[00:00] Olá, boas-vindas ao curso de **algoritmos com JavaScript**.

[00:07] Nesse curso, nós vamos conversar sobre o que são algoritmos e por que o estudo de algoritmos é tão importante em programação. Vamos começar analisando alguns problemas comuns de programação e de que forma podemos resolvê-los com algoritmos, algoritmos diversos.

[00:23] Esse curso está voltado para quem está começando do começo em programação e escolheu o JavaScript como linguagem, mas que já teve contato com os fundamentos do JavaScript: o que são dados primitivos, número, string, booleano, arrays, objetos e funções.

[00:43] Onde nós começamos e até onde vamos? Vamos começar traduzindo problemas, às vezes problemas do dia a dia mesmo, que resolvemos de cabeça, como falamos.

[00:54] E vamos traduzir a resolução desses problemas para o passo a passo de um algoritmo. Vamos escrever esses algoritmos usando funções e métodos do JavaScript. Vamos transcrever esses passos, esses fluxos para código.

[01:11] Juntar algoritmos. Às vezes, nós precisamos, para resolver problemas, trabalhar com mais de um algoritmo de uma vez. Vamos ver como usamos isso, como isso funciona no código.

[01:21] E testes de mesa. Às vezes, nós sabemos que um bloco de código está funcionando, mas não entendemos totalmente de que forma isso está ocorrendo. Por exemplo, quando usamos um algoritmo pronto – vamos ver como isso funciona durante o curso – temos lá um código que resolver um problema, mas o que está acontecendo dentro desse código?

[01:41] O que está guardado nas variáveis, o que faz cada linha? Então, os testes de mesa nos ajudam a seguir o fluxo de um algoritmo, o fluxo de um bloco de código linha por linha, variável por variável, e nós conseguimos entender exatamente todos os passos. Então, vamos fazer alguns testes de mesa.

[02:00] Vamos comparar algoritmos também. Como sabemos se um algoritmo funciona bem para o nosso caso e o que é funcionar bem, quando falamos de algoritmos.

[02:10] Análise de complexidade porque, pensando que algoritmos executam instruções, instruções de código, como conseguimos saber quão complexo um algoritmo pode se tornar baseado na quantidade de dados que temos para trabalhar com ele?

[02:26] Então, tenho muitos dados, tenho uma lista com muitos livros, tenho vários candidatos de um vestibular, como eu uso algoritmos de busca para isso? Como uso um algoritmo de ordenação?

[02:39] E o quão complexo que o código desse algoritmo, que as instruções podem se tornar em termos de processamento, de acordo com a quantidade de dados que trabalhamos? Vamos ver como isso funciona também.

[02:51] Nesse curso, vamos trabalhar firme com algumas partes fundamentais do JavaScript. Tipos de dados, variáveis, funções, arrays e objetos. Mas não vamos parar para explicar em detalhes o funcionamento, o conceito por trás de cada uma dessas partes fundamentais.

[03:07] Você vai aproveitar melhor o curso se já tiver feito os cursos anteriores sobre fundamentos do JavaScript ou se você já tiver estudado sobre esses assuntos.

[03:17] Lembrando que, nesse curso, nós não vamos falar sobre assuntos específicos de front-end, mas tudo que vamos ver aqui você pode abstrair e aproveitar os conceitos da mesma forma.

[03:29] Depois desse curso, você pode estudar mais algoritmos por conta própria, entender como diversos algoritmos funcionam para diversos problemas.

[03:37] Quais as complexidades de cada um e como você consegue, por exemplo, pegar o conceito de um algoritmo escrito em pseudocódigo e passá-lo para JavaScript para você conseguir usar em um projeto ou testar e entender melhor como ele funciona.

[03:49] Eu sou a Ju Amoasei e vamos começar.

JavaScript é conhecida como a “linguagem da web”. Embora tenha se desenvolvido no front-end, atualmente cada vez mais a utilizamos no chamado back-end, ou server side (lado do servidor) das aplicações web. Este curso foca no uso do JavaScript para back-end.

Assim, precisaremos instalar duas ferramentas: o **NodeJS**, necessário para executar código JavaScript sem precisar do navegador, e o **Visual Studio Code** ou VSC, editor que usamos para escrever código.

## NodeJS

Caso já tenha feito algum curso anterior de JavaScript para back-end, o NodeJS pode já estar instalado. Se não tiver certeza, é possível conferir com os seguintes passos:

1) Abra uma janela do terminal em seu computador. Isso pode ser feito da seguinte forma:

**Windows**: Você pode acessar a busca no menu inicial e procurar por Prompt de Comando, ou acessar via Menu Iniciar > Sistema do Windows > Prompt de Comando.

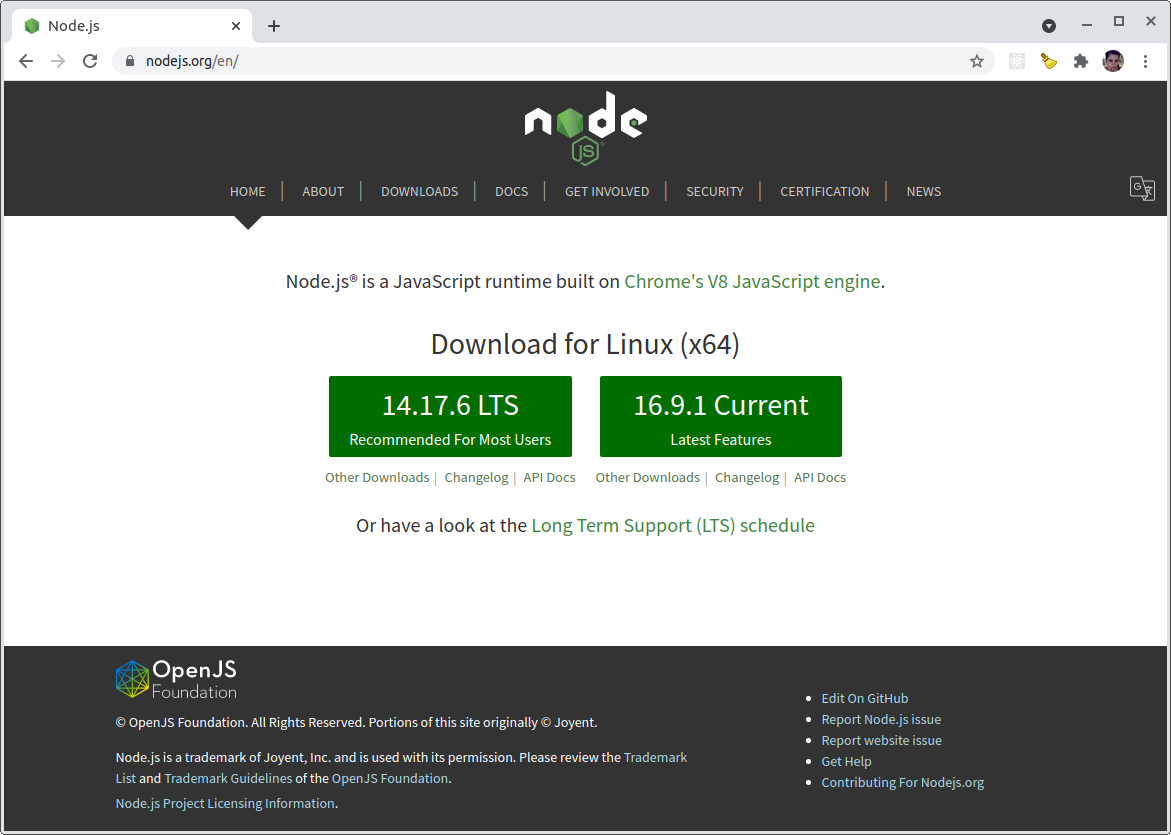
**MacOs**: O Terminal está disponível no menu de Aplicações, dentro da pasta de Utilitários.

**Linux (Ubuntu)**: O Terminal está disponível no menu de Programas/Aplicações. Caso não localize, pode estar dentro da pasta Utilitários.

2) No terminal, digite node --version ou node -v - caso retorne um número de versão, como por exemplo, v14.16.0, o NodeJS já está instalado.

Caso precise instalar, siga as instruções para cada sistema operacional que estão na [página inicial do NodeJS](https://nodejs.org/pt-br/). Este curso foi desenvolvido usando a versão 14.10.1.

O site do NodeJS oferece duas opções para download, a LTS e a "current" (atual). Você pode escolher a versão LTS (Long Term Support, ou Suporte a Longo Prazo) e clicar no botão correspondente para baixar e instalar normalmente como qualquer outro programa.



## Visual Studio Code

O Visual Studio Code é o editor que escolhemos utilizar para escrever nossos códigos durante este curso. Os links para baixar e instalar de acordo com o seu sistema operacional estão na [página inicial](https://code.visualstudio.com/) do VSC.

Existem vários outros editores, como o Atom, o Sublime e o Notepad++. Caso já tenha algum destes instalado ou preferência por outro, fique à vontade! Note apenas que alguns recursos e ferramentas variam de editor para editor, e podem não estar disponíveis ou funcionar de forma diferente.

Agora podemos começar!

[00:00] Boas-vindas ao curso de algoritmos com JavaScript. Vamos começar vendo qual é o nosso problema, o que nós vamos trabalhar nesse curso.

[00:10] Vamos supor que nós ganhamos um vale-presente em uma livraria de X, digamos, cinquenta reais, cem reais, e queremos ganhar esse vale-presente em livros de programação.

[00:20] Nós entramos no site da livraria, procuramos por livro de programação, e recebemos algumas opções. Só que o site da livraria trouxe para nós essas opções – um livro de JavaScript (25,00), um livro de PHP (15,00), um livro de Java (30,00), Elixir (50,00), Go (45,00) e Python (20,00) – que não estão em nenhuma ordem particular que consigamos ver.

[00:41] Não está em ordem alfabética e não está em ordem de preço, os preços estão misturados, os mais caros com mais baratos etc. Então, qual é o problema que nós temos? Quando temos um vale-presente, normalmente nós queremos fazer o melhor uso possível dele.

[00:54] Então, o que vale mais a pena? Tenho um vale de cinquenta reais, comprar cinco livros de dez reais ou comprar dois livros de vinte e cinco reais? Para conseguirmos fazer isso, temos que saber quais são os livros mais baratos, como eu consigo somar etc.

[01:08] Então, temos aqui a nossa busca de livros que trouxe uma lista desordenada para nós, não está em nenhuma ordem que nós conseguimos identificar. Essa lista de livros é pequena, então nós conseguimos ir guardando na memória, mais ou menos, a ordenação.

[01:26] Vamos ver qual é o mais barato? Eu olho aqui, o de JavaScript é o primeiro da lista, 25,00 reais, é o único livro que eu tenho. Depois tem o de PHP, que é 15,00. O livro de PHP é mais barato que o de JavaScript, então guardei isso na memória.

[01:39] Tem o de Java de 30,00 reais, o de Elixir de 50,00. Tenho na cabeça que o de PHP ainda é o mais barato, 15,00 reais. O de Go por 45,00 e de Python por 20,00. Então, dá para guardar na memória que os mais baratos são de PHP, de JavaScript e o de Python.

[01:58] O de PHP do 15,00, o de Python por 20,00 e o de JavaScript por 25,00. Temos 50 reais, mas dá para compensar o resto com o cartão. Então, vamos pensar no que fizemos.

[02:11] Na nossa cabeça, nós guardarmos alguns valores enquanto íamos percorrendo uma lista. Então, vamos dar uma olhada no que conseguimos fazer, de novo, anotando mais ou menos os passos que nós percorremos.

[02:30] Vou voltar minha lista ao que ela era. Vamos percorrer. Vamos dizer que essa lista de livros é um *array* de livros, podemos tratar dessa forma. Como que na nossa cabeça nós criamos mais ou menos um passo a passo de como resolver essa ordenação de livros?

[02:54] Começamos no que seria a primeira posição da nossa lista. Temos um valor só, não temos o que fazer com ele. Mas agora, passando para o segundo valor, eu tenho dois valores, 25,00 e 15,00, já consigo comparar.

[03:09] Então, eu consigo já passar, por exemplo, para o primeiro valor da lista o meu livro de 15,00 reais, então o livro de 15,00 está guardado como o mais barato. E, em seguida, ir passando para os próximos da lista.

[03:23] Então, agora eu sei que o livro de 15,00 é mais barato. Tenho em seguida um livro de 25,00, um livro de 30,00 – ok, por enquanto está ordenado –, um livro de 50,00 – por enquanto está ordenado –, depois do livro de 50,00, eu tenho o livro de 45,00.

[03:38] Ok, o livro de 45,00 é mais barato que o de 50,00, então já sei que o de 50,00 não está entre os livros mais baratos, posso trocá-los de lugar. E meu último livro da lista é um livro de 20,00, que eu já sei que é mais barato que é de 45,00, que o de 50,00, que o de 30,00 e que o de 25,00.

[03:59] Então, posso ficar trocando todos os meus livros de lugar até eu, na minha cabeça, fazer algo parecido com uma lista ordenada. Agora, se eu quiser, por exemplo, só o livro mais barato, é o livro de 15,00 reais; se eu quiser os três primeiros livros eu já sei que são os livros de PHP, Python e JavaScript, 15,00, 20,00 e 25,00 reais, e por aí vai.

[04:23] Então, vamos um pouco além de guardar as coisas na cabeça. Como vamos trabalhar com JavaScript, nós podemos supor que a nossa lista de livros é um *array*, como já tínhamos conversado. Os *arrays* no JavaScript começam com 0, então nós temos uma lista que vai do índice 0 até o índice 5.

[04:43] Como é que nós pensamos, na nossa cabeça, para guardar os valores? Nós meio que fomos guardando na nossa cabeça sempre qual era o valor mais barato.

[04:53] Então, na nossa cabeça, nós percorríamos a lista. Vi primeiro o livro de 25,00, então o livro de 25,00, quando nós começamos a percorrer essa lista, é o menor. Só tenho um valor, 25,00, é o único valor que eu tenho e também é o valor menor.

[05:11] Então, meu valor atual é o índice 0 e meu valor menor também é o índice 0. Passando para o próximo item da lista, o índice 1 do nosso *array*, agora eu tenho um livro de 15,00.

[05:24] 15,00 é menor do que 25,00, então o meu atual, o livro que estou verificando agora, é o livro que está no índice 1, e o meu menor também é o índice 1, porque 15,00 é menor do que 25,00.

[05:43] Eu sei que agora, passando para frente, estou no índice 2. O índice 2 do meu *array* é um livro de Java, que é de 30,00 reais, então o meu atual agora passa a ser 2, o terceiro elemento da minha lista.

[05:59] Ele não é menor do que 15,00, 30,00 não é menor do que 15,00, então o meu menor continua sendo o 15,00 reais. Eu mantenho isso na cabeça e mantenho isso na minha “variável mental” que estou chamando de menor.

[06:14] Passamos para o próximo, Elixir, que está no índice 3 do *array*, sendo o quarto item, o quarto elemento. 50,00 ainda não é menor do que 15,00, então meu item menor continua sendo o PHP, que está no índice 1.

[06:30] Passamos para a frente, agora vou para o próximo da lista, que é o livro de Go, no índice 4. Então, atualmente, estou analisando o 4. 45,00 também não é menor do que 15,00, então, na minha variável menor, continuo guardando o livro de PHP.

[06:47] E, por último, eu passo para o último elemento do *array*, que é o índice 5, é o sexto elemento, que é um livro de Python que é 20,00 reais. 20,00 não é menor do que 15,00 também, então continuo a minha variável menor, minha variável mental, continua guardando o livro que estava no segundo elemento, o índice de número 1 do *array*.

[07:12] Então, no final, o que nós fazemos na nossa cabeça sem perceber? Nós percorremos um *array*, percorremos uma lista de elementos e guardamos na cabeça, na nossa variável mental que eu chamei de menor, o índice – qual era o produto, na verdade, na cabeça nós não guardamos por índice –, qual era o produto que tinha o menor preço.

[07:35] Que é o de PHP, que no *array* de JavaScript está no índice 1, e por aí vai. Então, o que nós fizemos foi basicamente não compilar, porque o JavaScript não é uma linguagem compilada, mas nós interpretamos, fizemos um pequeno programa que rodou sem nem percebermos – nós fazemos isso muitas vezes por dia –, e o que nós criamos na nossa cabeça foi um algoritmo.

[08:03] Então, o que é um algoritmo? O que é essa sequência que nós criamos na cabeça e criamos com programas de computador? Segundo a definição da Wikipédia, podemos colocar aqui, o algoritmo é uma sequência finita de ações executáveis que visam obter uma solução para um determinado tipo de problema.

[08:23] Dá para traduzirmos essa explicação, traduzir algoritmo como basicamente tudo que nós fazemos, também no dia a dia, e as instruções que damos para o computador que envolvam sequências de passos definidas.

[08:38] Então, por exemplo, o que mais nós fazemos no dia a dia que podemos mudar para um algoritmo? Quando seguimos uma receita. Então, vamos ver, por exemplo, uma receita de bolo é uma sequência finita de ações executáveis, nós executamos todas as partes da receita, que visa obter uma solução para um determinado tipo de problema.

[09:00] No caso, falta de bolo, um aniversário, vontade de comer bolo, por exemplo. Na vida real, o que nós fazemos? Nós dividimos um algoritmo de uma receita de bolo em vários passos, nós não percebemos.

[09:14] Por exemplo, o primeiro passo seria conferir se temos todos os ingredientes. E tem condicionais nesses passos. Por exemplo, podemos dizer que, se não tiver ingredientes, o que nós fazemos? Nós saímos para comprar o que falta ou desencanamos?

[09:29] Se desencanamos de fazer o bolo, o restante do programa, que é fazer o bolo em si, nem é executado. E tem uma outra coisa que fazemos também, nós acabamos dividindo o algoritmo de fazer um bolo em várias partes pequenas sem perceber.

[09:43] Não é só as duas partes principais de ver sem tem todos os ingredientes, tem várias coisas que aprendemos no dia a dia de fazer bolo. Por exemplo, conferir os ingredientes é uma, para não esquecermos no final de botar o fermento que não tem fermento, e também algumas outras coisas.

[10:01] Se você faz bolo com ovos, por exemplo, você vai querer quebrar o ovo em um pote ao invés de quebrar o ovo na massa, porque o ovo pode estar podre. Essa é uma instrução que, normalmente, nós aprendemos, recebemos essa instrução pronta, alguém nos fala, alguém nos dá essa dica.

[10:16] Você pegar o ovo, tirar o ovo do fluxo do bolo, quebrar em um pote e voltar, é uma pequena parte que nós separamos. Então, na vida real, nós escrevemos algoritmos na cabeça, nós programamos esses algoritmos, terminamos as nossas sequências finitas de instruções – por exemplo, fazer um bolo, botar no forno etc. –, só que não percebemos que fazemos isso na cabeça.

[10:46] E não só nós seguimos essas sequências de passos, como também fazemos essas pequenas reparações, o que seriam algoritmos menores, algoritmos de ver se tem todos os produtos, se tem tudo que nós queremos, se não tem, compra, ou se não tem, desencana, não cumpre o resto.

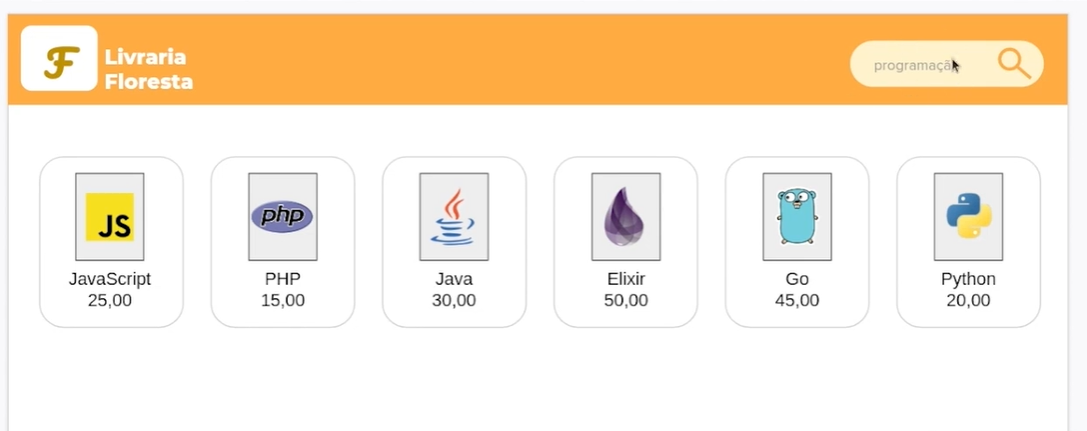
[11:03] Nós podemos traduzir todo esse rolê do bolo que nós fizemos para o nosso problema de encontrar o livro mais barato em uma lista desordenada. Essa história de quebrar o ovo no pote, de aprendermos a fazer bolo com o tempo, recebendo diversas dicas, é que existem vários algoritmos já estabelecidos, já prontos para vários problemas comuns de programação.

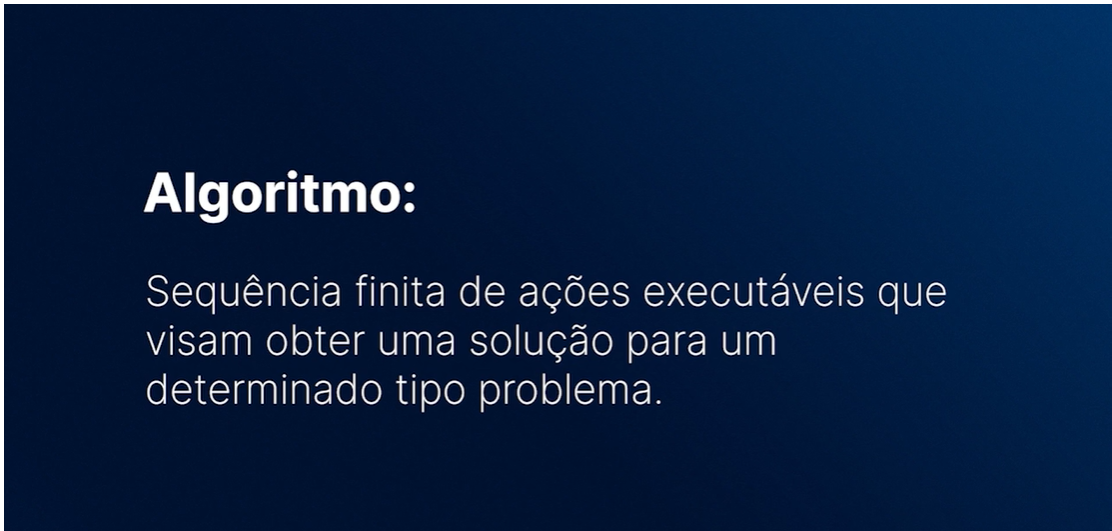
[11:28] Isso é mais ou menos o que vamos ver nesse curso: como pensamos em algoritmos, como testamos algoritmos, vamos ver alguns que já existem e se eles funcionam para nós, se eles são o melhor caso etc.

[11:41] Porque um mesmo problema, por exemplo, ordenar uma lista, fazer um bolo, pode ter mais de uma forma de ser resolvido, e a escolha errada também pode causar problemas.

[11:51] Por exemplo, eu tenho um algoritmo de me trocar e eu posso escolher colocar o tênis antes de colocar a calça. Funciona? Funciona, mas obviamente, colocar o tênis antes de colocar a calça vai ser muito mais trabalhoso do que colocar a calça antes de colocar o tênis.

[12:04] E por aí vai. Então, vamos dar uma olhada como transformamos esse algoritmo mental nosso em JavaScript, mas isso vamos ver no próximo vídeo.





[00:00] Agora que já temos o nosso "algoritmo mental" do fluxo que fizemos em JavaScript, vamos escrever esse código. Então, como fazemos para passar aquele fluxo que pensamos que criamos direto na página da nossa livraria? Vamos passar isso para o JavaScript, então.

[00:22] Primeira coisa que vamos precisar é criar um *array*, então vou criar const precosLivros. Por enquanto, criamos somente um *array* com o preço de cada livro, para começarmos em pequenos passos.

[00:37] Os preços são [25, 15, 30, 50, 45 20]. Então, um *array* em JavaScript. Nós precisamos guardar em algum lugar aquilo que eu estava chamando de variáveis mentais – agora não vai ser, agora vamos fazer no JavaScript.

[01:08] Uma que guarde o valor atual que estamos vendo, qual o produto atual que estamos analisando, vou chamar de let atual. Vou iniciá-la com 0, vai ser o primeiro índice do *array* que estamos trabalhando.

[01:24] E um outro let que vou chamar de maisBarato, que também vou iniciar com 0. É aquele onde vamos guardar - que estávamos guardando na cabeça - qual é o produto, qual é o índice do *array* que tem o menor valor. No caso do nosso *array*, teria que guardar o 15 mais do que qualquer outro.

[01:47] O que temos que fazer agora? Temos que percorrer toda essa lista de livros, confirmar qual é o mais barato de todos, porque sabemos na cabeça, mas agora o JavaScript tem que nos dizer.

[01:59] Podemos usar uma das várias formas que o JavaScript tem para trabalhar com *loops*, com laços de repetição, o que temos que fazer é percorrer listas, percorrer um *array*, que é um for. Então, vou criar um for que vai começar com let.

[02:16] Vou chamar a variável inicial desse *for* de atual, então for (let atual = 0;). Já que estamos iniciando, dentro do nosso *for*, uma let atual, nós não precisamos iniciá-la do lado de fora, nós podemos trabalhar só com a variável atual que está dentro do *for*, já vai fazer o mesmo trabalho.

[02:37] Vamos percorrer esse *array* enquanto atual for menor do que o tamanho do nosso *array* de livros, então (let atual = 0; atual < precosLivros.length;), com “gth” no final – sem errar, porque eu sempre erro e troco as últimas letras.

[02:58] Vamos atualizar a nossa variável atual, vamos incrementá-la de um em um, atual++, para ir pulando de índice em índice do *array*. E aqui dentro vamos criar o nosso //código.

[03:14] Se lembrarmos a forma como nós criamos esse algoritmo na nossa cabeça – vou voltar na nossa página –, o que tem que ser feito dentro desse *for* nós já falamos. Se o produto atual for menor, ele se torna o mais barato.

[03:33] Então, se o que nós estivermos vendo no momento na lista, por exemplo, “1” e “PHP”, ele é o que nós estamos vendo na lista? Ele é o de preço menor do que tínhamos antes? É. Então, ele se torna o menor, ele se torna o mais barato. E por aí vai.

[03:53] Nós meio que falamos, na cabeça, o que temos que transformar em código. Então, voltando para o nosso *for*, o que temos que fazer é uma condição, if.

[04:07] Se o produto atual, for o mais barato da lista, nós o guardamos dentro da nossa variável “preço menor”. No caso, eu chamei de maisBarato, então ele vai ser o mais barato e nós vamos guardar o número de índice dele dentro da nossa variável let maisBarato.

[04:27] A nossa condição é if (precosLivros), que é o nome do nosso *array*, na posição [atual], for menor do que nós tínhamos anteriormente – o que tínhamos anteriormente?

[04:44] Nós começamos com “0” para comparar, então o primeiro livro que nós comparamos, ainda não tem muito o que ser comparado. Se só temos um livro de 25, ele é o atual e é também o mais barato, então estamos começando tudo com 0.

[04:58] Então, vamos dizer que precosLivros na posição maisBarato. Lembrando que as variáveis guardam valores inteiros que traduzimos como índices do *array*. Então, precosLivros na variável atual é precosLivros na posição 0, ou seja, na posição 25.

[05:22] Vou até marcar com o comentário aqui, a posição 0. E precosLivros na posição maisBarato, o maisBarato também é uma variável que está guardando 0, e 0 é 25, então é onde estamos começando.

[05:37] Se a atual for menor do que já está guardado como maisBarato, vamos fazer alguma coisa, ou seja, temos que fazer com que o livro na posição atual se torne o mais barato – esquece o que tinha antes.

[05:53] Então, nós começamos na posição 0, mas, se a posição 1 for mais maisBarato, esquece o 0 que estava guardado dentro da variável e vamos colocar o 1 aqui.

[06:03] Então, o que tem que acontecer? maisBarato = atual. Nós declaramos uma let na nossa linha 4, ou seja, podemos substituir o valor dela de 0 para 1, de 1 para 2, de 2 para 3, de acordo com o índice onde está o menor valor do *array*.

[05:24] No caso, o 15, que já sabemos, olhando, que é o índice 1, mas o JavaScript ainda não sabe, ele vai ter que processar isso para nós.

[06:34] A última coisa a fazer, depois que sai desse *for*, é um console.log e olivro mais barato custa e podemos colocar, com *template strings* aqui, abrir cifrão e chaves, colocar ${precosLivros}, que é o nome do nosso *array*, em qual posição? Na posição maisBarato.

[07:04] Porque, lembrando, a partir do momento que o JavaScript entra no *for*, ele vai percorrer o *for* inteiro antes de continuar. Então, o console.log que nós colocamos por último, na linha 12, só vai ser executado depois que passar por todo o *array*.

[07:20] Então, nesse momento, nossa variável maisBarato teoricamente já está atualizada, passou pelo *array* inteiro, e já está guardando o que é realmente o menor valor da lista.

[07:30] Então, é precosLivros na posição que não sabemos – sabemos que é 1, mas o JavaScript ainda não. maisBarato que vai ter aqui um número entre 0 e 5 guardado dentro dessa variável.

[07:44] Vou salvar, venho no terminal e mando executar com node index.js, que é o nome que eu criei para esse arquivo. Então, o livro mais barato custa 15, ou seja, nós já sabíamos e o JavaScript agora também já sabe.

[08:01] Porque nós fizemos com que o JavaScript percorresse, com o código, exatamente o mesmo caminho que nós fizemos mentalmente usando o nosso site, usando as nossas variáveis.

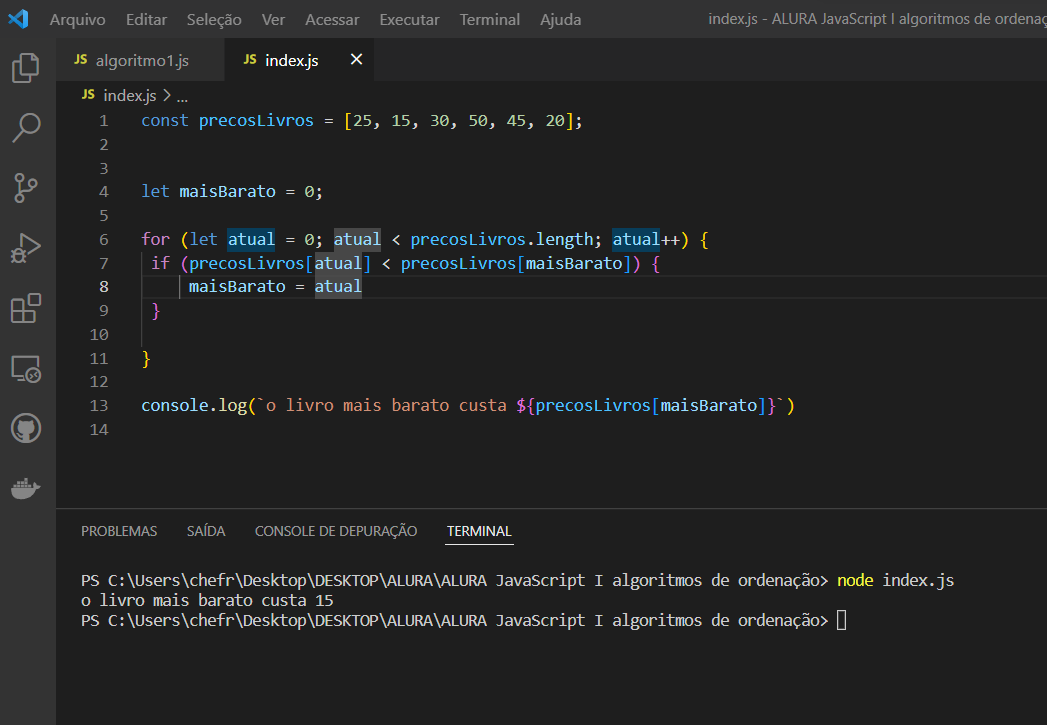
[08:20] Nós fizemos o que chamamos de “teste de mesa”, vou deixar um material extra para vocês sobre o que é um teste de mesa, mas, basicamente, nós fizemos uma interpretação do código na nossa cabeça e fomos marcando em cada momento do *loop*, em cada momento do processamento, o que tinha guardado em determinadas variáveis.

[08:37] Aqui eu chamei a variável de “menor”, no código eu acabei chamando de maisBarato, mas é a mesma variável. E a variável “atual”, que é a que está fazendo o controle desse *array*, de passar índice por índice.

[08:49] Então, o que nós fizemos foi criar o nosso primeiro algoritmo para resolver o nosso primeiro problema que é encontrar o livro mais barato em uma lista desordenada.

[08:58] Fizemos isso a partir da nossa cabeça e sem perceber, é uma coisa que fazemos todos os dias. Com o código, podemos passar os mesmos passos que fazemos na cabeça para o código, só fazendo de uma forma que o JavaScript entende.

[09:13] Mas, como falei antes, existe mais de uma forma de resolver um problema. Sempre ou quase sempre existe. Vamos dar uma olhada, continuaremos a estudar os nossos algoritmos.



[00:00] Antes de continuarmos, vamos refatorar esse *array* de livros que nós criamos para termos informações mais completas para trabalharmos. Porque trabalhar com *Array* de preços vai cumprir a missão, mas fica um pouco mais trabalhoso, porque vamos receber os valores sem saber de que livros estamos falando, e se tiver dois livros com o mesmo preço, como faz?

[00:22] Então, vamos fazer o seguinte, vou extrair esse *array* que chamei de precosLivros, vou criar outro arquivo, vou chamá-lo de “listaLivros.js”, na mesma pasta, e vou extrair essa *array* para esse arquivo.

[00:41] Então, vou continuar o chamando de precosLivros, tirá-lo do nosso “index.js” e passá-lo para a “listaLivros.js”, então “Ctrl + X” no nosso *array*, e “Ctrl + V” no nosso listaLivros.

[01:06] Então, “listaLivros.js” eu vou deixar com a variável precosLivros mesmo, e aqui, onde temos um *array* só de números, nós podemos ter um *array* de objetos, que cada objeto terá seu nome.

[01:25] Então, tenho a propriedade título: “JavaScript” para o primeiro livro, e a propriedade preco: 25 que, para JavaScript, é 25. A mesma coisa nós fazemos para todos os outros.

[01:45] O segundo elemento é PHP e a propriedade é 15: titulo: “PHP”, preco: 15.

[02:00] O terceiro eu já não lembro, nós olhamos, que é o de Java, 30,00. Então, titulo: “Java”, preco: 30. E, quase finalizando, o de Elixir é titulo: “Elixir”, preco: 50.

[02:40] Tem também o de Go e o de Python. titulo: “Go” e preco: 45; finalmente o último, mas não menos importante, que é titulo: “Python” e preco: 20.

[03:15] Agora temos um *array* de objetos e conseguimos identificar o que é o que, qual é o livro que custa 25, qual é o livro que custo 30 etc. Vamos dar uma última refatorada nesse código.

[03:30] O nosso arquivo “index.js”, agora preciso trazer o nosso *array* para cá. Vou exportar a nossa const precosLivros. Na última linha do arquivo "listaLivros.js", vou chamar module.exports = precosLivros.

[03:53] Podemos agora voltar para “index.js” e importar, na primeira linha do arquivo, antes de qualquer coisa, o const livros = require(‘./listaLivros’). Então, agora podemos chamar a const Livros que ela vai trazer o *array* para nós.

[04:25] E podemos refatorar essas partes do código do nosso *for* onde nós trabalhamos com os valores dos livros. Então, temos a nossa let atual, fica igual. Agora, precosLivros.length não é mais precosLivros porque estamos importando esse *array* de outro lugar, de livros.

[04:46] Então, precosLivros, em todos os lugares onde nós declaramos, agora é apenas livros. Lembrando que, agora, não adianta só trazermos livros[maisBarato], porque estamos trabalhando com objetos. Então, o que queremos comparar?

[05:05] Temos duas propriedades: a propriedade do titulo, que é do tipo *string*, e a propriedade preco, que é do tipo número. Então, temos que dizer no nosso *for* agora que o que nós vamos comparar é livros na posição [atual].preco e livros na posição [maisBarato].preco também.

[05:27] Senão o JavaScript não vai saber qual propriedade do objeto queremos usar para fazer essa comparação. No console.log(‘ o livro mais barato custa ${precosLivros} na posição [maisBarato].preco e podemos até adicionar um nome, e o título é, coloca outro *place holder* com cifrão e chaves, ${}.

[05:55] ${livros[maisBarato].titulo}, e aproveito e corrijo o meu console.log que não é mais (precosLivros[maisBarato].preco), agora é só (livros[maisBarato].preco) e (livros[maisBarato].titulo).

[06:18] Então, testamos novamente para ver se está tudo funcionando, limpar o terminal. E retornou: “o livro mais barato custa 15 e o título é PHP”. Tudo funcionando direito.

[06:33] O que nós fizemos aqui, na verdade, nós só demos uma melhorada no código para termos informações mais recheadas do que estamos trabalhando, e também começamos a extrair um pouco as partes do código para não ficar tudo nesse “index.js”.

[06:48] Vamos fazendo outros melhoramentos durante o curso, tem mais coisas para mexer, vamos extrair um pouco mais de código, separar um pouco mais de código, mas vamos fazendo isso aos poucos, vamos continuar vendo os algoritmos.