Módulo 6 - Algoritmos - Ejercicios para Practicar

UTILIDADES

Antes de empezar con los ejercicios, una pequeña explicación. En el codigo de estos ejercicios encontrarás lo siguiente:

Crear un array vacío de longitud n:

```
var myEmptyArray = new Array(3); // Array vacio de 3 posiciones.
console.log(myEmptyArray); // [empty × 3]
```

Rellenar todas las posiciones de un array con el mismo valor:

```
var myArray = new Array(3); // Array vacio de 3 posiciones.
myArray.fill("Hola");
console.log(myArray); // ["Hola", "Hola", "Hola]
```

En una sola línea equivaldría a:

```
var myArray = new Array(3).fill("Hola");
console.log(myArray); // ["Hola","Hola","Hola"]
```

Agenda

Vamos a implementar una agenda para un equipo de personas, de modo que nos indique que horas tienen libres y que horas están ocupados. Además, vamos a generar un algoritmo que haga una búsqueda en un equipo y determine cual es la primera hora en que todos los miembros del equipo están libres, para asi poder establecer una reunión. Es decir, que busque el primer hueco disponible en sus agendas.

Interfaz

Para este ejemplo usaremos la consola así que no necesitaremos interfaz gráfica.

Datos

Te proponemos que partas de las siguientes estructuras de datos:

```
// Constantes
var WORK_HOURS = [
  "08:00 - 09:00",
  "09:00 - 10:00",
  "10:00 - 11:00",
  "11:00 - 12:00",
  "12:00 - 13:00",
  "13:00 - 14:00",
  "15:00 - 16:00",
  "16:00 - 17:00"
];

// Datos
var myTeam = [
  {
    name: "María",
    availability: new Array(8).fill(true)
  },
  {
    name: "Pedro",
    availability: new Array(8).fill(true)
  },
  {
```

```
    name: "Esther",
    availability: new Array(8).fill(true)
  },
  {
    name: "Marcos",
    availability: new Array(8).fill(true)
  },
];
```

`WORK_HOURS`

Como ves, `WORK_HOURS` es un array que representa las posibles franjas horarias de cada miembro del equipo. En total son 8 (8 horas laborables) y si te fijas se ha excluido la franja de 14h a 15h ya que se reserva para el almuerzo. Este array de horas laborables podría servirte para mostrar por consola cada una de las franjas de cada miembro del equipo y si está libre o no.

`myTeam`

El equipo lo vamos a representar con un array de miembros llamado `myTeam`. Dicho array contiene objetos que representan a cada miembro, y que tienen 2 propiedades:

- El nombre del miembro (propiedad `name`).
- La disponibilidad de esa persona (propiedad `availability`).

La disponibilidad de cada miembro (`availability`) se ha modelado como un array de 8 posiciones (8 franjas horarias en total) que por defecto se ha inicializado a `true` para todas las franjas horarias. Es decir, inicialmente, todos los miembros están disponibles todas las horas.

Algoritmo

Vamos a dividirlo en 2 apartados:

1. Generación aleatoria de la disponibilidad

Como primer apartado, vamos a generar aleatoriamente la disponibilidad para todos los miembros del equipo. Se trata de recorrer todos los miembros del equipo, y a su vez, para cada miembro, todas las franjas horarias de su disponibilidad, e ir asignando aleatoriamente si está disponible o no en dicha franja.

De esta forma generamos un equipo con una agenda completamente aleatoria.

Sugerencia

Una vez hayas generado tu agenda aleatoria, muestrala por consola. Un ejemplo de salida por consola podría ser este:

```
Disponibilidad de María
    08:00 - 09:00: Si
    09:00 - 10:00: Si
    10:00 - 11:00: Si
    11:00 - 12:00: Si
    12:00 - 13:00: No
    13:00 - 14:00: No
    15:00 - 16:00: Si
    16:00 - 17:00: Si
Disponibilidad de Pedro
    08:00 - 09:00: No
    09:00 - 10:00: No
    10:00 - 11:00: Si
    11:00 - 12:00: No
    12:00 - 13:00: Si
    13:00 - 14:00: Si
    15:00 - 16:00: Si
    16:00 - 17:00: Si
Disponibilidad de Esther
    08:00 - 09:00: Si
    09:00 - 10:00: No
    10:00 - 11:00: Si
    11:00 - 12:00: Si
    12:00 - 13:00: No
    13:00 - 14:00: No
    15:00 - 16:00: Si
```

```
    16:00 - 17:00: Si
Disponibilidad de Marcos
    08:00 - 09:00: Si
    09:00 - 10:00: No
    10:00 - 11:00: No
    11:00 - 12:00: Si
    12:00 - 13:00: Si
    13:00 - 14:00: No
    15:00 - 16:00: Si
    16:00 - 17:00: No
```

2. Buscar hueco libre

Para buscar el primer hueco libre habrá que comprobar la primera franja horaria en la que todos los miembros del equipo están disponibles.

Sugerencia

Muestra por consola la franja horaria encontrada donde hay hueco, si es que lo hay, y en caso de que no exista hueco, infórmalo también. Por ejemplo:

```
// Ejemplo de hueco disponible
Hueco encontrado en el horario 15:00 - 16:00

// Ejemplo de hueco no existente
Lo siento. No hay hueco disponible en el equipo.
```

Calculadora de cambio óptimo de billetes y monedas

Vamos a implementar una calculadora de cambio óptimo en base a un importe de compra y la cantidad entregada por el cliente.

Interfaz

Debemos permitir que el usuario introduzca una cantidad con el importe total de la compra y una cantidad con el dinero que nos entregan.

- Input numérico para el importe total.
- Input numérico para la cantidad que se entrega.
- Botón de calcular.

Al presionar el botón calcular, un algoritmo debe procesar los cálculos necesarios para que devuelva un resultado con el número de billetes y monedas necesarias para hacer la devolución.

Algoritmo

Dentro de nuestra tienda, podremos dar cambio de billetes de 5 €, 10 €, 20 €, 50 €, 100 € y 200 € (no daremos cambio de 500 €). Asímismo, tendremos monedas de 1, 2, 5, 10, 20 y 50 céntimos; y 1 y 2 euros.

Pensemos cómo haríamos si estuviéramos a cargo de la caja de nuestra tienda. Si el importe de la compra fuese 152 euros y el cliente nos pagase con un billete de 200 euros. ¿Qué deberíamos hacer?

- Primero calculamos la diferencia para saber cuánto debemos devolver: `48 euros`.
- Seguidamente, intentaríamos dar cambio con billetes de 200 euros: `importe devolución / 200 = 0.24`.
- Vemos que NO podemos dar billetes de 200 euros (parte entera de 0.24 igual a 0).
- Lo intentamos con el siguiente billete, 100 euros. `importe devolución / 100 = 0.48`.
- Vemos que NO podemos dar billetes de 100 euros (parte entera de 0.48 igual a 0).
- Seguimos intentándolo con el resto de billetes, repitiendo los pasos hasta que damos con el correcto, 20 euros.
- Calculamos el cambio con billetes de 20 euros: `importe devolución / 20 = 2.4`.
- Vemos que podemos dar 2 billetes de 20 euros, así que lo restamos del importe de la devolución `importe devolución = importe devolución - 2 * 20`.
- Tenemos como importe actualizado 8 euros.

- Repetimos los pasos con el resto de billetes y monedas.

Al final del algoritmo debe quedarnos la cantidad de billetes y monedas de cada tipo que debemos devolver:

- 2 billetes de 20 euros.
- 1 billete de 5 euros.
- 1 moneda de 2 euros.
- 1 moneda de 1 euro.

Challenge

Vamos a complicar nuestro problema. Ahora debemos hacer el cálculo según los billetes y monedas que tengamos en nuestra caja.

Para ello necesitaremos de alguna forma, indicar qué cantidad de billetes y/o moneda tenemos disponibles.

Calculadora de sueldo neto

Vamos a implementar una calculadora de sueldo neto.

Interfaz

Para ello, debemos permitir que el usuario introduzca el sueldo bruto anual. Por tanto, nuestra **interfaz gráfica** puede ser tan sencilla como:

- Input numérico para el sueldo bruto.
- Botón de calcular.

Al presionar calcular, un algoritmo debe procesar el sueldo bruto introducido y devolver un informe de sueldo neto.

Algoritmo

Para calcular el salario neto a partir de una cantidad bruta anual, ten en cuenta las siguientes premisas:

- Al sueldo bruto hay que restarle los 2 impuestos principales por los que se ve gravado en nuestro pais: Seguridad Social e IRPF.
- **Seguridad Social**. Simplificando, diremos que todo trabajador debe abonar el 6,35% de sus ingresos brutos en concepto de seguridad social, con una base máxima anual de cotización de 45014.4€.
  - Esto significa, que si el salario bruto sobrepasa dicha base máxima de cotización, solo se tiene en cuenta dicha base máxima para el cálculo del impuesto.
  - Ejemplo: un salario de 60.000€ o de 100.000€ pagarán la misma cantidad de Seguridad Social, correspondiente al 6.35% de la base máxima (45014,4€).
- **IRPF**. Aunque este impuesto depende de multitud de factores, situaciones familiares, minusvalias, ascendiente y descendientes, etc., vamos a simplificar su cálculo del siguiente modo:
  - La base imponible sobre la que se calcula el impuesto consiste en restar a los ingresos brutos:
    - Seguridad Social.
    - 2.000€ en concepto de gastos generales deducibles.
  - Una vez obtenida la base, el impuesto se calcula en tramos:
    - Tramo 0: Los primeros 5.550€ no tributan (0%). Se considera el mínimo personal y familiar.
    - Tramo 1: Desde esos 5.550€ del tramo anterior hasta los 12.450€ se aplica un gravamen del 19%.
    - Tramo 2: Desde 12.450€ hasta 20.200€ el porcentaje asciende al 24%.
    - Tramo 3: Desde 20.200€ hasta 35.200€ se cotiza al 30%.
    - Tramo 4: Desde 35.200€ hasta 60.000€ abonamos el 37%.
    - Tramo 5: Finalmente, toda cantidad restante que supere los 60.000€ esta gravada al 45%.
- Una vez calculado los 2 impuestos, muestra un escueto informe al usuario que presente:
  - Salario bruto anual.
  - Cantidad destinada a Seguridad Social.
  - Cantidad destinada a IRPF.
  - Salario neto anual.

- Salario neto mensual (en 12 o 14 pagas).

(*) Los datos para el cálculo de Seguridad Social e IRPF aqui presentados están extraidos para la campaña de 2018. Se podrían revisar y actualizar para futuras campañas.