Universidad de La Laguna. Escuela Técnica Superior de Ingeniería Informática
Tercero del Grado de Informática
DESARROLLO DE SISTEMAS INFORMÁTICOS: 2ª PARTE
04/04/2017 8 páginas

Nombre: _____
Alu: _____ GitHub Id: _____ GitHub Team: _____

Preguntas de Repaso de Heroku

1. Una vez instalado el Heroku cli 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?

10. Explique los pasos para desplegar una aplicación en Heroku

11. Explique como resolver los problemas que pueden surgir cuando la aplicación desplegada en Heroku no funciona correctamente

Preguntas de Repaso de Expresiones Regulares

1. Escriba expresiones regulares que casen con las siguientes especificaciones:

    1. car and cat
    2. pop and prop
    3. ferret, ferry, and ferrari
    4. Any word ending in ious
    5. A whitespace character followed by a dot, comma, colon, or semicolon
    6. A word longer than six letters
    7. A word without the letter e

2. Escriba una expresión regular que reconozca las cadenas de doble comillas. Debe permitir la presencia de comillas y caracteres escapados.

3. Escriba una expresión regular que reconozca los números en punto flotante (por ejemplo `-2.3e-1`, `-3e2`, `23`, `3.2`). numbers = /^ ... $/, matching exacto

4. Escriba una expresión regular que case con los números no primos expresados en unario. Pruebe con `1111`, `111`, `111111`, `1111111`, ...

5. Escriba una expresión regular que case con los comentarios JavaScript.

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

7. ¿Cual es la salida?

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

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

    Justifique su respuesta.

8. 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 ____________;
  });
```

2. Se quiere poner un espacio en blanco después de la aparición de cada coma:

```
> x = "a,b,c,1,2,d, e,f"
'a,b,c,1,2,d, e,f'
> x.replace(/,/g,", ")
'a, b, c, 1, 2, d,  e, f'
```

   Se pide que si hay ya un espacio después de la coma, no se duplique.

3. Se pide una expresión regular que case con expresiones del tipo `identifier = number` y retorne con cada paréntesis el identificador y el número. Pruebe con `h = 4`, `temp = 5.6`, `x23= -2.3e1` y `z += 3`

4. Imagine you have written a story and used single quotation marks throughout to mark pieces of dialogue. Now you want to replace all the dialogue quotes with double quotes, while keeping the single quotes used in contractions like *aren't*. Think of a pattern that distinguishes these two kinds of quote usage and craft a call to the replace method that does the proper replacement.

```
var text = "'I'm the cook,' he said, 'it's my job.'";
// Change this call.
var result = text.replace(/.../g, '...');
console.log(result);
var expected = `"I'm the cook," he said, "it's my job."`;
if (expected === result) console.log("OK")
else console.log("ERROR!");
```

Preguntas de Repaso de Cookies

1. Defina y explique el concepto de Cookie
2. Defina y explique el concepto de Cookie para la Gestión de una Sesión
3. Defina y explique el concepto de Cookie para la Autenticación en un Website
4. ¿Que es un secure cookie? (cookie seguro)

5. ¿Como se llama el middleware express que me facilita el manejo de los cookies? ¿Cual es el código para poner en funcionamiento dicho middleware?
6. Dentro de un middleware express ¿Cómo se llama el método del objeto `res` que me permite establecer un cookie? ¿Que argumentos recibe?
7. ¿Que método del objeto `request` me permite ver los cookies establecidos?
8. ¿Como borro un cookie en el servidor express?
9. ¿Que es la Query String? Ponga un ejemplo de Query String
10. If a form is embedded in an HTML page as follows:

```
<form action="/hello" method="get">
  <input type="text" name="first" />
  <input type="text" name="second" />
  <input type="submit" />
</form>
```

and the user inserts the strings `this is a field` and `was it clear already` in the two text fields and presses the `submit` button, the handler of the route `/hello` (the route specified by the `action` attribute of the form element in the above example) will receive a query string. Write down the query string it receives

Rutas en Express

1. Escriba un servidor que sirva ficheros estáticos desde el directorio `/public`
2. El servidor deberá responder a requests `GET` en las rutas `/user/nombredeusuario` (donde `nombredeusuario` varía) con una página que diga `Hola nombredeusuario` usando el método `render` del objeto `response`

- La página se elaborara con una vista que debe estar en el directorio `views/` usando el motor de vistas `ejs`
- La página elaborada en la respuesta tendrá un tag `img` referenciando a una imagen que está en `public/`

3. Escriba un middleware que intercepte en las rutas `/user/nombredeusuario` y que vuelque en la consola información sobre el request: (por ejemplo los atributos `method`, `path`, etc.)
4. Explique como se puede aislar el código anterior en un fichero `routes/user.js` que sea cargado desde el programa principal
5. Explique que hay que hacer para desplegar la aplicación en la máquina virtual del iaas

Preguntas de Repaso de Pruebas con Mocha, Chai y Should

1. ¿Como creamos el directorio con el esqueleto inicial para las pruebas con mocha?
2. En este ejemplo se usa Chai assert. Rellene lo que falta en estas pruebas del código del conversor de temperatura:

```
var assert = chai._____;

suite('temperature', function() {
    test('[1,{a: 2}] == [1,{a: 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 en un navegador. Rellene las partes que faltan
    - Sugerencias: El id es el usado por mocha para producir su salida de las pruebas, es necesario cargar `chai` y `mocha` y establecer el estilo de pruebas (`tdd`, `bdd`, etc.) y por útlimo ejecutar `mocha`

```
[~/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.

6. El siguiente ejemplo corresponde al ejemplo de pruebas que vimos para la renderización de una tabla correspondiente al capítulo *The Secret Life Of Objects* que usa `mocha`y `should`. Rellena las partes que faltan:

```
________("drawTable", function() {
  __("must draw the checkerboard correctly", function() {
    /* There are 5 columns and 5 rows and a white space between columns*/
    drawTable(checkerboard()).should._____(/^(([# ]{2}(\s|$)){5}){5}$/);
  })
});
```

Preguntas de Programación Orientada a Objetos

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

2. Escriba una clase `Programador` que hereda de `Persona`
3. Escriba un código ECMA6 que defina una clase `Persona` con atributos `nombre` y `apellidos` y que disponga de un método `saluda`.
4. Usando ECMA6 escriba una clase `Programador` que hereda de `Persona`
5. Explique las diferencias en la salida entre este código

```
function Person() {
  this.age = 0;

  function growUp() {
    this.age += 10;
  }
  growUp();
  console.log(this.age);
}
var p = new Person();
```

y este otro:

```
function Person() {
  this.age = 0;

  var growUp = () => {
    this.age += 10;
  }
  growUp();
  console.log(this.age);
}

var p = new Person();
```

Justifique su respuesta.

2. Explique que hacen los métodos `bind`, `apply` y `call` y cuales son sus similitudes y diferencias
3. ¿Cual es el significado del primer argumento del método `Object.cretate`? ¿Y el segundo?
4. Todo objeto JavaScript tiene una propiedad `"prototype"` ¿verdadero o falso?
5. La propiedad `prototype` de una función es un objeto de tipo `Function` ¿verdadero o falso?

6. El `prototype` de una función es un objeto de tipo `Function` ¿verdadero o falso?
7. ¿Cual es el problema con este código? ¿Como se arregla el problema?

```
Object.prototype.nonsense = "hi";
for (var name in map)
  console.log(name);
```

8. ¿Que significa que una propiedad es no-enumerable?
9. ¿Como puedo crear un objeto que carezca de prototipo?

10 El argumento `descriptor` del método

```
Object.defineProperty(obj, prop, descriptor)
```

puede ser de uno de dos tipos: **data descriptors** o **accessor descriptors**.

- Un `data descriptor` es una propiedad que tiene un `value`, que puede o no ser `writable`.
- Un `accessor descriptor` es una propiedad que describe un par de funciones getter-setter.

Un descriptor puede ser de uno de estos tipos pero no puede ser ambos.

Explique cuales de estas propiedades pertenecen a que tipo, cual es su valor por defecto y que describen:

1. `configurable`
2. `enumerable`
3. `value`
4. `writable`
5. `get`
6. `set`