Imersão Dev - Aula 5:

Rafaella: Olá, Dev! Estou aqui ansiosíssima para te dar as boas vindas ao quinto dia de Imersão Dev. Ontem aprendemos alguns conceitos importantíssimos para qualquer desenvolvedor e desenvolvedora que deseja trabalhar na área. Estamos prontos para poder aprofundar um pouco mais.

Hoje veremos algo que vai mudar a sua vida na programação e eu tenho certeza que você vai utilizar muito no futuro. Eu não queria dar muitos spoilers, mas o projeto está sensacional. Não esqueça que no final desta aula você já pode postar o seu projeto no GitHub! Se você não conhece a ferramenta, deixaremos um conteúdo extra para você conhecer. Você também pode compartilhar conosco em suas redes sociais, como LinkedIn ou Instagram e não esqueça de nos marcar para acompanharmos os avanços.

Participe da nossa comunidade! Se você tiver qualquer dúvida, jogue lá. Se quiser responder dúvidas, faça isso. Uma das melhores formas de aprendermos é ensinando e compartilhando nossos resultados. Teremos um canal específico para isso. Pegue o seu café, abra seu CodePen e bora para a quinta-aula.

Até o momento, vimos diversas coisas, como a variavel `if` e agora `for`, como na aula passada. Nessa aula vamos rever todos esses conceitos e fazer um projeto muito legal com tudo isso, além de alguns conteúdos novos.

Guilherme: Eu já estou com o modelo base que vamos utilizar. Nessa aula vamos fazer algo incrível, criaremos um Aluraflix numa versão 2.0 . A ideia é: temos um campo onde vamos inserir o endereço da imagem que queremos inserir em nosso catálogo e um botão para adicionarmos o filme. O HTML e o CSS já foram feitos pelo para focarmos totalmente no JavaScript.

Para isso, vou dar uma “garfada” no projeto do Fork no canto inferior. Estou logado com a minha conta e faço uma cópia desse projeto para minha conta @guilimadev. Vamos começar a botar a mão na massa aqui e desenvolver esse nosso projeto. A primeira coisa que temos que ter em mente é que quando eu clico no botão "Adicionar Filme" eu quero uma determinada ação. Vejamos como ele está no HTML. Temos o botão e tem a

função `onClick` e a função `adicionarFilme()`, que o pessoal que desenvolveu o layout já colocou para nós.

Então eu vou criar uma função no JavaScript para que quando eu clique nesse botão, uma mensagem seja exibida no console para garantirmos que a ação está acontecendo de fato.

Vou colocar aqui `function adicionarFilme()` e um `console.log` com a mensagem `“clicou”` .

```
function adicionarFilme() {
        console.log("clicou")
}
```

Guilherme: Em “Settings” vou adicionar as tags "aluraflix, imersãodev, alura". Não deixe de colocar essas tags para conseguirmos ver esses seus projetos também e não esqueça de compartilhar nas suas redes sociais o que você tem feito marcando sempre a Alura, eu, a Rafa e o Paulo.

Salvei aqui o projeto, vou rodar. Como não temos nenhuma ação ainda para o campo não vou dar ênfase nele, vou clicar no “Adicionar Filme” e aparece o “clicou”. Maravilha!

Rafa, o que podemos fazer na sequência, porque vimos que podemos trabalhar com listas então não faz muito sentido criarmos uma variável para cada filme que queremos adicionar. O que podemos fazer?

Rafaella: Então, podemos criar uma lista, mas primeiro acho que podíamos ir guardando todas as informações que vamos digitando no campo do input.

Guilherme: Vamos lá.

Rafaella: Vamos salvando as informações que estamos colocando. Vamos pegar em nosso HTML o id para fazermos o `getElementById` do nosso input. Qual o id dele?

Guilherme: id="filme"

Rafaella: Então vamos fazer aquele `getElementById` do `id="filme"`. Vamos criar uma nova variável para a gente guardar informação do que estamos colocando aí dentro. Criamos um `var` nova que pode ser `campoFilmeFavorito`.

Rafaella: E dentro vamos fazer o `getElementById` e o `id="filme"`. Aí não vai ser só a tag porque quando pegamos aquele `getElementById` nós puxamos a tag inteira e quando damos `console.log` vemos que na verdade é o input e não é isso que queremos, queremos apenas o valor que é o tal do "value" então podemos colocar `.value` no final.

```
function adicionarFilme() {
        var campoFilmeFavorito =
document.getElementById("filme").value;
        console.log("clicou");
}
```

Rafaella: Perfeito. Então assim nós já pegamos as informações que estamos recebendo no nosso input e está passando para o `campoFilmeFavorito` e além disso precisamos além de puxar e guardar essa informação imprimir na tela. Gui, como fazemos para imprimir na tela? Podemos criar uma outra função, por exemplo, que seja talvez listar na tela, o que você acha?

Guilherme: Acho que inicialmente podemos realizar todas essas operações, depois vamos quebrando o nosso código. Algo que vai ser muito comum: escrevemos um determinado código, um programa e temos certeza que esse programa e esse código vão mudar. E uma das formas que temos de garantir que essa mudança seja sustentável é deixar claro que essa função que adiciona o filme tem um determinado comportamento e outras funções terão outros comportamentos.

Primeira coisa, vamos só garantir que estamos pegando o nome do filme certinho. Vou colocar o campoFilmeFavorito, salvo e executo. Como estamos testando, não vou colocar o endereço de uma imagem de um filme mesmo. Vou escrever "Está funcionando " e clicar para adicionar. Está funcionando, beleza. O campoFilmeFavorito está pegando exatamente o conteúdo que queremos ali.

```
function adicionarFilme() {
        var campoFilmeFavorito = document.getElementById("filme").value;
        console.log(campoFilmeFavorito)
}
```

Guilherme: Agora queremos pegar esse conteúdo do campo do filme favorito, da imagem, e exibir aquela imagem que colocamos em nossa lista direto no JavaScript. Mas em qual lugar do JavaScript queremos mostrar isso? Utilizamos na aula passada uma tag img em que nós passávamos o caminho. Eu vou fazer o mesmo passo, pode ser, Rafa?

Rafaella: Pode ser.

Guilherme: Vou criar uma nova variável chamada elementoFilmeFavorito eu vou armazenar aquela nossa tag img. Vamos escrever o `src` passando o caminhoFilmeFavorito que queremos colocar ali. Vou concatenar com o nome, com o campo que a Rafa fez para nós e vou só fechar a nossa tag img com o sinal de maior.

```
function.adicionarFilme() {
        var campoFilmeFavorito = document.getElementById("filme").value;
        var elementoFilmeFavorito = "<img src" + campoFilmeFavorito + ">"
}
```

Rafaella: Como vimos na aula anterior, é exatamente a mesma coisa que estamos fazendo para criar o nosso elemento de imagem e adicionar no HTML.

Guilherme: Isso aí. Agora, o que eu preciso fazer é o seguinte: Onde eu quero exibir esse filme favorito? Existe um local em nosso HTML que deixamos separado para exibir a nossa lista de filmes. Vamos dar uma olhada?

```
<div id="listaFilmes"></div>
```

Guilherme: Scrolando aqui, reparem que temos uma `div` que é como vimos lá no comecinho é uma parte, uma divisão dos nosso HTML, então vai ter essa `listaFilmes` e dentro dela quero colocar todas as capas dos filmes preferidos que vamos adicionando em nosso projeto. Vou copiar ela para garantirmos o mesmo nome e recuperar esse valor. Quero pegar esse valor `listaFilme`, então vai chamar `var listaFilmes` e vou pegar esse valor dele com `documet.getElementById()` e aqui eu passo o `listaFilmes`.

```
function adicionarFilme() {
        var campoFilmeFavorito = document.getElementById("filme").value;
        var elementoFilmeFavorito = "<img src=" + campoFilmeFavorito +">"
        var listaFilmes = document.getElementById("listaFilmes")
}
```

Guilherme: O que eu preciso fazer agora é colocar dentro de `listaFilmes` o `filmeFavorito`, esse `elementoFilmeFavorito` com a imagem, com o campo da imagem que temos certinho. Para isso, vamos usar o `innerHTML` O que era mesmo o `inner`, Rafa?

Rafaella: Para relembrarmos o que tínhamos feito, colocamos dentro daquele elemento que pegávamos e era interno. Nós tínhamos a tag h2, por exemplo, que que usamos e

colocava o `innerHTML` e a informação que queríamos colocar dentro daquela tag. Então dessa vez vamos fazer a mesma coisa, vamos usar o mesmo innerHTML .

Guilherme: Então chamamos aqui essa variável de elementoListaFilmes. Vou pegar essa variável e eu quero colocar um determinado elemento, um conteúdo que criamos so `elementoFilmeFavorito` dentro dele. Para isso, vamos usar o `innerHTML`. E vou falar que ele vai ser igual ao `elementoFilme`. Pode ser, Rafa?

```
function adicionarFilme() {
        var campoFilmeFavorito = document.getElementById("filme").value;
        var elementoFilmeFavorito = "<img src=" + campoFilmeFavorito +">"
        var listaFilmes = document.getElementById("listaFilmes")
        elementoListaFilmes.innerHTML = elementoFilmeFavorito
}
```

Rafaella: Pode ser. Então você vai colocar dentro do `elementoListaFilmes` o `elementoFilmeFavorito`, é isso?

Guilherme: É isso aí. Vamos pegar um filme, então? Falamos bastante de Harry Potter, então vou pegar ele.

Paulo: Não, vamos pegar um outro. Esse já era na aula passada, agora vamos pegar outro.

Guilherme: Fala aí, Paulo, um filme.

Paulo: Matrix.

Guilherme: Beleza, Matrix. Vamos pegar um poster, copiar o endereço da imagem e voltamos. Vamos executar o nosso projeto, salvei, executei e , colocamos em nosso

espaço em branco com “jpg” no final, “Adicionar Filme”. Pronto, aparece Matrix. Vamos pegar outro filme, Rafa?

Rafaella: Vamos

Guilherme: Pode falar.

Rafaella: Shrek.

Guilherme: Shrek. Escolho um pôster, clico com o botão direito na imagem e copio o endereço dela e repetimos o processo. Aqui temos algo interessante, quando adicionamos um filme, seria legal se depois que adicionamos ele, esse conteúdo, esse campo, ficasse vazio.
Rafaella: Isso é verdade. Podemos fazer daqui a pouco isso.

Guilherme: Vamos fazer. Enfim, vou colocar o Shrek e clico em adicionar. Aconteceu alguma coisa, Rafa. Ele adicionou o Shrek meio que em cima do Matrix. Isso acontece porque toda vez que o botão “Adicionar Filme” for pressionado, executamos a função que pega o conteúdo do campo, criamos a tag `elementoFilmeFavorito`, vai até `elementoFilmeFavorito` , que é onde queremos de fato colocar esses filmes para serem exibidos na tela. Depois temos `elementoFilmes` e vamos pegar esse elemento do HTML. Então temos uma tag que sinalizamos o filme favorito e quando clicamos novamente não estamos fazendo uma atribuição baseada nesse elemento lista em que vamos adicionar vários filmes favoritos.

O que precisamos fazer, portanto, é muito similar ao índice que tínhamos quando usamos o ++, ou seja, pegar o valor do índice e somar um. Teremos o mesmo comportamento aqui. Então eu vou pegar o conteúdo que já está no `elementoListaFilmes.innerHTML` e vou somar com o conteúdo que está ali.

```
function adicionarFilme() {
      var campoFilmeFavorito = document.getElementById("filme").value;
      var elementoFilmeFavorito = "<img src=" + campoFilmeFavorito +
">";
      var elementoListaFilmes =
```

```
document.getElementById("listaFilmes");
      elementoListaFilmes.innerHTML = elementoListaFilmes.innerHTML +
elementoFilmeFavorito;
}
```

Guilherme: Para o elementoFilmeFavorito, vou pegar os conteúdos que já existem, ou seja, os filmes que já estão presentes e colocar o filme que eu estou criando agora. Só por teste, vou colocar duas vezes o Shrek que eu já tenho aqui, pode ser?

Rafaella: Pode. Gui, só uma coisa, é um pouco diferente nesse caso do índice porque no índice podíamos fazer aquele índice ++ , mas só podemos fazer o ++ quando você está, literalmente, adicionando uma unidade. Aqui, por exemplo, se colocássemos ++ não ia ficar legal. Na verdade, estamos adicionando, somando, com outra coisa, com elementoFilmeFavorito então por isso que precisamos escrever dessa forma mais extensa.
Guilherme: Muito boa colocação. Continuando, vou adicionar o Shrek aqui novamente e agora sim, se isso aqui ficar adicionando aqui, vão aparecer vários Shrek em nossa página. Isso ficou legal, já é um começo, né Rafa?

Rafaella: Nós podíamos limpar agora então o campo, né? Toda vez que adicionamos ele limpa, o que você acha? Para isso, vamos utilizar aquela variável que fizemos com o `getElementById(“filme”)` e vamos tornar o valor dela, ou seja, estamos com a variável `campoFilmeFavorito` com esse valor e podemos transformar esse valor em zero. Na verdade não é zero, é vazio.

Para fazermos isso, vamos colocar o campoFilmeFavorito.value que queremos transformar e vamos igualar ao vazio. Colocamos duas strings e nada dentro delas. É como se quiséssemos limpar, se escrevêssemos "a" dentro das string, colocaríamos "a" mas não é isso que queremos, queremos realmente alterar o valor desse input do nosso campo para nada, não ter nada escrito.

```
function adicionarFilme() {
```

```
        var campoFilmeFavorito = document.getElementById("filme").value;
        var elementoFilmeFavorito = "<img src" + campoFilmeFavorito + ">";
        var elementoListaFilmes = document.getElementById("listaFilmes");
        elementoListaFilmes.innerHTML =
                elementoListaFilmes.innerHTML + elementoFilmeFavorito;
}
```

Paulo: Bom, para nós limparmos essa parte com toda vez que alguém adicionar, queremos limpar só para deixar mais redondo, né? Então podemos pegar esse elemento que é esse tal de input text que é o elemento que tem o id `filme` e alterar. Dessa vez eu não quero pegar o `value` dele, eu quero alterá-lo.

Lembra que na linha número dois fizemos o `document.getElementById("filme").value`? Agora vamos fazer a mesma coisa, vamos pegar essa linha e alterar e fazer igual a `""` , que é o igual a nada. Não é bem nada, porque nada pode ser zero, pode ser *undefined* pode ser nulo e tantos outros. Nesse caso, é outra coisa, estamos pegando um elemento do HTML e alterando o valor dele para essas duas aspas `""`.

```
function adicionarFilme() {
        var campoFilmeFavorito = document.getElementById("filme").value;
        var elementoFilmeFavorito = "<img src" + campoFilmeFavorito + ">";
        var elementoListaFilmes = document.getElementById("listaFilmes");
        elementoListaFilmes.innerHTML =
                elementoListaFilmes.innerHTML + elementoFilmeFavorito;
        document.getElementById("filme").value = "";
}
```

Guilherme: Pode ser um texto aqui, Paulo?

Paulo: Pode ser. É que aí ele vai quebrar a imagem, mas pode ser.

Guilherme: Só agora.

Paulo: As pessoas já devem ter testado isso, né? Se você colocar uma imagem que não é um endereço de imagem correta, ele vai dar uma imagem quebrada porque o seu carregador tentou carregar a imagem nesse endereço que não existe. Mss repare que depois fica aquele textinho esbranquiçado que chamamos de placeholder, aquele texto é como se ele tivesse zerado e deixado o valor padrão daquele campo HTML e com isso deixamos mais redondinho a experiência das pessoas que usam o Aluraflix.

Rafaella: Exato. Se você quiser, você vai olhar ali no HTML e vai achar esse placeholder que está escrito acima. Mas o valor dele, aquele que quando nós realmente escrevemos fica mais escuro, deixamos zerado e é isso que queríamos mesmo.

Guilherme: Acho que isso já responde a outra pergunta. Olha que interessante. Aqui eu venho no filme do Shrek coloco "Copiar endereço da imagem", se eu coloco o endereço certinho e dou um “Adicionar”, ele adiciona legal, temos aquela imagem quebrada então aqui está funcionando. Mas se eu coloco qualquer endereço de imagem que não é um .jpg ou .png e dou adicionar, ele coloca um quadradinho de uma imagem quebrada, isso não está legal. Vamos melhorar isso, Rafa? O que podemos fazer?

Rafaella: É como colocamos lá o source que você tinha escrito. Aquela tag img que nós fizemos quando passamos o `src` de source ele espera uma imagem porque a tag é img, se passarmos qualquer outra coisa ele quebra, não tem jeito. Então podemos fazer uma validação, podemos usar uma condição: se a url não terminar com .jpg, damos uma mensagem de erro e se funcionar, imprimimos a imagem.

Guilherme: Vamos lá. Então depois que pegamos o valor da imagem acho que essa validação aconteceria na linha três. Isso seria um *if*?

Rafaella: Exatamente. Aí colocamos o filmeFavorito

Guilherme: Aqui seria campoFilmeFavorito, né?

Rafaella: Isso.

Guilherme: Acho que até poderíamos mudar para `filmeFavorito`, o que vocês acham, faz sentido?

Paulo: Faz. Nsse caso, em específico, não é o campo, porque já pegamos o `.value`. Você tem razão, é até melhor. Perfeito, bem observado, Guilherme.

Guilherme: Boa.

```
function adicionarFilme() {
        var filmeFavorito = document.getElementById("filme").value;
        if (filmeFavorito)
        var elementoFilmeFavorito = "<img src=" filmeFavorito + ">"
        var elementoListaFilmes = document.getElementById("listaFilmes");
        elementoListaFilmes.innerHTML =
                elementoListaFilmes.innerHTML + elementoFilmeFavorito;
        document.getElementById("filme").value = ""
```

Rafaella: Existe também um método que podemos utilizar para verificarmos dentro de uma string, algum texto. Aquela variável do tipo que nós colocamos as aspas, se ela termina com alguma coisa ou não. E é literalmente isso que vamos escrever. Se o `filmeFavorito` terminar com, ou seja, `endsWith()` e passamos o valor `.jpg`. Vamos abrir as chaves e colocar os comandos que queremos executar. Se a nossa string, que no caso é a url, terminar com jpg vamos fazer todos os comandos que tem abaixo. Se tiver certo, nós vamos listar.

```
function adicionarFile() {
      var filmeFavorito = document.getElementById("filme").value;
      if(filmeFavorito.endsWith(".jpg")) {
            var elementoFilmeFavorito = "<img src=" + filmeFavorito +
">";
            var elementoListaFilmes =
document.getElementById("listaFilmes");
```

```
            elementoListaFilmes.innerHTML =
                  elementoListaFilmes.innerHTML +
elementoFilmeFavorito;
      }
      document.getElementById("filme").value = "";
}
```

Vocês viram que o Gui deixou aquele negócio de deixarmos em branco fora do *if*? Então isso será executado se entrar ou não no *if*, porque mesmo que esteja errado ele quer limpar o campo.

Além do *if*, vamos colocar um *else*, para, caso a imagem que a pessoa entrou - a URL que a pessoa entrou - não seja .jpg, consigamos avisar que a imagem não é .jpg e que o endereço do filme é inválido. Então, colocamos um console.error(). Nós temos o tipo console.log(), que vai imprimir a mensagem no console, e, também, um formato interessante, o console.error, que além de imprimir a mensagem que queremos passar dentro dos parênteses, imprime em um formato de erro, na cor vermelha.

Isso é bem interessante quando queremos testar se algo está correto ou não, já que fica mais evidente onde está o erro. Imprimimos no console para verificarmos se algo está errado e escrevemos "endereço de filme inválido”, console.error("Endereço de filme inválido"). Vamos ver como está no console.

Guilherme: Nós começamos com inválido ou válido o nosso teste?

Rafaella: Vamos fazer um válido e depois erramos. E já podemos abrir o console para verificarmos se aparecerá alguma coisa.

Guilherme: Vamos rodar. Já abri o console, tem uma *defined* de algum momento em que testamos. Temos o .jpg, vamos adicionar e aparecerá o Shrek. Tudo certo. Agora faremos o mesmo endereço e tiraremos apenas o final. Pode ser, Rafa?

Rafaella: Pode.

Guilherme: Então, tirarei o .jpg e ficará uma *string* qualquer sem ele. Vamos apertar o botão adicionar e aparecerá "Endereço de filme inválido". Ficou simples e interessante. Nós conseguimos garantir que o nosso projeto não terá imagens quebradas. Eu só estou incomodado com uma coisa. Paulo e Rafa, vocês que já desenvolvem, acham que essa estrutura de código está fácil de ser mantida? Ou poderíamos isolar todas as ações, dividir. O que vocês acham?

Rafaella: Eu estava pensando nisso também, Gui, porque, querendo ou não, como havia comentado, nós não estamos apenas adicionando um filme, por exemplo, adicionamos e ele aparece na tela, são duas coisas: nós estamos guardando a informação e, depois, estamos imprimindo na tela. São duas ações diferentes que podemos dividir em blocos de códigos, funções, que chamamos quando queremos executar.

Paulo: Nós já estudamos um pouco de *function*, porque precisamos lidar com o evento - disparo de evento - que algumas pessoas chamam de *callback,* a chamada de volta para algo que aconteceu. Nós escrevemos várias *functions* - abre parêntese, fecha parêntese, abre chaves - sem saber direito o que é. As *functions* são trechos de código executados em algum momento, quando alguém chama.

Quando nosso código começa a ficar grande, eu acho que é dessa sensação de que vocês estão falando, e ele faz muitas coisas, é responsável por diversas coisas e começa a ter muitas variáveis, *ifs, fors,* entre outros, nós costumamos, para atribuir uma única responsabilidade para cada trecho de código, ir quebrando.

Não há regras de quando exatamente devemos quebrar um código em partes menores, e nós já comentamos sobre isso, a Rafa mostrou duas vezes, em uma delas, nós tínhamos três variáveis para fazer a mesma coisa e depois fizemos tudo em uma linha só. Então, quando quebramos em mais linhas, em blocos, fazemos mais *ifs* e *elses* para deixar mais legível? Essa percepção vem com o tempo.

No começo, nós pecamos pelo excesso, deixamos tudo muito "linguição e macarrônico". Em um segundo momento da nossa carreira de Dev, pecamos por outro excesso, queremos deixar tudo perfeito, super organizado, quebrado em mil módulos diferentes e isso gera outro problema em que, a todo momento, precisamos nos situar sobre o que está acontecendo no código.

Portanto, considero uma boa ideia essa que vocês estão sugerindo: como poderíamos quebrar esse código em várias partes? Vale notar que não é necessário, nesse caso do "Adicionar filmes", quebrar em mais de um bloco. Nós vamos aprender, porque usaremos em outros casos.

Guilherme: Dessa reflexão sobre a quebra do código em partes, eu diria que se trata de uma boa prática de programação. Considero que esse é o ponto principal.

Rafaella: Sim, é uma boa prática. Então, vamos fazer isso: separar em adicionar filmes e listar filmes na tela, duas coisas diferentes que queremos fazer. Primeiro, podemos criar essa nossa nova função. Nós já temos o adicionarFilme .

```
function adicionarFilme()
 var filmeFavorito = document.getElementById("filme").value;
 if (filmefavorito.endsWith(".jpg")) '{
```

```
    var elementoFilmeFavorito = "<img src=" + filmeFavorito + ">";
    var elementoListarFilmes = document.getElementById("ListarFilmes");

   elementoListaFilmes.innerHTML =
    elementoListaFilmes.innerHTML + elementoFilmeFavorito;
 } else {
   console.error("Endereço de filme inválido");
 }
 document.getElementById("filme").value = "";
}

function ListarFilmesNaTela() {

}
```

E podemos criar uma nova função chamada listarFilmesNaTela(). Vamos colocar os parênteses, abrir as chaves.

Nós estamos validando se o filme terminará com .jpg. Se ele termina com .jpg, nós listaremos na tela. Tudo que vem dentro dessa validação, nós poderíamos fazer nessa função listarFilmesNaTela(). Por exemplo, nós estamos pegando a *tag* de imagem, o getElementById para colocar em um lugar específico. Então, vamos copiar essas três linhas de código - na verdade, são quatro, uma está quebrada - colar e separar em duas funções diferentes: adicionarFilme() e listarFilmesNaTela().

```
function ListarFilmesNaTela() {
var elementoFilmeFavorito = "<img src=" + filmeFavorito + ">";
    var elementoListarFilmes = document.getElementById("ListarFilmes");

   elementoListaFilmes.innerHTML =
    elementoListaFilmes.innerHTML + elementoFilmeFavorito;

}

Para conseguirmos chamar o listarFilmesNaTela(), dentro do if, vamos exexutar essa
função que será listarFilmesNaTela().

 if (filmefavorito.endsWith(".jpg")) '{
  listarFilmesNaTela()
```

Vamos recapitular o que está acontecendo. Quando apertamos o botão "Adicionar Filme", pegamos a imagem e a URL que queremos, nós chamamos automaticamente a função adicionarFilme(), porque ela está no nosso *onclick*. Chamando essa função,

criamos uma variável, onde colocaremos o valor que colocamos no nosso campo da URL e, depois disso, validaremos se esse valor termina com .jpg ou não.

Se ele não termina com .jpg, nós daremos uma mensagem de erro e, embaixo, limparemos o que estava escrito no campo. Mas, se termina com .jpg, vamos listar filmes na tela. Então, estamos chamando a nossa segunda função, que fará o trabalho de listar. Um detalhe importante é que as funções são independentes umas das outras, logo, nem tudo que criamos dentro de uma função está sendo criado em outra.

Caso você queira testar, Gui, podemos ver se a variável que está na função adicionarFilme(), a filmeFavorito, consegue ser impressa no listarFilmesNaTela(). Vamos colocar um console.log(filmeFavorito) no listarFilmesNaTela() e imprimir a nossa filmeFavorito, que é a variável criada na outra função. Em seguida, analisaremos se o console.log() sabe o que é essa variável, para isso, precisamos salvar e rodar.

Guilherme: Um exemplo que vai dar certo?

Rafaella: Isso.

Guilherme: Peguei um .jpg, apertei o botão "Adicionar Filme" e estourou.

Rafaella: Recebemos a mensagem "Uncaught ReferenceError: filmeFavorito is not defined". Então, ele não foi definido, mas nós definimos, colocamos em outra função. É por isso que eu digo que as funções têm elementos criados nelas que não aparecem nas outras. Sendo assim, precisamos, ou criar uma nova variável na segunda função com um valor, mas não é o que queremos, pois, teríamos que fazer tudo em função só, enfim, teríamos que criar tudo de novo na mesma função. Ou, podemos passar uma informação de uma função para outra. Como podemos fazer isso?

Nos parênteses em listarFilmesNaTela(), que até agora não utilizamos, podemos passar as variáveis que queremos mandar para outros lugares. No function listarFilmesNaTela(), podemos receber a variável filme. Então, podemos colocar, dentro dos parênteses, filmeFavorito.

Guilherme: Filme favorito ou filme?

Paulo: Vamos começar com filme, Rafa, porque depois explicamos que o nome pode ser o mesmo.

Rafaella: Está bem, então, listarFilmesNaTela(**filme**).

Paulo: Está declarando que precisamos receber alguém que se chama filme. Então, não usamos filmeFavorito e, sim, filme, porque é a variável que está dentro dessa função. Costumamos falar que está no **escopo** dessa função. Deu erro, anteriormente, o filmeFavorito, porque é uma variável que nasceu dentro do adicionarFilme() e morreu

dentro dele, porque foi criada lá. Ela estava no escopo daquela função, não estava no escopo de fora.

Nós até vimos isso, uma variável que apareceu em outra aula e declaramos fora, mas ela estava no escopo das outras. Nesse caso, não, porque foi declarada dentro do adicionarFilme().

Rafaella: Exato! Existe o escopo global e local. O global, é quando declaramos fora das funções. Então, se jogamos o var antes de começar as funções, ele estará disponível para as outras duas funções utilizarem. Mas, como declaramos a variável dentro de uma função, ela não está disponível, está apenas local, portanto, precisamos passá-la por meio do parâmetro, esses parênteses que estamos criando.

Nós colocamos apenas a descrição do que está sendo recebido na nossa função. O listarFilmesNaTela(filme) recebe um valor filme e estamos utilizando esse valor para fazer algumas coisas. Precisamos também enviar essa informação, porque o listarFilmesNaTela() sabe o que quer receber, mas o adicionarFilme() não sabe o que tem que enviar, se é o campo *input*, se é um valor.

Então, quando chamamos a função abaixo, já passamos, nos parênteses, no momento em que estamos executando-a, o que queremos passar. Dentro do if, onde estamos chamando a função listarFilmesNaTela(), vamos colocar, nos parênteses, a variável que queremos mandar para essa função, que é o filmeFavorito.

```
function adicionarFilme()
  var filmeFavorito = document.getElementById("filme").value;
  if (filmefavorito.endsWith(".jpg")) '{
    listarFilmesNaTela(filmeFavorito);
  } else {
```

Vamos testar. Antes de tudo, vamos investigar se está funcionando o que queremos. Gui, acho que você pode colocar um console.log() de novo, agora do filme. O que você acha?

Guilherme: Sim, pode ser, console.log(filme);.

Rafaella: Vamos analisar se o valor que estamos pegando da outra é realmente o que estamos puxando. Agora, precisamos rodar.

Guilherme: Não me lembro se eu executei. Vou rodar mais uma vez, só para ter certeza. Agora sim: .jpg; um caso de sucesso; adicionar; deu certo.

Rafaella: Estamos recebendo, o console.log() está na segunda função, o valor dele estava na primeira e nós conseguimos passar de um para o outro. Um detalhe importante, é que poderíamos escolher o nome que quiséssemos para a variável dentro

dos parênteses na segunda função, a que recebe. Nós escolhemos filme, mas poderíamos ter colocado filmeFavorito, que, na verdade, é o mesmo nome da variável de cima.

Algo que precisamos prestar atenção em relação a essa variável, a esse parâmetro que estamos recebendo, é que tudo que executarmos dentro da função com ele - por exemplo, neste trecho de código, estamos passando o nosso console.log(), ou estamos imprimindo com a *tag* img - nós precisamos referenciar a variável que estamos recebendo, porque se referenciarmos ao nome da variável da outra função, não funcionará de novo.

```
function listarFilmesNaTela(filme) {
  console.log(filme);
  var elementoFilmeFavorito = "<img src=" + filmeFavorito + ">";
  var elementoListarFilmes = document.getElementById("ListarFilmes");

   elementoListaFilmes.innerHTML =
    elementoListaFilmes.innerHTML + elementoFilmeFavorito;
}
```

É como se estivéssemos transformando uma variável em outra para a função abaixo usar essa outra. É bem simples, e, agora, conseguimos dividir o nosso código em duas partes, deixando muito mais perceptível o que cada parte do código está fazendo.

```
function adicionarFilme()
  var filmeFavorito = document.getElementById("filme").value;
  if (filmefavorito.endsWith(".jpg")) '{
    listarFilmesNaTela(filmeFavorito);
  } else {
    console.error("Endereço de filme inválido");
  }
  document.getElementById("filme").value = "";
 }

function listarFilmesNaTela(filme) {
  console.log(filme);
  var elementoFilmeFavorito = "<img src=" + filmeFavorito + ">";
  var elementoListarFilmes = document.getElementById("ListarFilmes");
   elementoListaFilmes.innerHTML =
    elementoListaFilmes.innerHTML + elementoFilmeFavorito;
}
```

Paulo: Rafa, Guilherme, quando eu sugeri, no momento em que vocês foram colocar o entre parênteses, para receber o filme, e a Rafa ia colocar filmeFavorito, é porque pode.

Nós poderíamos, nessa função que se chama listarFilmesNaTela(), receber, ao invés de uma variável que se chama filme, receber uma com o mesmo nome filmeFavorito e não há conflito com a variável declarada mais acima, porque ela está em um escopo diferente.

Eu pedi para colocarmos a variável filme de propósito, para ficar bem compreensível que são coisas diferentes. Porém, se o nome for igual, eles continuariam isolados. É até comum o nome ser idêntico, mas, como já disse, deixamos assim de propósito, para testarmos essas mudanças.

Guilherme: Meu desafio é: nós temos um botão para adicionar um filme, com um campo em que escrevemos, eu gostaria que existisse um jeito de removermos algum filme da tela. Portanto, determinado filme, queremos remover, sendo assim, passamos o endereço desse filme ou algo do tipo e tentamos remover o filme dessa lista.

Paulo: Não é trivial, porque precisamos pegar toda a lista de filmes e o HTML. Fazer isso manualmente não é trivial.

Rafaella: Não é trivial. Boa sorte a quem for fazer esse desafio! O meu, é: além da imagem, colocar também o nome do filme embaixo. Nós estamos pegando apenas a URL, mas seria interessante se pudéssemos criar outro *input*, outro campo, e colocar o nome. Depois, salvar essas informações e imprimir as duas na tela.

Paulo: Esse é muito bacana! Nele, é necessário trabalhar com duas listas ou guardar as duas coisas e colocar um peso, um H2, um *div* para aparecer o nome. Bem interessante.

Paulo: Eu tenho também um desafio: tentar misturar, além de guardar. Quer dizer, nós estamos, a todo momento, inserindo na tela, no HTML mesmo, eu queria guardar também em uma lista, como fizemos em outra aula, guardar em uma lista todos que foram adicionados, simplesmente para ter em dois lugares. Às vezes temos a informação numa estrutura de dados, numa variável e na tela.

O nosso "mostrar tela" poderia mudar. Ao invés de alterar manualmente, com dificuldade, ele percorreria toda a lista com o *for* que aprendemos e vai retornando na tela. A todo momento, ele refaz do zero o componente de lista de filmes e adiciona todas as imagens. Nosso caso é diferente, quando apertamos o botão "Adicionar Filme" ele faz lista de imagens igual a lista de imagens antiga mais a imagem nova.

Outra forma de fazer isso é, toda vez que alguém aperta "Adicionar Filme", simplesmente fazemos .push das variáveis e, no listarFilmes, podemos fazer um *for,* iteramos um *for it*, para quem quiser ir além, iteramos entre todos os elementos da nossa lista e, então, colocamos. Isto é, redesenhamos tudo, todas as vezes. Pode surgir o questionamento se não é muito lento, mas, esse não é o ponto. O importante é a oportunidade de praticar algo diferente.

Sobre os desafios, acho que, para o do Guilherme, é necessário estudar um pouco mais. Como estamos chegando ao fim da primeira semana, para quem está acompanhando o cronograma, e nos aproximamos do final de semana, vou trazer um desafio maior, um projeto, e, para ele, será necessário estudar um pouco a mais duas coisas que ainda não exercitamos. Se lembram do nosso conversor de dólar para real? Que também pode existir de euro para real ou dólar para várias outras coisas.

Eu gostaria que vocês praticassem, pode ser em um projeto novo, primeiro sem o conversor, e criassem várias funções, por exemplo, a função que se chama `converteDolarParaReal()`, que recebe quantos dólares. Também a função `converteDolarParaEuro()` e `converteEuroParaBitcoin()`. Enfim, várias dessas, e toda vez que você recebe um valor, será possível fazer um `console.log()` do valor, e, então, escreve um código que testa essas funções.

Sobre isso vocês já têm conhecimento e conseguem fazer, mas eu quero dar dois passos. O primeiro, toda vez que chamamos uma função e passa argumentos para ela, quantos dólares ou qual é o filme, também podemos devolver, para quem chamou, um valor, que é o valor de retorno, existe até um comando que se chama *return*.

Então, durante o final de semana, quem já chegou até aqui, fez todos os exercícios e quer mais, só para quem de fato está acompanhando tudo. Sendo assim, para as pessoas que fizeram tudo e estão empolgadas, procurem pelo *return* para entender e fazer essas várias funções, que são curtas: multiplica, divide, faz algumas contas, arredonda, entre outros.

O outro passo tem a ver um pouco com a página web, com o HTML. No nosso conversor que tem o playstation, além do *input,* quero que você coloque outro *input* para perguntar "você quer converter esses dólares para quê?" e a pessoa pode escolher: real; euro; ou bitcoin. Sabe os *radio buttons,* aquelas “bolinhas” que têm opções. Procurando no HTML *radio button*, você compreenderá que ele tem um *input* desse tipo e, então, você pode clicar, pegar qual o usuário ou usuária escolheu e, dentro do código, fazer um *if.*

Portanto, "*if* a pessoa escolheu converter para euro, chama a função `converteDolarParaEuro()`, passando o valor colocado". Seguindo, "*else if* escolheu a opção converter para bitcoin, chama a função `converteDolarParaBitcoin()`, pega o valor, recebe o retorno e atualiza" e assim por diante. Com isso, vocês praticarão muito as funções, conhecerão melhor o *return,* que em algum momento aparecerá nas nossas aulas, e praticarão um pouco HTML de mais elementos, porque os *Input Types* você escreverá.

É um projeto novo, não estudamos tudo isso. É um extra mesmo para quem está com muita vontade de praticar no final de semana. Para quem está chegando agora, está no meio do final de semana, aproveite para se dedicar ao que foi passado na parte essencial. Esse, de fato, é um *plus* para quem quer ir além, baseado na ideia que a Rafa e o Guilherme trouxeram.

É possível fazer sim, depende de um tanto de pesquisa, vamos deixar alguns links disponíveis para o *return* e para o *Input Type*, mas vocês têm condições, sim, de realizar durante o final de semana, dependendo do tempo de aula que você acompanhou.

Parabéns por finalizar essa primeira etapa. Nas próximas aulas, misturaremos mais conteúdos e criar projetos mais estruturados, complexos, que envolvem códigos maiores. Então, pratique bastante, tire suas dúvidas: "E se fosse assim? E se eu testasse isso que o Guilherme falou com a Rafa, mais o Paulo, e misturasse tudo? Que erro dá? Por que dá esse erro?" As perguntas fazem parte do trabalho. Isso acontecerá no dia a dia, mesmo com experiência.

Então, encare a mensagem de erro no console, encare conversar no Discord da comunidade, "googlar" naquele Mozilla, perguntar para o amigo, para a amiga, ver o streaming no tweet do **Marco Bruno**, o streaming das pessoas que estão direto nos canais do Discord.

Enfim, esse canal do Discord, que também pode ser o Hangout ou o Zoom, acontece nas empresas também. Sempre falamos, "conseguimos parear?" Chamamos isso de "parear", "conseguimos sentar juntos e fazer juntos? Eu não estou conseguindo resolver esse problema. Você me ajuda?", então, uma pessoa ajuda a outra. Estamos apresentando algumas situações muito reais, mesmo que você não perceba agora. As situações são muito reais, inclusive nos exercícios.

Então, até a próxima aula!! Obrigado!!

Rafaella: Até mais!!