1 Parte I - Linguagem

Esse conjunto de questões avalia noções da sintaxe e conceitos do JavaScript.

1.1 Tarefa 1 - Variáveis e Funções

Defina as seguintes variáveis e funções:

- ”fib” : Uma função que toma como argumento um inteiro ”n” e retorna o ”n”-ésimo número de fibonacci (fib(1)=fib(2)=1, fib(3)=2 etc)
- ”i” : Um inteiro com valor 1984
- ”v” : Um vetor de inteiros contendo os valores (nessa ordem) 0,1,2 e 3.
- ”caneta” : Um objeto com as propriedades/métodos
  - ”cor” : Uma string.
  - ”fabricante” : Outra string.
  - ”escrever” : Uma função que toma como argumento uma string e a escreve no console.

2 Parte II - Detalhes e Especialidades do JS

2.1 Tarefa 1 - Funções anônimas

Em JS, as funções são tratadas como objetos (funções de primeira ordem), mas há situações onde precisamos de referências a funções mas não de seus nomes, como em eventos e callbacks. Nessas situações, podemos usar das chamadas **Funções Lambda** (ou funções anônimas).
Declaramos-nas das seguintes formas:

- `function(argumentos){expressões} ( function(a){return 2*a;} )`
- `argumento=>valor ( a=>2*a )`
- `(argumentos)=>valor ( (a,b)=>a+b )`
- `(argumentos)=>{expressões} ( (a,b)=>{a+=b;return 2*b;} )`

Um exemplo de uso simples: eventos de DOM. Queremos que um elemento responda a um clique com uma mensagem ”Clicado!” no console.
Ao invés de definir uma função externamente e depois definí-la como o que é chamado pelo evento, podemos simplesmente declarar:

```
elemento.onclick = ()=>console.log("Clicado!")
```

No contexto dessa questão, existem funções do tipo

`"Testar_<índice>(função)"`

Que avaliam a corretude de sua função.

Complete o código a seguir (usando funções anônimas) de modo que:

- `A primeira função (argumento de Testar_1) receba um inteiro e retorne seu quadrado`
- `A segunda função receba duas Strings e retorne a primeira concatenada à segunda`
- `A terceira função receba uma função f e retorne o resultado de f(1,2)`

```
Testar_1(|sua função anônima 1|);
Testar_2(|sua função anônima 2|);
Testar_2(|sua função anônima 3|);
```

2.2 Tarefa 2 - Objetos, chaves e comparação

Em JS não podemos comparar diretamente objetos que não sejam strings,números, booleanos ou outros tipos simples/tipos com comparadores definidos. Eis aqui algumas situações contraintuitivas onde isso pode ser um problema:

- `[1,2,3]==[1,2,3] => (false)`
- `a={p:10};b={p:10};a==b; => (false)`

Para resolver esse problema, precisamos comparar todas as propriedades dos objetos que queremos comparar. Para obter essa lista de propriedades, podemos utilizar um loop for(propriedade in objeto) ou a função Object.keys(objeto).
Implemente uma função ”Comparar(a,b)” que cheque se **b** possui **todas** as propriedades de a **com os mesmos valores**.
Importante: Se um dos membros de ”a” for um Objeto (e o membro existir e for um objeto em ”b”), sua função deve comparar os dois objetos também.

3 Parte III - Aplicações, DOM e JQuery

Esse conjunto de questões avalia o entendimento e solução de requisitos (e questões menos diretas).

3.1 Tarefa 1 - Classes

Edson pretende criar um novo serviço de streaming de música. Após consultar alguns sites, ele decidiu fazê-lo em JavaScript. Edson, porém, é somente um visionário, não programador; assim, ele delegou a você a importante tarefa de começar o sistema.
Defina uma classe/função construtora ”Musica”, que possua as seguintes propriedades/comportamentos

- ”total” : Número total de músicas (Musica.total)
- Suas instâncias (new Musica()) devem possuir:
  - ”nome” : Nome da música
  - ”tags” : Um vetor de strings
  - ”compos” : Nome de quem compôs
  - ”id” : Um número identificador único (desde que seja único, pode gerá-lo de qualquer forma)
  - ”tocado” : Um número que começa nulo
  - ”tocar” : Um método que incrementa a propriedade ”tocado”
- Toda vez que uma nova instância é criada, ”total” deve ser incrementado
- O construtor deve tomar como argumentos ”nome”,”tags”, ”compos” (nessa ordem)

3.2 Tarefa 2 - Loops

Edson não ficou satisfeito com sua classe; afinal, ele não é programador e na prática tudo o que seus objetos ”Musica” fazem é contar o número de vezes que foram ”tocados”.
Para provar que sua estrutura é útil, implemente uma função ”selecionar_por_tag” que tome como argumentos ”tags” (uma lista de strings) e ”musicas”, uma lista de objetos ”Musica”.
Sua função deve retornar uma lista/vetor dos objetos ”Musica” que possuem todas as tags ”tags” em sua lista de tags (propriedade ’tags’).

Dica: Vetores possuem métodos (como indexOf, find e findIndex) para buscar índices de entradas, além de outros como filter e map que facilitam o processamento. Use-os, assim não é necessário implementá-los.

3.3 Tarefa 3 - DOM Básico

Edson ainda não está satisfeito. Ele é uma pessoa bastante visual e não valoriza tanto um programa que não mostra nada na tela.
Faça uma função ”mostrar_por_tag” que tome como argumento ”musicas”, uma lista de músicas.
Sua função deve

- Capturar o valor (value) digitado pelo usuário no elemento com id ”entrada”.
- Buscar as músicas na lista ”musicas” que possuam a tag especificada pelo usuário (utilizando sua função selecionar_por_tag).
- Limpar o conteúdo da div ”saida”.
- Gerar, para cada música, uma div com fundo da cor ”#101010”, texto de cor ”#f0f0f0” e com conteúdo bruto (innerHTML) sendo o nome da música.
- Adicionar cada div criada à saída (div ”saida”).

O seu programa tem acesso às funções definidas nas questões anteriores para facilitar a implementação (você pode usar sua selecionar_por_tag)

3.4 Tarefa 4 - Teste final/JQuery

Edson finalmente está convencido de sua capacidade para desenvolver e lhe cedeu agora a tarefa de trabalhar num pequeno protótipo.
Seu primo Nelson já escreveu uma página simples para o serviço e Edson insiste que você não faça alterações diretas nela, mesmo que seu design seja questionável.
Sua tarefa é implementar as seguintes funcionalidades com JS+JQuery:

- Encontrar automaticamente quais são os filtros existentes (baseado no innerHTML dos membros da div ”col”).
- Guardar uma lista de filtros que estão ativados.
- Quando o usuário clicar em um dos filtros de tag:
  - O botão do filtro deve ser destacado ativando a classe ”ativo” se estiver ativo e a classe deve ser removida se não estiver ativo.
  - Se o filtro estiver desativado, ele deve ser ativado e vice-versa.
  - Atualizar as músicas que estão sendo mostradas (vide próximo ítem).
- Esconder as músicas que não possuem **todas** as tags listadas nos filtros ativos.
- Todos os filtros devem começar desativados.

As divs de música possuem id=”m”+(identificador único da música)
Existe uma variável global chamada ”músicas”, um vetor que contém objetos ”Música”

Dicas:

- Use os seletores do JQuery ($(’{query}’)) ao invés do DOM puro.
- Você pode fazer sub-seleções no JQuery com $(’{query1}’).find(’{query2}’).
- $(’id’).click(função) !== document.getElementById(’id’).onclick=função.
- $(’{query}’).addClass(”classe”) e $(’{query}’).removeClass(”classe”) existem.
- Use o inspetor de página do seu navegador para analisar a estrutura da página criada por Nelson.

Caso você não tenha acesso à página do corretor automático, o código de Nelson está disponível a seguir:

```
<!--HTML do Nelson-->
<!--CSS fictício do Nelson-->
    <style>
        .container{
            width:90%;
            height:50vh;
            position:relative;
            left:50%;
            transform:translate(-50%);
            border: solid black 1px;
        }
        .head{
            width:100%;
            height:10%;
            position:absolute;
            top:0;
            left:0;
            text-align:center;
            background:black;
            color:white;
            display:flex;
            align-items:center;
        }
        .col{
            width:20%;
            height:90%;
            position:absolute;
            left:0;
            top:10%;
            background: grey;
        }
        .show{
            width:80%;
            height:90%;
            position:absolute;
            left:20%;
            top:10%;
            overflow-y:scroll;
            overflow-x:hidden;
        }
        .name{
            color:white;
            width:100%;
            text-align:center;
        }
        .autor{
            color:lightgrey;
            width:100%;
        }
        .autor:before{
            content:"By: ";
        }
        .song{
            height:32%;
            width:32%;
            background:#9f719f;
            border-radius:9%;
            display:inline-grid;
            align-items:center;
            border:solid white 1px;
```

```
        }
        .filtro{
            width:96%;
            position:relative;
            left:50%;
            transform:translate(-50%);
            border-radius:9%;
            color:white;
            background:black;
            border:solid white 1px;
            text-align:center;
            transition:all .5s;
        }
        .ativo{
            color:black;
            background:white;
            border:solid black 1px;
        }
        //Isso aqui é para facilitar a marcação dos filtros selecionados
        //Isso é, caso alguém faça isso algum dia, né.
        //Assinado: Nelson.
    </style>
    <div class="container">
            <div class="head">
                    <pre style="width:100%">Nelsongs/Edsonic</pre>
                </div>
                <div class="col">
                    <div style="width:100%;background:darkgrey;text-align:center">
                        Filtros
                    </div>
                    <div class="filtro ativo" id="World">World</div>
                    <div class="filtro" id="Bossa">Bossa</div>
                    <div class="filtro" id="Jazz">Jazz</div>
                    <div class="filtro ativo" id="Misc">Misc</div>
                    <div class="filtro" id="Brazil">Brazil</div>
                    <div class="filtro" id="Classic">Classic</div>
                </div>
                <div class="show" id="show">
                    <div id="m91" class="song">
                        <div class="name">As músicas aparecerão</div>
                        <div class="autor">Eu</div>
                    </div>
                    <div id="m37" class="song">
                        <div class="name">Durante o teste do código.</div>
                        <div class="autor">Eu</div>
                    </div>
                    <div id="m38" class="song">
                        <div class="name">Os elementos existirão</div>
                        <div class="autor">Eu</div>
                    </div>
                    <div id="m54" class="song">
                        <div class="name">Antes do início da execução.</div>
                        <div class="autor">Eu</div>
                    </div>
                    <div id="m42" class="song">
                        <div class="name">(O gerador usa suas respostas)</div>
                        <div class="autor">Eu</div>
                    </div>
            <div>
    <\div>
```