML, en la hoja de estilo y en el JavaScript?
6. Rellene los estilos para los elementos de las clases para que su visibilidad case con la que su nombre indica:

```
.hidden      { display: ____; }
.unhidden    { display: _____; }
```

14. Mocha, Chai

1. ¿Como creamos el directorio con el esqueleto inicial para las pruebas con mocha?

2. Rellene lo que falta:

```
[~/srcPLgrado/temperature/tests(master)]$ cat tests.js
var assert = chai._____;

suite('temperature', function() {
    test('[1,{a:2}] == [1,2]', function() {
      assert._________([1, {a:2}], [1, {a:2}]);
    });
    test('5X = error', function() {
        original.value = "5X";
        calculate();
        assert._____(converted.innerHTML, /ERROR/);
    });
});
```

3. Este es un fichero `test/index.html` apto para ejecutar las pruebas con Mocha y Chai en la práctica de la Temperatura. Rellene las partes que faltan:

```
[~/srcPLgrado/temperature(karma)]$ cat tests/index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Mocha</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="mocha.css" />
  </head>
  <body>
    <div id="_____"></div>   <!-- para la salida de las pruebas -->
    <input id="original" placeholder="32F" size="50">
    <span class="output" id="converted"></span>

    <script src="________"></script>
    <script src="________"></script>
    <script src="../temperature.js"></script>
    <script>mocha._____('___')</script>
    <script src="tests.js"></script>

    <script>
      mocha.___();
    </script>
  </body>
</html>
```

4. Rellene las partes que faltan del fichero con las pruebas TDD en Mocha y Chai para la práctica de la temperatura:

```
[~/srcPLgrado/temperature(karma)]$ cat tests/tests.js
var assert = ____.assert;
```

```
_____('temperature', function() {

    ____('32F = 0C', function() {
        original.value = "32F";
        calculate();
        assert._________(converted.innerHTML, "0.0 Celsius");
    });
});
```

5. ¿Como puedo ejecutar las pruebas escritas usando Mocha y Chai usando el comando `npm test`?. (no se asume el uso de Karma en esta pregunta) Explique como hacerlo.

15. Gulp

1. Complete las partes que faltan del siguiente `gulpfile.js` en el que se lleva a cabo una tarea para la optimización (uglify/minify) de la aplicación de la práctica de la temperatura:

```
/tmp/pl-grado-temperature-converter(karma)]$ cat gulpfile.js
var gulp    = require('gulp'),
    gutil   = require('gulp-util'),
    uglify  = require('gulp-uglify'),
    concat  = require('gulp-concat');
var minifyHTML = require('gulp-minify-html');
var minifyCSS  = require('gulp-minify-css');

gulp.____('minify', function () {
  gulp.___('temperature.js')
  .____(uglify())
  .___(gulp.____('minified'));

  gulp.__('./index.html')
    .___(minifyHTML())
    .___(gulp.___('./minified/'))

  gulp.__('./*.css')
   .___(minifyCSS({keepBreaks:true}))
   .___(gulp.___('./minified/'))
});
```

16. npm y package.json

1. ¿Con que comando creo el fichero `package.json`?

2. Explique en consiste el versionado semántico/semantic versioning. ¿Cual es el nombre en inglés de los tres números de version? ¿Como cambian?

3. ¿Que se guarda en el campo `"dependencies": {}` de `package.json`?

4. ¿Que opción debo añadir al comando `npm install` para que la librería instalada se añada como dependencia en el fichero `package.json`?

5. ¿Que se guarda en el campo `"devDependencies": {}` de `package.json`?

6. ¿Que opción debo añadir al comando `npm install` para que la librería instalada se añada como `"devDependencies"` en el fichero `package.json`?

7. Explique que significan en los objetos que describen las dependencias dentro `package.json` las siguientes notaciones:

   a) `*`

   b) `latest`

8. ¿Cual es la salida? ¿Como actúa el operador `~`?

```
> var semver = require('semver')
undefined
> semver.toComparators('~1.2.3')
[ [ '_______', '______' ] ]
```

9. ¿Cual es la salida? ¿Como actúa el operador `^`?

```
> var semver = require('semver')
undefined
> semver.toComparators('^1.2.3')
[ [ '_______', '______' ] ]
```

17. Karma

1. Explique como funciona Karma

2. ¿Con que comando puedo crear el fichero de configuración de Karma?

3. ¿Que debemos poner en la entrada `frameworks` de karma para el ejemplo de la temperatura?

```
    frameworks: ['_____'],
```

4. La librería/plugin `karma-mocha` provee el adapter de Karma para Mocha. ¿Como le pasamos opciones para configurar Mocha desde Karma? Rellene las partes que faltan:

```
client: {
  args: ['--grep', 'pattern'], // solo pruebas que casan con pattern
  mocha: {
    __: '___'
  }
},
```

5. Explique que debe ponerse (y que no) en la sección `files` del fichero de configuración de Karma ¿Donde son cargados dichos ficheros?:

```
    files: [ ... ],
```

6. Los preprocesadores en Karma nos permiten procesar los ficheros en `files` antes de que sean cargados en el navegador.

```
preprocessors = {
  '**/*.coffee': 'coffee',
  '**/*.html': 'html2js'
};
```

¿Que hace el preprocesador `html2js`? ¿Que hace el preprocesador `coffee`?

7. Complete la función `setup` de las pruebas de la práctica de la temepratura con Mocha, Chai y Karma:

```
setup(function(){
   if (typeof ________ !== 'undefined') {
       document.body.innerHTML = ________['tests/test.html'];
       original = document._____________('original');
       converted = document._____________('converted');
   }
 });
```

   ¿Como se llama la variable en la que se guardan los HTML de los ficheros cargados en los navegadores?

8. ¿Que es PhantomJS? ¿Como funciona?