ACTIVIDADES 2 – UT5 ALMACENAMIENTO INFO. JAVASCRIPT – Arrays JS

NORMAS GENERALES PARA LAS ACTIVIDADES

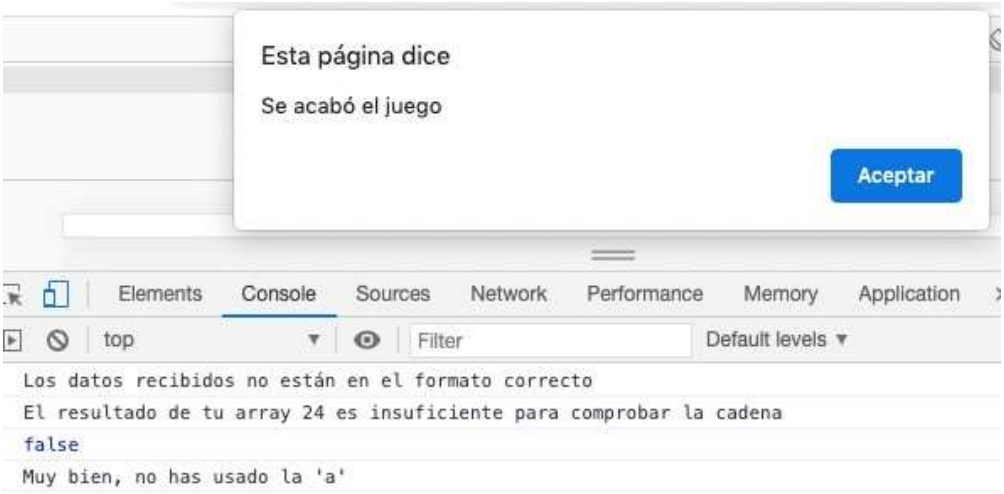
- Condiciones de entrega
  - Se entregarán en la fecha indicada en la plataforma. No se admitirán ejercicios entregados después de esa fecha.
  - La entrega de todas las actividades se hará a través de GitHub y AULES.
  - En GitHub, crea un repositorio LM si no lo tienes ya y subirás en él un directorio que deberá nombrarse con el nombre y primer apellido del alumno seguido de la frase "-actividades2UT5". *El nombre y los apellidos deben ir separados por una mayúscula*. En AULES, entrega un enlace a ese repositorio (si tienes problemas también se acepta un zip con la nomenclatura descrita anteriormente con todos los .html y los .js necesarios con la solución de los ejercicios).
- Condiciones de corrección
  - Las actividades se deben realizar con un editor (por ejemplo: Visual Studio Code, Atom, Notepad++)
  - Se deben entregar los ficheros .html y .js que se generen.
  - Si se detecta copia en alguna actividad se suspenderá automáticamente la unidad de didáctica a todos los alumnos implicados.
  - Si se detecta copia de alguna página web de internet, automáticamente se suspenderá la actividad copiada.
- Calificación
  - Existen cuatro actividades. Todas las actividades tienen la misma puntuación.
  - Las actividades se puntuarán dentro del apartado de procedimientos que es un 15% de la nota de la unidad.

EL CÓDIGO DEBE SER LO MÁS COMPACTO POSIBLE : la idea de estas actividades es que las realicéis evitando usar demasiados bucles y usando en su lugar métodos de la API de JavaScript para que el código quede lo más compacto y óptimo posible y se aprendan a utilizar las características propias del lenguaje.

Ejercicio 1. Crea una función que reciba una cadena, un booleano, una función y un array numérico por parámetro. Si el tipo de algún parámetro no es el esperado debes imprimir un error (suponemos que si nos pasan un array sí que en todas sus posiciones habrá un dato numérico). Si es lo esperado, si el valor del booleano es *true*, recorre el array con la estructura *foreach*. Si el producto de todos los ítems en mayor a 100 entonces comprueba si en la cadena hay alguna "a". Si hay alguna a muestra por pantalla un mensaje diciendo que la "a" no está permitida. Si no la hay, muestra por pantalla un mensaje diciendo, "Muy bien, no has usado la 'a'". En caso de que el producto del array fuera menor de 100 informa al usuario: "El resultado de tu array es insuficiente para comprobar la cadena". Si el valor del booleano es *false* entonces ejecuta la función recibida por parámetro.

Ejemplo:

```
Ejercicio1()
Ejercicio1("hola mundo",true,[1,2,3,4],()=>{alert("Se acabó el juego")})
Ejercicio1("Sí",true,[10,20,30,40],()=>{alert("Se acabó el juego")})
Ejercicio1("Sí",false,[10,20,30,40],()=>{alert("Se acabó el juego")})
```



Documentos aparte, añado comentarios en el código.

Ejercicio 2. Crea una función llamada verAsignaturas. Esta función va a recibir un número indeterminado de alumnos. De cada alumno va a recibir un array. En ese array estará almacenado el nombre, el curso y las asignaturas de las que está matriculado (una asignatura en cada posición). Muestra por pantalla el nombre del alumno – el curso – asignaturas: y el nombre de las asignaturas separadas por un /. Si el número de datos de alumnos es 0 debes mostrar la cadena "No hay datos para mostrar". Debes usar el operador rest, la desestructuración de arrays y el código lo más compacto posible.

Ejemplo:

```
Ejercicio2(["Sara", "DAMA", "Programación", "ED"],["Martín", "DAMB", "Programación", "LM", "ED", "BBDD", "FOL", "SI"],["Emma",
"ASIR","ISO","BBDD","LM"])
Ejercicio2(["Álvaro","Semi","BBDD"])
Ejercicio2()
```

```
Sara-DAMA-asignaturas:Programación/ED                    ejercicio2.js:10
Martín-DAMB-asignaturas:Programación/LM/ED/BBDD/FOL/SI   ejercicio2.js:10
Emma-ASIR-asignaturas:ISO/BBDD/LM                        ejercicio2.js:10
Álvaro-Semi-asignaturas:BBDD                             ejercicio2.js:10
No hay datos para mostrar                                ejercicio2.js:13
```

Documentos aparte, añado comentarios en el código.

Ejercicio 3. Pide datos al usuario y crea un array insertando los datos al principio. Los datos tienen que ser de tipo numérico, si introduce uno que no sea de tipo numérico dejarás de pedir datos (puedes utilizar isNaN(*caracter*) que te devuelve un booleano indicando si el carácter definido es un número o no). Si estamos ante un número de vez (iteración) par en la que pedimos datos al usuario, utiliza el operador + para convertir a número, si no conviértelo mediante el constructor Number(). Seguidamente, ordena el array de mayor a menor conservando del array original sólo los múltiplos de 3 (usa el método filter). Muestra ambos por pantalla. Por último, muestra los dos arrays por pantalla.

Documentos aparte.

Ejercicio 4. Realiza con funciones el siguiente ejercicio:

Supón que trabajas en un pequeño negocio familiar de imprenta. El gerente te pide que escribas un programa que calcule el salario semanal de una serie de trabajadores. Para ello tendrás un array y en cada posición del array los datos de cada trabajador: Un empleado introducirá el nombre del trabajador, el número de horas que ha trabajado (no están permitidas las horas extras) y el precio hora que cobra el trabajador. Tu

programa deberá calcular el Impuesto de Hacienda que se le retiene (un 20 por ciento del salario bruto) y el Impuesto de la Seguridad Social (un 8 por ciento del salario bruto).

El programa deberá mostrar en líneas separadas la siguiente información: el nombre del trabajador, el salario bruto, la cantidad retenida para el pago del Impuesto de Hacienda, la cantidad correspondiente al pago del impuesto de la Seguridad Social y el salario neto del trabajador. Recuerda, cuanto más comprimido el código, mejor.

Documentos aparte.