Pregunta A Interrogación IIC2513 2021-1

SECCIÓN: Fundamentos

Solución

Para esta pregunta usted deberá elegir **solamente 5 (cinco) preguntas de entre las 8 de este listado de preguntas**.

Si por alguna razón usted responde más de 5 preguntas, sólo se corregirán cinco escogidas arbitrariamente.

Se especifica un máximo de líneas para su respuesta. Se les pide que respeten ese límite. Si se extienden, la pregunta puede considerarse mala.

P1.- Dado el siguiente trozo de código, ¿por qué al realizar http://127.0.0.1:3000, el retorno es un código de error 404? (2 líneas máx.)

```
const Koa = require('koa');
const app = new Koa();

const port = 3000;
app.listen(port, () => {console.log(`Aplicacion atendiendo
en puerto ${port}`)});
```

Solución:

Porque no existe ningún middleware agregado a la aplicación Koa **[1 pt]**. Por defecto, si ningún middleware setea ctx.body o cambia el status code, entonces Koa responde con 404 **[1 pt]** (referencia: https://koajs.com/#response-status).

P2.- Dado el siguiente trozo de código, ¿por qué la salida de console.log(a) es de esa forma? (3 líneas máximo.)

```
a = { llave1: "1",
      llave2: "2"
    };
b = a;

b.llave1 = "Hola Mundo";

console.log(a);
```

Solución:

En la línea “b = a” lo que se está haciendo es asignarle a “b” el mismo valor al que apunta “a”. Ese valor es un objeto, que corresponde a un valor no primitivo (mutable) **[1 pt]**. Entonces al cambiar la propiedad “llave1” desde “b”, se está mutando el mismo objeto al que apunta “a” **[1 pt]**.

Nota: es válido también utilizar conceptos de paso por valor y referencia.

P3. ¿Cuál es la diferencia entre = = y = = = ? (3 líneas máx.)

Solución:

“**==**” es el operador de igualdad, el cual, en el caso de comparar valores de distinto tipo, intenta convertir el tipo de un valor al del otro **[1 pt]**, y eso genera algunos casos contraintuitivos. En cambio, “**===**” compara igualdad estricta, considerando igualdad de tipos también **[1 pt]**.

P4. Explique el resultado que se muestra en los console.log del siguiente código (1 línea máx. por cada console.log (2 en total))

```
a = 1;
b = “1”;
console.log(a - b);
console.log(a + b);
```

Solución:

En el primero, al ser valores de distinto tipo, se convierte “b” a número y se restan ambos, obteniendo 0 **[1 pt]**. En el segundo, se convierte “a” a string, en cuyo caso el operador suma es considerado una concatenación, resultando en “11” **[1 pt]**.

P5.- En el código a continuación, se tiene un arreglo de números *array*, cuyos números (contenido) se quieren imprimir en el mismo orden en que se ingresan al definir el arreglo. Para ello, se usa la función **`printNumber`**, pero todo fracasa ya que los números se imprimen siempre en orden ascendente. ¿A qué se debe ese comportamiento? ¿Podría arreglarlo? ¿Cómo o por qué no? Para esta última pregunta, considere que NO puede intervenir el contenido del arreglo ni alterar el setTimeout de la función

(3 líneas máx. para explicar el comportamiento y 2 líneas máx. para explicar si lo podría arreglar y cómo, o bien, por qué no)

```
printNumber = (number) => {
  setTimeout( () => { console.log(number) },
number*1000);
```

```
}

const array = [3, 7, 5, 1, 2];

array.forEach(element => printNumber(element));
```

Solución:

Se debe a que printNumber llama a la función setTimeout, que es asíncrona y cuyo callback realiza el console.log. Sin embargo, se esperará al menos la cantidad de segundos especificada en “number” antes de ejecutar console.log, lo que causa que siempre sea ascendente **[1 pt]**.

No se puede arreglar sin intervenir los elementos restringidos, debido a que las llamadas a printNumber llamarán inevitablemente a setTimeout con los segundos especificados por number **[1 pt]**.

Nota: existe un fix algo rebuscado, que también se puede considerar válido:

Se podría tener un array “buffer” inicializado al principio como vacío, dentro de printNumber agregar “number” a “buffer” y retornar inmediatamente (así no se ejecuta setTimeout), y finalmente hacer sort de buffer e imprimir cada elemento en consola **[1 pt]**.

P6.- Explique qué hace el siguiente trozo de código, con n un número entero (3 líneas máx.)

```
const digitize = n => [...`${n}`].map(i => parseInt(i))
```

Solución:

Primero separa los dígitos del “n” en un arreglo de strings, utilizando el spread operator **[1 pt]**, y luego para cada elemento, los convierte a enteros, retornando un arreglo de números enteros correspondientes a los dígitos de “n” **[1 pt]**

P7.- ¿Qué significa -en forma precisa- cuando se habla de que se está usando ES6? (2 líneas máx.)

Solución:

Significa que se está utilizando la versión de ECMAScript lanzada en el año 2015 **[1 pt]**, la cual introdujo features relevantes como clases, keywords const y let, destructuring, spread operator **[1 pt]**. ES6 suele también referirse a cualquier versión desde 2015 en adelante

P8.- Comente sobre la veracidad de la siguiente afirmación "*HTML es un lenguaje de programación complementario a JavaScript*" (2 líneas máx.)

Solución:

HTML no es un lenguaje de programación (a diferencia de JavaScript) **[1 pt]**, sino que un lenguaje de marcado que permite representar contenido en la Web, marcándolo para darle significado **[1 pt]**.

Nota: sí podría ser considerado complementario a JavaScript, pues es necesario que exista un documento HTML para manejar el DOM asociado. Asignar **0.5 pts** si sólo se menciona esto.

Entrega:
Debe entregar un único archivo PDF indicando las preguntas (P1, P2, etc.) con sus respectivas respuestas. El archivo PDF debe estar dentro del archivo .zip que se especifica en el enunciado general de la interrogación.