**01Apresentação**

Mônica: Olá, estudante! Boas-vindas ao curso de Lógica de Programação. Sou a Mônica Hillman e serei sua instrutora!

Audiodescrição: Mônica é uma mulher de pele clara. Tem olhos e cabelos castanhos, mechas loiras e franja. Usa um óculos de grau com armação redonda e piercing no septo. Está com uma camiseta rosa e usa um microfone de mesa. Ao fundo, uma iluminação da cor azul.

Guilherme: Boas-vindas! Que bom te encontrar nesse curso de lógica de programação. Eu sou o Guilherme Lima e também serei seu instrutor.

Audiodescrição: Guilherme é um homem de pele clara. Tem olhos castanhos, cabelos curtos e barba também castanhos. Usa óculos quadrados de armação preta e está com uma camisa cinza. Ao fundo, uma parede lisa e ao lado direito uma prateleira com livros.

Guilherme: Nesse curso vamos aprender o motivo pelo qual a Lógica de Programação é importante para nossa carreira como pessoas desenvolvedoras de software. Conheceremos conceitos comuns para a maioria das linguagens.

Mônica: Para isso, desenvolveremos um projeto, o jogo do número secreto.

Nele, a pessoa usuária precisará inserir alguns números para adivinhar qual foi o número aleatório gerado pelo computador.

Guilherme: Para desenvolver esse jogo, trabalharemos com variáveis, loops, condicionais, if e else. Conceitos usados no mercado de trabalho.

Além disso, mostraremos como o mercado escreveria este código que também estamos desenvolvendo, seguindo boas práticas de programação.

Vamos começar?

# 02Preparando o ambiente

O principal objetivo desta atividade é preparar o ambiente de desenvolvimento que vamos utilizar durante este curso. Nela, mostramos o que é necessário instalar, disponibilizamos alguns arquivos de código ou material necessário para garantir a experiência de aprendizado incrível e sustentável.

Material necessário

Para acompanhar este curso, clique neste link e faça o download do projeto base. Saber trabalhar com um código que foi desenvolvido por outras pessoas é de extrema importância para a carreira de desenvolvimento de software.

Nos projetos da maioria das empresas, é muito comum colaborar em partes de código que não foram escritas estritamente por você. Nesses momentos, a capacidade de compreender o raciocínio por trás do código, sua estrutura e funcionamento é crucial para fazer alterações, corrigir bugs e adicionar novos recursos de forma eficiente.

Mas… onde vamos escrever nossos códigos?

Quando pensamos em desenvolver códigos e criar programas de software, temos ferramentas que nos auxiliam nessa criação e edição. Uma delas, é o Visual Studio Code, que é um editor de código gratuito e de código aberto desenvolvido pela Microsoft. Ele é usado por pessoas desenvolvedoras de todo o mundo com a finalidade de escrever e editar código para uma variedade de plataformas e linguagens.

Faça o download do Visual Studio Code de acordo com seu sistema operacional (Windows, Linux ou Mac OS) neste link.

Como Abrir um Projeto no Visual Studio Code?

Localizar e Descompactar o Projeto: Navegue até o local onde você salvou o projeto baixado em 'Material Necessário'. Se o arquivo ainda estiver compactado, clique com o botão direito sobre ele e escolha 'Extrair aqui' ou 'Descompactar', dependendo do seu sistema operacional.

Selecionar a Pasta do Projeto no Visual Studio Code: Após descompactar, abra o Visual Studio Code. No menu superior, você encontrará diversas opções, como 'Arquivo', 'Editar', 'Selecionar'. Escolha 'Arquivo' ou ‘File’ em inglês. Clique em 'Arquivo' e, em seguida, escolha 'Abrir pasta' ou ‘Open Folder’. Isso abrirá uma janela do explorador de arquivos do seu sistema operacional, selecione a pasta do projeto e clique em 'Abrir'. Isso carregará todo o conteúdo do projeto no Visual Studio Code.

Sabendo disso, vamos iniciar nossos estudos em lógica de programação?

# 03Carregando o projeto base

Guilherme: Vamos iniciar nossos estudos!

Fizemos o download do projeto básico Games.js que utilizaremos nesse curso. Por padrão, existem diversas ferramentas que podemos usar para criar o nosso código, editá-lo, compilá-lo e fazer várias outras coisas. Usaremos uma bem conhecida no mercado: o Visual Studio Code.

Na atividade anterior a esse vídeo, você encontra o passo a passo necessário para rodar os códigos do curso. Além disso, também temos conteúdos que comparam outros editores de código e o motivo pelo qual preferimos o VS Code.

Guilherme: Nesse temos alguns arquivos.

img, vamos usá-lo para editar as imagens do projeto;

app.js, que ainda não tem nenhuma linha de código;

index.html, referente a parte visual que vamos explorar em breve;

style.css, os estilos.

Tudo isso foi criado pelo time da Moni.

Mônica: Isso mesmo, Gui! Como o HTML e o CSS não são linguagens de programação e esse é o foco desse curso, focaremos no app.js.

Entendendo o Jogo do Número Secreto

Guilherme: Na pasta do projeto, clicamos com o botão direito no arquivo index.html e abri-lo no navegador. Feito isso, na tela visualizamos o desenho de um robô e ao lado o texto "Você acertou! Você descobriu o número secreto".

Vamos entender melhor o que faremos nesse curso. Para isso, darei uma folha escrita "número secreto" e uma caneta para que a Môni escolha um número entre 1 e 100 sem me mostrar.

Mônica: Pronto!

Guilherme: Agora, vou fazer alguns chutes. O número secreto é 42?

Mônica: Não, é menor!

Guilherme: É 20?

Mônica: É maior que 20.

Guilherme: Então é 30?

Mônica: Na verdade, é menor.

Guilherme: 25, 28,29?

Mônica: É 29!

Guilherme: Descobrimos o número secreto! Nosso desafio é criar um jogo de adivinhação, exatamente como fizemos.

Para isso, será necessário que o programa escolha um número secreto e a gente interaja com a pessoa que jogará, sempre perguntando qual número ela acha que é. Nisso, a pessoa irá fazer os chutes e daremos dicas se o número é maior ou menor.

Começando a criação do código

Então, no Visual Studio Code vamos começar a escrever o código. Apertamos Ctrl+ B para visualizarmos apenas o app.js.

No navegador, repare que quando o jogo começa não é exibido nada. Queremos exibir uma mensagem na tela. Na programação existem diversos comandos, funções, atalhos e ferramentas que utilizamos para resolver esse desafio.

Usaremos um código em Javascript, vamos entender um pouco sobre os padrões e como ele funciona.

Para passar uma mensagem para a pessoa que iniciou o jogo, usaremos a palavra alert seguido de parênteses.

Lembrando que usamos alert e não alerta, pois o Javascript foi desenvolvido utilizando o idioma inglês.

Ao fazer isso, repare que o cursor fica entre os parênteses. Isso significa que podemos passar uma mensagem nesse espaço. Porém, existe uma regra.

Para passar essa mensagem, adicionamos aspas.

Mônica: No caso do Javascript podemos usar tanto aspas simples como duplas. Sendo assim, a única preocupação é em manter sempre uma padronização em todo o código. Se você começar usando aspas simples, mantenha em todo o projeto.

Guilherme: Em algumas linguagens, quando passamos uma mensagem é preciso utilizar aspas duplas, já outras permitem as aspas simples. No Javascript usaremos aspas simples.

Em seguida, passaremos uma mensagem simples.

Mônica: Podemos colocar a mensagem mais tradicional para as pessoas que estão iniciando em programação, "Olá Mundo".

Guilherme: Ótimo! Feito isso, no fim da linha adicionamos ;.

No Javascript, o ponto e vírgula é inserido em um momento específico de interpretação. Esse não é nosso foco agora. No mundo real do desenvolvimento, existem ferramentas que inserem o ponto e vírgula quando nós não o fazemos.

Como não usaremos nada disso, usaremos o ; durante todo o curso. Então, Moni, se eu esquecer, por favor, me lembre.

alert('Olá mundo');

COPIAR CÓDIGO

Mônica: A falta do ponto e vírgula não daria erro no código, mas cada linguagem tem sua particularidade e no Javascript, para identificar que o comando terminou é recomendado utilizar o ponto e vírgula.

Guilherme: Agora, salvamos o projeto apertando o comando Ctrl + S.

Em seguida, acessamos o aplicativo no navegador e atualizamos apertando "F5". Feito isso, aparece uma mensagem escrita "Olá, Mundo". Clicamos no botão "Ok" e a tela exibe a imagem do robô.

Ficou melhor! Mas, trocaremos a mensagem para "Boas-vindas ao jogo do Número Secreto". Novamente apertamos Ctrl + S para salvar.

alert('Boas-vindas ao jogo do Número Secreto');

COPIAR CÓDIGO

Lembre-se sempre de salvar todas as alterações de código!

Atualizamos a página e visualizamos a nova mensagem. Clicamos em "Ok" e a tela é exibida.

Mônica: Daremos continuidade no vídeo seguinte. Até lá!

# 04Exibindo mensagens na tela

**Guilherme:** Môni, o número que você havia escrito era mesmo 29?

**Mônica:** Sim!

**Guilherme:** Que bacana! Te pedi para memorizar esse número, pois precisaremos desse mesmo conceito na programação. Não queremos que o computador mude o número ou esqueça dele, certo?

**Mônica:** O jogo deve ser justo.

**Guilherme:** Exatamente! Então, para conseguirmos armazenar informações no programa, utilizamos um conceito chamado **variáveis**.

**Criando variáveis**

**Guilherme:** Em cada linguagem de programação, há uma forma diferente de criar uma variável. No Javascript, para criarmos variáveis e armazenar valores nelas, usamos a palavra-chave let.

Escrevemos isso no app.js, na linha abaixo de alert(). Isso significa que reservaremos um espaço na memória para guardar uma informação.

O conteúdo que queremos armazenar é o número secreto. Para isso, existe um padrão para declarar as variáveis, então, após let, escrevemos numeroSecreto. Sem acento e com a primeira letra da segunda palavra em maiúsculo.

**Mônica:** Também precisamos considerar o sentido prático do nome da variável e de outros conceitos que aprenderemos no decorrer do curso. Poderíamos colocar simplesmente número, mas seria mais útil usar NúmeroSecreto, sugerindo que esse é o número que queremos acertar no jogo.

**Guilherme:** Ótimo ponto! Quando pensamos no desenvolvimento de software de um programa, os nomes que atribuímos, especialmente para as variáveis e para outros conceitos, precisam ser claros.

As outras pessoas que forem trabalhar com esse projeto precisam saber o que está acontecendo no código.

**Mônica:** Até porque teremos muitos números no código, como os usados na tentativa de acerto, o número secreto e, se você quiser tornar o jogo mais complexo, outras variáveis que também serão números.

**Guilherme:** Vou utilizar o seu exemplo e definir o número secreto como 29. Para isso, na mesma linha de código, adicionamos espaço, o sinal de igual e o número 29.

**alert**('Boas-vindas ao jogo **do** número secreto');

**let** numeroSecreto = 29;

COPIAR CÓDIGO

Dessa forma, temos o número secreto **armazenado**. O próximo desafio será perguntar para a pessoa qual é o valor do número secreto.

Se voltarmos no navegador, repare que o alert é responsável por exibir uma mensagem na tela. Não existe um campo para escrevermos o número secreto.

Após clicar em "Ok" a tela do fim do jogo aparece. Para usarmos um comando que possamos fazer uma pergunta, ou pedir para a pessoa chutar um número, usaremos o comando prompt.

Adicionamos parênteses, aspas simples e escrevemos a mensagem Escolha um número entre 1 e 30.

**alert**('Boas-vindas ao jogo **do** número secreto');

**let** numeroSecreto = 29;

**prompt**('Escolha um número entre 1 e 30');

COPIAR CÓDIGO

Note que, sempre que estamos lidando com mensagens ou textos, usamos aspas simples.

**Mônica:** Inclusive, quando definimos uma variável para receber um número, não utilizamos aspas, inserimos o número diretamente. Dessa forma, o JavaScript interpreta esse 29 como um número e não como texto.

**Guilherme:** Salvamos o código e voltamos no navegador. Ao atualizar a página, aparece a mensagem se boas-vindas. Ao clicar em"Ok" aparece outra mensagem escrita "Escolha um número entre 1 e 30" com um campo de texto abaixo.

Digitamos 29 e clicamos em "Ok". Feito isso, somos direcionados para a página final.

Mas, será que o nosso jogo está realmente funcionando conforme o esperado? Para isso, mudaremos o número secreto para 5 e definir um intervalo de 1 a 10 e salvamos. Depois, vamos errar propositalmente para verificar o que acontece.

**alert**('Boas-vindas ao jogo **do** número secreto');

**let** numeroSecreto = 5;

**prompt**('Escolha um número entre 1 e 10');

COPIAR CÓDIGO

É sempre muito importante testar os erros. Em fase de teste, sempre simulamos alguns caminhos.

Feito isso, voltamos no navegador e atualizamos a página. Clicamos no "Ok" de boas-vindas e chutamos o número 4. Ao clicar em "Ok" encontramos um bug no nosso jogo. A mensagem exibida ainda é "Parabéns, você descobriu o número secreto".

Vamos corrigir esse problema no próximo vídeo!

**Guilherme:** Moni, será mesmo que estamos comparando os valores inseridos no prompt com o número secreto?

**Mônica:** Acho que não, pois quando testamos inserindo o número 4, ainda assim ganhamos o jogo. E isso não deveria acontecer.

**Guilherme:** Então, vamos corrigir isso!

# 05Comparando variáveis

Para realizarmos uma comparação, existe um comando em JavaScript que podemos utilizar para verificar se um valor é igual a outro.

**Mônica:** Neste caso, não seria de fato uma comparação, mas sim uma condição, certo?

**Guilherme:** Exato! Então, damos espaço no código e no fim escrevemos if, seguido de espaço.

Agora, precisaremos seguir uma regra do JavaScript, que é adicionar parênteses e dentro a condição que queremos verificar.

Então, escrevemos numeroSecreto seguido de =. Porém, repare que há algo diferente. Anteriormente, usamos o sinal de igual para atribuir um valor a uma variável. No entanto, quando queremos fazer uma comparação, usamos um sinal de igual duplo ==.

Mas temos um problema. Como vamos comparar o valor do prompt? Afinal, ele não armazena o valor sozinho, certo?

**Mônica:** Para guardar esse valor precisamos de uma "caixa", que é o conceito de variáveis que acabamos de aprender.

**Guilherme:** Então, na Linha 3, antes de prompt, escrevemos let, pois quero que o valor digitado seja armazenado em algum lugar. Também precisamos de um nome para essa variável, que nesse caso será chute. Em seguida, adicionamos o sinal de igual.

**alert**('Boas-vindas ao jogo **do** número secreto');

**let** numeroSecreto = 5;

**let** chute = **prompt**('Escolha um número entre 1 e 10');

COPIAR CÓDIGO

Agora conseguimos comparar. Então, na linha 5, se o chute for igual ao número inserido, podemos comparar. Então, após o sinal de igual duplo, escrevemos chute.

Assim, se o número digitado for igual ao número secreto, algo acontecerá. Caso for verdade, adicionamos chaves no código.

Repare que ao adicionar uma chave o VS Code autocompleta com a segunda. Esse software possui vários atalhos, especialmente quando estamos escrevendo código em JavaScript.

Dentro das chaves pressionamos "Enter". Tudo o que escrevermos nesse espaço entre as chaves será executado apenas se o número secreto for igual ao chute.

Existe outro comando em JavaScript onde podemos testar e visualizar, como se fosse um ambiente quase secreto para desenvolvedoras e desenvolvedores. Esse ambiente é chamado de console.log().

Então, nas chaves escrevemos console.log('Isso ai! Você descobriu o número secreto (5)').

**alert**('Boas-vindas ao jogo do número secreto');

**let** numeroSecreto = 5;

**let** chute = **prompt**('Escolha um número entre 1 e 10');

**if** (chute == numeroSecreto) {

console.**log**("Isso aí! você descobriu o número secreto (5)");

}

COPIAR CÓDIGO

Lembrando que quando usamos o if na linha 5, essa instrução específica não leva ponto e vírgula. Mas, quando criamos o código que será executado, caso a condição seja verdadeira, nós temos essa mensagem.

Salvamos o código e abrimos o navegador para testar. Atualizamos a página e clicamos em "Ok". Depois, chutamos o número 5, acertaremos propositalmente. Feito isso, aparece a mensagem Você acertou!".

**Mônica:** Gui, mas essa imagem irá aparecer mesmo se errarmos, porque configuramos como condição para exibir uma mensagem no console, mas não na tela.

**Guilherme:** Nós vamos corrigir isso até o final do jogo! A minha única dúvida nesse momento é onde está essa mensagem que colocamos no console.log.

Para isso, clicando com o botão direito em qualquer parte da tela do navegador e depois em inspecionar. Feito isso, encontramos uma parte do HTML e do CSS correspondente ao nosso código. Isso significa que é possível conseguirmos ver o código HTML e CSS de outras aplicações também.

Repare que na lateral superior direita encontramos um botão chamado "Console". Ao clicá-lo, encontramos o texto "Isso ai! Você descobriu o número secreto (5)".

Agora, testaremos errando propositalmente. Dessa vez, digitamos o número 3. Ao clicar em "Ok" repare que a mensagem não é exibida na janela de inspeção.

Podemos inserir algumas dicas para a pessoa, como, por exemplo, indicar que ela errou o número. **Faremos isso a seguir!**

# 06Alert e prompt

Em um projeto prático dentro do curso "Iniciando com JavaScript", você está desenvolvendo um jogo no qual os usuários têm que adivinhar o número secreto. Você escreveu um código que mostra uma mensagem de boas-vindas aos jogadores, pedindo que escolham um número entre 1 e 10, e apresenta uma resposta se eles acertarem, conforme a sequência a seguir:

**alert**('Boas-vindas ao jogo **do** número secreto')

**let** chute = **prompt**('Escolha um número entre 1 e 10')

**let** numeroSecreto = 4

**if** (chute == numeroSecreto) {

**alert**('Acertou')

}

Qual variável você precisaria alterar para mudar o número secreto a cada rodada do jogo?

let numeroSecreto = 4

**Onde está o bug?**

Luis é apaixonado por tecnologia e está dando seus primeiros passos no mundo da programação. Para praticar lógica de programação, ele decidiu desenvolver um sistema de login de usuário único em que é necessário apenas informar a senha para ver o conteúdo do restante do projeto.

No entanto, ao testar o seu projeto, percebeu que não importava a senha passada para o sistema, o acesso sempre era permitido.

Sabendo que você também está estudando lógica de programação, Luis pediu sua ajuda para que vocês consigam encontrar o bug juntos. Para isso, ele disponibilizou o código que escreveu:

**let** senhaDoSistema = "senhaTeste!";

**let** senha = **prompt**("Digite a senha do sistema:");

**if** (senha = senhaDoSistema){

**alert**("Acesso ao sistema garantido");

}

De acordo com o código disponibilizado, qual é o problema?

r. o problema está no if (senha == senhaDoSistema)

**Você acertou em cheio!**

Adriano, sua resposta está correta, parabéns! O problema no código é que o operador de comparação utilizado no if está incorreto. Ao invés de usar o operador de atribuição (=), você deve usar o operador de igualdade (==) para comparar se a senha digitada pelo usuário é igual à senha do sistema. O operador de atribuição (=) serve para atribuir um valor a uma variável, enquanto o operador de igualdade (==) é usado para verificar se dois valores são iguais. Então, no código fornecido, a condição if (senha = senhaDoSistema) sempre será verdadeira, pois a senha digitada pelo usuário está sendo atribuída à variável senha, e não comparada com a senha do sistema. Parabéns por ter identificado corretamente o problema! Continue praticando e aprimorando seus conhecimentos em lógica de programação, você está no caminho certo.

**Desafio: hora da prática**

Praticar a lógica de programação, incluindo conceitos como variáveis, condicionais (if), alert, prompt, é essencial para seu desenvolvimento. Esses fundamentos fornecem a base para resolver problemas de forma estruturada, tomar decisões no código, criar interações controladas e interagir eficazmente com as pessoas.

Compreender esses conceitos não apenas facilita o aprendizado de novas linguagens e tecnologias, mas também capacita você a criar soluções inovadoras, depurar eficientemente e manter a qualidade ao longo do ciclo de vida do software. Portanto, investir tempo nesses princípios desde cedo é fundamental para construir uma trajetória bem-sucedida no campo da programação.

Pensando nisso, criamos uma lista de atividades (não obrigatórias) focada em prática para melhorar ainda mais sua experiência de aprendizagem. Bora praticar então?

**Desafios**

1. Mostre um alerta com a mensagem "Boas vindas ao nosso site!".
2. Declare uma variável chamada nome e atribua a ela o valor "Lua".
3. Crie uma variável chamada idade e atribua a ela o valor 25.
4. Defina uma variável numeroDeVendas e atribua a ela o valor 50.
5. Defina uma variável saldoDisponivel e atribua a ela o valor 1000.
6. Exiba um alerta com o texto "Erro! Preencha todos os campos"
7. Declare uma variável chamada mensagemDeErro e atribua a ela o valor "Erro! Preencha todos os campos" Agora exiba um alerta com o valor da variável mensagemDeErro.
8. Use um prompt para perguntar o nome do usuário e armazená-lo na variável nome.
9. Peça ao usuário para digitar sua idade usando um prompt e armazene-a na variável idade.
10. Agora, caso a idade seja maior ou igual que 18, exiba um alerta com a mensagem "Pode tirar a habilitação!".

Caso precise de ajuda, opções de solução das atividades estarão disponíveis na seção “Opinião da pessoa instrutora”.

**Resolução dos desafios do curso de lógica de programação**

Praticar a lógica de programação, incluindo conceitos como variáveis, condicionais (if-else), loops (while) e interações com o usuário (alert, prompt), é essencial para sua carreira de desenvolvimento de software. Esses fundamentos fornecem a base para resolver problemas de forma estruturada, tomar decisões no código, criar iterações controladas e interagir eficazmente com os usuários.

Compreender esses conceitos não apenas facilita o aprendizado de novas linguagens e tecnologias, mas também capacita você a criar soluções inovadoras, depurar eficientemente e manter a qualidade ao longo do ciclo de vida do software. Portanto, investir tempo nesses princípios desde cedo é fundamental para construir uma base sólida e bem-sucedida no campo da programação.

**Desafios - Respostas**

1. Mostre um alerta com a mensagem "Bem-vindo ao nosso site!".

alert('Bem-vindo ao nosso site!');

1. Declare uma variável chamada nome e atribua a ela o valor "Lua".

let nome = "Lua";

1. Crie uma variável chamada idade e atribua a ela o valor 25.

let idade = 25;

1. Defina uma variável numeroDeVendas e atribua a ela o valor 50.

let numeroDeVendas = 50;

1. Defina uma variável saldoDisponivel e atribua a ela o valor 1000.

let saldoDisponivel = 1000;

1. Exiba um alerta com o texto "Erro! Preencha todos os campos."

alert('Erro! Preencha todos os campos');

1. Declare uma variável chamada mensagemDeErro e atribua a ela o valor "Erro! Preencha todos os campos." Agora exiba um alerta com o valor da variável mensagemDeErro.

let mensagemDeErro = 'Erro! Preencha todos os campos';

alert(mensagemDeErro);

1. Use um prompt para perguntar o nome do usuário e armazená-lo na variável nome.

let nome = prompt('Digite o seu nome');

1. Peça ao usuário para digitar sua idade usando um prompt e armazene-a na variável idade.

let idade = prompt('Digite a sua idade');

1. Peça ao usuário para digitar sua idade usando um prompt e armazene-a na variável idade. Agora, a idade seja maior ou igual que 18, exiba um alerta com a mensagem "Pode tirar a habilitação!".

let idade = prompt('Digite a sua idade');

if (idade >= 18) {

alert('Pode tirar a habilitação!');

}

# 10Para saber mais: documentação

A leitura da documentação oficial de uma linguagem de programação é crucial para que uma pessoa desenvolvedora de software possa utilizá-la de forma eficaz.

No caso do JavaScript, essa documentação fornece uma ampla gama de informações, incluindo conceitos sobre sintaxe, as bibliotecas e as funcionalidades disponíveis na linguagem.

Alguns links úteis para a documentação oficial do JavaScript incluem:

* [A documentação da linguagem de programação JavaScript](https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript)
* [Guia de JavaScript: o que é e como aprender a linguagem mais popular do mundo?](https://www.alura.com.br/artigos/javascript)

Ao ler a documentação oficial, temos uma compreensão mais profunda da linguagem e de suas funcionalidades. Isso facilita a escrita de códigos mais limpos, claros e seguros. A documentação também pode ajudar o desenvolvedor ou desenvolvedora a economizar tempo, além de evitar a necessidade de pesquisar em fóruns ou outras fontes de informação não confiáveis.

# 11O que aprendemos?

**Aqui está um resumo dos tópicos ensinados nesta aula:**

* Preparamos o ambiente de desenvolvimento com a instalação do Visual Studio Code para criar programas utilizando a linguagem JavaScript;
* Entendemos o conceito de variável para guardar informações, como números ou palavras, para usar mais tarde no nosso programa;
* Usamos o alert para exibir uma mensagem passando algumas instruções sobre o programa e usamos o prompt para interagir com a pessoa permitindo inserir um valor e armazenando em uma variável;
* Criamos um if, que é uma instrução em programação que permite ao computador tomar decisões ao executar um bloco de código apenas se uma condição específica for verdadeira.

**Na próxima aula:**

Vamos seguir as boas práticas de programação e aprofundar ainda mais nossos conhecimentos!

# 02Else e comentários

**Guilherme:** O próximo desafio é criar uma condição que nos permita exibir uma mensagem para que a pessoa usuária entenda se acertou ou errou.

Só para reforçar o que estamos aprendendo,no Javascript, existe uma forma de escrever um texto sem que esse texto seja interpretado como um comando. Seja o alert, o let, o if que escrevemos anteriormente, todos são interpretados de maneira diferente pelo código, cada um tem uma cor de fonte correspondente ao que eles fazem.

Para eu conseguir criar um trecho de código que seja interpretado como um comentário, que o Javascript não entenda como comando, usarei o sinal de barra dupla//. Se escrevermos algo depois deste sinal //, salvarmos e executarmos o código, veremos que não influencia em nada no nosso projeto.

Existem palavras que o Javascript entende como um comando ou código, ou seja, reconhece como uma instrução a ser executada. E existem outros termos ou símbolos que ele não entende ou reconhece.

Eu quero fazer um comentário nesse código para justamente informar o que nós estamos comparando. O comando if em português significa "Se". Ou seja, estamos falando "se o chute for igual ao número secreto, podemos executar alguma coisa. Vamos deixar um comentário acima do if informando isso.

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

console.**log**('Isso ai! Você descobriu o número secreto (5)');

}

COPIAR CÓDIGO

Mas, se queremos que uma ação seja executada quando a condição não for alcançada, ou seja, caso o número não seja igual ao número secreto, precisamos de um comando que faça esse "contraponto".

**Mônica:** Para fazer isso também precisamos de uma palavra especial. A gente pode escrever, por exemplo, o termo "senão". Dá pra ver até pela cor azulada que ele não vai funcionar. O if é um comando que funciona e está com uma cor rosada, como ele está considerando o "senao" como azul, é como se estivéssemos criando alguma coisa nova.

**Guilherme:** No Javascript, para representar o "se não", usamos a palavra else para isso, "else" em português significa "se não".

Ao escrevermos "else", note que ele mudou a cor, a fonte está com uma cor rosada agora.

Quando escrevemos else, precisamos criar um bloco de instruções, assim como fizemos com if. Por exemplo, posso exibir um alerta, um alert, informando que a pessoa errou. Neste caso, a mensagem será: "Você errou :(".

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

console.**log**('Isso ai! Você descobriu o número secreto (5)');

} **else** {

**alert**('Você errou :(')

}

COPIAR CÓDIGO

Podemos salvar e atualizar a página do navegador para testarmos o caminho feliz e o caminho triste.

No navegador, ao digitarmos, por exemplo, "3" no campo de texto da caixa de "Escolha um número entre 1 e 10". Vai aparecer a mensagem "Você errou :(".

Podemos fazer uma coisa diferente, em vez de exibir no console quando a pessoa acertar o número, que tal exibir um alerta assim como fazemos quando a pessoa erra?

**Mônica:** Acredito que é até melhor! Isso tornará a interação ainda mais amigável e interativa. Porque olhe tudo que a pessoa teve que fazer para visualizar o que está no console. Ele é uma ferramenta aprimorada para visualizar se há alguma interação no seu código, se está funcionando, se você conseguiu resgatar um valor corretamente.

Porém, não é recomendado para fazer essa interação com o usuário porque é uma ferramenta da pessoa desenvolvedora e não uma ferramenta direta para o usuário. Isso é do que se trata a parte do console que estamos observando.

**Guilherme:** Você me deu uma ideia. O que você acha se colocarmos no console apenas o valor do número secreto?

Por exemplo, vou jogar com a minha família. Dou uma olhada no console, vejo qual é o número secreto e chamo as pessoas. Legal, né? Gostei dessa ideia.

Portanto, o que vou fazer? Irei fazer uma troca aqui. Vou substituir console.log por alert e quando definimos o número secreto, vou exibir no console apenas o valor do número secreto. Com o console.log(). Como se fosse um recurso que só nós conhecemos.

E aqui tem algo diferente. Se colocarmos nesse console.log(), entre aspas simples, 'numeroSecreto', veja o que vai acontecer no nosso programa.

**alert**('Boas vindas ao jogo do número secreto');

**let** numeroSecreto = 5;

console.**log**('numeroSecreto')

**let** chute = **prompt**('Escolha um número entre 1 e 10');

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

**alert**('Isso ai! Você descobriu o número secreto (5)');

} **else** {

**alert**('Você errou :(')

}

COPIAR CÓDIGO

Ao executarmos a aplicação no navegador, só apareceu no console a informação:

numeroSecreto

Por quê? O que fizemos de diferente aqui?

**Mônica:** Você colocou as aspas simples, então ele está interpretando como um texto. Até me lembro da questão do número que comentei. Quando usamos, não usamos aspas para ele entender que é um número. E nesse console.log() é o mesmo caso. Queremos pegar o valor da variável e não um texto em si.

**Guilherme:** Correto. Sempre que mencionamos especificamente a variável, não precisamos usar as aspas simples ,certo?

**Mônica:** Também gostaria de fazer uma observação, pois ele já mostrou o número secreto escrito antes de acertarmos ou erramos. Porque o JavaScript lê o nosso código linha por linha. O console.log() estava sendo lido antes do pedido de escolher um número.

**Guilherme:** Vamos executar mais uma vez. Vamos atualizar.

Agora apareceu o número 5 no console. Eu vi antes das demais pessoas. Vou palpitar o número 5. Olha aí, acertei. Venci as demais pessoas.

Vamos testar agora o caso que não dá certo. E ele dá a mensagem informando que a pessoa errou.

Legal. O próximo passo que precisamos realizar no nosso código é o seguinte: caso a pessoa erre, queremos dar uma esperança para ela no nosso projeto.

Todo mundo pode ter a capacidade de acertar o jogo do número secreto, assim como foi o meu caso.

**Mônica:** Sim. Você fez várias tentativas e eu fui dando dicas até chegar no resultado correto.

**Guilherme:** Então, é isso que vamos fazer na sequência!

# 03Concatenação

**Mônica:** Gui, nós já testamos várias vezes o caminho feliz e o caminho triste e todas as vezes o número secreto era 5. Para deixar mais empolgante, vamos trocar isso? O que você acha?

**Guilherme:** Vamos trocar sim! O que você sugere, Moni?

**Mônica:** Podemos mudar para o número 4, que é o meu número de sorte.

**Guilherme:** Beleza. Substituiremos o número 5 pelo número 4.

Vamos salvar e executar no navegador para verificar como a nossa aplicação está funcionando.

Com frequência nós temos que ficar atualizando a página, notou? Vamos corrigir isso mais adiante.

De volta ao nosso jogo, a mensagem de "Boas-vindas ao jogo do número secreto" aparece. Vamos pelo caminho feliz primeiro. A dica que adicionamos em nosso código foi exibida no console, ficou incrível, não é?

Mas apareceu a seguinte mensagem, o que ficou estranho já que mudamos o número secreto para 4:

Isso ai! Você descobriu o número secreto (5)

Vamos arrumar isso.

Percebe como, apesar de nós alterarmos uma linha específica no nosso código, esperávamos que outras partes deveriam ter sido alteradas automaticamente, como a linha 8 no alert.

No momento, estamos colocando o número manualmente nesse alert. Para funcionar como queremos, devemos alterar manualmente aqui também de 5 para 4.

alert('Isso ai! Você descobriu o número secreto 4');

COPIAR CÓDIGO

Nós não deveríamos ter que alterar manualmente o valor do número secreto no nosso projeto. Vamos supor que o número secreto mudou. Jogamos uma semana inteira com toda a família, ganhamos todas as vezes porque sabemos qual é o número secreto. Agora o número secreto será 8. Teríamos que ir lá no código e alterar novamente para o número 8.

Isso não é ideal, temos o que costumamos chamar de um **acoplamento** muito grande: nosso código está muito interligado, uma alteração feita deveria refletir automaticamente em outras partes. Então nós precisamos fazer uma pequena mudança.

Precisamos, de alguma forma, mudar esse trecho que informa qual é o número secreto. Queremos que o valor presente nessa parte do código fosse o valor do número secreto que foi definido na parte de cima do código. Então vou escrever aqui: numeroSecreto.

alert('Isso ai! Você descobriu o número secreto numeroSecreto');

COPIAR CÓDIGO

Porém, se rodarmos o jogo novamente, o número secreto é 8, mas quando acertamos, aparece a mensagem:

Isso ai! Você descobriu o número secreto numeroSecreto

Isso não funcionou, Moni. Como podemos fazer para que sejamos capazes de combinar textos com valores de variáveis e até com diferentes números?

**Template Strings**

**Mônica:** Para juntarmos textos com valores de variáveis e até com números diferentes, existem várias maneiras de fazer isso em Javascript , e nós vamos optar por uma que se chama ***Template Strings***.

**Guilherme:** O primeiro passo dessa modificação é substituir as aspas simples por crases, também conhecida como acento grave (`).

**Mônica:** Quando insiro acentos graves, perceba que nada parece ter mudado na linha 8. Ele continua interpretando a linha como se fosse um texto, uma *string*. Precisamos adicionar mais um detalhe: antes do nome da variável numeroSecreto, vamos inserir um sinal de cifrão ($) e colocar a variável dentro de chaves {}.

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

COPIAR CÓDIGO

Nosso código da condição if vai ficar assim:

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

} **else** {

**alert**('Você errou :(')

}

COPIAR CÓDIGO

**Guilherme:** Agora, vamos salvar as alterações e verificar se funciona corretamente.

Vamos mais uma vez ao navegador para testar. Vamos testar o caminho feliz. O número secreto será 8. O que esperamos? Que seja exibida uma mensagem dizendo "Você descobriu o número secreto 8".

E foi o que apareceu ao colocarmos o número 8.

Isso ai! Você descobriu o número secreto 8

Agora, vamos alterar o número secreto para 4, let numeroSecreto = 4;. Salvar e testar novamente no navegador.

Ai inserirmos o número 4, ele exibe a seguinte mensagem:

Isso ai! Você descobriu o número secreto 4

**Mônica:** Ótimo! Agora, a nossa mensagem está bastante dinâmica.

Continuaremos no próximo vídeo.

# 04Live server

**Guilherme:** Eu gostei do aspecto dinâmico que deixamos na frase que informa o número secreto do nosso jogo.

Mas tem um ponto que me deixa um pouco triste no nosso jogo, todas as vezes que fazemos uma alteração no Javascript, precisamos atualizar a página antes de poder começar a jogar.

**Mônica:** E o pior é que é muito comum esquecermos de atualizar e, por isso, ao testar acharmos que o que fizemos no código não está funcionando. Tem como facilitar isso?

**Guilherme:** Vamos solucionar isso e facilitar a vida das pessoas.

O que nós queremos? Queremos modificar nosso código e quando acessarmos o navegador já estar com o código atualizado. Assim, quando formos ao navegador, teremos a versão mais recente possível do nosso código.

Há várias ferramentas que podem nos auxiliar com isso. No Visual Studio Code, que é o editor que recomendamos para o uso, o que farei é o seguinte:

Note que, na barra lateral do VS Code, temos a funcionalidade **Extensions (Extensões)** que também pode ser acessada com o atalho "Ctrl + Shift + X".

Ao clicar no menu "Extensions", será aberto um painel onde podemos buscar extensões para o VS Code.

**Mônica:** Extensões são algo a mais que colocaremos dentro do VS Code para facilitar nossa vida. É muito comum as pessoas usarem extensões no navegador também para facilitar algum processo durante a navegação.

**Guilherme:** Uma das extensões que usaremos no VS Code chama-se **Live Server**. Vamos pesquisar por "live server" no campo de busca de extensões.

Podemos clicar em "Live Server" e, em seguida, clicar no botão "Install" para instalar. Será instalada, e se bem me lembro, será solicitado um recarregamento do Visual Studio Code. Sei que seria mais fácil mencionar pelo ícone do Servidor ao Vivo que usaremos.

**Mônica:** O Live Server que usaremos é a extensão criada pelo Ritwick Dey, ela é a opção que tem mais downloads, então fica em primeiro na lista de busca de extensões.

O VS Code não pediu para reiniciar, já diz que está habilitado. Ótimo!

**Guilherme:** Agora, podemos clicar em "Explorer" ou usar o atalho "Ctrl + Shift + E" para visualizarmos a estrutura de todos os arquivos que temos.

No Explorer, clicaremos com o botão direito no arquivo index.html e selecionaremos a opção "Open with Live Server".

O que isso vai fazer? Isso abrirá um servidor local. Se você não sabe o que é um servidor, tudo bem. Não é nosso foco no momento. Mas ele abrirá um servidor que sempre atualizará automaticamente sempre que fizermos uma alteração no HTML, Javascript ou em qualquer outra parte da aplicação.

Ao clicarmos em "Open with Live Server", ele vai abrir uma página do navegador.

Já podemos também abrir o console de inspeção dessa página. No console, está sendo exibida uma mensagem informando que ele não encontrou o ícone do Live Server. Isso não é um problema. Podemos corrigir isso mais tarde. E veja só, o número 4 está sendo exibido no console. Não vou jogar. Só vamos mudar os valores Vou mudar o valor de numeroSecreto para 2, por exemplo. Podemos salvar o código.

**Mônica:** Note que, agora, o navegador que abrimos com o Live Server exibe uma série de números na barra de endereços, não mais o diretório de pastas, assim fica até mais fácil para identificarmos que é essa página que devemos usar daqui em diante. Na barra de endereços está sendo exibido: 127.0.0.1:5500/index.html.

**Guilherme:** Ao testarmos a aplicação, ele reconheceu a alteração que fizemos e já atualizou automaticamente.

Dessa forma, tudo que alterarmos no código será visto.

Mas, como o Live Server faz isso? Não precisamos saber especificamente os detalhes aprofundados, mas se pedirmos para inspecionar o código, observamos que na parte de *script* tem uma linha escrita assim: <!-- Code injected by live server -->, ou seja, "Código injetado pelo Live Server".

Este código que ele injeta é um código Javascript com o qual não precisamos nos preocupar no momento. Esse código é responsável por verificar as alterações, as mudanças que fazemos em nosso projeto para mantê-lo atualizado.

Para garantir que de fato está atualizando, vamos alterar o número secreto novamente. Agora o número secreto será 9. Podemos salvar o código e ir para o navegador que abrimos com o Live Server.

Perceba que o navegador já carregou a aplicação sem precisarmos atualizar. Ele executou o alerta automaticamente. E se acessarmos o console veremos a mensagem com o número nove. Tudo certo, a aplicação está funcionando corretamente.

Continuaremos no próximo vídeo!

# 05Alterando a mensagem do alert

Você é uma pessoa desenvolvedora de software em uma clínica médica chamada Médica Voll. A empresa deseja que você crie um pequeno jogo de adivinhação para seus pacientes enquanto eles estão na sala de espera. Então, você decide criar um jogo do número secreto usando JavaScript, conforme a seguir:

**alert**('Bem-vindo ao jogo do número secreto')

**let** chute = **prompt**('Escolha um número entre 1 e 10')

**let** numeroSecreto = 4

console.**log**(chute == numeroSecreto)

**if** (chute == numeroSecreto) {

**alert**('Acertou')

} **else** {

**alert**('O número secreto era ' + numeroSecreto)

}

COPIAR CÓDIGO

Dada essa estrutura básica de código que você criou, como você modificaria a mensagem do alerta para incluir o número que o usuário escolheu, caso não acertasse o número secreto? Escolha a alternativa correta:

**Obs:** Note que o "+" usado serve como uma alternativa ao `-> template de strings já apresentado para realizarmos concatenação de strings/caracteres/valores de varíaveis no JavaScript.

alert('O número secreto era ' + numeroSecreto + ', mas você escolheu ' + chute);

# 06Detran

Você é uma pessoa desenvolvedora de software que desenvolve aplicações para o Detran, que é o órgão responsável pela fiscalização e regulamentação do trânsito, habilitação de condutores, registro e licenciamento de veículos no Brasil.

A idade mínima para dirigir no Brasil é 18 anos. Sabendo disso, uma equipe criou o seguinte programa:

**let** idade = **prompt**("Digite sua idade:");

**if** (idade >= 18) {

console.**log**("Você é maior de idade.");

} **else** {

console.**log**("Você é menor de idade.");

}

COPIAR CÓDIGO

Com essa base, analise as afirmações abaixo e marque apenas as verdadeiras.

 Caso a idade seja igual a 18, a mensagem do console que será exibida será: Você é maior de idade.

Esta afirmação está correta. Se a idade digitada for igual a 18, a condição no if será verdadeira (idade >= 18 é verdadeiro), e a mensagem "Você é maior de idade" será exibida, pois o operador >= inclui o valor 18. O bloco de código dentro do else será ignorado.

 Alternativa correta



Caso a idade seja menor que 18, a mensagem do console que será exibida será: Você é menor de idade.

Esta afirmação está correta. Se a idade digitada for menor que 18, a condição no if será falsa (idade >= 18 é falso), e o fluxo de execução seguirá para o bloco do else, exibindo a mensagem "Você é menor de idade".

# 07Trabalhando com condicionais

Você é uma pessoa recém-chegada à equipe de desenvolvimento do Jornada Milhas, um site de compra de pacotes de viagens para os principais destinos do mundo.

Como primeira tarefa, sua liderança pediu para que você corrigisse um bug que tem acontecido quando uma pessoa cliente do site tenta utilizar suas milhas do cartão de crédito para obter desconto nos pacotes de viagem. Para isso, foi apresentado o seguinte código:

**let** porcentagemDesconto = 0;

**if**(quantidadeMilhas > 5000){

porcentagemDesconto = porcentagemDesconto + 10;

}

**if**(quantidadeMilhas >= 30000){

porcentagemDesconto = porcentagemDesconto + 20;

} **else** {

porcentagemDesconto = 0;

}

COPIAR CÓDIGO

Para te ajudar a compreender a origem do bug, sua liderança explicou a regra de negócio utilizada para compras de pacotes de viagens com milhas:

* Pessoas com uma quantidade de milhas **inferior a 5.000 não recebem desconto.**
* Pessoas com uma quantidade de milhas **igual ou superior a 30.000 recebem 20% de desconto.**
* Pessoas com uma quantidade de milhas **superior a 5.000, mas inferior a 30.000, recebem 10% de desconto.**

Mas, segundo relatos de clientes, ao tentar utilizar o desconto referente a 5.000 milhas, não ocorre nenhum desconto. Já clientes com mais de 30.000 milhas estão recebendo desconto excedente ao que deveriam receber.

Como você poderia modificar o código para que ele funcione corretamente? Selecione a opção correta:

**let** porcentagemDesconto = 0;

**if**(quantidadeMilhas >= 30000){

porcentagemDesconto = porcentagemDesconto + 20;

} **else** {

**if**(quantidadeMilhas > 5000){

porcentagemDesconto = porcentagemDesconto + 10;

}

}

A utilização do bloco if dentro do bloco else garante que, caso a quantidade de milhas for superior a 30000, o desconto será de 20%, e caso não for superior a este valor, mas for superior a 5000, o desconto será de 10%. Para o caso da quantidade de milhas ser inferior a 5000, o desconto permanecerá zerado, pois não existe a quantidade mínima para ganhar desconto.

let porcentagemDesconto = 0;

**if**(quantidadeMilhas >= 30000){

porcentagemDesconto = porcentagemDesconto + 20;

} **else** {

**if**(quantidadeMilhas > 5000){

porcentagemDesconto = porcentagemDesconto + 10;

}

}Aqui está o que o código está fazendo:

1. Inicialmente, a variável porcentagemDesconto é definida como 0.
2. O primeiro if verifica se a quantidadeMilhas é maior ou igual a 30.000. Se for, o desconto é definido como 20% (porcentagemDesconto = porcentagemDesconto + 20;).
3. Caso a condição do primeiro if não seja atendida (ou seja, a quantidadeMilhas é menor que 30.000), o código entra no bloco else. Dentro desse bloco else, há outro if que verifica se a quantidadeMilhas é maior que 5.000. Se for, o desconto é definido como 10% (porcentagemDesconto = porcentagemDesconto + 10;).

Essa solução atende corretamente às regras de negócio apresentadas:

* Pessoas com uma quantidade de milhas **inferior a 5.000 não recebem desconto** (pois nenhuma das condições é atendida).
* Pessoas com uma quantidade de milhas **igual ou superior a 30.000 recebem 20% de desconto**.
* Pessoas com uma quantidade de milhas **superior a 5.000, mas inferior a 30.000, recebem 10% de desconto**.

Portanto, o código que você propôs está correto e resolve o problema apresentado. Bom trabalho!

# 08Faça como eu fiz: console.log

O console.log é uma função muito importante em linguagens de programação, especialmente quando se trabalha com JavaScript. Sua principal função é imprimir mensagens no console do ambiente de desenvolvimento, o que permite testar informações relevantes durante a execução de um programa.

Sua vez: inclua comandos console.log em diferentes partes do código para verificar o fluxo do programa, os valores das variáveis e outras informações relevantes durante a fase de desenvolvimento.

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

No código a seguir temos uma sugestão da inclusão do comando console.log adicionado para verificar se os valores estão sendo atribuídos corretamente às variáveis e se as condições estão sendo avaliadas como esperado. Confira:

alert('Boas vindas ao jogo do número secreto');

// Adicione um console.log para verificar o valor de "chute" após a entrada **do** usuário

let chute = prompt('Escolha um número entre 1 e 10');

console.log('Valor do chute:', chute);

let numeroSecreto = 4;

// Adicione um console.log para verificar a comparação entre "chute" e "numeroSecreto"

console.log('Resultado da comparação:', chute == numeroSecreto);

**if** (chute == numeroSecreto) {

alert('Acertou');

} **else** {

// Adicione um console.log para verificar o valor de "numeroSecreto" quando o jogador erra

console.log('Valor do número secreto:', numeroSecreto);

alert('O número secreto era ' + numeroSecreto);

}

COPIAR CÓDIGO

É importante lembrar que, quando o projeto ficar disponível para todas as pessoas, o que chamamos no contexto de software de ambiente de produção, os comandos console.log devem ser removidos ou desabilitados, pois podem causar problemas de desempenho e segurança.

# 09Desafio: hora da prática

Programação é prática! Criamos mais uma lista de atividades (não obrigatórias) para você exercitar e reforçar ainda mais seu aprendizado.Bora praticar?

**Desafios**

1. Pergunte ao usuário qual é o dia da semana. Se a resposta for "Sábado" ou "Domingo", mostre "Bom fim de semana!". Caso contrário, mostre "Boa semana!".
2. Verifique se um número digitado pelo usuário é positivo ou negativo. Mostre um alerta informando.
3. Crie um sistema de pontuação para um jogo. Se a pontuação for maior ou igual a 100, mostre "Parabéns, você venceu!". Caso contrário, mostre "Tente novamente para ganhar.".
4. Crie uma mensagem que informa o usuário sobre o saldo da conta, usando uma template string para incluir o valor do saldo.
5. Peça ao usuário para inserir seu nome usando prompt. Em seguida, mostre um alerta de boas-vindas usando esse nome.

Caso precise de ajuda, opções de solução das atividades estarão disponíveis na seção “Opinião da pessoa instrutora”.

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

Para te ajudar a verificar seus códigos, deixo disponibilizado uma possível [resolução dos desafios neste link](https://github.com/alura-cursos/logica-js/tree/desafio_2) para você construir ou validar suas soluções.

Boa sorte nos estudos!

**Para saber mais: ponto e vírgula no JavaScript**

No JavaScript, o uso do ponto e vírgula (;) é uma prática recomendada. A linguagem possui um mecanismo chamado "inserção automática de ponto e vírgula" (automatic semicolon insertion - ASI), que tenta adicionar ponto e vírgulas automaticamente em certos pontos do código onde eles são ausentes.

Isso significa que, em alguns casos, o JavaScript tentará "corrigir" a falta de ponto e vírgula inserindo-o automaticamente. No entanto, a interpretação do ASI pode levar a comportamentos inesperados e erros sutis, especialmente quando as regras não são claras.

Portanto, para evitar possíveis problemas e garantir a clareza do código, muitos desenvolvedores preferem adicionar explicitamente ponto e vírgula em seus programas.

Apesar da inserção automática de ponto e vírgula poder ajudar a mitigar erros de sintaxe, é uma boa prática adicionar ponto e vírgula manualmente para evitar ambiguidades e problemas de interpretação. Isso é particularmente importante em situações como quando várias instruções estão em uma mesma linha, ao usar declarações de retorno de valor ou ao minificar o código.

Em projetos colaborativos ou de grande escala, a consistência no estilo de codificação e a clareza do código são cruciais, e o uso explícito de ponto e vírgula contribui para um código mais legível e menos sujeito a erros de interpretação por parte dos programadores e do próprio mecanismo de ASI.

* [Leitura complementar da documentação sobre o JavaScript](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Lexical_grammar#automatic_semicolon_insertion).
* Quer saber o que a comunidade recomenda? [Leia mais neste link do Stack Overflow](https://pt.stackoverflow.com/questions/3341/utilizar-ou-n%C3%A3o-ponto-e-v%C3%ADrgula-no-fim-das-linhas-em-javascript).

# 11O que aprendemos?

**Nesta aula:**

* Utilizamos o console para testar e depurar nosso código, exibindo mensagens e valores durante a execução do programa;
* Aprendemos a utilizar estruturas condicionais (if/else) para criar lógicas que permitem ao programa tomar decisões com base em condições específicas;
* Implementamos um bloco de código para exibir uma mensagem caso o chute do usuário não seja igual ao número secreto;
* Usamos template strings para concatenar o número secreto com o valor armazenado em uma variável e exibir uma mensagem personalizada.

**Na próxima aula:**

Vamos aprender como trabalhar com estruturas de repetição!

03 Loops e tentativas

# 02Novas condições

**Guilherme:** Nosso jogo agora está similar ao que fizemos no início, quando você escolheu o número 29. Não tínhamos combinado essa escolha.

Agora, ao errar, exibimos uma mensagem de erro, lembrando da frase "Errou!" Seria interessante fornecer uma dica à pessoa jogadora, concorda? Quando testei isso contigo, você indicou se o número era menor ou maior. Foram dadas dicas até que você encontrasse o número secreto.

Vamos criar essas dicas?

**Mônica:** Vamos.

**Criando as dicas**

[Arquivo app.js no repositório do GitHub](https://github.com/guilhermeonrails/logica-js/blob/aula_4/app.js)

**Guilherme:** Primeiramente, precisamos decidir em que ponto do nosso código introduziremos essas dicas. Vamos ao arquivo app.js.

app.js

alert('Boas vindas ao jogo do número secreto');

**let** numeroSecreto = 2;

console.log(numeroSecreto)

**let** chute = prompt('Escolha um número entre 1 e 10');

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

alert(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

} **else** {

alert('Você errou :(' )

}

Se o palpite coincidir com o número correto, o jogo se encerra; não ocorre mais nada. Caso contrário, poderíamos implementar novas condições para informar se o número secreto é maior ou menor. Concorda com essa abordagem?

Vamos estabelecer mais uma condição dentro do else, podemos remover o alert(). Utilizaremos o if(), abrindo parênteses e inserindo a condição.

Se o número secreto for maior do que o chute, exibimos uma mensagem assim informando que o número secreto é maior. Faremos isso através de um alert(). Usando as aspas simples, que é nosso padrão nesse projeto, escreveria "O número secreto é maior que", e incluiria o valor.

Que tal aplicar uma template string? Usamos: "O número secreto é maior que ${chute}" Usaremos o cifrão, as chaves, a palavra "chute", e fechamos as chaves. Quando lidamos com template strings, empregamos crases e encerraremos a condição com um ponto e vírgula.

app.js

// código omitido

} **else** {

**if** (numeroSecreto > chute) {

**alert**(`O número secreto é maior que ${chute}`);

}

}

COPIAR CÓDIGO

Se o chute for menor que o número secreto, entramos em outra condição. Após o fechamento das chaves, implementamos mais um else e utilizamos um alert() para informar que o número secreto é menor.

Para manter a padronização, o termo "chute" será sempre a primeira variável. Vamos alterar o código anterior. Portanto, se o chute for maior que o numeroSecreto o número secreto é **menor**:

app.js

// código omitido

} else {

if (chute > numeroSecreto) {

alert(`O número secreto é menor que ${chute}`);

} else {

alert()

}

}

# 03Enquanto...

**Mônica:** Quando estávamos interagindo com o jogo de forma mais manual, usando um pequeno pedaço de papel, você conseguiu fazer várias tentativas até acertar. No entanto, atualmente, estamos limitados a uma única tentativa em nosso computador. É essencial abordar essa questão.

**Guilherme:** Na programação, existe uma maneira de indicar: "Execute essas instruções apenas enquanto uma determinada condição permanecer verdadeira". Esse conceito é conhecido como *loop* ou, em inglês, while. A estrutura do while é bastante similar à do if.

Para empregar o while, escrevemos a palavra seguida por parênteses e chaves para delinear as condições. Dentro do "enquanto", o que ocorre?

[Arquivo app.js no repositório do GitHub](https://github.com/guilhermeonrails/logica-js/blob/aula_4/app.js)

app.js

// código omitido

// enquanto

**while** (condição) {

// instruções a serem executadas

}

// código omitido

COPIAR CÓDIGO

**Mônica:** Aconselhamos a fazer uma anotação do que significa cada uma desses comandos em JavaScript, que também se aplica a outras linguagens de programação. Isso será útil quando precisar procurar como fazer essas coisas, lembrando do que estamos falando.

**Guilherme:** No nosso jogo, como regra, queremos que as tentativas continuem apenas **enquanto o chute não for igual ao número secreto**. Caso contrário, se acertar, o jogo termina.

app.js

// código omitido

// enquanto chute não for igual ao n.s.

**while** () {

}

// código omitido

COPIAR CÓDIGO

Então, como vamos aplicar isso na lógica da nossa aplicação?

Vamos pedir um chute para a pessoa jogadora quando estivermos dentro da condição do while ou ainda não entramos?

**Mônica:** Já entramos, porque queremos iniciar o jogo, fazer a pergunta, a pessoa dá o chute, fazemos a pergunta de novo.

**Guilherme:** Então, dentro da variável let chute, vamos remover a parte do prompt('Escolha um número entre 1 e 10'), deixando a variável let chute; vazia, indicando um espaço na memória onde armazenaremos o chute, que ainda não foi preenchido.

Dentro do while(){} passamos que o valor do chute será o valor que pedimos na condição.

app.js

// código omitido

// enquanto chute não for igual ao n.s.

**while** () {

chute = prompt('Escolha um número entre 1 e 100');

}

// código omitido

COPIAR CÓDIGO

Agora, necessitamos verificar a validade da condição. Como determinamos se a executamos ou não? Até quando repetimos? Vamos escrever essa parte agora no código.

Vamos compor com base nas informações anteriores. O termo "enquanto" equivale ao while. A menção ao chute diz respeito ao chute que foi mencionado anteriormente dentro do while().

O "Não for igual" tem um código específico em JavaScript que podemos usar para representar essa expressão. Portanto, para representar "não é igual", usamos != em JavaScript.

Portanto, enquanto o chute não for igual ao número secreto, manteremos essa repetição. Por exemplo, pergunte e o chute, pergunte e o chute.

app.js

// código omitido

// enquanto chute não for igual ao n.s.

**while** (chute != numeroSecreto) {

chute = prompt('Escolha um número entre 1 e 100');

}

// código omitido

COPIAR CÓDIGO

Agora, se eu executar isso agora (**não execute o programa assim!**), recebemos um *loop* infinito de solicitações de chutes.

Ele pergunta, o chute, o chute, o chute, e assim por diante, infinitamente. Passaríamos o resto da nossa vida dizendo qual é o valor do palpite e nada mais aconteceria.

**Mônica:** Por causa da execução de códigos sequenciais, quando ele chega à linha o while, repete indefinidamente porque a condição nunca é atendida, então ele nunca avança para o próximo número.

**Guilherme:** Para corrigir isso, precisamos pegar as condições if que estão fora do while e movê-las para dentro do nosso laço while.

**Mônica:** Portanto, vamos copiar da linha 10 à linha 19. Com "CTRL + X" e depois da linha 8, damos um enter e na linha 9 colamos tudo que copiamos.

Por enquanto, temos:

app.js

// código omitido

**alert**('Boas vindas ao jogo do número secreto');

**let** numeroSecreto = 5;

console.**log**(numeroSecreto)

**let** chute

// enquanto chute não for igual ao n.s.

**while** (chute != numeroSecreto) {

chute = **prompt**('Escolha um número entre 1 e 10');

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

} **else** {

**if** (chute > numeroSecreto) {

**alert**(`O número secreto é menor que ${chute}`);

} **else** {

**alert**(`0 número secreto é maior que ${chute}`)

}

// código omitido

COPIAR CÓDIGO

**Guilherme:** Isso leva a um ponto importante.

Vamos analisar o código. Notem que sempre que começamos uma nova instrução, iniciamos no início da linha à esquerda, está bem alinhado à esquerda. No âmbito do while, onde normalmente usamos chaves {} para delimitar esse escopo, há um recuo à esquerda.

Essa prática é recomendada para garantir a **legibilidade** do código. No entanto, ao copiar ("Ctrl+C") e colar ("Ctrl+V") dentro do if(), os códigos não foram inseridos dentro do escopo apropriado, seja do while ou do if. Consequentemente, eles ficaram nesse formato. Isso não é apropriado.

Selecionamos da linha 10 até a linha 18 (começo do if() até o else()). Pressionamos "Shift" e o caractere "{" ("Shift + {"), o caractere "{" foi inserido corretamente.

app.js

// código omitido

// enquanto chute não for igual ao n.s.

**while** (chute != numeroSecreto) {

chute = **prompt**('Escolha um número entre 1 e 100');

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

} **else** {

**if** (chute > numeroSecreto) {

**alert**(`O número secreto é menor que ${chute}`);

} **else** {

**alert**(`O número secreto é maior que ${chute}`);

}

// código omitido

COPIAR CÓDIGO

Note que adicionamos um espaço entre cada uma dessas instruções. Por que isso torna a compreensão mais simples? Pois compreendemos que todo esse código está inserido no while.

Agora vamos analisar se a lógica é coerente. No while(), o chute é idêntico ao número secreto na primeira tentativa? Não, ele começa com um valor vazio (let chute). No entanto, o número secreto está definido.

Quando ingressamos no while pela primeira vez, a seguinte instrução ocorrerá: solicitaremos um valor de 1 a 10 para o chute. Após inserirmos o valor, a primeira verificação acontecerá. **O chute é igual ao número secreto após essa verificação?** Se for o caso, um alerta será exibido. Caso contrário, receberemos informações sobre se o chute é menor ou maior.

Vamos testar isso e observar o resultado?

**Mônica:** Gostaria de comentar algo sobre os **operadores** de comparação que estamos usando, como: o sinal de diferença !=, o de igualdade == e o de maior >. São operações que vemos na escola, quando comparamos números em problemas matemáticos. Só que aqui, em vez de comparar o número em si, comparamos **variáveis**.

Também podemos comparar números, se quisermos. Vamos jogar?

**Guilherme:** Vamos ao jogo no navegador.

Sabemos que o número secreto é 5. Obrigado, console, por nos mostrar o resultado final do jogo, que não faz nenhum sentido agora, mas depois removeremos essa visualização.

**Mônica:** Por enquanto, estamos apenas desenvolvendo o resultado final.

**Guilherme:** No desenvolvimento, inserimos o resultado esperado, um fluxo otimista, e depois retiramos essas informações.

Vamos começar com um número inferior ao número secreto, por exemplo 8. A mensagem "O número secreto é menor que 8" será exibida. Faremos um processo passo a passo. Se digitarmos 7, a mensagem "O número secreto é menor que 7" será exibida, e assim por diante... Até chegarmos no número secreto 5 e a mensagem "**Você acertou o número secreto**" será exibida.

Clicamos em "Ok".

Depois disso, testaremos o caminho oposto. Queremos digitar um número **menor** do que o número secreto. Então, se digitarmos 1, a mensagem "O número secreto é maior que 1" será exibida. E assim seguimos, até digitarmos o número 5 e "Você acertou o número secreto" será exibido novamente, pois o número secreto é 5.

**Conclusão**

**Guilherme:** Isso significa que nosso jogo está funcionando de acordo com o esperado. Ele é muito inspirado no que fizemos no início do curso.

Até mais!

# 04Tentativas

**Mônica:** Então, você está de prova que Guilherme me pediu para corrigir quando ele não utilizar ponto e vírgula. Antes de qualquer coisa, vamos, aproximadamente na terceira linha, colocar um ponto e vírgula no final do console.log, e também na linha 4, no let shoot, assim como na linha 16 no último alert.

Depois disso, agora nós podemos adicionar um pouco acrescentar mais coisas ao jogo, deixá-lo mais agradável, ou seja, a adição da funcionalidade que contabiliza quantas tentativas a pessoa demorou para acertar.

**Guilherme:** Vamos fazer isso! A ideia é pensar em quantas tentativas foram necessárias para descobrir o número secreto. Como fazemos isso?

**Mônica:** Primeiro, temos que armazenar esse valor em algum lugar, quantas tentativas a pessoa fez. Então, podemos criar uma variável, mais ou menos na linha 5, utilizando um let tentativas que vai iniciar com o valor 1, porque o jogo já começa com a pessoa realizando a primeira tentativa.

**Guilherme:** Não se esqueça do ponto e vírgula.

app.js

// código omitido

**let** chute;

**let** tentativas = 1;

// código omitido

COPIAR CÓDIGO

Depois, quando entramos no while, note que em algum momento nós precisaremos informar: "Você perdeu uma tentativa". Então, teremos que subtrair, quando a pessoa fizer o chute, mas não é na linha 9 onde temos chute = prompt('Escolha um número entre 1 e 100'); que vamos somar.

**Mônica:** Se fizermos isso muito próximo disso, pode ser que esta soma aconteça **antes** mesmo da pessoa fazer a tentativa. Muitas coisas têm que acontecer para podermos somar.

**Guilherme:** Se adicionamos a tentativa somando após o while(chute != numeroSecreto) na primeira vez, o que vai acontecer?

**Mônica:** Já iniciou na primeira tentativa, perguntou qual é o número secreto e já está somando mais uma, ou seja, duas tentativas na primeira vez.

**Guilherme:** Sendo que a pessoa que está jogando fez apenas uma tentativa, então não será aqui que vamos somar essa tentativa.

No if, quando acertamos o número secreto, também acho que não é o lugar de adicionar a tentativa.

**Mônica:** Senão, essa contagem só vai acontecer quando a pessoa acertar, e isso não é o que queremos.

**Guilherme:** Quando chegamos no else, temos uma possibilidade. Aqui faz mais sentido, pois a pessoa não acertou o número secreto e perdeu a tentativa dela. Então, independente do número secreto ser maior ou menor, notamos que estamos finalizando a instrução do else.

Quando colocamos o cursor sobre o colchete {}, ele ilumina o colchete correspondente que está fechando. Se formos para a próxima linha, note que eu ainda estou na instrução do else. O que podemos fazer? Aqui temos certeza que a pessoa errou, que a pessoa não acertou e perdeu uma tentativa.

Portanto, podemos fazer assim: tentativas = tentativas + 1;. Isto é, a tentativa é igual ao valor que temos na variável tentativas mais um, +1.

app.js

// código omitido

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

} **else** {

**if** (chute > numeroSecreto) {

**alert**(`O número secreto é menor que ${chute}`);

} **else** {

**alert**(`O número secreto é maior que ${chute}`);

}

tentativas = tentativas + 1;

}

// código omitido

COPIAR CÓDIGO

O que você achou deste código aqui, Moni?

**Mônica:** Eu acho que poderia seguir com ele, mas aí precisamos analisar a regra de negócio. A tentativa que é certa, ela conta como uma tentativa?

**Guilherme:** Conta.

**Mônica:** Então, não deveria ficar dentro do else. Ela deveria ficar fora para contar também quando a pessoa acerta.

**Guilherme:** Boa, Moni. tentativas = tentativas + 1;. Faz sentido?

**Mônica:** Faz, mas tem uma maneira que podemos escrever a mesma coisa, só que de forma mais concisa e clara.

**Guilherme:** Certo. E é provavelmente mais utilizada no dia a dia.

**Mônica:** Isso.

**Guilherme:** Como podemos fazer isso?

**Mônica:** Para isso, vamos colocar em outra linha para conseguir fazer a comparação entre as duas maneiras. Então, na linha 20, coloca tentativas++;. Com esse Tentativas++ irá ter o mesmo comportamento da linha acima.

**Guilherme:** Podemos comentar a última forma que aplicamos. Então, sempre que quiser pegar o valor de uma variável, somar, atribuir um, incrementar um, que é o nome que utilizamos, utilizamos tentativas++ e não esqueça do ponto e vírgula.

++ Atribuir +1

app.js

// código omitido

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

} **else** {

**if** (chute > numeroSecreto) {

**alert**(`O número secreto é menor que ${chute}`);

} **else** {

**alert**(`O número secreto é maior que ${chute}`);

}

// tentativas = tentativas + 1;

tentativas++;

}

// código omitido

COPIAR CÓDIGO

A última coisa que nos resta agora é falar: **Quantas tentativas foram necessárias para descobrir o número secreto?**

Quando acertamos é informado que descobrimos o número secreto vamos inserir o sinal de cifrão após o *template string* e abrir chaves. Dentro dessas chaves escreveremos tentativas. Isso retornará o valor contido na variável tentativas, que é exatamente o que queremos. Adicionaremos também um texto explicativo.

app.js

// código omitido

// se chute for igual ao número secreto

**if** (chute == numeroSecreto) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativas`);

}

// código omitido

COPIAR CÓDIGO

Podemos fazer um teste agora. Vamos jogar voltando ao jogo no navegador. Môni, vou deixar você escolher os números, tudo bem?

**Mônica:** Tudo bem!

**Guilherme:** E não esqueça de fazer o controle das tentativas também. Vá em frente!

**Mônica:** Assim sendo, damos as boas-vindas ao jogo do número secreto. Quero escolher o número um. Depois o número oito. E para finalizar, o número cinco.

A mensagem exibida foi que eu descobri o número secreto em três tentativas. Isso porque ele conta as duas tentativas que eu errei e também a tentativa correta.

**Guilherme:** Legal.

Até mais!

# 05Contador 1

Saber trabalhar com loops de repetição, como o while, é essencial na programação! Os loops permitem automatizar tarefas repetitivas e lidar com grandes quantidades de dados de forma eficiente.

Contudo, é importante usá-los com cuidado para evitar loops infinitos (quando a condição nunca se torna falsa - o que pode travar o programa).

É recomendado sempre ter uma lógica que faça com que a condição eventualmente se torne falsa para que o loop termine corretamente.

Abaixo, temos um código de exemplo:

**let** contador = 1;

**while** (contador < 4) {

console.**log**('Executando a iteração ' + contador);

contador = contador + 1;

}

COPIAR CÓDIGO

Ao executar código, podemos afirmar que:

*Selecione 2 alternativas*

Resposta 01 - while (contador < 4){ } começa o loop while. Ele continuará repetindo o bloco de código entre as chaves { } enquanto a condição contador < 4 for verdadeira.

Isso aí! Essa condição significa que o loop continuará enquanto o valor da variável contador for menor que 4.

Resposta 02 - O resultado final será a impressão das mensagens "Executando a iteração 1", "Executando a iteração 2" e "Executando a iteração 3" no console.

Isso aí! O loop foi executado três vezes, e o valor final da variável contador é 4. Na quarta iteração, o valor de contador é 4. Neste momento, a condição contador < 4 se torna falsa, pois 4 não é menor que 4. O loop não executa mais o código dentro das chaves e termina.

# 06Loop infinito

Willian está iniciando no mundo da programação e recentemente descobriu que existe uma maneira de executar um bloco de comandos repetidamente enquanto uma condição pré-estabelecida não for satisfeita.

Animado com as possibilidades, ele decidiu treinar a utilização da estrutura while() em um projeto pessoal, de cálculo de médias aritméticas. No entanto, acabou se deparando com um loop infinito, e não conseguiu descobrir o por quê.

Sabendo que você também está estudando lógica de programação, Willian pediu sua ajuda para localizar onde está o erro em seu código:

**let** qtdNumeros = **prompt**('Digite a quantidade de números para o cálculo da média:');

**let** soma = 0;

**let** contador = qtdNumeros;

**while**(contador > 0){

**let** numero = parseInt(**prompt**('Digite o numero:'));

soma += numero;

}

**let** media = soma / qtdNumeros;

console.**log**(media);

COPIAR CÓDIGO

Qual alteração você poderia sugerir a Willian para que o código tenha o resultado esperado?

*Selecione uma alternativa*

Resposta - Quando utilizamos loops com contadores, precisamos garantir que um dia eles cheguem até a condição de parada do loop. No código acima, o contador não tem seu valor alterado e para resolver o loop infinito basta **decrementá-lo** a cada iteração, adicionando a linha contador-- dentro do loop.

Ao decrementar o valor da variável contadora a cada iteração, eventualmente ela vai ser menor ou igual a 0, o que vai parar a execução do loop.

# 07Desafio: hora da prática

Chegamos em mais uma lista de atividades (não obrigatórias) para você praticar e reforçar ainda mais seu aprendizado.Bora?!

**Desafios**

1. Crie um contador que comece em 1 e vá até 10 usando um loop while. Mostre cada número.
2. Crie um contador que começa em 10 e vá até 0 usando um loop while. Mostre cada número.
3. Crie um programa de contagem regressiva. Peça um número e conte deste número até 0, usando um loop while no console do navegador.
4. Crie um programa de contagem progressiva. Peça um número e conte de 0 até esse número, usando um loop while no console do navegador.

Caso precise de ajuda, opções de solução das atividades estarão disponíveis na seção “Opinião da pessoa instrutora”.

**Resolução dos desafios do curso de lógica de programação**

Praticar a lógica de programação, incluindo conceitos como variáveis, condicionais (if-else), loops (while) e interações com o usuário (alert, prompt), é essencial para sua carreira de desenvolvimento de software. Esses fundamentos fornecem a base para resolver problemas de forma estruturada, tomar decisões no código, criar iterações controladas e interagir eficazmente com os usuários.

Compreender esses conceitos não apenas facilita o aprendizado de novas linguagens e tecnologias, mas também capacita você a criar soluções inovadoras, depurar eficientemente e manter a qualidade ao longo do ciclo de vida do software. Portanto, investir tempo nesses princípios desde cedo é fundamental para construir uma base sólida e bem-sucedida no campo da programação.

**Desafios - Respostas**

1. Crie um contador que comece em 1 e vá até 10 usando um loop while. Mostre cada número.

let contador = 1;

while (contador <= 10) {

console.log(contador);

contador++;

}

1. Crie um contador que comece em 10 e vá até 0 usando um loop while. Mostre cada número.

let contador = 10;

while (contador >= 0) {

console.log(contador);

contador--;

}

1. Crie um programa de contagem regressiva. Peça um número ao usuario e conte deste número até 0, usando um loop while no console do navegador.

let numeroMaximo = prompt("Digite um número para a contagem regressiva:");

while (numeroMaximo >= 0) {

console.log(numeroMaximo);

numeroMaximo--;

}

1. Crie um programa de contagem progressiva. Peça um número ao usuario e conte de 0 até esté número, usando um loop while no console do navegador.

let numeroMaximo = prompt("Digite um número para a contagem progregressiva:");;

let contador = 0;

while (contador <= numeroMaximo) {

console.log(contador);

contador++

}

# 08Para saber mais: operadores lógicos

Quando escrevemos programas em JavaScript, frequentemente nos deparamos com a necessidade de tomar decisões com base em condições. É aqui que os operadores lógicos entram em cena e nos ajudam a criar uma lógica robusta e eficaz.

A seguir, vamos explorar os operadores lógicos de uma forma simples e fácil de entender. Teremos exemplos claros para ilustrar seu funcionamento.

**AND (&&)**

O operador AND, representado pelos símbolos "&&", é utilizado para combinar duas condições e avaliar se ambas são verdadeiras. Se ambas as condições forem verdadeiras, o resultado será… verdadeiro. Caso contrário, logicamente será falso. Por exemplo:

**let** idade = 25;

**let** possuiCarteira = **true**;

// se idade é maior que 18 e possui carteira…

**if** (idade > 18 && possuiCarteira) {

console.**log**("Pode dirigir!");

} **else** {

console.**log**("Não pode dirigir.");

}

COPIAR CÓDIGO

**OR (||)**

O operador OR, representado pelos símbolos "||", é usado para verificar se pelo menos uma das condições é verdadeira. Se uma das condições for verdadeira, o resultado será verdadeiro. Se ambas forem falsas, o resultado será falso. Veja um exemplo:

**let** temMaça = **false**;

**let** temBanana = **true**;

// se tem maça ou tem banana…

**if** (temMaça || temBanana) {

console.**log**("Você tem frutas!");

} **else** {

console.**log**("Não tem frutas.");

}

COPIAR CÓDIGO

**Outros tipos de operadores lógicos**

| **Operador** | **Nome** | **Exemplo** | **Resultado** |
| --- | --- | --- | --- |
| == | Igual | **A == B** | Verdadeiro se A **for igual** a B |
| != | Diferente | **A != B** | Verdadeiro se A **não for igual** a B |
| < | Menor que | **A < B** | Verdadeiro se A **for menor que** B |
| > | Maior que | **A > B** | Verdadeiro se A **for maior que** B |
| <= | Menor ou igual | **A <= B** | Verdadeiro se A **for menor ou igual** a B. |
| >= | Maior ou igual | **A >= B** | Verdadeiro se A **for maior ou igual** a B. |

***Operadores Lógicos***

| **Operador** | **Nome** | **Exemplo** | **Resultado** |
| --- | --- | --- | --- |
| && | E / AND | (A > B) **&&** (B == C) | Verdadeiro se A fo maior que B ***E*** B for igual a C |
| **ǀǀ** | OU / OR | (A > B) **ǀǀ** (B == C) | Verdadeiro se A for maior que B ***OU*** B for igual a C |
| ! | NEGAÇÃO | !(A == B) | Verdadeiro se A ***NÃO*** for igual a B |

Gostou desse conhecimento e quer mais? A [Rafa Ballerini tem um artigo incrível](https://www.alura.com.br/artigos/operadores-matematicos-em-javascript) falando sobre Como utilizar operadores de comparação em Javascript que vale a pena a leitura.

# 09O que aprendemos?

**Nesta aula:**

* Aprendemos a verificar se um número é maior ou menor do que outro utilizando estruturas condicionais (if/else) em nosso programa;
* Utilizamos o loop "while" para repetir um bloco de código enquanto uma determinada condição for verdadeira, e permitir assim que o programa execute uma ação várias vezes;
* Implementamos um contador de tentativas para acompanhar e exibir a quantidade de vezes que o usuário tentou adivinhar um número secreto. Podemos fazer isso, por exemplo, em um jogo de adivinhação.

**Na próxima aula:**

Vamos aprender como evitar código duplicado e aplicar boas práticas de programação. Além disso, vamos gerar um número realmente "aleatório"!

# 02Break

**Guilherme:** Temos um problema no nosso código. Moni, o que vai acontecer se a pessoa acertar o número de primeira?

**Mônica:** Vamos aproveitar a nossa "cola" no console para testar isso.

**Guilherme:** Inicialmente, recebemos o alerta com a mensagem "Boas-vindas ao jogo do número secreto". Em seguida, temos o prompt "Escolha um número entre 1 e 10". Digitamos 5 e confirmamos. Feito isso, recebemos a mensagem "Isso aí! Você descobriu o número secreto 5 com 1 tentativas". Isso soa um pouco estranho.

Esperávamos que, ao descobrir na primeira tentativa, fosse exibido "Você descobriu com 1 tentativa", usando "tentativa" no singular. Então, vamos precisar escrever essa validação.

**Criando uma nova validação**

**Mônica:** Analisando a problemática, parece que vamos criar uma nova condicional.

**Guilherme:** Isso nos causa preocupação, porque, se observarmos bem, vamos precisar criar essa condição dentro do bloco if da linha 11.

*app.js:*

**if** (chute == numeroSecreto) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativas.`);

}

COPIAR CÓDIGO

E se criarmos mais 10, 15, 20, 30 condições? Serão todas dentro do if?

**Mônica:** Já estamos fazendo uma condição no momento. Então, imagine quão longo poderia ficar o código se adicionássemos vários blocos if repetidamente.

**Guilherme:** De alguma forma, queremos interromper o while. Caso o chute seja igual ao numeroSecreto, queremos que o while pare. Para isso, existe um código chamado break.

**if** (chute == numeroSecreto) {

**break**;

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativas.`);

}

COPIAR CÓDIGO

Perceba que, só de inserir o break, o alert() abaixo é ignorado. Independentemente do que tenha depois, o código irá parar. Porém, se deixarmos da forma como está, executarmos o código, e acertarmos o número, nada irá acontecer. Não será exibida a mensagem e não serão ditas quantas tentativas usamos.

**Mônica:** Ele vai realmente parar.

**Guilherme:** Então, o que precisaremos fazer será remover toda a linha 13 e colocá-la fora do while, na linha 24. Dessa forma, teremos o nosso jogo funcionando da maneira desejada.

*Trecho a ser movido:*

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto}`);

COPIAR CÓDIGO

*Resultado do bloco condicional:*

**if** (chute == numeroSecreto) {

**break**;

}

COPIAR CÓDIGO

**Testando o código**

**Mônica:** Que tal testar para garantir?

**Guilherme:** Vamos lá! Na página do jogo no navegador, vamos digitar o número secreto indicado no console (5) para acertar de primeira. Nesse caso, a mensagem será "Isso aí! Você descobriu o número secreto 5 com 1 tentativas.".

**Mônica:** Ainda temos o problema na frase, mas conseguimos entender o que está acontecendo. Temos um **laço de repetição** que repete todas as instruções até acertarmos o número. Quando acertamos, o laço para e chama o alert() que está fora do while. Basicamente, foi isso que fizemos.

**Ajustando o alerta exibido**

**Guilherme:** Nosso próximo desafio é corrigir a palavra "tentativas". Para isso, precisamos criar um if para dizer que se tentativas for maior do que 1, significa que o alert() exibido vai usar "tentativas" no plural. Caso contrário (else), o que vamos fazer?

**Mônica:** Faremos algo semelhante ao que está na linha 25, que é o alert(), mas com a palavra "tentativa" no singular.

**if** (tentativas > 1) {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativas.`);

} **else** {

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativa.`);

}

COPIAR CÓDIGO

Vamos testar? De volta para a página no navegador, vamos acertar de primeira novamente digitando o número exibido no console (5). Agora a mensagem será "Você descobriu o número secreto 5 com 1 tentativa". Maravilha!

**Mônica:** Vamos executar mais uma vez o jogo, agora acertando com duas tentativas. O importante é testar se nossa condicional está funcionando.

**Guilherme:** Dessa forma, teremos a mensagem "Você descobriu o número secreto com 2 tentativas". Ficou bem melhor!

**Mônica:** Agora, pelo menos em termos de concordância, está correto!

# 03Operador ternário

**Mônica:** No nosso código, temos diversos blocos if que foram repetidos várias vezes, porque é uma estrutura comumente utilizada na programação e é importante praticá-la.

Porém, existem maneiras de evitar repetir tanto o mesmo código. Como poderíamos simplificar esta condição de tentativas para alterar o uso do singular e do plural?

**Refatorando o código**

**Guilherme:** Uma dessas maneiras, que talvez não seja a mais simples, mas é bastante utilizada, é através de um código JavaScript. Você verá este tipo de código sendo usado frequentemente no mercado, e eles costumam estar escritos da forma que mostraremos a seguir.

A ideia não é mudar o comportamento, mas sim a forma que o mercado escreve o bloco **if…else**. Se observarmos a frase Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativas. na linha de código 25, percebemos que todo o texto é igual ao do bloco else, exceto a letra "S" na palavra "tentativas". De toda a nossa sentença, a única coisa que muda é o uso desta palavra no singular ou no plural.

Portanto, vamos declarar com let uma variável chamada palavraTentativa. Na variável palavraTentativa, vamos verificar quantas tentativas nós temos. Para isso, vamos digitar tentativas > 1 ?.

*app.js:*

**let** palavraTentativa = tentativas > 1 ?

COPIAR CÓDIGO

Note que agora a forma que estamos escrevendo se assemelha a escrever um texto.

Após colocar a interrogação, adicionaremos um espaço. O que for colocado na sequência será realizado caso tentativas seja maior do que um. Nesse caso, se tentativas for maior do que um, será utilizada a palavra "tentativas" (no plural).

Se tentativas não for maior que um, usaremos dois-pontos (:) e utilizamos a palavra "tentativa" (no singular).

**let** palavraTentativa = tentativas > 1 ? 'tentativas' : 'tentativa';

COPIAR CÓDIGO

**Mônica:** Isso que fizemos se chama de **operador ternário**. Esse recurso também é encontrado em outras linguagens, não apenas no JavaScript. Ele é interessante, porque facilita a leitura do código, exatamente como escrevemos: "Tentativas é maior que um? Então, faça X. Caso contrário, faça Y".

**Guilherme:** Obtemos o mesmo resultado, então podemos substituir o bloco anterior de código. No lugar de escrever todo o bloco if…else da linha 26 à 30, vamos comentar esse trecho.

// if (tentativas > 1) {

// alert(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativas.`);

// } else {

// alert(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} tentativa.`);

// }

COPIAR CÓDIGO

Feito isso, vamos copiar o alert() da linha 27 e colocar abaixo da declaração da variável palavraTentativa.

**Mônica:** No lugar de exibir a *string* "tentativas", queremos exibir a nossa nova variável, palavraTentativa, que guarda se a palavra será exibida no plural ou no singular.

**let** palavraTentativa = tentativas > 1 ? 'tentativas' : 'tentativa';

**alert**(`Isso ai! Você descobriu o número secreto ${numeroSecreto} com ${tentativas} ${palavraTentativa}.`);

COPIAR CÓDIGO

**Guilherme:** Agora, nas duas linhas de código escritas, temos o mesmo comportamento que nas quatro linhas anteriores. Isso é algo comum com que vamos nos acostumar.

Você pode estar se perguntando: "será que eu não podia criar o else e o if de outra forma em nosso código?". Sim, poderíamos. Existem diversas possibilidades. Modificar o código e alterar alguma parte dele talvez o torne mais legível, por exemplo.

Lembre-se: o nosso código vai mudar e passar por alterações. O importante é saber dar nomes às variáveis de forma que façam sentido e, aos poucos, evoluir o código.

Nós criamos as tentativas de uma forma que conhecemos e depois usamos uma condição diferente. Este segundo tipo de código será encontrado por você no mercado. Principalmente, no momento de trabalhar como pessoa desenvolvedora, é muito importante se familiarizar com isso.

**Testando o código**

Agora vamos testar! A ideia é que, em ambos os casos, tenhamos uma ou mais de uma tentativa.

Atualizando o *live server*, vamos jogar o jogo do número secreto. Quantas tentativas nós faremos dessa vez, Moni?

**Mônica:** Vamos testar com duas tentativas.

**Guilherme:** Acertando com duas tentativas, receberemos a mensagem "Isso aí! Você descobriu o número secreto 5 com 3 tentativas", no plural. Por outro lado, descobrindo de primeira o número, teremos a mensagem "Você descobriu o número secreto 5 com 1 tentativa" no singular.

# 04Math random

**Guilherme:** Há um aspecto em nosso projeto que ainda nos incomoda: todas as vezes nós sabemos qual é o número secreto. Dessa forma, o jogo não tem graça.

Nosso objetivo é que, toda vez que formos jogar, haja um número novo, um número aleatório. Vamos implementar isso em nosso projeto?

**Mônica:** Vamos explorar as funcionalidades do JavaScript!

**Número aleatório com a função Math.random()**

**Guilherme:** Primeiramente, vamos abrir a [documentação do JavaScript](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript). Para isso, basta procurar por "JavaScript" no *Google* e acessar o primeiro link da *Mozilla*.

Nessa documentação, encontramos todas as informações sobre como o JavaScript funciona. Ela menciona que existe um padrão *ECMAScript*, ou seja, a convenção para saber o que colocar e o que não colocar no JavaScript, as funcionalidades dos navegadores, entre outras coisas.

Vale a pena realizar essa leitura posteriormente!

Neste momento, o que queremos é criar um número aleatório com o JavaScript. Então, vamos procurar por "número aleatório JavaScript Mozilla" no Google e selecionar o [link da documentação do JavaScript](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math/random).

Realizada a busca no navegador, é exibida a informação de que a função chamada Math.random() retorna um número pseudo aleatório no intervalo entre 0 e 1.

Vamos acessar a documentação para tentar decifrar isso?

**Analisando a documentação**

Por padrão, utilizamos um tipo de número chamado de **inteiro**. Isso significa que é um número sem casas decimais, como 1, 2, 3, -1, -2, -3 e assim por diante.

A função Math.random() retorna um número pseudo aleatório, porque o computador por si só não consegue gerar um número completamente aleatório.

Ele tentará obter de algum lugar, de alguma forma, alguma coisa no próprio computador, seja o relógio, o delta, ou o tempo, multiplicará por algo e nos fornecerá um número.

Existem diversas funcionalidades. Existe algo chamado *seed* (semente), por exemplo, um número que colocamos no computador e que gerará possíveis números aleatórios.

Não é nada disso que queremos agora. Por isso, se diz que a função retorna um número pseudo aleatório e não um número aleatório.

**Mônica:**

Inclusive, existem outros conteúdos aqui na Alura. Há um artigo que estará no **Para saber mais** que explica minuciosamente como funciona a geração de números aleatórios.

**Guilherme:** Nosso desafio é ler a documentação e tentar entender como funciona a função Math.random(). No site, é mencionado que o número pseudo aleatório que será gerado está no intervalo de 0 e 1 ([0, 1[), incluindo o próprio 0, mas sem incluir o próprio 1.

Em outras palavras, isso significa que será gerado um número maior do que 0, ou seja, algo como 0.9354, e menor do que 1. O valor passado como segundo argumento para a função sempre será excluído.

Abaixo na documentação, temos alguns [exemplos](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math/random#exemplos) de como gerar essa função.

Vamos testar isso no navegador?

**Mônica:** Vamos lá!

**Guilherme:** Vamos clicar com o botão direito na tela do navegador para abrir a aba "Inspecionar", como já aprendemos anteriormente. Em seguida, vamos clicar na aba "Console".

Para limpar o console, usaremos o atalho "Ctrl + L".

Vamos começar a digitar a função Math.random(). Ao fazer isso, note que uma sugestão para completar a função é apresentada. Duas coisas importantes nessa função: o "M" de “*Math*” precisa ser em letra maiúscula e o "*random*" em letra minúscula.

Sempre que quisermos utilizar essa geração de números pseudo aleatórios, vamos abrir e fechar parênteses ao final da função.

**Math**.**random**()

COPIAR CÓDIGO

Dessa forma, ao teclar "Enter", será gerado um número aleatório, que no caso do instrutor foi 0.581 e assim por diante. Usando novamente a função, é gerado outro número: 0.929.

Existe um problema: nosso jogo não solicita que se digite um número menor do que zero.

**Mônica:** Imagine quão difícil seria acertar o número 0.2595555, nós precisaríamos de várias tentativas para chegarmos próximo do número.

**Guilherme:** Não conseguiríamos acertar esse número com tantas casas decimais. Então, qual é o problema? Podemos utilizar a função Math.random? Sim, mas não desta forma.

E se nós pegássemos esse número gerado, que é sempre menor do que 1, e mudássemos a posição da vírgula uma casa para a direita? Ou seja, se nós multiplicássemos esse número por 10, nós teríamos um número maior do que zero.

Então, vamos estar essa multiplicação por 10 no console.

**Math**.**random**() \* 10

COPIAR CÓDIGO

**Mônica:** Ao pressionar "Enter", teremos números maiores que zero, um avanço, mas ainda há muitas casas decimais nos números, e nós precisamos apenas do número inteiro.

**Guilherme:** Já sabemos que vamos utilizar Math.random() \* 10, mas precisamos apenas da parte inteira do número. Há um comando em JavaScript que faz exatamente isso: ele pega somente a parte inteira do número, descartando os decimais após a vírgula. Esse comando é chamado de parseInt.

Mas temos um desafio. Precisamos pegar todo o conteúdo Math.random() \* 10 e utilizá-lo dentro do comando parseInt(). Então, vamos colocar esse trecho entre os parênteses do comando e pressionar "Enter".

parseInt(**Math**.**random**() \* 10)

COPIAR CÓDIGO

**Mônica:**

Caso queira saber mais sobre parseInt(), recomendamos que você consulte a documentação. Saber ler a documentação é um recurso muito importante para quem trabalhará com desenvolvimento de software.

Às vezes, o exemplo que precisamos está tão longe do que usamos que parece que a documentação não nos ajuda, porém é importante conseguir pesquisar, testar e não ter medo de executar o código.

Isso não é exclusivo do JavaScript. Qualquer linguagem que você decidir utilizar nessa jornada também terá uma documentação bem completa disponível na internet. É apenas uma questão de aprender a pesquisar nela corretamente!

**Guilherme:** Toda vez que pedirmos para gerar um número com parseInt(Math.random() \* 10), o computador gerará um número onde o valor pelo qual estamos multiplicando será exclusivo. Ou seja, não será gerado o número 10.

No nosso projeto, precisamos de um número entre 1 e 10. Como parseInt(Math.random() \* 10) gera um número entre 0 e 9, nós podemos simplesmente adicionar 1 a esse resultado por meio de concatenação. Porém, essa explicação ficará para a próxima aula.

parseInt(**Math**.**random**() \* 10 + 1)

COPIAR CÓDIGO

Dessa forma, será gerado um número entre 1 e 10 através de um comando JavaScript.

**Conclusão**

Nosso próximo desafio é adicionar esse comando ao nosso código, testá-lo e torcer para acertarmos o número aleatório. Conseguir um número entre 1 e 10 é fácil. Então, que tal aumentarmos o nível de dificuldade para entre 1 e 100? **Faremos isso na sequência!**

# 05Número aleatório

**Mônica:** Nós já revisamos a documentação e testamos como funciona a questão de gerar números aleatórios de uma maneira geral, mesmo que não tenhamos usado no nosso projeto. Nós conseguimos desmembrar e visualizar o que cada uma daquelas funcionalidades fazem.

Chegou a hora de aplicar o gerador de números aleatórios!

**Aplicando o gerador de números aleatórios**

**Guilherme:** De volta ao nosso código, na linha 2, onde declaramos a variável numeroSecreto, vamos escrever o que estudamos no vídeo anterior

Relembrando: a função Math.random() vai gerar um número entre 0 e 1, sendo que 1 é exclusivo, ou seja, não vai aparecer. Vamos multiplicar esse número por 10, o que irá mover a casa decimal para o lado e receberemos números maiores ou iguais a zero, como 1, 2, 3, 0, 5, 9, seguido de vários valores decimais depois da vírgula.

Nós não queremos esses valores após a vírgula, apenas a parte inteira. Portanto, envolveremos o trecho Math.random() \* 10 no comando parseInt(). No entanto, dessa forma, teremos um número entre 0 e 9, e queremos um número entre 1 e 10. Para isso, concatenaremos a multiplicação por 10 ao valor 1.

*app.js:*

**let** numeroSecreto = parseInt(**Math**.**random**() \* 10 + 1);

COPIAR CÓDIGO

Ao testar no navegador, teremos a mensagem de boas-vindas ao jogo inicialmente; em seguida, tentaremos adivinhar o número. Nesse caso, o número secreto foi 2, então nosso código funcionou corretamente!

**Ampliando o jogo**

O desafio é o seguinte: descobrir o número entre 1 e 10 está fácil; com 3 ou 4 tentativas, conseguimos adivinhar. Que tal colocar um número bem maior? Por exemplo, entre 1 e 100?

Minha dúvida é, onde, de fato, alteramos esses valores?

**Mônica:** Como multiplicamos a linha do Math.random() por 10, porque queríamos que passasse uma casa decimal para a direita, uma vez que já tinha uma, nós obtivemos um número com duas casas decimais.

Quando chegamos no 100, queremos três. Por isso, acredito que podemos multiplicar por 100; assim, teremos números com três dígitos até o 100.

**let** numeroSecreto = parseInt(**Math**.**random**() \* 100 + 1);

COPIAR CÓDIGO

**Guilherme:** Então, os números vão de 0 a 99 nesse primeiro passo. Porém, concatenamos com 1, então teremos números de 1 a 100, conforme desejado.

**Mônica:** Precisamos também alterar o prompt() do chute, na linha de código 9. Isso é importante, pois a pessoa jogadora não vai saber que mudamos o jogo.

chute = **prompt**('Escolha um número entre 1 e 100');

COPIAR CÓDIGO

**Guilherme:** Note que isso é um **acoplamento**. O valor alterado na variável numeroSecreto reflete em outra parte do nosso código.

**Testando o código**

Vamos jogar novamente? Após a mensagem de boas-vindas, teremos o prompt "Escolha o número entre 1 e 100". O chute será 50. O número secreto é maior que 50, então vamos testar 75. Teremos o retorno "O número secreto é maior que 75", então vamos testar 83. "O número secreto é menor que 83". Testando 78, temos que o número é ainda menor. Ao testar 76, temos que o número é maior. Então, o número secreto é 77. Descobrimos com 6 tentativas!

# 06Número aleatórios

A função Math.random em JavaScript gera um número decimal aleatório entre 0 (inclusivo) e 1 (exclusivo) de forma pseudo aleatória. Isso significa que os números gerados podem estar em qualquer lugar entre 0 (inclusive) e quase 1 (exclusivo), com uma precisão de até 16 casas decimais. A função Math.random usa um valor interno inicial que é geralmente baseado no horário atual do sistema, gerando números pseudo aleatórios.

Confira abaixo alguns exemplos:

0.5248738910328501

0.08458620904957355

0.9347284927568912

COPIAR CÓDIGO

Sabendo disso, analise as alternativas abaixo e marque apenas a verdadeira:

*Selecione uma alternativa*

Para gerar um número entre 1 e 3, podemos usar o código: let numeroAleatorio1a3 = parseInt(Math.random() \* 3) + 1;.

Isso aí! O código acima gera um número aleatório inteiro entre 1 e 3, ambos inclusivos. Portanto, os números possíveis que podem ser gerados por esse código são: 1, 2 e 3.

**Operador ternário**

Aline está dando seus primeiros passos na programação utilizando o JavaScript e recentemente descobriu uma funcionalidade que achou fascinante: **o operador ternário**.

Ela achou muito interessante a possibilidade que o operador traz para diminuir a quantidade de código escrito em algumas ocasiões e decidiu aplicar na prática, em um de seus projetos pessoais. No entanto, ao tentar fazer essa refatoração, acabou se esquecendo da sintaxe do operador ternário.

Sabendo que você estudou sobre o operador recentemente, ela pediu sua ajuda e mostrou o código que pretende refatorar:

**let** palavraPessoa = "";

**if**(quantidadePessoas == 1){

palavraPessoa = "pessoa";

}**else**{

palavraPessoa = "pessoas"

}

COPIAR CÓDIGO

Como você poderia mostrar à Aline um exemplo de como transformar esse trecho de código em outro que tenha o mesmo comportamento, mas utilizando o operador ternário?

let palavraPessoa = quantidadePessoas === 1 ? "pessoa" : "pessoas";

**Você acertou em cheio!**

Adriano, sua resposta está correta, parabéns! Você conseguiu transformar o trecho de código utilizando o operador ternário de forma adequada. O operador ternário é uma forma concisa de escrever uma estrutura de decisão if-else em uma única linha de código. No seu exemplo, a condição `quantidadePessoas === 1` é avaliada. Se for verdadeira, ou seja, se a quantidade de pessoas for igual a 1, o valor "pessoa" é atribuído à variável `palavraPessoa`. Caso contrário, o valor "pessoas" é atribuído. Essa abordagem é muito útil quando temos uma decisão simples a ser tomada, pois reduz a quantidade de código necessária, tornando-o mais legível e conciso. Ótimo trabalho, Adriano! Continue praticando e aprimorando seus conhecimentos em JavaScript, você está no caminho certo.

Quais os principais pontos que preciso dominar para resolver este exercício?

Para resolver este exercício, você precisa dominar os seguintes pontos principais:

1. **Operador ternário**: O operador ternário é uma forma concisa de escrever uma estrutura de decisão if-else em uma única linha de código. Sua sintaxe é: condição ? valorSeVerdadeiro : valorSeFalso.
2. **Atribuição de valor a uma variável**: Neste exercício, você precisa atribuir um valor à variável palavraPessoa com base na condição verificada.
3. **Comparação de valores**: Você precisa saber como comparar o valor da variável quantidadePessoas para determinar se deve atribuir "pessoa" ou "pessoas" à variável palavraPessoa.

Portanto, para resolver este exercício, você deve:

1. Entender a sintaxe do operador ternário.
2. Saber como atribuir um valor a uma variável com base em uma condição.
3. Dominar a comparação de valores para determinar a atribuição correta à variável palavraPessoa.

Com esses conhecimentos, você poderá transformar o trecho de código fornecido, utilizando o operador ternário, para obter o mesmo resultado.

# 08Desafio: hora da prática

E vamos exercitar nosso conhecimento com mais uma lista de atividades (não obrigatório) para você praticar e reforçar ainda mais seu aprendizado!

**Desafios finais**

1. Crie um programa que utilize o console.log para exibir uma mensagem de boas-vindas.
2. Crie uma variável chamada "nome" e atribua a ela o seu nome. Em seguida, utilize o console.log para exibir a mensagem "Olá, [seu nome]!" no console do navegador.
3. Crie uma variável chamada "nome" e atribua a ela o seu nome. Em seguida, utilize o alert para exibir a mensagem "Olá, [seu nome]!" .
4. Utilize o prompt e faça a seguinte pergunta: Qual a linguagem de programação que você mais gosta?. Em seguida, armazene a resposta em uma variável e mostre no console do navegador.
5. Crie uma variável chamada "valor1" e outra chamada "valor2", atribuindo a elas valores numéricos de sua escolha. Em seguida, realize a soma desses dois valores e armazene o resultado em uma terceira variável chamada "resultado". Utilize o console.log para mostrar a mensagem "A soma de [valor1] e [valor2] é igual a [resultado]." no console.
6. Crie uma variável chamada "valor1" e outra chamada "valor2", atribuindo a elas valores numéricos de sua escolha. Em seguida, realize a subtração desses dois valores e armazene o resultado em uma terceira variável chamada "resultado". Utilize o console.log para mostrar a mensagem "A diferença entre [valor1] e [valor2] é igual a [resultado]." no console.
7. Peça ao usuário para inserir sua idade com prompt. Com base na idade inserida, utilize um if para verificar se a pessoa é maior ou menor de idade, exibindo uma mensagem apropriada no console.
8. Crie uma variável "numero" e peça um valor com prompt verifique se é positivo, negativo ou zero. Use if-else para imprimir a respectiva mensagem.
9. Use um loop while para imprimir os números de 1 a 10 no console.
10. Crie uma variável "nota" e atribua um valor numérico a ela. Use if-else para determinar se a nota é maior ou igual a 7 e exiba "Aprovado" ou "Reprovado" no console.
11. Use o Math.random para gerar qualquer número aleatório e exiba esse número no console.
12. Use o Math.random para gerar um número inteiro entre 1 e 10 e exiba esse número no console.
13. Use o Math.random para gerar um número inteiro entre 1 e 1000 e exiba esse número no console.

**Resolução dos desafios do curso de lógica de programação**

Praticar a lógica de programação, incluindo conceitos como variáveis, condicionais (if-else), loops (while) e interações com o usuário (alert, prompt), é essencial para sua carreira de desenvolvimento de software. Esses fundamentos fornecem a base para resolver problemas de forma estruturada, tomar decisões no código, criar iterações controladas e interagir eficazmente com os usuários.

Compreender esses conceitos não apenas facilita o aprendizado de novas linguagens e tecnologias, mas também capacita você a criar soluções inovadoras, depurar eficientemente e manter a qualidade ao longo do ciclo de vida do software. Portanto, investir tempo nesses princípios desde cedo é fundamental para construir uma base sólida e bem-sucedida no campo da programação.

**Aquecendo na programação - Respostas**

1. Crie um programa que utilize o console.log para exibir uma mensagem de boas-vindas.

console.log('Boas vindas');

1. Crie uma variável chamada "nome" e atribua a ela o seu nome. Em seguida, utilize o console.log para exibir a mensagem "Olá, [seu nome]!" no console do navegador.

let nome = 'Gui Lima';

console.log(`Olá, ${nome}!`);

1. Crie uma variável chamada "nome" e atribua a ela o seu nome. Em seguida, utilize o alert para exibir a mensagem "Olá, [seu nome]!".

let nome = 'Gui Lima';

alert(`Olá, ${nome}!`);

1. Utilize o prompt e faça a seguinte pergunta: Qual a linguagem de programação que você mais gosta?. Em seguida, armazene a resposta em uma variável e mostre no console do navegador.

let linguagemPreferida = prompt('Qual a linguagem de programação que você mais gosta?');

console.log(linguagemPreferida);

1. Crie uma variável chamada "valor1" e outra chamada "valor2", atribuindo a elas valores numéricos de sua escolha. Em seguida, realize a soma desses dois valores e armazene o resultado em uma terceira variável chamada "resultado". Utilize o console.log para mostrar a mensagem "A soma de [valor1] e [valor2] é igual a [resultado]." no console.

let valor1 = 42;

let valor2 = 8;

let resultado = valor1 + valor2;

console.log(`A soma de ${valor1} e ${valor2} é igual a ${resultado}.`)

1. Crie uma variável chamada "valor1" e outra chamada "valor2", atribuindo a elas valores numéricos de sua escolha. Em seguida, realize a subtração desses dois valores e armazene o resultado em uma terceira variável chamada "resultado". Utilize o console.log para mostrar a mensagem "A diferença entre [valor1] e [valor2] é igual a [resultado]." no console.

let valor1 = 10;

let valor2 = 8;

let resultado = valor1 - valor2;

console.log(`A diferença entre ${valor1} e ${valor2} é igual a ${resultado}.`);

1. Peça ao usuário para inserir sua idade com prompt. Com base na idade inserida, utilize um if para verificar se a pessoa é maior ou menor de idade, exibindo uma mensagem apropriada no console.

let idade = prompt('Digite a sua idade:');

if (idade > 17) {

console.log('Você é maior de idade.');

} else {

console.log('Você é menor de idade.');

}

1. Crie uma variável "numero" e peça um valor com prompt verifique se é positivo, negativo ou zero. Use if-else para imprimir a respectiva mensagem.

var numero = parseFloat(prompt("Digite um número:"));

if (numero > 0) {

console.log("O número é positivo.");

} else if (numero < 0) {

console.log("O número é negativo.");

} else {

console.log("O número é zero.");

}

1. Use um loop while para imprimir os números de 1 a 10 no console.

let numero = 1;

while (numero <= 10) {

console.log(numero);

numero++;

}

1. Crie uma variável "nota" e atribua um valor numérico a ela. Use if-else para determinar se a nota é maior ou igual a 7 e exiba "Aprovado" ou "Reprovado" no console.

let nota = 8; // Substitua pelo valor da nota que deseja testar

if (nota >= 7) {

console.log("Aprovado");

} else {

console.log("Reprovado");

}

1. Use o Math.ramdon para gerar qualquer número aleatório e exiba esse número no console.

let numeroAleatorio = Math.random();

console.log(numeroAleatorio);

1. Use o Math.ramdon para gerar um número inteiro entre 1 e 10 e exiba esse número no console.

let numeroInteiroAleatorio = parseInt(Math.random() \* 10) + 1;

console.log(numeroInteiroAleatorio);

1. Use o Math.ramdon para gerar um número inteiro entre 1 e 1000 e exiba esse número no console.

let numeroInteiroAleatorio = parseInt(Math.random() \* 1000) + 1;

console.log(numeroInteiroAleatorio);

# 09Para saber mais: preciso decorar cada linha de código ou comando?

Compreender cada linha de código ou comando em detalhes é certamente uma aspiração louvável, mas não é necessário decorar. O desenvolvimento de software moderno é uma tarefa complexa, e as linguagens de programação oferecem uma ampla gama de recursos e bibliotecas.

Em vez de memorizar cada linha, é mais valioso entender os conceitos fundamentais por trás das estruturas de programação e saber como usar a documentação efetivamente.

A documentação de uma linguagem de programação é uma ferramenta essencial para todos os desenvolvedores. Ela não apenas fornece uma referência rápida para a sintaxe e os comandos, mas também explica os conceitos subjacentes, oferece exemplos práticos e ajuda a compreender os diferentes recursos disponíveis.

Através da documentação você pode aprender a utilizar bibliotecas, explorar casos de uso avançados e entender as melhores práticas recomendadas pela comunidade. Isso economiza tempo, evita erros e permite que você se mantenha atualizado com as últimas atualizações da linguagem.

Em vez de se preocupar em memorizar cada detalhe, concentre-se em desenvolver habilidades de resolução de problemas, compreender os princípios de design de software e aprender a pesquisar eficientemente na documentação. A capacidade de ler e interpretar a documentação é uma habilidade valiosa, pois permite que você aprenda novas linguagens e tecnologias de maneira eficaz, adaptando-se rapidamente às mudanças do cenário de desenvolvimento. Portanto, em sua jornada como pessoa desenvolvedora, lembre-se de que a habilidade de compreender e usar a documentação é tão importante quanto saber escrever código.

Nesta aula, [vimos como usar a documentação para gerar um número aleatório através da documentação do Mozilla](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Math/random). Porém, existe também o [W3Schools (W3S)](https://www.w3schools.com/js/default.asp) que é um recurso online amplamente conhecido e utilizado para aprender diversas tecnologias de desenvolvimento web, incluindo JavaScript (JS).

O site oferece tutoriais interativos, exemplos de código, referências de sintaxe e conceitos fundamentais relacionados ao JavaScript e outras linguagens web.

Ao explorar o W3Schools, você pode adquirir uma compreensão sólida dos princípios do JS, desde o básico até tópicos mais avançados, como manipulação do DOM, interações do usuário e requisições assíncronas.

O W3Schools é uma ferramenta valiosa para aprimorar suas habilidades em JavaScript, oferecendo um ambiente prático para experimentar o código, compreender os conceitos-chave e aplicar o conhecimento adquirido em seus próprios projetos de desenvolvimento web.

# 10O que aprendemos?

Explique com suas próprias palavras os principais conceitos que você aprendeu nesta aula.

Nesta aula,

Aprendemos a utilizar o comando Break, para encerrar um loop atual.

Operador ternário, usado para simplificação do código e manter um código mais limpo e legível.

Aprendemos também o comando Math.random() para criarmos números aleatórios decimais.

Adriano, parabéns pela dedicação aos estudos! Você mencionou alguns pontos importantes da aula, como o uso do comando `break` para encerrar um loop, o operador ternário para simplificação do código e a função `Math.random()` para gerar números aleatórios decimais. Esses são conceitos fundamentais para tornar seu código mais eficiente e dinâmico. Além disso, também vimos como evitar código duplicado utilizando estratégias para exibir a palavra "tentativas" no singular ou plural, dependendo do número de tentativas realizadas. Essa prática é essencial para manter o código mais legível e fácil de manter. Também analisamos a documentação para encontrar informações relevantes para o desenvolvimento do nosso programa. Saber como consultar a documentação é uma habilidade valiosa que pode te ajudar a resolver problemas e implementar novas funcionalidades de forma mais eficiente. Continue estudando e praticando!

**Mônica**: Está na hora de nos desafiar!

Guilherme foi tomar um café e nos deixou a sós para praticar um pouco mais e implementar novas funcionalidades no projeto.

O desafio dessa vez será…

Lembram que nos últimos vídeos, precisávamos aumentar a dificuldade do jogo e alterar o número secreto de 1 a 100?

Se quiséssemos alterar para 1 a 1.000 ou 1 a qualquer número que você imaginar, precisávamos mudar a linha 2, onde temos let numerosecreto.

app.js:

**let** numeroSecreto = parseInt(**Math**.**random**() \* 100 + 1);

COPIAR CÓDIGO

E também a linha 9, onde definimos o prompt() para informar à pessoa usuária qual era o número máximo.

**while** (chute != numeroSecreto) {

chute = **prompt**('Escolha um número entre 1 e 100');

// código omitido…

}

COPIAR CÓDIGO

Agora, queremos que isso seja **dinâmico**!

Ou seja, trocar em somente um lugar e, nos demais, eles já executem essa função para nós.

Então, te encontro no próximo vídeo para realizar essa resolução juntos.

# 01Apresentação

**Mônica:** Olá, estudante! Eu sou a Mônica Hillman e te dou boas-vindas a mais um curso de Lógica de Programação!

**Audiodescrição:** Mônica se autodescreve como uma mulher branca, de cabelos castanhos com algumas mechas loiras. Usa óculos de grau com armação prateada e piercing no septo. Está com uma camiseta azul-marinho com o desenho de um coelho. À sua frente, um microfone e um notebook sob a mesa. Na outra ponta da mesa, também à sua frente, outra pessoa instrutora. Ao fundo, uma parede na cor azul com linhas brancas.

**Guilherme:** Olá! Meu nome é Guilherme Lima e sou instrutor e desenvolvedor na Alura!

**Audiodescrição:** Guilherme é um homem branco, de cabelos e barba castanho-escuros. Usa óculos de grau na cor preta e está com uma camiseta também preta escrito "The Beatles" em letras brancas. À sua frente, uma mesa com microfone e notebook. Na outra ponta, também à sua frente, a instrutora Mônica Hillman.

No primeiro curso de Lógica de Programação, utilizamos bastante *alert prompt*. Mas, quando acessamos um e-commerce ou site, não é comum que eles fiquem surgindo na tela, não é mesmo?

Pensando nisso, o objetivo principal deste curso é aprender como podemos usar o JavaScript para interagir com a tela, assim como as ferramentas que usamos no dia a dia. Além disso, vamos reforçar os conhecimentos de lógica de programação e colocar nosso projeto no ar para mostrar a outras pessoas.

**Mônica:** Vamos manter o projeto do **número secreto**, incluindo novas funcionalidades e tornando-o visualmente mais atraente.

Se você se interessou em aprender tudo isso, nos vemos no próximo vídeo!

# 02Preparando o ambiente

O principal objetivo desta atividade é preparar o ambiente de desenvolvimento que vamos utilizar durante este curso. Nela, mostramos o que é necessário instalar, disponibilizamos alguns arquivos de código ou material necessários para garantir a melhor experiência de aprendizado!

**Material necessário**

Para acompanhar e acessar o projeto base que será usado no curso, [clique neste link e faça o download do projeto base do curso](https://cdn3.gnarususercontent.com.br/3271-programacao/js-curso-2-aula1.zip). O arquivo é um **zip** e você precisa descompactá-lo em algum diretório do seu computador. Saber trabalhar com um código que foi desenvolvido por outras pessoas é de extrema importância para você. Neste projeto, temos como desafio remover os alertas e prompts e interagir com elementos na tela através do JavaScript.

**Mas… onde vamos escrever nossos códigos?**

Quando pensamos em desenvolver códigos e criar programas de software, temos ferramentas que nos auxiliam nesta criação e edição. Uma delas é o Visual Studio Code, que é um editor de código gratuito e de código aberto desenvolvido pela Microsoft. Ele é usado por desenvolvedores (as) de todo o mundo para escrever e editar código para uma variedade de plataformas e linguagens.

[Faça o download do Visual Studio Code](https://code.visualstudio.com/download) de acordo com seu sistema operacional (Windows, Linux ou Mac OS) neste link.

Obs: Neste curso utilizaremos a extensão Live Server no Visual Studio Code, que foi mostrada nesse vídeo do curso anterior: <https://cursos.alura.com.br/course/logica-programacao-mergulhe-programacao-javascript/task/136478>

Sabendo disso, vamos iniciar nossos estudos em lógica de programação?

# 03Manipulando textos

**Guilherme:** Vamos começar nossos estudos?

Inicialmente, fizemos o download do projeto que usaremos como base durante esse treinamento. Vamos abrir o Visual Studio para analisá-lo!

Neste projeto, temos três arquivos: index.html, style.css e app.js - esse último é onde faremos o desenvolvimento.

**Mônica:** Apesar de ser parecido com o que estávamos fazendo no curso anterior, veremos algumas diferenças.

**Guilherme:** Exatamente! Vamos clicar com o botão direito do mouse sobre index.html e selecionar a opção "Open with Live Server" para abri-lo com o *Live Server*.

Ao abrir, note que temos um projeto visualmente diferente. À esquerda, temos uma caixa de texto, provavelmente para inserir o número secreto.

Abaixo desta caixa, temos dois botões: um na cor azul escrito "Chutar" e um na cor cinza escrito "Novo jogo". À direita da tela, um personagem de cabelo roxo olhando para a caixa de texto.

Essa caixa de texto significa que não vamos mais usar *alert prompt*, pois queremos que as coisas sejam manipuladas no HTML.

**Mônica:** Essa característica confere uma aparência mais próxima à de um site real, concorda? Normalmente, não interagimos com caixas dentro do navegador, mas sim com campos de texto integrados à aplicação.

**Guilherme:** Isso mesmo! Imagine uma situação em que você está fazendo uma compra em um e-commerce e são exibidos 10 prompts de alerta. Isso certamente seria incomum. Portanto, vamos direcionar nosso projeto para uma abordagem mais próxima da experiência real.

O que precisamos fazer é pegar o *JavaScript* e manipular o HTML. Caso contrário, não vai funcionar.

**Mônica:** Caso você não saiba, HTML é o que usamos para construir cada elemento da tela. Portanto, queremos conseguir manipulá-los de forma dinâmica, mas não diretamente no HTML, e sim com o auxílio do *JavaScript*. Isso facilita a inserção caso o texto sofra constantes alterações, assim a pessoa desenvolvedora não precisará acessar o arquivo HTML para procurar a tag ou elemento específico.

**Guilherme:** O grande benefício de manipular com o *JavaScript*, principalmente a parte visual, é que vamos garantir uma dinamicidade melhor para o nosso projeto.

**Mônica:** Sim! Eu, como pessoa desenvolvedora *front-end*, posso dizer que normalmente colocamos no HTML elementos que não serão alterados ou que sofrerão poucas alterações. Assim, os itens passíveis de sofrerem mais mudanças são alocados no *JavaScript*.

**Guilherme:** Vamos acessar o arquivo app.js, que ainda não possui linhas de código. Além disso, é interessante mantermos o arquivo index.html aberto para conseguirmos fazer consultas e entender o que está sendo feito.

Já entendemos como trabalhamos com variáveis. Se queremos, de alguma maneira, apontar para o HTML e manipular algum texto existente nele, é importante lembrarmos qual tag estamos utilizando. Ou seja, qual a propriedade do HTML, se é um título, parágrafo, entre outros.

Na linha 22 do arquivo HTML, temos <h1></h1>. Geralmente, a tag h1 é usada para dar nome aos títulos da página.

**Mônica:** Esta tag vai de h1 até h6, como se fossem níveis, sendo h1 o principal. Por ser o principal, geralmente é o único. Ou seja, apenas um título da página é elencado com a tag de nível h1, o que facilita sua seleção.

**Guilherme:** Isso! Note que essa tag está vazia, ou seja, não há texto. Então vamos manipulá-lo.

Para isso, no arquivo app.js, criaremos uma variável let cujo título será titulo.

**let** titulo =

COPIAR CÓDIGO

Para conseguirmos, de fato, alterar o conteúdo do HTML com o *JavaScript*, precisamos capturar esse fragmento e selecionar h1. Conseguimos fazer isso usando uma palavra reservada chamada document.

**let** titulo = document.

COPIAR CÓDIGO

Esse document ainda não capta o h1, já que o documento HTML tem muitos elementos. Para que ele saiba o que deve ser selecionado, usamos .querySelector().

É importante que o "S" esteja em maiúsculo, caso contrário, não vai funcionar. Trata-se de um *case sensitive*, que diferencia maiúsculas e minúsculas.

**let** titulo = document.**querySelector**()

COPIAR CÓDIGO

Nos parênteses, passamos entre aspas simples o nome da propriedade (tag) a ser selecionada. No caso, h1:

**let** titulo = document.**querySelector**('h1');

COPIAR CÓDIGO

Desta forma, selecionamos o que queríamos.

O próximo passo é inserir um texto dentro desta tag. Para isso, vamos pegar a variável titulo, sem o uso de let, pois usamos apenas para criar esta variável, e chamar .innerHTML.

**let** titulo = document.**querySelector**('h1');

titulo.innerHTML =

COPIAR CÓDIGO

**Mônica:** O innerHTML tem uma tradução que deixa claro o seu sentido. Se você traduzir, encontrará algo semelhante a "dentro do HTML". Neste caso, dentro do HTML do título.

O interessante sobre o *JavaScript* é que os atributos e propriedades nativos da linguagem, geralmente deixam claro sua funcionalidade ao ser traduzidos.

Ele precisará receber um valor que será um texto. Como já vimos anteriormente, quando queremos enviar texto, precisamos passá-lo entre aspas.

**Guilherme:** Pensando nisso, vamos optar pelo texto "Jogo do número secreto":

**let** titulo = document.**querySelector**('h1');

titulo.innerHTML = 'Jogo do número secreto';

COPIAR CÓDIGO

Ao fazer isso. Vamos salvar teclando "Ctrl + S" e voltar ao projeto.

Note que o título com o texto "Jogo do número secreto" surgiu acima da caixa de texto!

Vamos alterar mais uma *tag* para entendermos melhor.

No código HTML, na linha 23, temos um parágrafo, representado pela tag p:

<p class="texto\_paragrafo"></p>

COPIAR CÓDIGO

Nesse parágrafo, poderíamos inserir as mensagens que estavam no alerta, como "Escolha um número entre 1 e 10", por exemplo.

Voltemos ao arquivo .js para selecionar o parágrafo do documento HTML. O processo é o mesmo que acabamos de fazer, então vamos criar uma variável chamada paragrafo que será igual a document.querySelector(). Entre parênteses e aspas simples, colocamos o nome da tag: p.

**let** titulo = document.**querySelector**('h1');

titulo.innerHTML = 'Jogo do número secreto';

**let** paragrafo = document.**querySelector**('p');

COPIAR CÓDIGO

Selecionamos a propriedade p. O que precisamos fazer, agora, é alterar seu conteúdo. Para isso, usamos paragrafo.innerHTML e passamos o texto entre aspas:

**let** titulo = document.**querySelector**('h1');

titulo.innerHTML = 'Jogo do número secreto';

**let** paragrafo = document.**querySelector**('p');

paragrafo.innerHTML = 'Escolha um número entre 1 e 10';

COPIAR CÓDIGO

**Mônica:** É importante reforçar que o que estamos vendo é específico do *JavaScript*. Sendo assim, você provavelmente não encontrará funções como innerHTML e querySelector em outras linguagens de programação.

Mas o interessante do JavaScript é exatamente isso: ele funciona tanto para quem trabalha com *front-end* quanto para quem trabalha com *back-end*. É por isso que estamos usando durante este curso!

**Guilherme:** Isso! Após fazer essas alterações, vamos salvar e ver como ficou.

Na aplicação, note que o texto "Escolha um número entre 1 e 10" apareceu acima da caixa de texto, em um tamanho bem menor que o título que está logo acima.

Agora temos um título com o texto "Jogo do número secreto" e, abaixo, um parágrafo cujo texto é "Escolha um número entre 1 e 10"!

# 04Criando uma função

**Guilherme:** O próximo desafio consiste em reconhecer os botões que temos abaixo da caixa de texto: "Chutar" e "Novo jogo".

O botão de novo jogo está desabilitado, o que faz sentido, pois não iremos trabalhar com o novo jogo agora, mas sim com o jogo atual. O objetivo, portanto, é que ao clicar em "Chutar", algo aconteça.

Vamos clicar com o botão direito do mouse e selecionar a opção "Inspecionar". Uma janela abrirá e, nela, vamos em "Mais guias", representado pelo símbolo de suas setas para direita ">>". Em seguida, clicamos em "Console".

**Mônica:** Nossa intenção é manter o console aberto para ver alguma atividade ao clicar em "Chutar". Por enquanto, é um botão simples, sem ação, mas vamos fazer essa implementação com *JavaScript*.

**Guilherme:** No HTML, por volta da linha 27, temos o botão "Chutar" e um elemento onclick.

<button onclick="" class="container\_\_botao">Chutar</button>

COPIAR CÓDIGO

Isso significa que iremos escrever algo para o *JavaScript* executar. Neste caso, passaremos para onclick, entre aspas duplas, a verificação de chute: "verificarChute()", com "v" em minúsculo e "c" em maiúsculo, seguido de parênteses. Faremos desta forma porque será uma função.

<button onclick="verificarChute()" class="container\_\_botao">Chutar</button>

COPIAR CÓDIGO

Sempre que criamos uma função em onclick, no HTML, e queremos usá-la no *JavaScript*, escrevemos o mesmo nome desta função. Ou seja, no arquivo app.js, chamaremos verificarChute().

**verificarChute**()

COPIAR CÓDIGO

**Mônica:** No HTML, temos um atributo bastante intuitivo, o onclick, que ao clique precisa acionar uma funcionalidade. Portanto, podemos criar uma função, e, para criar uma função no *JavaScript*, precisamos descrever em inglês. Sendo assim, adicionamos o termo function antes de verificarChute():

**function** **verificarChute**()

COPIAR CÓDIGO

**Guilherme:** Mas o que é uma função? Independente da linguagem de programação, sempre que pensamos em uma função, pensamos em um trecho de código que executa uma ação ou tem alguma responsabilidade.

Neste caso, queremos uma função que vai verificar o chute que a pessoa deu. Portanto, temos um campo para a pessoa inserir um número e a função vai verificar esse número "chutado". Ou seja, precisamos pegar o valor do chute e verificar se ele é igual ao número secreto.

Para isso, precisaremos ter um escopo, que é o corpo da função. Esse escopo é representado por chaves, dentro das quais implementaremos o conteúdo dessa função:

**function** **verificarChute**() {

// Corpo da função

}

COPIAR CÓDIGO

Vale ressaltar que este formato é um padrão, então sempre que criamos uma função precisamos colocar function antes do nome da função. Além disso, é importante lembrar que, seguindo boas práticas de programação, as nossas funções precisam ter nomes descritivos. O nome "v", por exemplo, é muito vago e não nos dá ideia do que uma função faz. Já verificarChute é bem descritivo e qualquer pessoa entenderá do que se trata.

**Mônica:** A regra que seguimos para declarar variáveis, também funciona para funções. Funções com nomes pouco descritivos são exemplos de má prática e devem ser evitadas!

**Guilherme:** A função é responsável por executar alguma ação dentro do nosso programa. Por padrão e por convenção, é ideal determinar que as funções tenham apenas uma responsabilidade. Logo, a responsabilidade do botão que verifica o chute, como o nome sugere, é verificar o chute.

No escopo da função, vamos incluir um console.log() com a mensagem "O botão foi clicado!":

**function** **verificarChute**() {

console.**log**('O botão foi clicado!')

}

COPIAR CÓDIGO

Vamos salvar e voltar para a aplicação.

Cada vez que clicamos no botão "Chutar", a função é chamada e a frase "O botão foi clicado!" é impressa no console. Isso significa que o botão está funcionando corretamente!

Em seguida, vamos nos aprofundar na questão de funções e começar a trabalhar em nosso jogo.

**Manipulando conteúdo HTML com JavaScript**

Você está lidando com a interação entre o HTML e o JavaScript para tornar a interface de usuário mais dinâmica e interativa. No entanto, você gostaria de mudar o título do seu aplicativo de forma dinâmica com o JavaScript.

Qual código JavaScript você usaria para alterar o texto exibido no cabeçalho da sua página (elemento h1) para Bem-vindo ao mundo dev?

let titulo = document.querySelector("h1");

titulo.innerHTML = "Bem-vindo ao mundo dev";

**Você acertou em cheio!**

Adriano, sua resposta está correta, parabéns! O código que você forneceu é exatamente o que é necessário para alterar o texto exibido no cabeçalho da página (elemento h1) para "Bem-vindo ao mundo dev". Vamos entender passo a passo: 1. `let titulo = document.querySelector('h1')`: Essa linha seleciona o elemento HTML com a tag `>` e armazena uma referência a ele na variável `titulo`. 2. `titulo.innerHTML = 'Bem-vindo ao mundo dev'`: Essa linha altera o conteúdo interno (innerHTML) do elemento `<h1>` selecionado, substituindo o texto atual pelo novo texto "Bem-vindo ao mundo dev". Dessa forma, você consegue manipular dinamicamente o conteúdo do cabeçalho da sua página usando JavaScript, tornando a interface mais interativa e personalizada para os usuários. Parabéns, Adriano! Você demonstrou um ótimo entendimento da interação entre HTML e JavaScript. Continue assim, praticando e aprimorando seus conhecimentos. Você está no caminho certo para se tornar um desenvolvedor web ainda mais habilidoso!

**Para saber mais: diferenças entre html, css e Js**

HTML, CSS e JavaScript (JS) são três tecnologias fundamentais para o desenvolvimento web.

O **HTML** (HyperText Markup Language) é a **linguagem de marcação** utilizada para estruturar o conteúdo de uma página web. Essa linguagem usa tags para definir a hierarquia dos elementos de uma página. Originado em 1991 por Tim Berners-Lee, o HTML é como o esqueleto de uma página web: trabalha organizando o conteúdo, como títulos, parágrafos, imagens e links, para que tudo fique bem arrumado e no lugar certo.

O **CSS** (Cascading Style Sheets), criado em 1995, é destacado como uma **linguagem de estilos**. Ele separa a apresentação da estrutura: é responsável pela apresentação e **estilização** dos elementos, pois permite controlar cores, fontes, layouts e outros aspectos visuais. É como vestir a estrutura HTML com roupas elegantes para que a página pareça exatamente como você deseja.

O **JavaScript**, por sua vez, é a única linguagem de programação das três. É responsável por adicionar **interatividade e dinamismo às páginas**. Possibilita a criação de funcionalidades como animações, validações de formulários e atualizações de conteúdo em tempo real.

O papel de cada tecnologia dessas na construção de aplicações web, de forma bem resumida, é: o HTML estrutura o conteúdo, o CSS define o estilo e o layout, e o JavaScript adiciona funcionalidades dinâmicas.

| **Tecnologia** | **Função** | **Exemplos de Uso** |
| --- | --- | --- |
| HTML | Linguagem de marcação para estruturar conteúdo. | **Título**  Parágrafo |
| CSS | Linguagem de estilos para apresentação e estilização. | p { color: blue; } div { background: #f2f2f2; } |
| JavaScript (JS) | Linguagem de programação para interatividade. | function iniciarJogo() { ... } |

A aprendizagem dessas tecnologias é crucial para qualquer pessoa interessada em desenvolvimento web, uma vez que elas formam a base do ambiente digital. Compreender o HTML é necessário para estruturar e organizar o conteúdo de uma página de maneira semântica, o que é vital para a acessibilidade, indexação por motores de busca e uma experiência consistente entre diferentes dispositivos.

O CSS permite que pessoas desenvolvedoras controlem o design e a aparência de seus sites, possibilitando a criação de layouts atraentes e responsivos, que se adaptam a diferentes tamanhos de tela.

Por fim, o JavaScript capacita os desenvolvedores e as desenvolvedoras a criar experiências interativas e dinâmicas, e melhorar a usabilidade ao oferecer às pessoas usuárias funcionalidades mais avançadas.

A proficiência em HTML, CSS e JavaScript permite materializar ideias de maneira criativa e funcional, o que proporciona aos usuários finais uma experiência envolvente e intuitiva.

Além disso, essas habilidades são altamente valorizadas no mercado de trabalho, visto que a demanda por pessoas desenvolvedoras web qualificadas é constante, dada a importância contínua da presença online para empresas e indivíduos. Portanto, aprender e aprimorar essas tecnologias é um investimento significativo para qualquer pessoa que deseja se destacar na área de desenvolvimento web.

Para saber mais ainda sobre as características dessas três tecnologias a [Rafa Ballerini escreveu um artigo enfatizando essas diferenças](https://www.alura.com.br/artigos/html-css-e-js-definicoes).

**Desafio: hora da prática**

Em uma carreira de desenvolvimento de software, a prática consistente da lógica de programação desempenha um papel fundamental na construção de bases sólidas.

A lógica de programação não apenas permite a criação de algoritmos eficientes e soluções elegantes, mas também desenvolve a capacidade de pensar de forma estruturada e analítica.Essa habilidade é essencial para enfrentar desafios complexos e transformar problemas abstratos em implementações tangíveis.

Pensando nisso, criamos uma lista de atividades (não obrigatórias) focada em prática para melhorar ainda mais sua experiência de aprendizagem. Bora praticar então?

**Desafios**

1. Faça o [download de outro projeto clicando neste link](https://github.com/alura-cursos/js-curso-2/tree/desafio_1) e abra no Visual Studio Code.
2. Altere o conteúdo da tag h1 com document.querySelector e atribua o seguinte texto: Hora do Desafio.
3. Crie uma função que exiba no console a mensagem O botão foi clicado sempre que o botão Console for pressionado.
4. Crie uma função que exiba um alerta com a mensagem: Eu amo JS, sempre que o botão Alerta for pressionado.
5. Crie uma função que é executada quando o botão prompt é clicado, perguntando o nome de uma cidade do Brasil. Em seguida, exiba um alerta com a mensagem concatenando a resposta com o texto: Estive em {cidade} e lembrei de você.
6. Ao clicar no botão soma, peça 2 números inteiros e exiba o resultado da soma em um alerta.

Caso precise de ajuda, opções de solução das atividades estarão disponíveis na seção “Opinião da pessoa instrutora”.

**Resolução dos desafios 1**

Em uma carreira de desenvolvimento de software, a prática consistente da lógica de programação desempenha um papel fundamental na construção de bases sólidas. A lógica de programação não apenas permite a criação de algoritmos eficientes e soluções elegantes, mas também desenvolve a capacidade de pensar de forma estruturada e analítica. Essa habilidade é essencial para enfrentar desafios complexos e transformar problemas abstratos em implementações tangíveis.

Pensando nisso, criamos uma lista de atividades (não obrigatórias) focada em prática para melhorar ainda mais sua experiência de aprendizagem.

**Sugestões de respostas**

1. Altere o conteúdo da tag h1 com document.querySelector e atribua o seguinte texto: Hora do Desafio:

let titulo = document.querySelector('h1');

titulo.innerHTML = 'Hora do Desafio';

1. Crie uma função que exiba no console a mensagem O botão foi clicado sempre que o botão Console for pressionado:

No index.html, adicionamos no onclick o seguinte código:

<button onclick="exibirMensagemNoConsole()" class="button">Console</button>

* Já no app.js:

function exibirMensagemNoConsole() {

console.log('O botão foi clicado!')

}

1. Crie uma função que exiba um alerta com a mensagem: Eu amo Js, sempre que o botão Alerta for pressionado:

No index.html, adicionamos no onclick o seguinte código:

<button onclick="exibirAlerta()" class="button">Alert</button>

* Já no app.js:

function exibirAlerta() {

alert('Eu amo Js')

}

1. Crie uma função que é executada quando o botão prompt é clicado, perguntando o nome de uma cidade do Brasil. Em seguida, exiba um alerta com a mensagem concatenando a resposta com o texto: Estive em {cidade} e lembrei de você:

No index.html, adicionamos no onclick o seguinte código:

<button onclick="exibirPrompt()" class="button">Prompt</button>

* Já no app.js:

function exibirPrompt() {

let nomeDaCidade = prompt('Digite o nome de uma cidade do Brasil que você gosta muito:')

alert(`Estive em ${nomeDaCidade} e lembrei de você`)

}

1. Ao clicar no botão soma, peça 2 números e exiba o resultado da soma em um alerta.

No index.html, adicionamos no onclick o seguinte código:

<button onclick="somandoDoisNumeros()" class="button">Soma</button>

* Já no app.js:

function somandoDoisNumeros() {

let primeiroNumero = parseInt(prompt('Digite o primeiro número'));

let segundoNumero = parseInt(prompt('Digite o segundo número'));

let resultado = primeiroNumero + segundoNumero;

alert(`${primeiroNumero} + ${segundoNumero} = ${resultado}`)

}

# 08O que aprendemos?

**Aqui está um resumo dos tópicos ensinados nesta aula:**

* Carregamos o projeto com o HTML e o CSS para criar a estrutura e o estilo da página que utilizaremos durante o curso;
* Aprendemos a manipular o conteúdo dos elementos HTML, como o H1 e o parágrafo (p), utilizando JavaScript para alterar dinamicamente o texto exibido na página;
* Entendemos a importância da interação entre o HTML e o JavaScript para criar páginas web dinâmicas e interativas.

**Na próxima aula:**

Vamos aprender o que são funções de forma prática e qual a sua importância na carreira de desenvolvimento de software!

# 02Funções com parâmetros

**Guilherme**: Vamos aprofundar ainda mais em relação a funções.

**Funções com parâmetros**

**Guilherme**: Moni, se você observar o nosso código, especificamente nas linhas 1 e 2 e nas linhas 4 e 5, você pode pensar que ambas são muito diferentes e não têm qualquer semelhança.

app.js:

**let** titulo = document.**querySelector**('h1');

titulo.innerHTML = 'Jogo do número secreto';

**let** paragrafo = document.**querySelector**('p');

paragrafo.innerHTML = 'Escolha um número entre 1 e 10';

COPIAR CÓDIGO

Em uma, pegamos o conteúdo do h1 e, na outra, pegamos o conteúdo do parágrafo, criando variáveis e assim por diante.

**Mônica**: No entanto, elas estão bem semelhantes. Apesar de pegar elementos diferentes, elas fazem a mesma ação. Elas pegam o elemento, armazenam em uma variável e também alteram seu texto.

**Guilherme**: Vamos analisar isso detalhadamente? Primeiro, temos let titulo, let paragrafo. Qual é a diferença entre eles?

No caso de let titulo, criamos uma variável para guardar a seleção do h1. Enquanto em let paragrafo, criamos uma variável para guardar o elemento do parágrafo.

Basicamente, estamos fazendo algo do tipo, let campo, let tag ou let elemento, seguindo essa ideia. Um desses campos é o título, o outro é o parágrafo.

Após isso, colocamos um igual à document.querySelector(). Dentro os parênteses, tem algo diferente. No titulo, buscamos pelo h1, no outro, buscamos por uma *tag* p.

Portanto, a única diferença aqui é a *tag*.

Na linha seguinte, pegamos a valor desse campo, que um é título e o outro é parágrafo, e fazemos campo.innerHTML para atribuir outro valor.

A diferença está no texto de cada um. O texto do título é "jogo do número secreto" e o texto do parágrafo é "escolha um número entre 1 e 10".

Exemplo

**let** campo - document.**querySelector**(tag)

campo.innerHTML = texto

COPIAR CÓDIGO

Você pode perguntar: "Guilherme, isso não vai funcionar. O que você está fazendo?".

A explicação é a seguinte:

Sempre que temos **padrão** de código ou um código muito parecido, onde apenas alguns detalhes são modificados, podemos **isolar esse comportamento em uma função**.

Em vez de escrever quatro linhas de código, escrevemos menos linhas.

**Mônica**: Sim, é uma boa prática **evitar a repetição de código** com funções!

**Guilherme**: Nesse caso, utilizamos dois seletores. O seletor do h1 e o seletor do p.

Mas imagine se no nosso projeto tivéssemos que lidar com 50 seletores, como h1, h2, parágrafo e outros diversos seletores? Seria bastante difícil e trabalhoso escrever o mesmo código 50 vezes.

**Mônica**: Agora, você pode estar se perguntando: "Por que primeiro você mostra o modo errado e depois o modo correto?". Bom, não é errado, estamos apenas aprimorando uma boa prática.

**Guilherme**: Na verdade, o código que temos está em evolução. Isso é comum. Lembre-se, a maneira atual não é a maneira errada, pois o código ainda funciona.

No contexto em que estamos inseridos, a estrutura atual funciona. A questão é: "Como podemos tornar esse código independente do campo, onde passamos uma *tag* e passamos um texto?". Poderemos fazer isso usando **funções**.

Por isso, vamos criar uma nova função a partir da linha 7, usando o termo function. Esta função terá um nome, por exemplo, exibirTextoNaTela().

**Mônica**: Para seguir o padrão de boas práticas, a primeira letra de cada palavra deve ser maiúscula, exceto a primeira.

**Guilherme**: Abrimos e fechamos chaves e recortamos a lógica do campo dentro do bloco da função.

**Mônica**: Também, vamos adicionar o ponto e vírgula no final dessas duas linhas.

**Guilherme**: Sim, Moni! Essa é a razão pela qual o *Pair Programming* (programação em dupla) é tão útil.

**Mônica**: Pronto, na linha 14 após o console.log(), falta um ponto e vírgula também. Portanto, vamos conferindo e fazendo as alterações necessárias à medida que avançamos.

**function** **exibirTextoNaTela**() {

**let** campo = document.**querySelector**(tag);

campo.innerHTML = texto;

}

**function** **verificarChute**() {

console.**log**('O botão foi clicado!');

}

COPIAR CÓDIGO

**Guilherme**: Agora vem a parte desafiadora. Precisamos fazer com que o código funcione usando uma função.

O que é diferente é a tag, o h1 e o p. Por isso, na hora de invocar essa função, podemos especificar que o JavaScript deve usar a função exibirTextoNaTela() especificamente para o h1. Posteriormente, o JavaScript deverá usar a função exibirTextoNaTela() especificamente para o parágrafo.

Vamos comentar a linha do let titulo ao paragro.innerHTML, adicionando // no começo. Assim, poderemos usar a função que criamos.

// let titulo = document.querySelector('h1');

// titulo.innerHTML = 'Jogo do número secreto';

// let paragrafo = document.querySelector('p');

// paragrafo.innerHTML = 'Escolha um número entre 1 e 10';

COPIAR CÓDIGO

Na linha abaixo da função, vamos digitar exibirTextoNaTela(). Assim, primeiro criamos a função e, agora, estamos de fato executando esta função.

Queremos exibir o texto na tela. Porém, o texto na tela refere-se ao h1. Para conseguir capturar o h1 e passar dentro da tag no querySelector(), vamos passar a tag como parâmetro da função. Ou seja, escrevemos tag entre os parênteses de function exibirTextoNaTela().

Com isso, na chamada da função exibirTextoNaTela(), passados o h1 entre aspas. Pois, queremos que esse valor de tag seja substituído pela string h1. Dessa forma, todos os lugares onde existir tag serão substituídos por h1.

Mas, queremos fazer o mesmo para o parágrafo também. Em uma nova linha, basta escrever exibirTextoNaTela() para o parágrafo, ou seja, informando o p entre aspas.

**function** **exibirTextoNaTela**(tag) {

**let** campo = document.**querySelector**(tag);

campo.innerHTML = texto;

}

**exibirTextoNaTela**('h1');

**exibirTextoNaTela**('p');

COPIAR CÓDIGO

**Mônica**: Através desses parênteses após o nome da função, conseguimos nos comunicar com aquela função sobre o que queremos que aconteça lá dentro.

**Guilherme**: Perceba que na primeira vez que chamamos a função exibirTextoNaTela(), passando o h1, todo lugar que tiver tag será substituído pelo h1.

Na segunda vez que chamamos essa função exibirTextoNaTela(), passando p, todo lugar que tiver tag será substituído pelo valor p.

**Mônica**: Sim, mas também temos que informar o texto para o campo.innerHTML. O que é texto? Também é algo que precisaremos comunicar para essa função. Então, dentro dos parênteses da função na linha 7, vamos adicionar uma vírgula para informar outro parâmetro que essa função vai receber e adicionamos o texto.

Agora, onde chamamos o exibirTextoNaTela(), vamos informar o texto que queremos. Após enviar o primeiro parâmetro, adicionamos uma vírgula, e enviamos o segundo parâmetro.

Na primeira chamada da função, o segundo parâmetro será Jogo do número secreto entre aspas. Na segunda chamada, será o texto Escolha um número entre 1 e 10 também entre aspas.

**function** **exibirTextoNaTela**(tag, texto) {

**let** campo = document.**querySelector**(tag);

campo.innerHTML = texto;

}

**exibirTextoNaTela**('h1', 'Jogo do número secreto');

**exibirTextoNaTela**('p', 'Escolha um número entre 1 e 10');

COPIAR CÓDIGO

No curso anterior, mencionamos muito que o JavaScript lê linha a linha. Por isso, estamos criando a função e chamando-a depois.

Mas, se colocássemos a função depois, também funcionaria. O JavaScript pega tudo que está declarado como função, guarda em algum lugar para ser lido somente quando invocamos (chamamos) essa função.

**Guilherme**: Lembrando que isso é algo específico do JavaScript, tanto que não estamos fazendo desta forma. Preferimos declarar a função e, abaixo, usar essa função.

Podemos apagar da linha 1 a 5, onde tínhamos os trechos comentados de código. Dessa forma, evoluímos o nosso código.

Qual é o desafio agora? Precisamos testar para verificar se as alterações funcionam. Vamos voltar no navegador. Ainda temos os textos: "Jogo do nome secreto" e "Escolha um número entre 1 e 10". Excelente!.

Ao clicar em "Chutar", aparece a mensagem no console:

O botão foi clicado.

**Mônica**: Portanto, aprendemos dois tipos de funções: **com parâmetros** e **sem parâmetros**. No entanto, essas não são as únicas operações que podemos realizar com funções.

# 03Funções com retorno

**Mônica**: Já criamos dois tipos de funções: uma para exibir textos na tela e outra para verificar a interação com o botão.

Agora, precisamos gerar um número aleatório. Como isso é uma nova funcionalidade, podemos implementá-la utilizando funções.

**Funções com retorno**

**Guilherme**: Vamos criar uma nova função na linha 13 do nosso código, após verificarChute().

Repare que quando começamos a escrever function, o Visual Studio Code já sugere a criação de um *function statement*. Quando confirmamos, pressionando "Enter", a estrutura básica de uma função é criada automaticamente.

app.js:

**function** **name**(params) {

}

COPIAR CÓDIGO

Aproveite as sugestões da IDE!

No lugar de name, podemos nomear a função da forma que achamos mais adequado. Vamos chamá-la de gerarNumeroAleatorio.

**Mônica**: Há uma sugestão para que um parâmetro seja adicionado à função. Mas, neste caso, não precisamos disso, pois desejamos que a máquina gere um número aleatório sem a necessidade de receber qualquer informação prévia.

Por isso, podemos apagar params que está entre os parênteses, deixando-os vazios.

**Guilherme**: Para gerar o número aleatório, utilizaremos o código que conhecemos: Math.random(). Será uma função que multiplica por 10 e soma 1 para ter um número entre 1 e 10.

**Mônica**: A função Math.random() retorna um valor decimal, mas queremos um número inteiro. Por isso, colocamos a expressão matemática entre parênteses e utilizamos parseInt() para converter o resultado em um número inteiro.

**function** **gerarNumeroAleatorio**() {

parseInt(**Math**.**random**() \* 10 + 1);

}

COPIAR CÓDIGO

**Guilherme**: Dessa forma, isolamos esse processo. Em outras palavras, existe uma parte da nossa aplicação que é capaz de gerar esse número aleatório.

Imagine se tivéssemos um código com 500 linhas. Encontrar onde é gerado um número aleatório poderia ser bem complicado. Ao utilizar a função, a aplicação fica mais fácil de ser compreendida.

Não nos aprofundamos em Math.random(), porque já aprendemos essa função anteriormente. Dúvidas sobre essa função? Revisite o curso anterior!

**Mônica**: Agora, precisamos guardar o valor do número aleatório em algum lugar para posteriormente fazer sua verificação.

**Guilherme**: Por isso, vamos criar na linha 1 a variável let numeroSecreto que vai ser igual à função gerarNumeroAleatorio().

**let** numeroSecreto = **gerarNumeroAleatorio**();

COPIAR CÓDIGO

Mas, temos um problema. Diferente das outras funções, como exibirTextoNaTela() e verificarChute(), esperamos que a função gerarNumeroAleatorio() nos **retorne uma informação**.

**Mônica**: Atualmente, o que acontece em nosso código? Essa função é chamada na linha 1, ela é executada e gera um número aleatório. Mas para onde vai essa informação? Ela precisa ser armazenada na memória.

**Guilherme**: Ao gerar o número aleatório, queremos que a função nos retorne esse número que terá valor entre 1 e 10. Para garantir esse comportamento, precisamos informar que queremos este retorno utilizando a palavra reservada return na linha 16, antes de toda a expressão matemática.

**function** **gerarNumeroAleatorio**() {

**return** parseInt(**Math**.**random**() \* 10 + 1);

}

COPIAR CÓDIGO

**Mônica**: Com isso, o valor gerado será atribuído à variável numeroSecreto através do retorno da função.

**Guilherme**: Nesse ponto, podemos perceber que existem diferentes modelos de funções. Por exemplo, temos uma função que não possui parâmetros e nem retorno, que é a função verificarChute().

Temos a função exibirTextoNaTela(). Ela realiza a tarefa de exibir um texto na tela, mas não esperamos que ela nos devolva uma informação. Por isso, ela possui parâmetros que são as informações que queremos exibir, mas não possui um retorno.

Na função de gerarNumeroAleatorio(), não definimos nenhum parâmetro. Por isso, não tem nenhuma informação entre os parênteses. Contudo, há um retorno. Qual é o retorno? A geração de um número entre 1 e 10.

E o que fazemos quando essa função é chamada, ou seja, quando a utilizamos na linha let numeroSecreto?

Note que não estamos utilizando a função quando a declaramos como function. Nesse momento, apenas estamos declarando que a função existe. Para efetivamente usar a função, precisamos invocá-la como gerarNumeroAleatorio().

Com isso, pegamos o resultado retornado pela função e o armazenamos na variável chamada numeroSecreto.

Agora, vamos testar a geração desse número secreto para certificar que a função funciona corretamente.

**Mônica**: Precisamos imprimir esse número em algum lugar.

**Guilherme**: Podemos imprimi-lo no console. Podemos substituir o texto do console.log() da função verificarChute() pela variável numeroSecreto.

**function** **verificarChute**() {

console.**log**(numeroSecreto);

}

COPIAR CÓDIGO

Após salvar o arquivo, voltamos ao projeto no navegador e atualizamos a página. Desse modo, o número secreto só será exibido quando clicarmos no botão "Chutar".

Vamos limpar o console com "CTRL + L" e clicar em "Chutar". Foi gerado o 8.

8

Clicamos de novo, foi gerado novamente o número 8.

**Mônica**: O número 8 não está sendo gerado repetidamente, o que acontece é que chamamos a função de geração de número aleatório apenas uma vez. Essa primeira vez gerou o número 8 aleatoriamente. Nas vezes seguintes, estamos apenas imprimindo o mesmo número.

**Guilherme**: Se atualizamos a página e limpamos o console, podemos clicar em "Chutar" de novo. Ai, sim, teremos outro número gerado.

5

**Conclusão**

**Guilherme**: Já criamos três tipos de funções: funções com retorno, funções com parâmetros e funções sem retorno e sem parâmetros.

**Mônica**: Além disso, exploramos algo novo. Antes, havíamos criado variáveis que recebiam valores do tipo *string* ou número. Agora, estamos atribuindo funções às variáveis.

# 04Tipo booleano

**Guilherme**: Agora é a hora da verdade. No HTML do nosso programa, temos uma propriedade chamada input após o parágrafo de "Escolha um número entre 1 e 10". O que ela faz?

index.html:

<input type="number" min="1" max="10" class="container\_\_input">

COPIAR CÓDIGO

Se observamos o jogo no navegador, esse campo *input* é onde a pessoa jogadora irá inserir um número entre 1 e 10.

Nesse campo, podemos digitar o valor 5, por exemplo. Para chutar esse valor, clicamos no botão "Chutar". Mas, o número aleatório gerado foi 6, erramos.

**Mônica**: No momento, só imprimimos o número aleatório, não o nosso chute.

**Guilherme**: Então, o que queremos fazer? Ao clicar em "Chutar", queremos pegar o valor especificado no *input* que é a entrada da pessoa usuária, e compará-lo com o número aleatório. Inicialmente, vamos mostrar no console para conferir se acertamos ou erramos o chute.

**Tipo booleano**

**Mônica**: Como queremos um elemento que está dentro do HTML, usaremos um modo muito semelhante ao que fizemos para pegar o parágrafo e o título.

**Guilherme**: Porém, temos uma diferença. Diferentemente do h1 e p, o que queremos é obter a informação inserida, não exibi-la. Portanto, não usaremos a função de exibirTextoNaTela().

Na função verificarChute(), vamos criar um campo novo. Por exemplo, let chute vai ser igual à document.querySelector(). Entre os parênteses, vamos escrever o input entre aspas.

Como sabemos que é a campo input? Porque verificamos a linha 25 do código HTML, onde escrito *input* do tipo número de 1 a 10. Esse é o campo que queremos.

Pegamos essa informação, porém queremos apenas o valor inserido no campo. Não queremos o HTML completo. Por exemplo, queremos apenas o valor 5 que foi digitado.

Quando temos um *input* e queremos apenas o valor inserido, utilizamos .value após a seleção que fizemos. Precisa ser escrito desta maneira. Lembre-se de colocar o ponto e vírgula ao final.

app.js:

**function** **verificarChute**() {

**let** chute = document.**querySelector**('input').value;

console.**log**(numeroSecreto);

}

COPIAR CÓDIGO

**Mônica**: E por qual motivo fazemos isso? Quando usamos o h1 e p, guardamos o texto direto dentro da variável. Mas o *input* não é um texto, é um valor inserido por alguém, portanto, usamos .value.

**Guilherme**: Em seguida, vamos fazer a comparação dentro do console para verificar que tipo de resposta vai ser gerada.

No console.log(), vamos escrever chute == numeroSecreto.

**function** **verificarChute**() {

**let** chute = document.**querySelector**('input').value;

console.**log**(chute == numeroSecreto);

}

COPIAR CÓDIGO

Por que utilizamos == e não =? Sempre que usamos =, queremos atribuir um valor. Quando usamos ==, queremos comparar um valor.

No JavaScript, existem outras formas de comparação. Mas, esses dois já são suficientes para nós por enquanto.

O que faremos no navegador? Iremos clicar no botão de "Novo jogo" para dar um *refresh* na tela. Em seguida, vamos inserir um valor e clicar no botão "Chutar". O que esse botão fará? Vai verificar o nosso chute. Ou seja, vai pegar o valor que inserimos e mostrará no console se o chute é igual ao número secreto.

Antes de tudo, vamos limpar o nosso console mais uma vez com "Ctrl + L". Agora, insira um número 2 no campo de *input* e apertamos "Chutar".

false

Apareceu algo diferente. O que é um false no JavaScript? Vamos pesquisar?

No buscador, vamos colocar false JavaScript e entrar na [documentação do JavaScript sobre booleanos](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Boolean). Olha que interessante.

O tipo **booleano** é um nome muito comum para outras linguagens de programação também. É um valor que pode ser **verdadeiro** ou **falso**.

Quando desenvolvemos o jogo pela primeira vez, o que ele esperava quando fazíamos as comparações? Se é verdadeiro ou se é falso.

Sempre que encontramos o nome booleano na programação, seja no JavaScript ou outras linguagens, esperamos que esse valor será verdadeiro ou falso.

Pode haver um valor ou outro diferente. Talvez o JavaScript trabalhe um pouco diferente, mas o que importante para nós é entender se esse valor é verdadeiro ou falso.

**Mônica**: Percebam como o conhecimento vai aumentando, porque já conhecemos vários tipos. Aprendemos sobre o tipo *string* (texto), o tipo *number* (número), e agora o tipo *boolean* (verdadeiro ou falso).

**Guilherme**: Podemos até manipular o jogo para dar verdadeiro. Vamos remover temporariamente a chamada da função gerarNumeroAleatorio() que guardamos na variável numeroSecreto. No lugar, vamos colocar um número qualquer, como o 5.

Exemplo

**let** numeroSecreto = 5;

COPIAR CÓDIGO

No navegador, vamos digitar o número 5 no campo de *input* para verificar qual tipo de retorno que será exibido no console.

true

É retornado o valor true que, em inglês, significa verdadeiro. E false significa falso.

Vamos desfazer a última alteração no código para ser gerado, de fato, um número aleatório.

**Conclusão**

Em resumo, sempre que fazemos uma comparação, esperamos um valor verdadeiro ou falso - mais conhecido como tipo booleano.

Na sequência, o que vamos fazer? Podemos usar uma alternativa ao console, para exibir na tela uma resposta que faça mais sentido.

# 05Anatomia de uma função

A anatomia de uma função em JavaScript é composta por diversas partes que definem seu comportamento. É importante compreender esses elementos para criar funções reutilizáveis e modularizar o código de forma eficiente.

Assim, conhecer a estrutura das funções em JavaScript é fundamental para dominar a linguagem e desenvolver aplicações complexas de forma organizada e funcional.

Com base no que vimos nesta aula, analise as afirmações abaixo e marque apenas as opções verdadeiras sobre funções.

*Selecione 3 alternativas*

As funções em JavaScript podem ter um nome.

As funções em JavaScript podem ter um nome.

Na verdade, é uma prática comum dar um nome descritivo às funções para facilitar a leitura e o entendimento do código. Esses nomes são usados para chamar a função posteriormente ou para se referir a ela dentro do próprio código, permitindo que você reutilize a função em diferentes partes do programa.

Palavra-chave function: É usada para declarar uma função em JavaScript.

Isso! Toda função em JavaScript é declarada utilizando a palavra-chave function. Essa palavra-chave é utilizada para criar uma função.

Parênteses (): Após o nome da função (se houver), os parênteses são usados para definir os parâmetros da função.

Os parênteses em uma função no JavaScript são usados para conter os argumentos que a função recebe, permitindo a passagem de informações para a função e influenciando seu comportamento e resultados.

# 06Função que verifica números

Você é uma pessoa desenvolvedora de software que trabalha com JavaScript e precisa criar uma função que verifique se um número é positivo, negativo ou zero. Sua tarefa é implementar uma função que recebe como parâmetro um número inteiro e exibe uma mensagem no console conforme as seguintes regras:

* Se o número for maior que zero, a mensagem deve ser: "O número é positivo."
* Se o número for menor que zero, a mensagem deve ser: "O número é negativo."
* Se o número for igual a zero, a mensagem deve ser: "O número é zero."

Você pode utilizar a estrutura de controle if-else para verificar as condições e exibir a mensagem correspondente no console. Sabendo disso, analise as alternativas abaixo e marque aquela que possui o código que atende todas as regras citadas acima.

*Selecione uma alternativa*

**function** **verificarNumero**(numero) {

**if** (numero > 0) {

console.**log**("O número é positivo.");

} **else** **if** (numero < 0) {

console.**log**("O número é negativo.");

} **else** {

console.**log**("O número é zero.");

}

}

Neste código, usamos a estrutura de controle if-else para verificar as três possibilidades: se o número é maior que zero, se é menor que zero ou se é igual a zero. Dependendo da condição, a função imprime a mensagem correspondente no console.

# 07Para saber mais: funções com JavaScript

As funções desempenham um papel central na programação e no desenvolvimento de software, permitindo encapsular blocos de código reutilizáveis e executáveis. Elas podem ser definidas para executar tarefas específicas, desde cálculos simples até manipulação complexa de dados. As funções em JavaScript podem ter ou não parâmetros, bem como retornar ou não valores.

| **Tipo de Função** | **Exemplo de Código** | **Uso** |
| --- | --- | --- |
| Sem retorno e sem parâmetro | function saudacao() { ... } | Execução de bloco de código simples. |
| Sem retorno e com parâmetro | function cumprimentar(nome) { ... } | Execução de bloco de código com argumentos. |
| Com retorno e sem parâmetro | function gerarNumeroAleatorio() { ... } | Cálculo e retorno de um valor específico. |
| Com retorno e com parâmetro | function somar(a, b) { ... } | Cálculo e retorno baseado em argumentos. |
| Função anônima | let saudacao = function() { ... }; | Definição de função sem nome localmente. |
| Arrow function | let quadrado = x => x \* x; | Definição concisa de funções curtas. |

Seu uso é essencial para criar aplicações dinâmicas, interativas e eficientes em JavaScript. Porém, **você não precisa criar todos os tipos de funções neste momento**. Acompanhe as aulas e saiba que essas variações existem e conforme a necessidade, vamos mergulhando cada vez mais profundamente em lógica e JavaScript.

# 08Desafio: hora da prática

Saber como usar funções é um dos conceitos fundamentais da programação e do desenvolvimento de software. As funções desempenham um papel crucial na organização, modularização e reutilização de código.

Pensando nisso, criamos uma lista de atividades (não obrigatórias) focada em prática para melhorar ainda mais sua experiência de aprendizagem. Bora praticar então?

**Desafios**

1. Criar uma função que exibe "Olá, mundo!" no console.
2. Criar uma função que recebe um nome como parâmetro e exibe "Olá, [nome]!" no console.
3. Criar uma função que recebe um número como parâmetro e retorna o dobro desse número.
4. Criar uma função que recebe três números como parâmetros e retorna a média deles.
5. Criar uma função que recebe dois números como parâmetros e retorna o maior deles.
6. Criar uma função que recebe um número como parâmetro e retorna o resultado da multiplicação desse número por ele mesmo

Caso precise de ajuda, opções de solução das atividades estarão disponíveis na seção “Opinião da pessoa instrutora”.

**Resolução dos desafios 2**

Em uma carreira de desenvolvimento de software, a prática consistente da lógica de programação desempenha um papel fundamental na construção de bases sólidas. A lógica de programação não apenas permite a criação de algoritmos eficientes e soluções elegantes, mas também desenvolve a capacidade de pensar de forma estruturada e analítica. Essa habilidade é essencial para enfrentar desafios complexos e transformar problemas abstratos em implementações tangíveis.

Pensando nisso, criamos uma lista de atividades (não obrigatórias) focada em prática para melhorar ainda mais sua experiência de aprendizagem.

**Sugestões de respostas**

1. Criar uma função que exibe "Olá, mundo!" no console.

function exibirOla() {

console.log("Olá, mundo!");

}

exibirOla();

1. Criar uma função que recebe um nome como parâmetro e exibe "Olá, [nome]!" no console.

function exibirOlaNome(nome) {

console.log(`Olá, ${nome}!`);

}

exibirOlaNome("Alice");

1. Criar uma função que recebe um número como parâmetro e retorna o dobro desse número.

function calcularDobro(numero) {

return numero \* 2;

}

let resultadoDobro = calcularDobro(5);

console.log(resultadoDobro);

1. Criar uma função que recebe três números como parâmetros e retorna a média deles.

function calcularMedia(a, b, c) {

return (a + b + c) / 3;

}

let media = calcularMedia(4, 7, 10);

console.log(media);

1. Criar uma função que recebe dois números como parâmetros e retorna o maior deles.

function encontrarMaior(a, b) {

return a > b ? a : b;

}

let maiorNumero = encontrarMaior(15, 9);

console.log(maiorNumero);

1. Criar uma função que recebe um número como parâmetro e retorna o resultado da multiplicação desse número por ele mesmo.

function quadrado(numero) {

return numero \* numero;

}

let resultado = quadrado(2);

console.log(resultado);

# 09O que aprendemos?

Explique com suas próprias palavras os principais conceitos que você aprendeu nesta aula.

Nesta aula aprendemos para que serve as funções na linguagem de programação, funções com parâmetros que são valores que podem passar para uma função quando chamada.

Função com retorno são aquelas que devolvem um valor após serem executados.

Os tipos booleanos que representa um valor lógico verdadeiro ou falso.

Adriano, parabéns pela dedicação aos estudos! Você mencionou que aprendeu sobre a importância das funções na linguagem de programação e como elas podem receber parâmetros. Além disso, você destacou a diferença entre funções com retorno, que devolvem um valor após serem executadas, e funções sem retorno. Além disso, vimos que criamos uma função em JavaScript que recebe 2 parâmetros para gerar um número aleatório. Utilizamos a palavra-chave "return" dentro da função para retornar o valor do número aleatório gerado. Também criamos outra função chamada `exibirMensagemInicialNaTela`, que não possui retorno nem parâmetros. Também vimos como os parâmetros e retornos são importantes para a comunicação entre funções e o restante do programa. Compreendemos a diferença entre funções com retorno e funções sem retorno e aprendemos como chamar e invocar a função para exibir uma mensagem inicial na tela. Continue estudando e praticando!

# 02Verificando o chute

**Guilherme:** Môni, acredito que deveríamos mostrar no console um valor *True* ou *False* para mostrar a criação do jogo para os amigos.

**Mônica:** É bom reforçar que o console é uma ferramenta para a pessoa desenvolvedora e não para a usuária.

**Guilherme:** Então, considerando que nós já discutimos a questão dos valores booleanos, vamos fazer um if() e else().

Em function verificarChute(), abaixo de let chute, escrevemos if(). Se o chute for igual ao número secreto, isso significa que acertamos. Então, nos parênteses passamos chute == numeroSecreto e adicionamos chaves.

Para passar a informação indicando que a pessoa acertou, podemos exibir o texto na tela no lugar do h1, passando outro valor, por exemplo, "acertou".

Para isso, nas chaves e na linha abaixo, escrevemos exibirTextoNaTela(). Nos parênteses passamos 'h1', 'Acertou!' e salvamos.

//Código omitido

**function** **verificarChute**() {

**let** chute = document.**querySelector**('input').value;

**if** (chute == numeroSecreto) {

**exibirTextoNaTela**('h1', 'Acertou!');

}

}

//Código omitido

COPIAR CÓDIGO

Será muito difícil acertar no chute, então, na primeira linha de código, em let numeroSecreto passamos o número 7.

**let** numeroSecreto = 7;

//Código omitido

COPIAR CÓDIGO

Para testar, acessamos o jogo, digitamos o número "7" e clicamos em "Chutar". Feito isso, o texto exibido muda de "Jogo do número secreto" para "Acertou!".

**Mônica:** Deu certo!

**Guilherme:** Mas, repare que a mensagem "Escolha um número entre 1 e 10" continua sendo exibida. Podemos definir uma mensagem melhor quando a pessoa acerta o número.

Para isso, na linha 16, escrevemos exibirTextoNaTela() e dentro dos parênteses passamos 'p', 'Você descobriu o número secreto!'.

//Código omitido

**function** **verificarChute**() {

**let** chute = document.**querySelector**('input').value;

**if** (chute == numeroSecreto) {

**exibirTextoNaTela**('h1', 'Acertou!');

**exibirTextoNaTela**('p', 'Você descobriu o número secreto!')

}

}

//Código omitido

COPIAR CÓDIGO

Para testar novamente, abrimos o jogo e passamos o número "7". Feito isso, visualizamos a mensagem "Acertou! Você descobriu o número secreto!". Ficou muito bom!

Nos casos de erro, precisamos informar se o número que foi chutado é maior ou menor que o número da sorte. Para isso, na linha 17 escrevemos else e adicionamos chaves. Dentro delas, passamos if(chute > numeroSecreto) seguido de chaves.

Dentro das chaves e na linha abaixo escrevemos exibirTextoNaTela(). Nos parênteses, passamos 'p', 'O número secreto é menor'.

Na linha seguinte escrevemos else {}. Nas chaves passamos 'p', 'O número secreto é maior'.

//Código omitido

**function** **verificarChute**() {

**let** chute = document.**querySelector**('input').value;

**if** (chute == numeroSecreto) {

**exibirTextoNaTela**('h1', 'Acertou!');

**exibirTextoNaTela**('p', 'Você descobriu o número secreto!');

} **else** {

**if** (chute > numeroSecreto) {

**exibirTextoNaTela**('p', 'O número secreto é menor');

} **else** {

**exibirTextoNaTela**('p', 'O número secreto é maior');

}

}

}

//Código omitido

COPIAR CÓDIGO

Feito isso, na primeira linha de código, apagamos o número 7 e escrevemos gerarNumeroAleatorio().

**let** numeroSecreto = **gerarNumeroAleatorio**();

//Código omitido

COPIAR CÓDIGO

Abrimos o jogo e testaremos novamente. Chute um número, Môni.

**Mônica:** Número 2.

**Guilherme:** Repare que a mensagem "O número secreto é maior" é exibida na tela.

Chutarei o número 8. A mensagem exibida é "O número secreto é menor".

**Mônica:** Número 4.

**Guilherme:** O número é maior. Vou tentar o número 6. Feito isso, a mensagem exibida é "Acertou! Você descobriu o número secreto!".

Deu certo! Para jogar de novo, atualizamos a página.

**Mônica:** Número 4.

**Guilherme:** O número é menor, 2 também.

**Mônica:** Então é o número 1.

**Guilherme:** Deu certo! Temos a ideia do jogo sem o *alert* e o *prompt*, entendendo como criar funções e usando retorno.

Repare que essa questão de termos uma função que exibe na tela, passando somente a tag, ficou muito interessante.

Mas, não concluímos. Ainda temos alguns desafios importantes pela frente.

**Até o próximo vídeo!**

# 03 Contando tentativas

**Mônica:** Seguindo a lógica do jogo que estávamos desenvolvendo com o prompt e o alert, falta exibir o número de tentativas.

**Guilherme:** Exatamente, Môni! É interessante sabermos que foram necessárias, por exemplo, 5 tentativas para descobrir o número secreto. Faremos isso agora!

**Implementando o número de tentativas**

Na linha 16, alteramos o conteúdo do texto da tag p para Você descobriu o número secreto com 5 tentativas!.

//Código omitido

**let** mensagemTentativas = `Você descobriu o número secreto com 5 tentativas!`;

COPIAR CÓDIGO

Para controlar o número de tentativas, na linha 2, criamos uma variável chamada tentativas e inicializamos com o valor 1. A lógica do nosso jogo é que, desde o início, caso o número seja acertado de primeira, já teremos ali uma tentativa.

//Código omitido

**let** tentativas = 1;

COPIAR CÓDIGO

Se passarmos uma *template string* diretamente para exibir o texto na tela, talvez o HTML não entenda. Isso porque ele espera uma string e não uma *template string*, que é algo que somente o JavaScript entende.

Sendo assim, na linha 17, criamos uma variável chamada let mensagemTentativas = Você descobriu o número secreto com 5 tentativas. Para indicar que essa string é uma *template string* usamos as crases no início e no final.

**Mônica:** Feito isso, vamos ao invés de deixar definido o número de tentativas, apagamos o 5 e passamos ${tentativas}.

//Código omitido

**if** (chute == numeroSecreto) {

**exibirTextoNaTela**('h1', 'Acertou!');

**let** mensagemTentativas = `Você descobriu o número secreto com ${tentativas} tentativas!`;

**exibirTextoNaTela**('p', 'Você descobriu o número secreto com 5 tentativas!');

} **else** {

**if** (chute > numeroSecreto) {

**exibirTextoNaTela**('p', 'O número secreto é menor');

} **else** {

**exibirTextoNaTela**('p', 'O número secreto é maior');

}

}

}

COPIAR CÓDIGO

**Guilherme:** No entanto, devemos nos lembrar que se o número for acertado de primeira, a frase será "Você descobriu o número secreto com 1 tentativas". Então, precisamos criar uma variável para tentativas.

Na linha 17 escrevemos let palavraTentativa igual à tentativas > 1 ? 'tentativas' : 'tentativa'. Em seguida, na linha 18, apagamos a palavra tentativas e passamos ${palavraTentativa}.

**Mônica:** Também precisamos trocar o parâmetro que enviava texto. Então, na linha 19 apagamos o texto e passamosmensagemTentativas.

//Código omitido

**if** (chute == numeroSecreto)

**exibirTextoNaTela**('h1', 'Acertou!');

**let** palavraTentativa = tentativas > 1 ? 'tentativas': 'tentativa';

**let** mensagemTentativas = `Você descobriu o número secreto com ${tentativas} ${palavraTentativa}!`;

**exibirTextoNaTela**('p', mensagemTentativas);

//Código omitido

COPIAR CÓDIGO

**Mônica:** O JavaScript é uma linguagem não-tipada, então quando informamos que vamos receber um parâmetro, não precisamos especificar qual o seu tipo. Isso nos permite trocar o tipo do parâmetro para uma variável.

**Guilherme:** Agora, precisamos atualizar a contagem de tentativas sempre que o chute for errado.

Para isso, no fim do else{} escrevemos tentativas++.

//Código omitido

**function** **verificarChute**() {

**let** chute = document.**querySelector**('input').value;

**if** (chute == numeroSecreto)

**exibirTextoNaTela**('h1', 'Acertou!');

**let** palavraTentativa = tentativas > 1 ? 'tentativas': 'tentativa';

**let** mensagemTentativas = `Você descobriu o número secreto com ${tentativas} ${palavraTentativa}!`;

**exibirTextoNaTela**('p', mensagemTentativas);

} **else** {

**if** (chute > numeroSecreto) { exibirTextoNaTela ('p', '0 número secreto é menor');

} **else** {

**exibirTextoNaTela**('p', 'O número secreto é maior');

}

tentativas++;

}

}

COPIAR CÓDIGO

**Guilherme:** Agora, vamos testar o jogo para verificar se tudo está funcionando corretamente. No navegador, atualizamos a página do jogo.

Primeiro tentamos o número "5". A mensagem exibida é "O número secreto é menor". Tentamos "2" e o número secreto é menor. Em seguida, tentamos o número "3". A mensagem exibida é "Você descobriu o número secreto com 3 tentativas!".

Agora, vamos testar da outra forma. Para isso, na primeira linha do código, apagamos gerarNumeroAleatorio() e escrevemos 2.

**let** numeroSecreto = 2;

//Código omitido

COPIAR CÓDIGO

Atualizamos o jogo e chutamos o número "2". Feito isso, aparece a mensagem "Você descobriu o número secreto com 1 tentativa!".

**Mônica:** Importante ressaltar que a contagem considera apenas a primeira tentativa de acerto. Em caso de acertos posteriores, não são consideradas uma tentativa. Somamos apenas os casos de erro.

**Guilherme:** Para finalizar, na primeira linha de código, apagamos o 2 e voltamos o gerarNumeroAleatório().

Agora, sim, temos as mensagens corretas no singular e plural, dependendo do número de tentativas realizadas para descobrir o número secreto.

Template strings, também conhecidas como template literals, são uma forma de criar strings em JavaScript que permite a interpolação de expressões. Elas são delimitadas por crases (`) ao invés de aspas simples ou duplas.As principais características das template strings são:

Interpolação de expressões: Você pode incluir expressões JavaScript dentro das strings, envolvendo-as com ${}. Essas expressões serão avaliadas e seu resultado será inserido na string.

Exemplo:

Javascript

Copiar código

const name = "João";

console.log(`Olá, ${name}!`); // Saída: Olá, João!

Strings multilinha: As template strings permitem que você crie strings que ocupem várias linhas, sem a necessidade de usar concatenação ou caracteres de escape.

Exemplo:

Javascript

Copiar código

const message = `Olá,

como você

está?`;

console.log(message);

Expressões de template: As template strings também permitem o uso de expressões de template, que são funções que recebem a string literal e as partes interpoladas como argumentos, permitindo manipulá-las de forma mais avançada.

Exemplo:

javascript

Copiar código

function upperCase(strings, ...values) {

return strings.map((str, i) => `${str}${values[i] ? values[i].toUpperCase() : ""}`).join('');

}

const name = "joão";

console.log(upperCase`Olá, ${name}!`); // Saída: Olá, JOÃO!

As template strings tornam o código mais legível e facilitam a manipulação de strings, especialmente quando você precisa incluir expressões dinâmicas em seus textos.

# 04Botão reiniciar

**Guilherme:** Tem uma parte do jogo que me deixa um pouco confuso. Vou digitar o número "5" e depois clicar em "Chutar". Feito isso aparece a mensagem dizendo que o número secreto é menor.

Perceba que o número 5 continua aparecendo no campo. Seria interessante que a cada tentativa o campo fosse limpo. Faremos isso agora!

**Implementando o campo vazio**

Analisando nosso código, percebemos que, quando acertamos o chute, deixamos o número exibindo na tela. Mas, quando erramos, é melhor deixarmos o campo de texto limpo.

Para isso, abaixo da linha tentativas++, escrevemos limparCampo. Como isso é algo que o Javascript não sabe o que é, podemos criar uma função.

**Mônica:** Inclusive, essa é uma prática que pode nos ajudar em outros casos futuros em que queiramos limpar o campo de texto.

**Guilherme:** Na linha 35, escrevemos function limparCampo(). Sua única responsabilidade será limpar o campo, dela não esperamos nenhum retorno.

Então, após os parênteses adicionamos chaves. Dentro, passamos a variável chute igual à document.querySelector('input').

Não colocamos .value porque não estamos querendo pegar o valor. Ao invés disso, na linha seguinte, passamos chute.value igual à '', ou seja, uma *string* vazia.

//Código omitido

**function** **limparCampo**() {

chute = document.**querySelector**('input');

chute.value = '';

}

COPIAR CÓDIGO

**Mônica:** *Strings* são sempre textos entre aspas. Como não definimos nenhum texto, o campo deve ficar vazio.

**Guilherme:** Vamos testar! Abrimos o jogo e atualizamos a página. Digitamos o número 5 e clicamos em "Chutar". Aparece a mensagem "O número secreto é menor" e o campo fica vazio.

Depois, chutamos o número "4" e a mensagem "Você descobriu o número secreto com 3 tentativas" aparece. Abaixo, o campo exibe o número4. Deu certo!

Se analisarmos o jogo, temos um botão chamado "Novo Jogo". Faremos com que quando o número for acertado ele seja habilitado. Depois podemos fazer a implementação.

index.html

**Mônica:** Podemos abrir o arquivo index.html para identificar como o botão está sendo desativado.

Na linha 28 temos o botão com clique vazio. No fim desse trecho de código, encontramos o disable. É isso o que está desativando o botão. Sendo assim precisamos editar esse atributo para ativá-lo.

**Habilitando o Botão Novo Jogo**

app.js

**Guilherme:** Não mexeremos no HTML. Então, voltamos para o arquivo app.js.

**Mônica:** O botão precisa ser ativado quando o número secreto for ativado e assim iniciar um novo jogo.

**Guilherme:** Exatamente! A primeira coisa que faremos é pegar esse botão do HTML.

Porém, temos um problema. Existem dois botões, então se definirmos document.querySelector("button"), como estávamos fazendo antes,o código provavelmente selecionará o primeiro botão ao buscar entre todos os elementos.

Repare que o segundo botão possui uma característica diferente, o id="reiniciar".

O id é um atributo que especifica o botão e deve ser único, não pode ser usado em outro lugar.

Então, na linha 20, escrevemos document.getElementById.

Lembrando que, por padrão, o "E", "B" e "I" precisam ser letras maiúsculas.

**Mônica:** Para especificar qual é o id que queremos, colocamos parênteses. Dentro adicionamos aspas seguido da palavra reiniciar, o nome do ID. Precisa ser igual ao HTML.

**Guilherme:** Se nessa mesma linha adicionarmos um ponto, que é como acessaremos esse elemento, a ferramenta sugere vários atributos.

**Mônica:** Segundo a lógica, queremos remover o atributo que faz com que o botão seja desativado. Nesse caso, irá funcionar se usarmos o removeAttribute. Existem várias funções, mas essa é a que queremos, remover o atributo e não o removeAttributeNS(Node).

**Guilherme:** Então, em após removeAttribute() adicionamos aspas simples e escrevemos disable.

//Código omitido

**function** **verificarChute**() {

**let** chute = document.**querySelector**('input').value;

**if** (chute == numeroSecreto) {

**exibirTextoNaTela**('h1', 'Acertou!');

**let** palavraTentativa = tentativas > 1 ? 'tentativas' : 'tentativa';

**let** mensagemTentativas = `Você descobriu o número secreto com ${tentativas} ${palavraTentativa}!`;

**exibirTextoNaTela**('p', mensagemTentativas);

document.**getElementById**('reiniciar').**removeAttribute**('disabled');

//Código omitido

COPIAR CÓDIGO

**Guilherme:** Agora, vamos testar. No jogo, chutamos o número "5" e a mensagem "Você descobriu o número secreto com 1 tentativa!" aparece. Logo em seguida, note que o botão "Novo jogo" é habilitado.

O próximo passo é fazermos com que esse botão funcione e comece um novo jogo.

**Até o próximo vídeo!**

# 05Reiniciando o Jogo

**Guilherme:** Agora definiremos uma funcionalidade para o botão "Novo Jogo". Ao clicá-lo, queremos que as tentativas sejam zeradas e um novo número seja sorteado.

**Mônica:** Como queremos que isso seja iniciado com um clique, então usaremos o onClick.

**Definindo a funcionalidade do botão Novo Jogo**

index.html

**Guilherme:** Se acessarmos o HTML, na linha 28 temos um onclick vazio. Após o sinal de igual, dentro de aspas duplas, criamos uma função chamada reiniciarJogo().

Lembrando que precisamos usar os parênteses para indicar que é uma função.

//Código omitido

**class**="container\_input">

<div class="chute container\_botoes">

<button onclick="verificarChute()"

class="container botao">Chutar</button>

<button onclick="reiniciarJogo()" id="reiniciar"

class="container\_botao" disabled>Novo jogo</

button>

</div>

</div>

COPIAR CÓDIGO

app.js

Feito isso, voltamos no app.js. No fim do código, na linha 41, criamos a função reiniciarJogo(), que não receberá nenhum parâmetro.

Vamos executar essa função quando o botão for clicado, diferente de como geralmente usamos funções no JavaScript, aqui não invocamos a função.

Ao iniciar o jogo queremos que um novo número secreto seja sorteado. Então, adicionamos chaves e dentro passamos numeroSecreto igual à gerarNumeroAleatorio().

**Mônica:** Em seguida, precisamos que o campo fique vazio e que as tentativas sejam iniciadas como 1.

**Guilherme:** Isso mesmo! Então, passamos limparCampo() e na linha abaixo tentativas = 1.

//Código omitido

function **reiniciarJogo**() {

numeroSecreto = gerarNumeroAleatorio();

limparCampo();

tentativas = 1;

}

COPIAR CÓDIGO

**Guilherme:** Vamos testar. Primeiro precisamos acertar o número secreto, então digitamos "5".

Depois, clicamos no botão "Novo jogo". Um novo número é sorteado, mas temos um problema, a mensagem "Acertou" e "Você descobriu o número secreto em 1 tentativa" continuam aparecendo. Precisamos mudar isso.

Para facilitar, copiamos as linhas 9 e 10, referente as funções exibirTextoNaTela() e colamos na linha 45. Agora, faremos as alterações necessárias.

Vamos testar de novo. Ao acertar o número secreto clicamos em "Novo jogo" e a mensagem muda para "Jogo do número secreto. Escolha um número entre 1 e 10".

**Mônica:** Deu certo! Está funcionando, mas há código repetido e isso não é uma boa prática.

**Guilherme:** Verdade, Môni! Suponhamos que o jogo foi atualizado para a versão 2.0. Nesse caso, se na linha 9, mudarmos o texto para "Jogo do número secreto 2.0" quando rodarmos o jogo pela segunda vez, essa mensagem não será exibida.

Fazer essa alteração nos dois trechos de código não é uma boa prática. Sendo assim, precisamos isolar esse comportamento em uma função.

Na linha 9, criamos a função exibirMensagemInicial(). A única responsabilidade dessa função será exibir as mensagens iniciais.

Então, colamos as duas funções exbirNaTela() e colamos logo abaixo.

function **exibirMensagemInicial**() {

exibirTextoNaTela('h1', 'Jogo do número secreto');

exibirTextoNaTela('p', 'Escolha um número entre 1 e 10'); I

}

//Código omitido

COPIAR CÓDIGO

**Guilherme:** Agora, precisamos colocar essa mensagem inicial na função reiniciarJogo(). Para isso, apagamos as linhas 48 e 49. No lugar, passamos a função exibirMensagemInicial().

//Código omitido

function **reiniciarJogo**() {

numeroSecreto = gerarNumeroAleatorio();

limparCampo();

tentativas = 1;

exibirMensagemInicial()

}

COPIAR CÓDIGO

**Mônica:** Em seguida, precisamos chamar a nova função fora de qualquer outra função para ser iniciada na primeira vez que o app.js for lido.

**Guilherme:** Exatamente! Então, na linha 14, escrevemos exibirMensagemInicial(). Lembrando que poderia ser em outra linha.

function exibirMensagemInicial() {

exibirTextoNaTela('h1', 'Jogo do número secreto');

exibirTextoNaTela('p', 'Escolha um número entre 1 e 10'); **I**

}

exibirMensagemInicial()

//Código omitido

COPIAR CÓDIGO

**Guilherme:** Vamos testar. Chutamos até acertar. Depois, clicamos em "Novo Jogo" e ele é iniciado. Deu certo!

Porém, repare que ainda temos um problema. Sempre que clicamos em "Novo Jogo", ele traz um novo jogo. Seria interessante se pudéssemos restaurar o status da nossa aplicação o campo desabilitado. Faremos isso.

No fim do código, na função reiniciarJogo() passamos document.getElementById(). Nos parênteses adicionamos aspas simples e escrevemos reiniciar seguido do atributo .setAttribute().

Dentro dos parênteses, passamos o nome do atributo que queremos setar, então 'disabled'. Adicionamos vírgula e passamos o segundo parâmetro true.

//Código omitido

**function** **reiniciarJogo**() {

numeroSecreto = **gerarNumeroAleatorio**();

**limparCampo**();

tentativas =1;

**exibirMensagemInicial**();

document.**getElementById**('reiniciar').**setAttribute**('disabled', **true**)

}

COPIAR CÓDIGO

Voltamos no jogo. Repare que começamos com o botão "Novo Jogo" desabilitado. Após acertar o número da sorte ele é habilitado. Ao clicá-lo ele desabilita novamente.

**Mônica:** Deu tudo certo!

# 06Calculadora de Média e Situação do Aluno

Você foi contratado (a) para desenvolver uma calculadora de média e verificar a situação de aprovação de um aluno com base em suas quatro notas. A média para aprovação é 5. Sua tarefa é implementar duas funções em JavaScript:

* calcularMedia(nota1, nota2, nota3, nota4)=> Esta função recebe as quatro notas do aluno como parâmetros e retorna a média calculada com base nessas notas.
* verificarAprovacao(media) => Esta função recebe a média do aluno como parâmetro e retorna "Aprovado" se a média for maior ou igual a 5, caso contrário, retorna "Reprovado".

Utilize esses valores para as notas:

let nota1 = 7;

let nota2 = 6;

let nota3 = 3;

let nota4 = 5;

COPIAR CÓDIGO

Posto disso, analise cada afirmação abaixo e marque apenas aquelas que mostram partes de código que seguem as regras de negócio citadas acima.

**function** **verificarAprovacao**(media) {

**return** media >= 5 ? "Aprovado" : "Reprovado";

}

Isso aí! A função verifica se a média passada como parâmetro é maior ou igual a 5 e retorna um valor em formato de texto informando se foi aprovado ou reprovado.

**function** **calcularMedia**(nota1, nota2, nota3, nota4) {

**let** media = (nota1 + nota2 + nota3 + nota4) / 4;

**return** media;

}

Certo! Esta função calcula a média aritmética das quatro notas fornecidas. Primeiro, ela soma as quatro notas e depois divide essa soma por 4, que é o número total de notas. E por fim retorna o resultado dessa operação, que é a média do aluno.

console.log(calcularMedia(nota1, nota2, nota3, nota4));

console.log(verificarAprovacao(calcularMedia(nota1, nota2, nota3, nota4)));

# 07Desafio: hora da prática

Saber como usar funções é um dos conceitos fundamentais da programação e do desenvolvimento de software. As funções desempenham um papel crucial na organização, modularização e reutilização de código.

Pensando nisso, criamos uma lista de atividades (não obrigatórias) focada em prática para melhorar ainda mais sua experiência de aprendizagem. Bora praticar então?

**Desafios**

1. Crie uma função que calcule o índice de massa corporal (IMC) de uma pessoa, a partir de sua altura, em metros, e peso, em quilogramas, que serão recebidos como parâmetro.
2. Crie uma função que calcule o valor do fatorial de um número passado como parâmetro.
3. Crie uma função que converte um valor em dólar, passado como parâmetro, e retorna o valor equivalente em reais. Para isso, considere a cotação do dólar igual a R$4,80.
4. Crie uma função que mostre na tela a área e o perímetro de uma sala retangular, utilizando altura e largura que serão dadas como parâmetro.
5. Crie uma função que mostre na tela a área e o perímetro de uma sala circular, utilizando seu raio que será fornecido como parâmetro. Considere Pi = 3,14.
6. Crie uma função que mostre na tela a tabuada de um número dado como parâmetro.

Caso precise de ajuda, opções de solução das atividades estarão disponíveis na seção “Opinião da pessoa instrutora”.

# Resolução dos desafios 2

Em uma carreira de desenvolvimento de software, a prática consistente da lógica de programação desempenha um papel fundamental na construção de bases sólidas. A lógica de programação não apenas permite a criação de algoritmos eficientes e soluções elegantes, mas também desenvolve a capacidade de pensar de forma estruturada e analítica. Essa habilidade é essencial para enfrentar desafios complexos e transformar problemas abstratos em implementações tangíveis.

Pensando nisso, criamos uma lista de atividades (não obrigatórias) focada em prática para melhorar ainda mais sua experiência de aprendizagem.

#### Sugestões de respostas

1. Crie uma função que calcule o índice de massa corporal (IMC) de uma pessoa, a partir de sua altura, em metros, e peso, em quilogramas, que serão recebidos como parâmetro:

function calculaIMC(altura, peso){

let imc = peso / (alturaMetros \* alturaMetros);

}

1. Crie uma função que calcule o valor do fatorial de um número passado como parâmetro.

function calcularFatorial(numero) {

if (numero === 0 || numero === 1) {

return 1;

}

let fatorial = 1;

for (let i = 2; i <= numero; i++) {

fatorial \*= i;

}

return fatorial;

}

// Exemplo de uso

let numero = 5;

let resultado = calcularFatorial(numero);

console.log(`O fatorial de ${numero} é ${resultado}`);

1. Crie uma função que converte um valor em dólar, passado como parâmetro, e retorna o valor equivalente em reais. Para isso, considere a cotação do dólar igual a R$ 4,80.

function converterDolarParaReal(valorEmDolar) {

let cotacaoDolar = 4.80;

let valorEmReais = valorEmDolar \* cotacaoDolar;

return valorEmReais.toFixed(2);

}

// Exemplo de uso

let valorEmDolar = 50;

let valorEmReais = converterDolarParaReal(valorEmDolar);

console.log(`${valorEmDolar} dólares equivalem a R$ ${valorEmReais}`);

1. Crie uma função que mostre na tela a área e o perímetro de uma sala retangular, utilizando altura e largura que serão dadas como parâmetro.

function calcularAreaPerimetroSalaRetangular(altura, largura) {

let area = altura \* largura;

let perimetro = 2 \* (altura + largura);

console.log(`Área da sala: ${area} metros quadrados`);

console.log(`Perímetro da sala: ${perimetro} metros`);

}

// Exemplo de uso

let altura = 3; // em metros

let largura = 5; // em metros

calcularAreaPerimetroSalaRetangular(altura, largura);

1. Crie uma função que mostre na tela a área e o perímetro de uma sala circular, utilizando seu raio que será fornecido como parâmetro. Considere Pi = 3,14.

function calcularAreaPerimetroSalaCircular(raio) {

let area = Math.PI \* raio \* raio;

let perimetro = 2 \* Math.PI \* raio;

console.log(`Área da sala circular: ${area.toFixed(2)} metros quadrados`);

console.log(`Perímetro da sala circular: ${perimetro.toFixed(2)} metros`);

}

// Exemplo de uso

let raio = 4; // em metros

calcularAreaPerimetroSalaCircular(raio);

1. Crie uma função que mostre na tela a tabuada de um número dado como parâmetro.

function mostrarTabuada(numero) {

for (let i = 1; i <= 10; i++) {

let resultado = numero \* i;

console.log(`${numero} x ${i} = ${resultado}`);

}

}

// Exemplo de uso

let numero = 7;

mostrarTabuada(numero);

# 08O que aprendemos?

**Nesta aula:**

* Aprendemos como criar um programa para verificar se o “chute” inserido é igual ao número secreto definido. Utilizamos estruturas condicionais para tomar decisões com base no resultado dessa comparação;
* Criamos uma variável para armazenar a quantidade de tentativas realizadas pelo usuário;
* Vimos a importância de consultar a documentação da linguagem e das bibliotecas utilizadas no desenvolvimento do programa. A documentação é uma fonte valiosa de informações que nos auxilia na compreensão de conceitos e no uso correto das funcionalidades disponíveis.

**Na próxima aula:**

Vamos aprender como funcionam as listas (ou arrays, em inglês) e como são fundamentais na carreira de desenvolvimento de software!