[00:00] Olá, o meu nome é Guilherme Lima e eu serei o seu instrutor nesse treinamento de métodos de array com JavaScript.

[00:05] Nesse curso vamos trabalhar no AluraBooks. Temos uma API que vai disponibilizar uma lista de objetos e o nosso desafio vai ser pegar as informações que estão naquela lista, exibir essas informações na tela, que começaremos com um projeto base, com uns arquivos estáticos.

[00:23] Criar filtros para livros de front-end, back-end, livros de dados, livros disponíveis, manipular elementos na tela através do DOM e o JavaScript. Ordenação de preços, tudo isso e vamos focar em alguns métodos especiais do JavaScript como: o método filter, forEach. map, reduce e sort.

[00:44] Esse curso voltado a pessoas que não tiveram contato com esses métodos especiais do JavaScript. E se você já tem contato também, eu recomendo que você faça esse treinamento, porque sempre tem algo muito interessante para aprendermos e analisarmos também.

[00:59] Esse curso está incrível, está com um leiaute muito bonito e ele está um curso bem leve. Espero que você goste, abriremos bastante a documentação, vamos deixar esses links da API para você conseguir praticar na sua casa também. E eu tenho a certeza de que vai ser um momento de aprendizado muito bacana entre nós. Vamos lá?

**Olá!**

É muito bom receber você neste curso de JavaScript: manipulação de array.

Espero que seja uma experiência de aprendizado incrível e que possamos vencer todos os desafios juntos. Neste curso, vamos aprender de forma prática como utilizar os métodos de array com JavaScript.

**Preparando ambiente**

Recomendo o uso do [VSCode](https://code.visualstudio.com/" \t "_blank) para editar o código e, para tornar o desenvolvimento ainda mais fácil, recomendo a [instalação da extensão do Live Server](https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer).

Para aprofundar nossos conhecimentos na linguagens JavaScript, vamos disponibilizar o [projeto base deste treinamento que está no GitHub](https://github.com/alura-cursos/alura_books/archive/refs/heads/projeto_inicial.zip).

A Alura é formada por pessoas que gostam de tecnologia e acreditam no poder da educação através dela. Somos uma comunidade que ama compartilhar conhecimento. Em caso de dúvida na instalação ou durante o curso, conte sempre com o fórum. Caso não tenha dúvidas, não deixe de participar do fórum ou interagindo no nosso servidor do Discord para ajudar outras pessoas e fazer da comunidade um lugar ainda melhor!

**Vamos começar?**

[00:00] Vamos iniciar os nossos estudos sobre métodos de *array* com JavaScript. Seu primeira passo é acessar o link "preparando ambiente" na atividade anterior a esse vídeo, lá vocês podem fazer *download* do projeto base que utilizaremos durante todo o nosso treinamento.

[00:14] Eu já fiz o download e o nome está "alura\_books - projeto\_inicial". Vou renomear esse download só para "alura\_books" e vou usar o Visual Studio Code para editar o nosso código. Então eu vou arrastar esse alura\_books para dentro e veremos o que é que temos para mexer nessa aplicação, nesse projeto.

[00:33] Então temos uma pasta, com várias imagens. Temos um "index.html". Logo mais entenderemos o que é cada etapa, o que são esses botões, como a estrutura está feita. Temos um arquivo que se chama "equisicao.txt", que eu já vou falar para você o que é. E o "style.css" para os estilos da nossa aplicação.

[00:52] Estou com o meu projeto base, com o *download*, o que eu preciso fazer agora é, de alguma forma, carregar esse projeto para conseguirmos visualizar o que mexeremos durante todo o nosso treinamento.

[01:03] Para isso, eu vou usar a extensão do Live Server, eu vou colocar *live*. Nesse primeiro link, Live Server. Eu vou instalar, o meu já está instalado, é só você instalar essa extensão. Eu vou clicar com o botão direito em cima - deixe-me fechar essa janela - do "index.html", eu vou colocar "abrir com o Live Server" ou "*open with Live Server*".

[01:24] Eu vou carregar essa aplicação. Carregou a aplicação, vamos ver o que é que temos. Temos um cabeçalho em cima com o título e subtítulo: "AluraBooks, melhores livros de programação, encontre na nossa estante o que você precisa para o seu desenvolvimento".

[01:39] E temos alguns botões: "Livros de front-end", "Livros de back-end", "Livros de dados". Eu clico nesses botões e não acontece absolutamente nada, daremos vida para esses botões com o JavaScript.

[01:50] E tem alguns livros, então tem o "Cangaceiro JavaScript", tem o outro "Cangaceiro", repara que ele está com uma opacidade diferente. Então ele está mais fraco para conseguirmos visualizar, provavelmente é porque esse livro não está disponível. Em baixo tem algumas informações.

[02:06] O título do livro, o autor, o valor desse livro e tem uma *tag* das categorias. Então tem *front-end*, *back-end*, repare que essas *tags* estão todas não verdadeiras. Por exemplo, temos uma *tag* para o livro de SASS, que é para os pré-processadores de CSS, com *back-end* também, não faz muito sentido. É meramente ilustrativo.

[02:28] E embaixo temos todos os livros disponíveis por 299 reais. Alura, mergulhe em tecnologia. Esse é o visual, todo esse visual que estamos vendo está estático. O que isso significa? Que ele está direto no HTML, olha só: temos os nossos botões e, quando tem a seção de livros, tem o "Cangaceiro JavaScript" no livro\_\_titulo, tudo isso está estático, ou seja, toda essa informação que estamos vendo na tela está diretamente no HTML.

[02:57] E no mundo real, não é isso que acontece. Geralmente buscamos essas informações de um banco de dados, ou de uma outra aplicação, ou de uma API. Então, o que é que faremos? No nosso treinamento, temos esse arquivo requisicao.txt. Repare que tem um endereço: https://guilhermeonrails.github.io/casadocodigo.livros.json.

[03:19] Eu vou clicar com o "Ctrl" e vou clicar em cima desse link para abrirmos e vermos o que é que acontece. Repara que, eu fazendo isso, ele me levou para uma outra página que tem: título, preço, autor, imagem, alt e categoria.

[03:32] Essas informações que estão aqui, elas estão do tipo json. Podemos até ver que tem esse nome livros.json. Então, geralmente, o que acontece? Um time focado no *back-end* vai criar todo o banco de dados, a comunicação. Vai falar: olha, vai ter uma tabela, chamada títulos, vai ter um autor que esse autor criou esse título desse livro, vai fazer toda aquela mágica no *back-end* conversando com o banco de dados.

[04:00] E para pegarmos essas informações exibirmos na tela, eu não quero exibir essas informações que já estão estáticas. Eu quero exibir tudo isso daqui, eu vou precisar fazer uma requisição. Então qual que vai ser o nosso desafio? Eu vou comentar todas essas linhas nossas de livro, então olha só, escrolando para baixo, quando começar a nossa seção, onde tem o id=livros, eu vou pegar do *id* <div class="livro">, vou clicar na linha 33, vou deixar a linha 33 clicada. Vamos escrolar.

[04:33] E repare que acabou todo meu conteúdo de livro naquele *id*. Eu vou apertar o "Shift" e vou selecionar a linha 264. Agora vem uma mágica: eu vou selecionar o "Ctrl + /" ou "Cmnd + / " para eu comentar todos esses livros estáticos.

[04:50] E se eu volto na minha aplicação do AluraBooks, olha só o que aconteceu? Não está nenhum livro sendo exibido. Por quê? O que queremos fazer agora é uma requisição para exibir, de fato, esses livros que estão aqui. Então eu quero pegar todos esses livros e construir, utilizando, manipulando o DOM, cada um desses livros no AluraBook. E é isso que faremos na sequência.

[00:00] Agora que comentamos os livros estáticos do HTML que tínhamos aqui, queremos realizar uma requisição para esse *endpoint* que está anotado em nosso "requisição.txt" e pegarmos esse json e colocar os livros dentro do nosso AluraBooks.

[00:20] Para isso, vamos fazer por etapas. Primeiro queremos realizar a requisição. Vamos criar uma pasta chama "app" e dentro dela vamos manter todo nosso código JavaScript. Durante o nosso treinamento vamos ter mais de um códido JavaScript, então vamos colocar todos os códigos dentro dessa pasta para mantermos o projeto organizado.

[00:40] Dentro dessa pasta, vamos criar um arquivo chamado "main.js". Esse arquivos será responsável por de fato realizar a requisição e dizer que temos a variável livros com base na requisição já feito.

[01:00] No primeiro passo será criar essa variável chamada livros que será igual a uma *array* vazia, let livros = []. Em seguida, criamos uma constante chamada endpointDaApi que será igual ao conteúdo que temos em "requisição.txt": const endpointDaApi = 'https://guilhermeonrails.github.io/casadocodigo/livos.json'

[01:36] Em seguida, fazemos um get para ele de fato fazer a requisição e buscar os livros da API. Então, getBuscarLivrosDaAPI(), ele será uma função.

[01:54] Queremos que essa função seja assíncrona. Para isso, precisamos definir a palavra async antes da palavra function. Em seguida, damos o nome da função: async function getBuscaLivrosDaAPI(). Não teremos parâmetros nessa função.

[02:07] Queremos que essa função seja assíncrona porque em nossa API estamos utilizando um projeto com alguns livros. Mas vamos supor que tenhamos onze milhões de livros cadastrados, essa função demoraria muito e provavelmente teríamos um erro.

[02:31] Para dizer que essa função precisa esperar uma promessa de receber todos os livros desse endpoint, vamos usar o await. Passamos uma requisição e obteremos uma resposta. Então vamos chamar res para termos uma resposta e deixaremos ela igual a await: const res = await. Ou seja: esper por uma promessa

[02:48] Nós sempre utilizamos para conseguirmos manipular as funções e essas requisições com HTTP esses APIs do fetch. Como queremos realizar uma função para o endpointDaAPI vamos passar ele como argumento: await: const res = await fetch(endpointDaAPI).

[03:04] Depois que pegarmos a resposta dessa aquisição, queremos atribuí-la a livros. E livros também será igual a await e a resposta dessa requisição queremos que seja transformada em json(): livros = wait res.json(). Em seguida, damos um console.log(livros) para exibirmos isso na tela.

[03:35] Como criamos um novo arquivo precisamos ir em "index.html" e adicionar o <script src="app/main.js"></script> antes de fechar o <body>, na linha 276.Assim, vinculamos esse HTML com o JavaScript que estamos criando.

[04:06] No navegador, clicamos com o botão direito do mouse e selecionamos "Inspecionar". Em "Console" temos uma *Array*(12), ou seja, os livros estão aqui. Porém, a forma como estamos visualizando esses livros não está muito legal.

[04:20] Podemos alterar a forma que vemos eles substituindo o console.log() por console.table(). Agora ele monta uma tabela com todos os livros que temos. Assim, conseguimos ver todos os campos que utilizamos e todas as outras informações.

[04:37] Para visualizarmos melhor, vou tirar o await de de livros. Ele vai realizar uma requisição, essa requisição vai esperar uma promessa. Ele vai terminar de fazer essa requisição, mas não vai dar tempo dele montar a resposta em json. Para conseguirmos de fato visualizar essa resposta dessa promessa na nossa variável de livros, vamos usar o await.

[05:05] Dessa forma temos os livros da API. Metade dos nossos problemas foram resolvidos. Trouxemos as informações que estavam no *endpoint* para nossa aplicação. Mas agora queremos montar na tela, manipular o DOM. Como faremos isso utilizando métodos especiais de *arrays* com JavaScript? Descobriremos na sequência!

[00:00] Nosso desafio agora é pegarmos as informações do json, que a colocamos na nossa variável de livros, e exibi-las na tela. A primeira ideia que nos vem à cabeça é a seguinte: já que temos a lista de livros, o que é a que podemos fazer? Poderíamos fazer um for para falar pegarmos o livro no índice 0, que tem o título 0, que tem o preço 0 e montar na tela, essa estrutura da classe de livro que comentamos.

[00:28] Então vamos criar uma nova classe de livro, essa classe vai ter uma imagem. Quando chegar no src, queremos pegar essa imagem que está dentro do src da nossa classe, esse HTTP que é um link que contém a imagem desse livro.

[00:41] Quando chegar na parte do nome do livro, temos o título do livro e assim por diante. Fazemos esse comportamento para todas as outras classes que temos. Só que, se eu faço isso na mão, isso vai ficar muito "verboso", principalmente utilizando o for.

[00:57] Assim que percebemos a utilidade e o benefício de utilizar métodos especiais de array. E eu quero pesquisar o primeiro método junto com vocês. Vou digitar forEach. Deixe-me colocar forEach mozilla, que conseguiremos até rodar um exemplo bacana dele.

[01:12] Vou clicar nesse primeiro link array.prototype.forEach(). Ele fala assim: o método forEach() executa uma função dada para cada elemento do array. Isso parece que faz sentido para nós. O que é que nós queremos pegar? Cada elemento do array e criar para ele uma classe de livro, com todas as características que ele tem.

[01:31] E o próximo elemento? Cria mais uma classe de livros e assim por diante. Então para nós parece que faz sentido. Vamos testar, só para vermos na prática acontecendo?

[01:41] Eu quero que você faça junto comigo. Na parte superior direita da tela, onde está escrito "Português (do Brasil)", clicaremos e selecionaremos a língua inglesa. Quando selecionamos a língua inglesa, aparece um *try it*, que seria para testarmos o nosso código.

[01:55] Eu vou escrever um código, se você quiser escrever junto comigo pode, mas eu recomendo que você assista e você preste atenção no que eu vou fazer.

[02:04] Primeiro de tudo, eu vou criar uma variável chamada livros e vou falar que ela vai ser igual a uma lista e vou atribuir já os valores. Eu tenho o livro de JavaScript, temos livros de Python e temos livros também de Go. Eu vou deixar em maiúsculo, para ficar certa a escrita e vou fechar.

[02:22] Tenho uma lista de livros com três elementos. O que é que eu faria da forma clássica, sem utilizar o forEach? Eu faria um for onde let i=0;) que é o clássico. Depois eu falaria assim: enquanto i for menor que livros.length, enquanto livros for menor que o tamanho de elementos, de itens que eu tenho nessa minha lista, você vai continuar incrementando.

[02:49] E aqui eu colocaria uma chave e outra chave fechando. E aqui eu faço, por exemplo, um console.log e aqui eu coloco livros, no plural, com o índice que ele está trabalhando e fecho o parênteses. Se eu for para baixo e rodar esse código, ele vai mostrar "Js", "Python" e "Go". Maravilha, conseguimos o resultado que queríamos.

[03:13] Só que, lembra que eu falei? Essa forma é muito verbosa, olha o tanto que eu tenho que escrever para conseguir esse resultado. O que eu vou fazer? Eu vou comentar essas três linhas e vou escrever agora utilizando esse método que estamos pesquisando, que é o forEach.

[03:28] Então temos alguns exemplos das sintaxes, como é que ele funciona, quais são os parâmetros que ele utiliza. E tem algumas formas. Eu posso fazer utilizando o arrow function, que é uma das formas mais comuns de encontrarmos nos códigos.

[03:42] Então eu vou passar o forEach, vou passar aqui dentro um elemento e o que eu quero que ele faça depois. Primeiro, eu vou pegar a minha lista, o meu array livros.forEach. Lembrando que forEach é uma palavra *case sensitive*, então vai fazer diferença eu escrever o "e" com maiúsculo ou minúsculo.

[04:00] Se você observar: "e" maiúsculo; em cima também: "e" maiúsculo. Então "for" minúsculo, "e" maiúsculo e "ach" minúsculo. forEach.

[04:12] Então dentro do forEach eu tenho uma lista de livros. O *for*, em inglês, significa para; *each* significa cada. Então para cada livro - eu vou escrever livro - dentro da minha lista de livros, eu quero fazer alguma coisa, eu quero realizar alguma ação.

[04:29] Então eu vou colocar uma arrow function abrindo => e vou colocar a nossa arrow function. E aqui eu posso colocar a mesma ideia que tínhamos ali, console.log(livro) que é a manipulação que já estamos utilizando.

[04:43] Vou rodar esse código e temos o mesmo resultado. Puxa Gui, você fez alguma coisa, alguma mágica, eu vou fazer o meu código quebrar, para vocês verem que isso está funcionando. Olha, console.log, tirei o parênteses, eu vou rodar e ele vai falar "opa, você esqueceu de colocar um parênteses depois do argumento".

[04:58] Então eu vou colocar o parênteses, quando eu rodo temos o mesmo resultado. E embaixo tem algumas formas, podemos receber mais parâmetros como o elemento que estamos utilizando, o index e a própria array que iremos utilizar.

[05:14] Se precisarmos de algum caso específico, olha, quando tiver um index eu quero que tenha tal comportamento, conseguimos fazer isso também.

[05:23] Podemos utilizar arrowFunction ou funções callback, forEach e uma outra função call-back para utilizar essa ação em todos os elementos. O que eu quero deixar claro no coração de vocês é o seguinte: forEach, ele vai pegar cada elemento da nossa lista, do nosso array e vai executar uma determinada função que passaremos.

[05:45] Gui, deu para entender, criei livros.forEach(livro => { console.log(livro) }. Mas como é que fazemos agora para o nosso código, para criarmos, de fato, uma função com o forEach para exibir os livros na tela? Veremos isso no próximo vídeo.