# Intensivo JavaScript – Fullture

Aula 02

**Temas**: resolução das tarefas; classe Map, iteração de listas com map, filter e reduce

## Resolução das tarefas da aula 01

**Tarefa 01)** Continuar o último código, adicionando função para percorrer a lista de objetos, contando quantos temas há por data.

```
const temasPorData = () => {
    let map = new Map()
    for(let item of lista){
        if(map.get(item.dia)){
            let antigo = map.get(item.dia)
            map.set(item.dia, [antigo, item.tema].flat())
        }else{
            map.set(item.dia, [item.tema])
        }
    }
    map.forEach((valor, chave) => {
        console.log(`Dia ${chave}: ${valor.length} tema(s)`)
    })
}
```

**Tarefa 02)** Implementar função temasLength, que utilize map e reduce.

```
const temasLength2 = () => {
    let aux1 = lista.map((obj) => obj.tema.length)
    let len = aux1.reduce((str, total) => str + total)
    console.log(`temasLength versão 02: ${len} caracteres.`)
}
```

**Tarefa 03)** Corrigir o botão de atualização da tabela, para deixar apenas uma única vez o conteúdo da lista ser adicionado na tabela do DOM.

```
const removeTr = () => {
    var tabela = document.querySelector('table');
    while(tabela.childNodes.length>2){
        tabela.removeChild(tabela.lastChild)
    }
}

const mapTable = () =>{
    removeTr() //remove todas as linhas antes de adicionar
    lista.map((obj) => addTr(obj))
}
```

# Aula 02

Considere os arquivos index.html e base.js disponíveis em:
https://drive.google.com/file/d/12_KICahzV71PTcWeu7IwaFvxsXiwk5sY/view?usp=sharing

**Questão 01)** Na função temasPorData, qual a necessidade de utilizar a função flat?

A. Adicionar novo elemento ao vetor.
B. Adicionar ao map sem sobrescrever o valor antigo.
C. Evitar criação de vetores n-dimensionais para visualização e acesso.
D. Evitar criação de vetores n-dimensionais para manter a contagem correta.

**Resposta**: C

**Questão 02)** Qual a necessidade do if/else na função temasPorData?

A. Remover o elemento antigo.
B. Permitir criar apenas uma lista por data.
C. Manter a lista de elementos da data (item.dia) atualizada.
D. Não é necessário, já que tanto no if quanto else é feito map.set.

**Resposta**: C

**Questão 03)** Na função removeTr por que todas as linhas são removidas?

A. Porque estava assim no Google.
B. Porque para usar o lista.map é necessária essa abordagem.
C. Porque seria inviável analisar linha por linha o que já foi plotado.
D. Porque é mais fácil apagar tudo para mapTable plotar todas novamente.

**Resposta**: D

**Questão 04)** Na função mapTable por que é utilizada a função map?

A. Porque é a forma certa de iterar uma lista.
B. Porque todas as linhas foram removidas do DOM antes.
C. Para simplificar a iteração da lista, sem usar o for ou foreach.
D. Para transformar no objeto Map, que evitará duplicar valores.

**Resposta**: C

**Questão 05)** O que se obtém como resultado de [...map] na função temasPorData?

A. Uma lista de map.
B. O elemento da última posição do map.
C. O elemento da primeira posição do map.
D. Uma lista com os elementos destruídos do map.

**Resposta**: D

**Questão 06)** Na função temasPorData há impressão dos valores do map. Sobre isso, assinale a alternativa correta:

A. Nenhuma das alternativas.
B. A função console.table poderia apresentar melhor os valores.
C. A função não poderia ser refatorada para apenas retornar o map.
D. Caso a função não fizesse a impressão, o javascript nunca executaria.

**Resposta**: A

**Questão 07)** Qual a vantagem de apagar todas as linhas TR do DOM e plotar novamente?

A. Menor uso de memória.
B. Menor uso de processador.
C. Maior compatibilidade com diferentes navegadores.
D. Mais fácil controlar o estado dos dados que interessam.

**Resposta**: D

**Questão 08) Para discussão em grupo:** O que poderia ser feito com algo similar a temasLength2? O que poderia ser feito com *map* e *reduce*, que seja mais útil?

## Tarefas

**Tarefa 01)** Simplifique a função temasPorData. Uma possibilidade é remover o if/else e fazer a chamada de map.set apenas uma vez. **Dica**: é possível sempre inserir um array na posição da chave, algo equivalente ao que fazemos atualizando a lista no localStorage.

**Tarefa 02)** Refatore a função temasPorData para:

1. Receber o objeto map por parâmetro;
2. Crie o map na função submissao, para passar como argumento da temasPorData;
3. Receber um objeto com tema e dia por parâmetro;
4. Adicionar apenas o objeto passado por parâmetro no map, ao invés de iterar a lista;
5. Retornar o map atualizado.

**Tarefa 03)** Atualize a função mapTable para receber o map por parâmetro, e atualizar o DOM pelo map. **Dica**: não existe a função map no objeto Map, mas o forEach pode ser usado no lugar.

**Tarefa 04)** Verifique se todas as vezes que a lista era chamada foram substituídas pelo map. Caso não tenha sido, passe a utilizar o objeto map ao invés da lista, fazendo as devidas adequações. Após verificar, remova todas as ocorrências da lista global, que já deve ter sido substituída pelo map, que é instanciado na função submissao.

**Tarefa 05)** Armazene o Map em localStorage a cada vez que for atualizado. Verifique que consegue obter o Map de forma correta, ao obter do localStorage. **Dica**: é necessário utilizar a destruição do objeto antes de armazenar no localStorage. Ou seja, o que deve ser armazenado no localStorage é: JSON.stringify([...map])

**Tarefa 06)** Crie uma função para retornar ou um Map vazio ou o Map armazenado em localStorage, já convertido para objeto Map. **Dica**: é necessário construir o objeto com algo como: map = new Map(JSON.parse(<variável obtida do localStorage>))

**Tarefa 07)** Crie uma função chamada init, que obtém o Map utilizando a função da Tarefa 06 e atualiza o DOM, assim que a página é aberta.

**Tarefa 08)** Formate os valores apresentados na coluna tema, no DOM, para que se tenha espaços entre as vírgulas e a próxima palavra. **Dica**: é necessário converter o “value” do Map para string usando a função .toString().