Ebook

Lógica de Programação com JavaScript

1. **ApresentaçãoTranscrição**

**Jacqueline:** Olá, boas-vindas a este curso! Eu sou Jacqueline Oliveira, engenheira de software e instrutora na Alura.

**Audiodescrição:** Jacqueline se descreve como uma mulher de pele branca, com cabelos longos e loiros lisos, olhos castanhos, vestindo uma camisa preta. Ela está sentada no estúdio da Alura, ao lado esquerdo de uma mesa preta com um notebook e um microfone pretos à sua frente, com uma parede clara ao fundo iluminada em gradiente de azul e lilás.

Porém, não estou sozinha. Estou bem acompanhada pelo Rodrigo.

**Rodrigo:** Te dou boas-vindas também! Sou Rodrigo Ferreira, um dos instrutores na escola de Programação da Alura.

**Audiodescrição:** Rodrigo se descreve como um homem de pele branca, com cabelo curto castanho, olhos castanhos, usando uma camisa azul. Ele também está sentado no estúdio da Alura, porém ao lado direito da mesa.

Lhe damos as boas-vindas a mais um curso para treinar e praticar **lógica de programação**!

**O que vamos aprender?**

**Jacqueline:** O objetivo do curso é reforçar todos os conceitos vistos em lógica de programação, como o uso de **variáveis**, **condicionais**, e **estruturas de repetição**, para consolidar os conhecimentos adquiridos.

**Rodrigo:** Exato! Para quem está começando agora a dar os primeiros passos com programação, é uma área nova, com muitos conceitos, alguns até um pouco mais difíceis, mas a melhor forma de aprender e se familiarizar é com a **prática**.

Então, nos cursos de lógica de programação, você aprenderá a desenvolver um **projeto**, e neste curso, proporemos alguns projetos distintos. Em cada aula, apresentaremos um projeto, explicaremos a ideia e você terá que tentar criar esse projeto por conta própria.

Caso tenha dificuldade ou dúvidas, também mostraremos uma solução implementando o passo a passo para que você tenha uma ideia de como seria o processo de cada um desses projetos.

**Jacqueline:** Trabalharemos com quatro novos projetos, todos com a parte de front-end já desenvolvida que disponibilizaremos para você. Não se preocupe, pois o objetivo não é mergulhar profundamente em front-end. O principal alvo é praticar a lógica, como resolver uma situação envolvida no projeto, isto é, o desafio do projeto.

**Rodrigo:** Cada projeto terá funcionalidades específicas e focaremos na parte de algoritmos, de lógica de programação com o uso de variáveis, condicionais, *loops*, *arrays* e funções, tudo o que foi visto em outros cursos e será reforçado aqui com diferentes projetos.

Assim, você poderá criar um portfólio com vários projetos, praticar bastante, e se familiarizar cada vez mais com a programação, para posteriormente prosseguir com outros cursos e formações da Alura.

**Jacqueline:** Estou ansiosa para começar! Vamos iniciar, Rodrigo?

**Rodrigo:** Vamos conhecer esses projetos e colocar a mão na massa para praticar bastante a lógica de programação.

2. Projeto do Curso

3. Apresentação do Projeto Sorteador

**Transcrição**

**Rodrigo:** Antes de começarmos o curso, tivemos uma mudança. Você deve ter percebido, estamos com um cenário diferente. Estamos nas nossas respectivas casas, pois decidimos fazer uma alteração nesse curso.

Como algumas pessoas estavam com dificuldades para aplicar os conhecimentos e resolver os desafios sozinhas, decidimos gravar uma ***aula extra***. No entanto, ***essa aula será a primeira***. A antiga primeira foi adiada, tornou-se a segunda, e assim por diante.

Nessa aula, teremos um projeto, mas ele não será um desafio para você tentar implementar sozinho. ***Vamos desenvolver um projeto para reforçar os conhecimentos*** de HTML, CSS, JavaScript, para que você consiga ter uma base mais sólida e, a partir da aula 2, consiga desenvolver os desafios por conta.

Mesmo que na formação tenhamos os dois primeiros cursos, cujo foco foi dar uma base inicial, percebemos que ainda era necessário reforçar esses conhecimentos. O nosso objetivo é esse, certo, Jacque?

**Jacqueline:** Exatamente, Rodrigo.

**Conhecendo o projeto**

Vamos começar. Trabalharemos com o projeto de um ***sorteador de números***. Quando você fizer o download do projeto, receberá a seguinte estrutura de arquivos: uma pasta "img", dentro da qual há algumas imagens, um arquivo app.js vazio, o index.html com o HTML e o style.css. É possível consultar essa estrutura pelo explorador lateral de arquivos do *VS Code*, com o projeto aberto.

Vamos conhecer o projeto e falar um pouco desses arquivos.

Estamos seguindo a linha dos cursos anteriores, então estamos usando a ***extensão Live Server*** do VS Code. Vamos clicar com o botão direito no index.html e clicar em "*Open With Live Server*". Isso abrirá o nosso projeto no navegador.

Na tela do nosso sorteador de números, vamos digitar no campo "Quantidade de números" um número que representará a quantidade de números que serão sorteados. Vamos digitar, por exemplo, um 3. Nos campos "Do número" e "Até o número", devemos escolher o número inicial e o número final do sorteio. Vamos sortear números de 1 a 10, por exemplo.

Quando clicarmos no botão "Sortear", abaixo dos três campos, ele vai mostrar o resultado — neste caso, ele sorteou 8, 7 e 4.

Você receberá esse front-end e nós vamos programar essa lógica em conjunto. Certo, Rodrigo?

**Rodrigo** É um projeto bem simples e seguindo a mesma ideia. Temos um HTML, já temos o CSS, que está com um layout bonito, com essas cores e imagens, mas o que interessa para nós é a funcionalidade.

Perceba que é um projeto bem simples. Temos um formulário com três campos. A pessoa que for usar, vai digitar os três valores e clicar no botão de sortear. Temos que ter um código JavaScript com a lógica de ler os valores digitados, o algoritmo de gerar os números aleatórios seguindo esses valores e a exibição na tela.

À direita do botão "Sortear", há um botão de reiniciar, que limpa os campos para começar um novo sorteio. Esse é o objetivo desse projeto.

Nessa primeira aula, nós vamos fazer juntos. A Jacque vai programar no computador dela, eu vou estar apoiando e você vai nos acompanhar.

**Jacqueline:** Exatamente. No VS Code, com o projeto sorteador de números abertos, vamos voltar ao explorador lateral. Se nós observarmos o interior da pasta "img", temos as imagens usadas no layout do front-end, do nosso site.

Temos utilizado projetos unindo front-end e back-end para te proporcionar um portfólio bacana e te manter mais próximo do que vemos no mercado, onde você sempre terá múltiplos projetos abertos, no front e no back. Reforçamos que você sempre trabalhará um pouco com cada um.

Na pasta "img", nós temos as imagens. Já no index.html, é onde temos a estrutura do site.

O HTML, relembrando, é uma ***linguagem de marcação***. Ele vai marcar através de *tags* o que deve acontecer e como a página deve estar estruturada. No interior do arquivo HTML temos, por exemplo, uma *tag* head (cabeçalho, em inglês), e entre a abertura e o fechamento dessa *tag*, temos referências, links e o título "Sorteador de números".

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="preconnect" href="https://fonts.googleapis.com">

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

<link href="https://fonts.googleapis.com/css2?family=Chakra+Petch:wght@700&family=Inter:wght@400;700&display=swap" rel="stylesheet">

<link rel="stylesheet" href="style.css">

<title>Sorteador de números</title>

</head>

Copiar código

No "corpo" (body) desse HTML, temos um campo label (rótulo) dentro de uma divisão, o qual possui o texto "Quantidade de números", e logo abaixo, o input que ele vai ter, cujo valor mínimo é 1.

<body>

<div class="container">

<!-- Código omitido -->

<div class="container\_\_campo">

<label class="texto\_\_paragrafo">Quantidade de números</label>

<input class="container\_\_input" id="quantidade" type="number" min="1">

</div>

<!-- Código omitido>

</body>

Copiar código

Isso corresponde exatamente ao que estamos vendo na página do navegador: um label chamado "Quantidade de números". Se pressionarmos a seta para baixo que existe no canto direito do campo "Quantidade de números", o valor diminui, mas nunca fica menor que 1.

O HTML está estruturando a página. Por isso, vamos acessá-lo muitas vezes para fazer referências e buscar esses elementos para interagir no JavaScript.

Voltando ao *VS Code*, se acessarmos o arquivo style.css que contém a parte de CSS, veremos que ele tem algumas classes, como .container\_\_campo, .container\_input e .container\_botao. Esse arquivo corresponde à parte de estilização, ou seja, doa as características e a forma como os elementos serão vistos na tela — por exemplo, qual o tamanho do ponteiro, se vai ter ponteiro, se vai estar bloqueado e que cor ele terá.

Se renomearmos esse arquivo, por exemplo, para styleteste.css, quando voltarmos ao nosso sorteador no navegador, a estilização terá sumido. Ele vai continuar funcionando porque nosso JavaScript ainda estará fazendo esse trabalho, porém o estilo não estará mais aplicado.

Você precisa entender essa divisão nitidamente: o HTML diz o que está na página, o CSS estiliza e deixa essa página bonita e o JavaScript interage dinamicamente, fazendo a lógica do negócio.

Vamos retornar o nome do arquivo alterado para style.css.

**Rodrigo:** Temos essa separação entre HTML, CSS, JavaScript, onde cada linguagem cuida de uma parte do nosso projeto. Nosso foco principal não é no HTML e no CSS, portanto, em todos os desafios, já teremos o HTML e o CSS pronto.

Nosso foco é na ***lógica de programação***, no ***código JavaScript***, que se relaciona com os eventos. Ou seja, quando clicarmos em determinado botão, queremos fazer alguma coisa. Vamos escrever o algoritmo, pegar uma variável, fazer um cálculo e alterar a página.

**Jacqueline:** Exatamente. No projeto atual, podemos observar que está tudo funcionando porque no nosso arquivo app.js, isso já está programado. Contudo, neste momento, apagaremos todo o conteúdo desse arquivo para fazermos tudo juntos, do zero. Vamos lá?

**Iniciando a construção do código**

Após esvaziar o arquivo app.js, nada mais funcionará no nosso sorteador. Se preenchermos os campos no navegador e clicarmos em "Sortear", ele não vai fazer nada e nada acontecerá.

Como decidiremos por onde começar, Rodrigo? Se queremos capturar aqueles inputs que a pessoa digitou para fazer alguma coisa com eles, qual será a nossa primeira estratégia?

**Rodrigo:** Teremos que ***escrever o código JavaScript***. Contudo, devemos lembrar de uma coisa importante: temos que olhar sempre a página HTML, porque nela está a estrutura da página, os elementos. No JavaScript, o que vamos fazer é manipular esses elementos, recuperar os campos, os seus valores e outros elementos.

Portanto, ***todo o código JavaScript se baseia no HTML***. Um bom ponto de partida seria pensar no que vai disparar a nossa lógica, que é o clique do botão "Sortear". Na nossa página HTML tem um botão "Sortear".

Quando ele for clicado, temos que chamar um código no JavaScript. Então, acho que esse é o primeiro ponto a ser considerado. Precisamos olhar no HTML onde está aquele botão de sortear e ver se ele tem uma função. Caso não tenha, devemos colocar uma função onClick (no clique) para disparar o nosso código JavaScript e começar a escrever o nosso algoritmo.

**Jacqueline:** Perfeito, Rodrigo. Enquanto você estava falando, eu estava abrindo o local do código a ser estudado, no index.html. Repare que temos uma div, que é o chute container\_\_botoes. Nela, temos dois botões, o sortear e o reiniciar. Eles têm respectivamente os IDs, btn-sortear e btn-reiniciar.

<div class="chute container\_\_botoes">

<button onclick="sortear()" id="btn-sortear" class="container\_\_botao">Sortear</button>

<button onclick="reiniciar()" id="btn-reiniciar" class="container\_\_botao-desabilitado" >Reiniciar</button>

</div>

Copiar código

Quando clicamos no botão de sortear, ele dispara uma função chamada sortear(). Isso significa que, no nosso app.js, precisaremos criar essa função. Esse é o primeiro passo. Vamos acessar o arquivo app.js pelo explorador lateral e faremos uma function em seu interior chamada sortear.

**function** **sortear**(){

}

Copiar código

O que queremos fazer quando sortearmos? Precisaremos recuperar aqueles campos. Como o Rodrigo falou, onde estão aqueles campos? Quais são os campos que queremos recuperar?

Queremos recuperar a quantidade que foi digitada dentro de três campos input do arquivo HTML, cujos IDs são quantidade, de e até, que representam os três campos do formulário.

<div class="container\_\_campo">

<label class="texto\_\_paragrafo">Quantidade de números</label>

<input class="container\_\_input" id="quantidade" type="number" min="1">

</div>

<div class="container\_\_campo">

<label class="texto\_\_paragrafo">Do número</label>

<input class="container\_\_input" id="de" type="number" min="1">

</div>

<div class="container\_\_campo">

<label class="texto\_\_paragrafo">Até o número</label>

<input class="container\_\_input" id="ate" type="number" min="1">

</div>

Copiar código

Recuperaremos esses três campos inicialmente para começar a fazer alguma coisa. Como faremos isso, Rodrigo?

Dentro da nossa function sortear(), podemos criar umas variáveis e buscar o conteúdo que tiver informado lá. Vamos fazer isso?

**Rodrigo:** Certo. Como a Jacque estava mostrando, sempre temos que olhar para o HTML. Temos que saber qual elemento queremos buscar e como vamos buscá-lo.

As *tags* do HTML têm propriedades, como id, class e várias outras. Aquela que nos ajuda a encontrar um elemento é o id, que é um identificador único. A Jacque mostrou que cada campo tem um id.

Podemos recuperar um elemento pelo seu id e guardar o conteúdo dele numa variável. Vamos relembrar como declarar uma variável , Jacque?

**Jacqueline:** Sim. Entre as chaves da função, vamos usar o let para criar uma variável. Podemos usar um nome bem intuitivo e parecido com o que queremos. Então, se queremos recuperar a quantidade, vamos criar uma variável chamada quantidade e adicionar um sinal de igual à sua frente.

**function** **sortear**(){

**let** quantidade =

}

Copiar código

O que fazemos para buscar essa variável? Como recuperamos o elemento do HTML, Rodrigo?

**Rodrigo:** Reforçando, no JavaScript já existe uma variável global — ou seja, a qual não precisamos declarar. Ela se chama document, e representa o documento. Vamos adicioná-la após o sinal de igual.

**function** **sortear**(){

**let** quantidade = document

}

Copiar código

O documento é como se fosse a página HTML. Nessa variável, podemos adicionar um ponto e chamar alguma função para ela fazer alguma coisa.

**function** **sortear**(){

**let** quantidade = document.

}

Copiar código

Existem diferentes formas de recuperar um elemento na página. Uma delas é com a função getElementById() que, como o nome diz, vai buscar um elemento pelo seu ID. Se sabemos o ID do elemento na página e queremos recuperá-lo, escreveremos document.getElementById() e passamos entre os parênteses dessa função, o ID do elemento que queremos recuperar como retorno, entre aspas simples (por se tratar de uma string).

**function** **sortear**(){

**let** quantidade = document.**getElementById**('quantidade')

}

Copiar código

**Jacqueline:** Exatamente. Como nós vimos, os IDs dos elementos que queremos são quantidade, de e ate. Primeiro, vamos pegar o quantidade e dizer que queremos pegar o valor que está contido naquele input. Para isso, vamos adicionar um .value após os parênteses e finalizar a linha com ponto e vírgula.

**function** **sortear**(){

**let** quantidade = document.**getElementById**('quantidade').value;

}

Copiar código

**Rodrigo:** Jacque, isso é importante. Quando vamos recuperar um elemento no HTML, chamamos o document.getElementById() e passamos o ID. ***Isso devolve a tag HTML***.

Se não adicionássemos o .value, estaríamos guardando na variável quantidade o elemento HTML, ou seja, a *tag* input. Contudo, não queremos o input inteiro, somente o valor que a pessoa digitou. ***O .value serve para indicar que queremos guardar o valor, e não o input***.

**Jacque:** Exatamente.

Para aplicar a mesma lógica nos outros dois campos, podemos copiar e colar a primeira linha duas vezes abaixo de si, substituindo respectivamente a string entre os parênteses de getElementById() para de e ate.

Além disso, vamos substituir o nome das duas variáveis para de e ate.

**function** **sortear**(){

**let** quantidade = document.**getElementById**('quantidade').value;

**let** de = document.**getElementById**('de').value;

**let** ate = document.**getElementById**('ate').value;

}

Copiar código

**Jacqueline:** Depois disso, vamos conferir se estamos recuperando corretamente, utilizando aquele recurso que já vimos algumas vezes, o alert.

Abaixo das variáveis, vamos pular uma linha e só dar um alert() na página para não se preocupar em exibir o resultado naquele campo de resultado por enquanto, e ver se a nossa variável vai ser exibida corretamente.

Entre os parênteses do alert, vamos adicionar a quantidade e a quantidade recuperada. Para isso, vamos usar aquele recurso para concatenar uma palavra e uma variável: um par de crases, entre as quais adicionaremos a string Quantidade: e ${quantidade}, entre cifrão e chaves, que servirá para referenciar a variável que queremos.

Com isso, queremos que ele concatene e mostre para nós o texto "Quantidade:" e a quantidade recolhida.

**function** **sortear**(){

**let** quantidade = document.**getElementById**('quantidade').value;

**let** de = document.**getElementById**('de').value;

**let** ate = document.**getElementById**('ate').value;

**alert**(`Quantidade: ${quantidade}`);

}

Copiar código

Faremos a mesma coisa para os outros campos, copiando essa linha com "Ctrl+C", e colando duas vezes abaixo de si com "Ctrl+V", substituindo respectivamente o conteúdo entre crases, de Quantidade: ${quantidade} para Do número: ${de} e Até o número: ${ate}.

**function** **sortear**(){

**let** quantidade = document.**getElementById**('quantidade').value;

**let** de = document.**getElementById**('de').value;

**let** ate = document.**getElementById**('ate').value;

**alert**(`Quantidade: ${quantidade}`);

**alert**(`Do número: ${de}`);

**alert**(`Até o número: ${ate}`);

}

Copiar código

**Jacqueline:** Já podemos ir no nosso site aberto no *Live Server* e ver se isso vai funcionar. A partir do momento que adicionarmos as quantidades e clicar em sortear, ele deve exibir o resultado para nós em formato de alert.

Voltando ao sorteador aberto no navegador, vamos preencher os campos dizendo que queremos cinco números, que vão do 2 ao 98. Quando clicarmos em "Sortear", ele vai mostrar o texto "Quantidade: 5" em formato de caixa de alerta, no topo da página. Ao clicar em "OK" nessa caixa, veremos outra com o texto "Do número: 2" e, ao clicar em "OK" novamente, veremos a terceira caixa com o texto "Até o número: 98". Podemos clicar em "OK" para fechá-la.

Conseguimos realizar a primeira parte. Recuperamos os valores para poder programar o resto da lógica, que consiste em fazer o sorteio e exibir os cinco números dentro do intervalo informado.

**04. Para saber mais: Relembrando as diferenças entre HTML, CSS E JS**

Para praticar lógica de programação de forma interessante e visual, optamos por trazer projetos completos, que irão compor seu portfólio e fazer com que a sua experiência em desenvolvimento de aplicativos seja mais próxima ao que é feito no mercado de trabalho.

Sendo assim, os projetos apresentados sempre têm três arquivos distintos em sua composição: um com extensão **HTML**, outro com extensão **CSS** e outro com extensão **JS**.

Vamos relembrar, por meio de uma atividade de um dos cursos que é pré-requisito a esse, a diferença entre esses três tipos de tecnologias:

* [Para saber mais: Diferenças entre HTML, CSS e JS](https://cursos.alura.com.br/course/logica-programacao-funcoes-listas/task/137213)

**05. Para saber mais: Entendendo o DOM**

Quando uma página da web é carregada em um navegador, esse navegador cria uma representação do **DOM** dessa página. Mas o que é esse DOM, afinal?

O DOM (*Document Object Model*) é uma representação da estrutura de um documento HTML, onde os elementos são organizados em uma árvore hierárquica e cada nó possui propriedades e métodos. Isso permite que desenvolvedores acessem, modifiquem, removam ou adicionem elementos e conteúdo HTML dinamicamente através de scripts.

É por isso que, em nosso código JavaScript, frequentemente acessamos o valor de um elemento na página por meio do objeto document, referenciando-o pelo identificador ou por outra propriedade. Um exemplo disso é o uso de document.getElementById().

No artigo [O que é o DOM?](https://www.alura.com.br/artigos/o-que-e-o-dom), da Ellen Pimentel, você poderá aprofundar seu conhecimento sobre DOM e a forma de manipulá-lo.

**06. Faça como eu fiz: Exibindo dados recuperados**

Agora é com você! Faça o mesmo procedimento que foi demonstrado no vídeo anterior, escrevendo o código para recuperar os valores inseridos nos campos ‘quantidade de números’, ‘do número’ e ‘até o número’.

Já fez todos os procedimentos para exibir os dados recuperados?

**Opinião do instrutor**

Primeiramente, você precisará declarar uma função chamada \*\*sortear\*\*, no arquivo \*\*app.js\*\*: ``` function sortear() { } ``` Agora, dentro dessa função, será necessário declarar variáveis que representam as informações recuperadas da página: ``` function sortear() { let quantidade = document.getElementById(’quantidade’).value; let de = document.getElementById('de').value; let ate = document.getElementById('ate').value; alert(`Quantidade de números: ${quantidade}`); alert(`Do número: ${de}`); alert(`Até: ${ate}`); } ``` Pronto! Abra a página do projeto no navegador e verifique se os valores serão exibidos na tela após clicar no botão.

**07. Criando a função para sortear**

**Transcrição**

**Rodrigo:** Já conseguimos dar o primeiro passo na nossa função de sorteio. Recuperamos os elementos da página HTML e fizemos um teste com um alerta para verificar se o valor estava chegando corretamente no código.

Também reforçamos a ideia de recuperar elementos na página. Para isso, precisamos conhecer a página HTML e saber qual é o elemento que queremos recuperar, seja um parágrafo, um botão, ou outro — no nosso caso, foi um input. Para recuperá-lo, precisamos ser capazes de referenciar esse elemento de alguma forma. Utilizamos o id, que geralmente é uma boa opção para recuperar um elemento.

As *tags* já tinham ids, caso contrário, poderíamos adicionar ids na *tag* input dos três inputs. Como não precisamos da *tag* no nosso código JavaScript, mas sim o valor que foi digitado, utilizamos a propriedade .value, que obtém o valor digitado na caixa de texto da página.

**Convertendo o texto em número**

Tudo funcionou bem, mas antes de continuar nosso código, acreditamos que podemos fazer uma melhoria para ter uma garantia.

Quando recuperamos um elemento com document.getElementById(), e o .value, ele vai recuperar o valor que foi digitado dentro da caixa de texto. No entanto, a caixa de texto do navegador, a *tag* input, geralmente aceita letras também. Nesse caso, ela não vai aceitar, mas poderiam ter letras ali.

Portanto, esse .value retorna uma *string* (texto). Mas no nosso algoritmo, sabemos que são números. Então, podemos usar uma função no JavaScript para forçar que essa variável seja do tipo numérico.

**Jacqueline:** Exatamente. Acessando as linhas em que declaramos as variáveis, vamos utilizar, a função parseInt(), de modo que seus parênteses envolvam o document.getElementById().value.

Temos também a parseFloat, que poderíamos usar para declarar que é um tipo float, mas vamos usar parseInt para indicar que vai chegar um número do tipo inteiro.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

**alert**(`Quantidade: ${quantidade}`);

**alert**(`Do número: ${de}`);

**alert**(`Até o número: ${ate}`);

}

Copiar código

Se posicionarmos o cursor em cima de uma das variáveis antes de adicionar o parseInt(), a IDE exibirá uma caixa de mensagem informando que a variável é do tipo any (qualquer), ou seja, pode vir qualquer coisa. Após o parseInt(), a IDE exibirá o tipo number, sendo, portanto, capaz de entender que virá um número.

**Rodrigo:** Temos, então, mais uma função, o parseInt(). Ela recebe como parâmetro o texto ou o elemento no qual faremos *parse* (conversão) para inteiro. Vamos fazer isso para as três variáveis ali, a quantidade, o de e o até.

Com isso, nosso código recupera o elemento e adiciona dentro do parseInt(), garantindo que o retorno será do tipo inteiro e não mais *string* no código JavaScript.

**Implementando o sorteio**

**Jacqueline:** Mas agora queremos ver o sorteio. Colocamos lá do número 2 ao 98, e queremos saber como vamos sortear esse número. Não precisamos reinventar a roda e ficar pensando como fazer esse sorteio, pois certamente temos funções prontas para essa tarefa.

**Rodrigo:** Esse é o próximo passo do nosso algoritmo. Já temos os valores, agora temos que fazer a parte mais difícil, que é escolher um número aleatório entre o de e o até. Podemos começar a fazer uma nova função para ter essa lógica, mas não precisamos implementá-la do zero, conseguimos pesquisar uma solução pronta.

**Jacqueline:** Exatamente. Abaixo das chaves da função sortear(), vamos criar uma nova função com o function, e podemos chamá-la de obterNumeroAleatorio, porque é isso que ela fará.

**function** **sortear**(){

// Codigo omitido

}

**function** **obterNumeroAleatorio**() {

}

Copiar código

Vamos ter que passar para essa função aquele intervalo entre o valor mínimo e o máximo como parâmetro, para ele nos dar um número dentro daquele intervalo. Para isso, vamos colocar um min e um max entre parênteses.

**function** **sortear**(){

// Codigo omitido

}

**function** **obterNumeroAleatorio**(min, max) {

}

Copiar código

**Rodrigo:** Criamos uma função separada, cujo objetivo é apenas gerar um número aleatório entre o mínimo e o máximo.

Na nossa função principal, a sortear(), podemos criar uma variável e guardar o retorno dessa função nessa variável. Onde temos os três alertas seguidos, vamos apagar os três e criar em seu lugar uma variável numero que vai receber o retorno da nova função. Portanto, vamos adicionar o igual na variável numero e vamos chamar o obterNumeroAleatorio(), passando entre parênteses o que foi capturado no de e no ate.

Na linha de baixo, para conferir, vamos fazer um alert, mostrando esse numero que foi sorteado.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

**let** numero = **obterNumeroAleatorio**(de, ate);

**alert**(numero);

}

**function** **obterNumeroAleatorio**(min, max) {

}

Copiar código

**Jacqueline:** Construímos a estrutura da função nova e fizemos sua chamada na função principal. Agora vamos construir a parte da inteligência, ou seja, a forma com a qual obtemos esse número aleatório.

**Rodrigo:** Para isso, vamos pesquisar. Em vez de implementar um algoritmo do zero, no qual temos dois números e uma lógica que gera um número aleatório entre eles, podemos pesquisar na internet, pois esse é um cálculo muito comum.

Vamos pesquisar no Google se tem algum site, blog ou fórum com essa informação, porque, com certeza, pessoas já precisaram fazer isso.

números aleatórios em javascript

Copiar código

Ao digitar a busca acima, encontramos duas páginas que indicam a função Math.random(). Uma delas é da Alura, a qual referenciaremos abaixo para que você acesse e conheça.

* [Acesse o artigo da Alura *A função Math.random() no JavaScript na criação de números pseudoaleatórios.*](https://www.alura.com.br/artigos/funcao-math-random-javascript-numeros-pseudoaleatorios?utm_term=&utm_campaign=&utm_source=adwords&utm_medium=ppc&hsa_acc=7964138385&hsa_cam=20987928442&hsa_grp=157916200306&hsa_ad=689395782879&hsa_src=g&hsa_tgt=dsa-19959388920&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gad_source=1&gclid=CjwKCAiAq4KuBhA6EiwArMAw1BQf5rfBLtmVJrY63CBCvjgij3w6l2KDeeXA_6oC71g_S14O1aboKhoCZAcQAvD_BwE)

No momento, vamos utilizar a [página que explica a função Math.random(), do site Developer Mozilla](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math/random).

Acessando essa página, vemos a indicação da função mencionada, que serve para pegar um número aleatório. A página mostra para nós como poderíamos escrever essa função, com um exemplo na [seção "Gerando um número aleatório entre dois valores"](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math/random#gerando_um_n%C3%BAmero_aleat%C3%B3rio_entre_dois_valores).

**function** **getRandomArbitrary**(min, max) {

**return** **Math**.**random**() \* (max - min) + min;

}

Copiar código

Vamos começar a "brincar" com isso e ver como vai ficar. No código acima, selecionaremos e copiaremos a linha return Math.random() \* (max - min) + min.

Voltando ao código, vamos colar entre as chaves de obterNumeroAleatorio(), e ver se vai funcionar.

**function** **obterNumeroAleatorio**(min, max) {

**return** **Math**.**random**() \* (max - min) + min;

}

Copiar código

Voltando ao nosso sorteador aberto no navegador, colocaremos que queremos um número entre 2 e 40, 10, por exemplo. Quando clicarmos em "Sortear", ele exibirá a primeira caixa de diálogo da resposta com números quebrados, com várias casas decimais.

6.916911577052566

O número não veio como inteiro, e não é exatamente assim que queremos. Nesse caso, queremos que seja exibido o número 7.

Nesse caso, temos que voltar à página da documentação para analisar. Em seu interior, na seção "Gerando um número aleatório entre dois valores", da qual copiamos o código, encontramos a seção "Gerando um número inteiro aleatório entre dois valores", que tem um Math.floor() para fazer um ajuste e não mostrar o número quebrado.

Ele vai retornar um número maior do que o min, mas nunca igual ao max, ou seja, ele nunca será igual ao número final. Contudo, esse ainda não é o código que queremos.

Desceremos mais a página até encontrar a [seção "Gerando um número inteiro aleatório entre dois valores, inclusive"](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math/random#gerando_um_n%C3%BAmero_inteiro_aleat%C3%B3rio_entre_dois_valores_inclusive). Vamos utilizar essa função porque queremos um valor entre dois números, incluindo o último possível. Se colocarmos de 2 a 10, por exemplo, queremos permitir que ele também retorne o 10.

**function** **getRandomIntInclusive**(min, max) {

min = **Math**.**ceil**(min);

max = **Math**.**floor**(max);

**return** **Math**.**floor**(**Math**.**random**() \* (max - min + 1)) + min;

}

Copiar código

Vamos copiar a linha return Math.floor(Math.random() \* (max - min + 1)) + min no código de exemplo dessa função. Podemos usá-la direto, sem o ceil() e o floor() anteriores.

**return** **Math**.**floor**(**Math**.**random**() \* (max - min + 1)) + min;

Copiar código

Vamos colar essa linha no lugar da outra, entre as chaves da nossa função de sorteador. O novo código possui um pequeno ajuste em relação ao anterior.

**function** **obterNumeroAleatorio**(min, max) {

**return** **Math**.**floor**(**Math**.**random**() \* (max - min + 1)) + min;

}

Copiar código

**Rodrigo:** Essa é a versão otimizada para o nosso cenário. Queremos que ele sorteie um número inteiro, não um número decimal, e que ele leve em consideração o mínimo e o máximo de maneira inclusiva.

Como foi mostrado, o site que acessamos possui uma documentação explicando a função Math.floor(). Perceba que já tem uma função que faz o que queremos, utilizando o Math.random() para escolher um número aleatório.

Portanto, não precisaríamos tentar fazer do zero essa lógica de gerar número aleatório, pois já temos isso pronto no JavaScript. Só precisamos recolher alguns exemplos no tutorial e usar o mais próximo do que precisamos para o nosso projeto.

**Jacqueline:** Vamos testar para ver se com essa alteração veremos o retorno sem aquele número quebrado. Voltando ao nosso sorteador aberto no navegador, colocaremos o valor 1 em "Quantidade de números", e os valores de 2 a 10 novamente. Quando sortearmos, a janela de diálogo retornará o número 8, um valor inteiro, conforme desejado.

Contudo, se alterarmos o valor de "Quantidade de números" para 5, ele continuará retornando um único valor. Vamos precisar fazer essa chamada ao método obterNumeroAleatorios() dentro de alguma repetição.

**Rodrigo:** Sim. Vamos sortear uma vez e usar aquela variável quantidade que, até então, não estava sendo usada para nada. Vamos ter que fazer uma repetição, ou seja, um *loop* (laço) para sortear um número de acordo com a quantidade que a pessoa digitar no campo "Quantidade de números".

**Jacqueline:** Exatamente. Para isso, podemos apagar as linhas let numero = obterNumeroAleatorio(de, ate) e alert(numero), entre as chaves da função sortear(), para recomeçar a estrutura.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

}

Copiar código

Vamos sortear com uma estrutura de repetição e depois vamos guardar isso em algum lugar para exibir. Para isso, vamos um *array* (arranjo). Não vamos ter um número só, mas sim um *array* onde haverá vários números — quantos a pessoa desejar.

No lugar do código removido, vamos criar uma variável sorteados que vai representar esse *array*. Vamos fazer um let sorteados e adicionar um sinal de igual. Para declarar o nosso *array*, vamos abrir e fechar os colchetes.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

**let** sorteados = [];

}

Copiar código

Qual é a estrutura de repetição que você nos recomenda, Rodrigo?

**Rodrigo:** Podemos utilizar um laço for, que vai percorrer do primeiro ao segundo número que a pessoa digitou. Se ela digitou 5, é de 1 a 5, se ela digitou 20, de 1 a 20. O laço for é um exemplo que pode ser usado para fazer a lógica de percorrer determinados elementos e fazer alguma coisa nesse intervalo percorrido.

**Jacqueline:** Vamos fazer isso aqui. Abaixo de let sorteados = [], vamos pular uma linha e fazer o for(). Entre seus parênteses, vamos colocar um let e a variável de iteração que, por padrão, chamamos de i. Nós a iniciamos com o valor 0, adicionamos um ponto e vírgula e o comando i < quantidade para dizer "enquanto i for menor do que a quantidade que a pessoa digitou.

Por fim, adicionamos outro ponto e vírgula e o comando i++ para incrementar a variável i de 1 em 1.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

**let** sorteados = [];

**for** (**let** i = 0; i < quantidade; i++) {

}

}

Copiar código

Entre as chaves desse for, aí sim, precisamos fazer atribuir obterNumeroAleatorio() a uma variável e colocar o valor dessa variável no *array*.

**Rodrigo:** Criamos um laço que vai de zero até a quantidade que a pessoa digitou. A estrutura do laço for tem o formato visto acima. Criamos uma variável i que vai de 0 até quantidade para ser o controlador do laço e evitar que ele execute infinitamente.

É dentro do laço que precisamos chamar o obterNumeroAleatorio(), e o número de vezes que vamos chamar essa função vai depender da quantidade que a pessoa digitou e que vamos usar no controle do for.

**Jacqueline:** Acima do laço, vamos criar a variável numero com let numero.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

**let** sorteados = [];

**let** numero;

**for** (**let** i = 0; i < quantidade; i++) {

}

}

Copiar código

Dentro das chaves do laço, por sua vez, vamos colocar o numero, que receberá obterNumeroAleatorio(de, ate). Ou seja, ele vai fazer isso quantas vezes forem necessárias.

Desceremos uma linha, na qual precisamos fazer mais alguma coisa. Uma vez que obtivemos o número, precisamos adicioná-lo ao nosso *array*. Para isso, vamos adicionar um sorteados.push() para colocar o valor. Então, vamos colocar numero entre parênteses.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

**let** sorteados = [];

**let** numero;

**for** (**let** i = 0; i < quantidade; i++) {

numero = **obterNumeroAleatorio**(de, ate);

sorteados.**push**(numero);

}

}

Copiar código

Teoricamente, se a pessoa escolher cinco números, ele vai rodar cinco vezes, chamando o número aleatório e guardando-o no *array*. E depois, para conferir, podemos imprimir esse *array*. Abaixo das chaves do for, vamos pular uma linha e fazer um alert() desse array sorteados.

**function** **sortear**(){

**let** quantidade = parseInt(document.**getElementById**('quantidade').value);

**let** de = parseInt(document.**getElementById**('de').value);

**let** ate = parseInt(document.**getElementById**('ate').value);

**let** sorteados = [];

**let** numero;

**for** (**let** i = 0; i < quantidade; i++) {

numero = **obterNumeroAleatorio**(de, ate);

sorteados.**push**(numero);

}

**alert**(sorteados);

}

Copiar código

Vamos para o nosso sorteador aberto no navegador. Vamos pedir para escolher 5 números, de 1 a 20, e clicaremos em "Sortear". O resultado será exibido na caixa de alerta no topo da página.

1, 2, 14, 4, 8.

Os números do resultado estão dentro da faixa, sempre antes de 20. Isso significa que o código funciona corretamente.

Até o momento, cumprimos nossa missão. Recuperamos os valores, fizemos uma função para obter um número aleatório e fizemos um *loop* para devolver os números aleatórios selecionados no campo "Quantidade de números".

A seguir, vamos melhorar nossa aplicação, exibindo no label os números sorteados e fazendo alguns outros tratamentos para torná-la mais coesa.

**Rodrigo:** Perfeito. Conseguimos fazer o segundo passo, que era gerar os números aleatórios. Conforme o alert que criamos para exibir o resultado, a nossa aplicação está funcionando bem.

Devemos reforçar que ***copiar funções de sites de documentações é bem comum no nosso dia a dia***. Futuramente, você vai precisar fazer alguma lógica ou algoritmo que outras pessoas já precisaram. Nesses casos, você pode pesquisar na internet por sites e fóruns ou usar ferramentas de inteligência artificial, como o *ChatGPT*.

Se você conhece o *ChatGPT*, você pode pedir a ele que gere uma função de sortear números aleatórios. Obviamente, ***nem sempre a ferramenta acerta***, então, é bom sempre testar para ver se o código gerado por ela vai funcionar.

A seguir, continuaremos o nosso projeto.

**08. Para saber mais: A função Math.random()**

A função Math.random() em JavaScript é uma função integrada que gera um número pseudoaleatório decimal entre 0 (inclusivo) e 1 (exclusivo). Isso significa que o número gerado estará em um intervalo de [0, 1), onde 0 é incluído, mas 1 não é.

Quando você chama Math.random(), obtém um número aleatório como resultado. Exemplo de código:

**let** numeroAleatorio = **Math**.**random**();

console.**log**(numeroAleatorio); // Isso imprimirá um número aleatório entre 0 (inclusivo) e 1 (exclusivo)

Copiar código

Esse recurso é frequentemente utilizado em diversas aplicações para gerar números aleatórios que podem ser usados em jogos, simulações, sorteios, geração de senhas aleatórias, entre outras funcionalidades onde a aleatoriedade é necessária.

Para criar números aleatórios em um intervalo diferente de [0, 1), você pode manipular o resultado de Math.random() multiplicando-o e adicionando valores conforme necessário, exatamente como fizemos em vídeo no nosso projeto **Sorteador de números**.

Para saber mais sobre essa função, não deixe de ler o artigo [A função Math.random() no JavaScript na criação de números pseudoaleatórios](https://www.alura.com.br/artigos/funcao-math-random-javascript-numeros-pseudoaleatorios), que contém exemplos de utilização e manipulação do valor.

**09. Exibindo os números sorteados**

**Rodrigo:** Conseguimos completar mais uma etapa. Agora nosso algoritmo é capaz de gerar números aleatórios. Adicionamos um alerta para exibir na tela e verificamos que tudo funcionou corretamente. Até pesquisamos uma função para gerar números aleatórios e a implementamos no site. Inclusive, poderíamos ter usado inteligência artificial nessa parte.

Nosso próximo passo é não exibir os números na página, ao invés de um alerta. Embaixo dos botões, há um texto onde pretendemos exibir esses valores.

**Exibindo os valores em forma de texto**

**Jacqueline:** Exatamente. Portanto, precisaremos conferir novamente o HTML para descobrirmos o elemento que precisamos buscar. No site, temos o texto "**números sorteados**", que é precisamente onde queremos colocar esses cinco números que estão aparecendo em forma de alerta.

Vamos novamente ao index.html, onde verificaremos onde está este texto. Ele está na linha 43, dentro da <label class="texto\_\_paragrafo">: "Números sorteados: nenhum até agora". Portanto, temos o elemento com o id resultado, e nele temos uma label com esse texto. Teremos que recuperar esse o resultado e colocar exatamente a tag da label nesse elemento. Inclusive, copiaremos a linha da label, para adicioná-la, e voltaremos para o arquivo app.js.

<label class="texto\_\_paragrafo">Números sorteados: nenhum até agora</label>

Copiar código

O **primeiro passo** é declarar o resultado para conseguirmos obter esse elemento. Então, no final da function sortear(), abaixo do for() criaremos a variável resultado codando: let resultado = document.getElementById('resultado').

**Rodrigo:** Nessa situação, queremos fazer o contrário da anterior. Anteriormente estávamos recuperando os elementos e recuperando o que a pessoa digitou na tela. Agora, não queremos ler uma informação da tela, **queremos colocar uma informação na tela**.

Temos a nossa variável resultado, no código JavaScript, e precisamos passar essa informação para página, concatenar dentro da página. Então, é o caminho inverso. Sendo assim, **não vamos usar o .value**, porque esse elemento que recuperamos não é uma tag input. **Usamos o document.getElementById()**, porque queremos o elemento HTML. Aquela tag era uma div, e nela vamos passar a tag label.

**Jacqueline:** Então, obtemos o elemento resultado do HTML e passamos para a variável resultado no JavaScript. Para colocar esse elemento na página HTML, na linha abaixo escrevemos resultado.innerHTML. Com o .innerHTML, conseguimos passar o elemento na página, que será justamente a linha que copiamos.

Porém, ao invés do texto "Números sorteados: nenhum até agora", mudaremos para Números sorteados: ${sorteados}, passando a array dos números sorteados. Contudo, como estamos concatenando com outra variável, precisamos deixar toda a tag <label> </label> entre crases.

Arquivo app.js

**function** **sortear**(){

//código omitido

**let** resultado = document.**getElementById**('resultado')

resultado.innerHTML = `<label class="texto\_\_paragrafo">Números sorteados: ${sorteados}</label>`

}

**function** **obterNumeroAleatorio**(min, max) {

**return** **Math**.**floor**(**Math**.**random**() \* (max - min + 1)) + min;

}

Copiar código

Teoricamente, esse código é o suficiente para exibirmos na tela. Já tiramos o alert(). Então, vamos testar lá se vai funcionar. Abrindo a página do sorteador, preencheremos os campos, indicando que queremos 5 números entre 1 e 10. Ao clicarmos no botão "Sortear", o resultado aparece na tela: "Números sorteados: 3, 5, 8, 3, 2". Deu certo.

Quantidade de números: 5

Do número: 1

Até o número: 10

Números sorteados: 3, 5, 8, 3, 2

**Conhecendo o .textContent**

**Rodrigo:** Percebemos que **para lermos um elemento da página, chamamos o .value**, se for um campo de texto. **Para colocarmos uma informação na página, temos o .innerHTML**. Isso é algo que vamos fazer bastante nos desafios das próximas aulas, então você precisará fazer essa lógica.

Tem uma outra propriedade também, que é o **.textContent, que é quando queremos colocar um texto dentro de um elemento na página**. Seria equivalente ao que fizemos, ou seja, podemos usar o .textContent ou o .innerHTML.

Geralmente, **usamos o .textContent quando temos um texto simples**: uma frase, uma palavra, uma *string*, um número. No caso que fizemos, não temos um texto simples, e sim uma tag HTML inteira. Então, tem a tag, label, ID e outras coisas, então usamos o .innerHTML, porque queremos colocar um trecho de código HTML internamente nesse elemento.

**Dica:** Para exibir textos simples na tela, podemos usar o .textContent, porém, quando precisamos manipular o conteúdo HTML de um elemento interno, usamos o .innerHTML.

**Corrigindo o sorteio de números repetidos**

**Rodrigo:** Deu certo, mas temos um problema nos números que foram sorteados.

**Jacqueline:** Reparei também, Rodrigo. Quando diminuí o range de 1 para 10, ele gerou números repetidos: **ele sorteou o 3 duas vezes** e não queremos sorteio repetido. Imagina, duas pessoas ganhando o mesmo presente? Isso é problema de família, não dá. Então, precisamos encontrar uma maneira de proteger quando o número tiver sido sorteado, para ele não ser sorteado novamente. Então, vamos fazer um ajuste no nosso método.

Acessando novamente o arquivo app.js, dentro da função sortear(), temos o for(). Nesse for(), temos a variável numero, que recebe o obterNumeroAleatorio().

**for** (**let** i = 0; i < quantidade; i++) {

numero = **obterNumeroAleatorio**(de, ate);

sorteados.**push**(numero);

}

Copiar código

Antes de efetivamente passar os números aleatórios para a array, podemos consultar se o número já existe na array. Se já estiver, mandamos chamar outro número aleatório, até que ele consiga gerar um número que ainda não está lá.

**Rodrigo:** Nessa função, obterNumeroAleatorio(), que até copiamos daquele site, passamos o de e o ate para ele gerar, porém, ele não sabe se aquele número já foi gerado antes. A única coisa que ele faz é gerar os números. Se há números repetidos que não queremos, precisamos reescrever o código para fazer esse controle.

Precisamos fazer um outro loop, porque já temos um loop for() que está fazendo essa lógica de 0 até a quantidade que a pessoa digitou. Nesse outro loop, enquanto o número que pedimos para ele gerar já tiver sido sorteado, ele precisa sortear outro, até gerar um número que não foi sorteado no momento. É isso que faremos.

**Jacqueline:** Exatamente. Como conhecemos vários métodos de repetição, podemos usar o **while()**, que é justamente essa palavra-chave **enquanto**. Enquanto o número sorteado já estiver na array, temos que sortear outro.

**for** (**let** i = 0; i < quantidade; i++) {

numero = **obterNumeroAleatorio**(de, ate);

**while** (sorteados.**includes**(numero))

sorteados.**push**(numero);

}

Copiar código

Então, no while(), para verificarmos se o número já está na array, usamos o **sorteados.includes()**, passando o numero entre os parênteses.

**Rodrigo:** Então, sorteados é a nossa variável array, e **todo array no JavaScript tem essa função chamada includes(), que devolve um booleano**: *true* (verdadeiro) ou *false* (falso). O que esse while() está dizendo é: "enquanto a variável sorteados tem incluído esse numero". Enquanto for *true*, vamos entrar no nosso loop e pedir para ele fazer um novo sorteio. Porque, se isso der verdadeiro, é porque aquele número que sorteado já está dentro do nosso array, o que não queremos.

**Jacqueline:** Então vamos fazer outro sorteio. Para isso, dentro do while(), chamamos novamente o numero recebendo o obterNumeroAleatorio(de, ate);. E ele ficará fazendo esse loop até que todos os números sejam exclusivos.

**for** (**let** i = 0; i < quantidade; i++) {

numero = **obterNumeroAleatorio**(de, ate);

**while** (sorteados.**includes**(numero)) {

numero = **obterNumeroAleatorio**(de, ate);

}

sorteados.**push**(numero);

}

Copiar código

Vamos testar para ver se agora ele não vai mais repetir? Voltamos ao sorteador e pedimos, mais uma vez, 5 números de 1 a 10. Ao clicarmos no botão "Sortear", temos: 8, 9, 3, 10, 2. Se clicarmos em "Sortear" novamente, aparece: 7, 1, 8, 9, 3. Fazendo isso mais algumas vezes, notamos que já parou de repetir números, ou seja, conseguimos corretamente proteger para não termos número repetido, fazendo esse loop dentro do for().

Parece uma coisa complexa, por termos um loop dentro de outro, mas não, faz todo sentido. Exatamente o que você falou: não podemos ficar alimentando um array enquanto esse já existir aquele número ali dentro.

**Conclusão**

**Rodrigo:** E perceba, então, que o array, que é um objeto do JavaScript, nos ajuda. Ele tem várias funções, e usamos duas delas: o includes e o push. Com o includes(), passamos um elemento e ele indica se aquele elemento já existe dentro do array, devolvendo um booleano: *true* ou *false*. E o push "empurra", ou seja, coloca o elemento dentro do array.

Há outros métodos também no array, que você pode pesquisar na documentação, para quando tiver alguma situação em que deseja fazer algo com um array. Por exemplo, colocar ou remover um elemento, verificar se um elemento já existe ou descobrir quantos elementos tem no array. **Para todas essas situações, tem funções no array que podemos utilizar**.

Então, conseguimos concluir mais uma etapa. Nosso sorteador está sorteando os números e está exibindo na tela, evitando repetição. A princípio, terminamos o nosso projeto, mas está faltando o botão "Reiniciar", que no momento não faz nada.

**Jacqueline:** Pois é. Ele está com uma propriedade que nem nos permite clicar em "Reiniciar". É algo que deveríamos poder fazer, já que tivemos um sorteio. Após acabar o sorteio e exibir o resultado, deveríamos poder reiniciar esse sorteio.

**Rodrigo:** Então, no próximo vídeo, vamos implementar o último botão, e a última lógica, para fechar o nosso projeto de sorteador de número secreto.