# Pensando Como um Programador

samuelamaro96746313

March 2021

Uma das coisas mais difíceis de aprender na programação não é a sintaxe que você precisa aprender, mas como aplicá-la para resolver problemas do mundo real. Você precisa começar a pensar como um programador - isso geralmente envolve olhar para as descrições do que seu programa precisa fazer e analisar como eles podem ser aplicados na solução real (prática), quais recursos de código são necessários para alcançar esse objetivo, e como fazê-los trabalhar em conjunto.

Isso requer um mistura de trabalho duro, experiência com a sintaxe de programação utilizada e prática, além de um pouco de criatividade, é claro. Quanto mais você programa, melhor programador se torna. Nós não prometemos transformar seu cérebro em um "cérebro de programador"em 5 minutos, mas vamos te dar todas as oportunidades para pensar na prática como um programador ao longo deste curso.

Com isso em mente, vejamos o exemplo que estaremos construindo neste artigo e analisaremos o processo geral de dissecá-lo em tarefas tangíveis.

## Exemplo - jogo adivinhe um número

Vamos imaginar que o seu chefe te deu as seguintes diretrizes para criar este jogo: Quero que você crie um jogo simples do tipo adivinhe um número. Ele deve gerar um número aleatório de 1 a 100, depois desafiar o jogador a adivinhar o número em 10 rodadas. A cada rodada deve ser dito ao jogador se ele está certo ou errado, se estiver errado, deve ser dito se o palpite é muito baixo ou muito alto. Também deve ser mostrado ao jogador os números que ele tentou adivinhar anteriormente. O jogo termina se o jogador acertar o número ou acabarem o número de tentativas. Quando o jogo acabar, deve ser dado ao jogador a opção de jogar novamente.

Olhando para o enunciado, a primeira coisa que devemos fazer é quebrá-

lo em pequenas tarefas, da forma mais parecida com o pensamento de um programador quanto possível:

1. Gerar um número aleatório entre 1 e 100.
2. Gravar o número do turno que o jogador está. Iniciar em 1.
3. Dar ao jogador uma forma de adivinhar o número.
4. Após a tentativa ter sido submetida, primeiro gravar em algum lugar para que o usuário possa ver as tentativas anteriores.
5. Depois, verificar se o palpite está correto.
6. Se estiver correto:
   (a) Escrever mensagem de parabéns.
   (b) Impedir que o jogador insira mais respostas (isso pode bugar o jogo).
   (c) Mostrar controle que permita ao jogador reiniciar o jogo.
7. Se o palpite estiver errado e o jogador ainda tem turnos sobrando:
   (a) Dizer ao jogador que ele está errado.
   (b) Permitir que ele insira outra resposta.
   (c) Incrementar o número do turno em 1
8. Se o jogador está errado mas não tem turnos sobrando:
   (a) Dizer ao jogador que o jogo acabou.
   (b) Impedir que o jogador insira mais respostas (isso pode bugar o jogo).
   (c) Mostrar controle que permita ao jogador reiniciar o jogo.
9. Quando reiniciar, tenha certeza de resetar todas as variáveis e a interface do jogo, então volte para o passo 1.

Então vamos em frente, olhando como podemos transformar esses passos em código, construindo esse exemplo e explorando as ferramentas do JavaScript ao longo do caminho.

## Adicionando variáveis para armazenar nossos dados

Vamos começar. Primeiramente, adicione as seguintes linhas na sua tag `<script>|`:

```
1
2       var numeroAleatorio= Math.floor(Math.random() * 100) + 1;
3
4       var palpites = document.querySelector('.palpites');
5       var ultimoResultado = document.querySelector('.
    ultimoResultado');
6       var baixoOuAlto = document.querySelector('.baixoOuAlto');
7
8       var envioPalpite = document.querySelector('.envioPalpite');
9       var campoPalpite = document.querySelector('.campoPalpite');
10
11      var contagemPalpites = 1;
12      var botaoReinicio;
13
14
```

Listing 1: Adicionando variaveis

Aqui estamos setando as variáveis que precisamos para guardar os dados que nosso programa irá utilizar. Variáveis são basicamente recipientes para valores (como números, ou strings ou textos). Variáveis são criadas com a palavra-chave `var` seguida de um nome para sua variável. Você pode atribuir um valor para sua variável com um sinal de igual `(=)` seguido do valor que você quer dar a ela.

No nosso exemplo:

- À primeira variável — `numeroAleatorio` — é atribuído um número aleatório entre 1 e 100, calculado usando um algoritmo matemático.
- As próximas três variáveis são criadas para guardar uma referência para os parágrafos resultantes em nosso HTML, e são usadas para inserir valores nos parágrafos no código:

```
1
2           <p class="palpites"></p>
3           <p class="ultimoResultado"></p>
4           <p class="baixoOuAlto"></p>
5
6
```

- As próximas duas variáveis armazenam referências para o campo de texto e o botão de envio e são usados para controlar o envio do palpite.

```
1
2           <label for="campoPalpite">Digite seu palpite: </
    label><input type="text" id="campoPalpite" class="
    campoPalpite">
3           <input type="submit" value="Enviar palpite" class
    ="envioPalpite">
4
5
```

- As últimas duas variáveis (contagemPalpites e botaoReinicio) são usadas para armazenar a contagem dos palpites do usuário, e o outro é uma referência para o botão de reset, que não existe ainda (mas irá existir).

## Funções

Em seguida, adicione o seguinte código abaixo do JavaScript anterior:

```
1
2         function conferirPalpite() {
3             alert('Eu sou um placeholder');
4         }
5
6
```

Funções são blocos reutilizáveis de código que você pode escrever uma vez e executá-lo de novo e de novo, eliminando a necessidade de repetir o código todas as vezes. Isso é realmente útil. Há várias formas de se definir funções, mas, por agora, vamos nos concentrar em um tipo simples. Aqui nós definimos uma função usando a palavra chave `function|`, seguida de um nome, com parênteses colocados na sequência. Depois disso nós colocamos duas chaves `({ })|`. Dentro das chaves vai todo o código que queremos executar sempre que chamarmos a função.

O código é executado digitando o nome da função seguido pelos parênteses.

Tente salvar o seu código agora e atualizá-lo no navegador.

## Operadores

Os operadores JavaScript nos permite realizar testes, fazer cálculos matemáticos, unir sequências de texto, e outras coisas do tipo.

Vamos salvar nosso código e atualizar a página exibida em nosso navegador. Abra o console JavaScript se você ainda não o tiver aberto, para que possamos digitar os exemplos mostrados abaixo — digite cada um exatamente como mostrado na coluna "Exemplo", pressionando Return/Enter na sequência, e veja quais resultados são retornados.

| Operador | Nome | Exemplo |
|---|---|---|
| + | Adição | 6 + 9 |
| - | Subtração | 20 - 15 |
| | Multiplicação | 3 * 7 |
| / | Divisão | 10 / 5 |

Você também pode usar o operador + para unir sequências de texto (isso é chamado de concatenação em programação). Tente inserir as seguintes linhas:

```
var nome = 'Bingo';
nome;
var ola = ' diz ol !';
ola;
var cumprimento = nome + ola;
cumprimento;
```

Há também alguns atalhos para operadores disponíveis, chamados de operadores de atribuição ampliada (ou atribuição composta). Por exemplo, se você quer adicionar uma nova sequência de texto à uma existente e retornar o resultado, você pode fazer o seguinte:

```
nome += 'Diz ol !';
```

Isso é equivalente a:

```
nome = nome + 'Diz ol !';
```

Quando estamos rodando testes de verdadeiro/falso (por exemplo, condicinais internas — veja abaixo, usamos operadores de comparação, por exemplo:

| Operador | Nome | Exemplo |
|---|---|---|
| === | Igualdade estrita(é estritamente o mesmo ?) | 5 === 2 + 4 |
| !== | Não-igualdade(não é o mesmo?) | 'Chris' !== 'Ch' + 'ris' |
| < | Menor que | 10 < 6 |
| > | Maior que | 10 > 20 |

## Condicionais

Voltando à nossa função `conferirPalpite()`, imagino que seja seguro dizer que não queremos que ela apenas exiba uma mensagem de teste (placeholder). Nós queremos verificar se o palpite do jogador está correto ou não, e responder apropriadamente.

```
1
2       function conferirPalpite() {
3             var palpiteUsuario = Number(campoPalpite.value);
4             if (contagemPalpites === 1) {
5               palpites.textContent = 'Palpites anteriores: ';
6             }
7             palpites.textContent += palpiteUsuario + ' ';
8
9             if (palpiteUsuario === numeroAleatorio) {
10              ultimoResultado.textContent = 'Parab ns! Voc
    acertou!';
11              ultimoResultado.style.backgroundColor = 'green';
12              baixoOuAlto.textContent = '';
13              configFimDeJogo();
14            } else if (contagemPalpites === 10) {
15              ultimoResultado.textContent = '!!!FIM DE JOGO!!!';
16              baixoOuAlto.textContent = '';
17              configFimDeJogo();
18            } else {
19              ultimoResultado.textContent = 'Errado!';
20              ultimoResultado.style.backgroundColor = 'red';
21              if(palpiteUsuario < numeroAleatorio) {
22                baixoOuAlto.textContent = 'Seu palpite est
    muito baixo!';
23              } else if(palpiteUsuario > numeroAleatorio) {
24                baixoOuAlto.textContent = 'Seu palpite est
    muito alto!';
25              }
26            }
27
28            contagemPalpites++;
29            campoPalpite.value = '';
30            campoPalpite.focus();
31      }
32
33
```

Isso é bastante código — ufa! Vamos abordar cada seção e explicar o que faz.

- A primeira linha (linha 2 no código acima) declara uma variável chamada palpiteUsuario e define seu valor igual ao valor inserido pelo jogador no campo de texto. Nós também rodamos esse valor através do método embutido `Number()`, apenas para ter certeza de que o valor inserido é um número.
- Em seguida, encontramos nosso primero bloco de código condicional (linhas 3–5 no código acima). Um bloco de código condicional lhe permite executar código seletivamente, dependendo se uma condição é verdadeira ou não. Se parece um pouco com uma função, mas não é. A forma mais simples de um bloco condicional começa com a palavra chave `if`, depois os parênteses, depois as chaves. Dentro dos parênteses nós incluímos um teste. Se o teste retornar true(verdadeiro), o código dentro das chaves é

executado. Caso contrário, não é executado, e seguimos para a próxima parte do código. Neste caso, o teste está verificando se a variável contagemPalpites é igual a 1 (isto é, se essa é ou não a primeira tentativa do jogador):

```
1
2             contagemPalpites === 1
3
4
```

Se a condição for verdadeira, nós tornamos o conteúdo do parágrafo de palpites, `<p class="palpites"></p>` igual a "Palpites anteriores: ". Caso contrário, o texto não é exibido.

- A linha 6 acrescenta o valor atual de palpiteUsuario ao final do parágrafo palpites, mais um espaço em branco para que haja espaçamento entre cada palpite mostrado.
- O próximo bloco (linhas 8–24 acima) fazem as seguintes conferências:
  - O primeiro `if(){ }` confere se o palpite do jogador é igual ao número aleatório (numeroAleatorio) definido no topo do nosso JavaScript. Se for, o jogador adivinhou corretamente o número e venceu o jogo. Então mostramos ao jogador uma mensagem de parabenização com uma agradável cor verde, limpamos o conteúdo do parágrado que informa sobre o palpite ser alto ou baixo `<p class="baixoOuAlto"></p>`, e executamos uma função chamada `configFimDeJogo()`, que iremos discutir mais tarde.
  - Agora nós encadeamos outro teste ao final deste anterior usando uma estrutura `else if(){ }`. Este confere se o palpite do jogador é sua última tentativa. Se for, o programa faz o mesmo que no bloco anterior, porém com uma mensagem de fim de jogo ao invés do texto de parabeninzação.
  - O bloco final encadeado ao final do código `(else { })` contém código que só é executado se nenhum dos outros dois testes retornar verdadeiro (ou seja, o jogador não acertou o número, porém ainda tem mais tentativas restantes). Neste caso nós dizemos a ele que está errado, e então rodamos outro teste condicional para checar se o palpite foi maior ou menor do que a resposta certa, exibindo então uma mensagem apropriada para informá-lo se foi maior ou menor.
- As próximas três linhas da função (linhas 26–28) nos deixa preparados para o próximo palpite ser submetido. Nós somamos 1 à variável contagemPalpites para que o jogador use sua tentativa (`++` é uma operação de incremento — incrementa em 1), e o campo de texto do formulário de inserção seja esvaziado e focado novamente, pronto para que o próximo palpite seja inserido.

## Eventos

Neste ponto temos uma função `conferirPalpite()` bem implementada, mas ela não irá fazer nada pois nós não a chamamos ainda. Idealmente nós queremos que ela seja acionada quando o botão "Enviar palpite"for pressionado, e para fazer isso precisamos usar um evento. Eventos são ações que acontencem no navegador, como um botão sendo clicado, ou uma página carregada, ou um vídeo tocando; ações as quais podemos responder executando blocos de código. Os construtores que monitoram os acontecimentos de eventos são chamados de **event listeners**, e os blocos de código executados em resposta ao acontecimento do evento são chamados de **event handlers**.

Adicione a seguinte linha abaixo da chave de fechamento da sua função `conferirPalpite()`:

```
1
2       envioPalpite.addEventListener('click', conferirPalpite);
3
4
```

Aqui nós estamos adicionando um **event listener** ao botão `envioPalpite`. Esse é um método que aceita a inserção de dois valores (chamados de argumentos) — o tipo de envento que estamos monitorando (neste caso o evento **click**) como um string (sequência de texto), e o código que queremos executar quando o evento ocorrer (neste caso a função `conferirPalpite()` — note que não temos que especificar os parênteses quando estivermos escrevendo dentro de `addEventListener()|`).

Tente agora salvar e atualizar seu código, e seu exemplo deve funcionar agora, até um ponto. O único problema agora é que se você acertar o palpite ou ficar sem mais tentativas o jogo irá falhar, porque ainda não definimos a função `configFimDeJogo()` que deve ser executada uma vez que o jogo terminar. Vamos adicionar agora o código restante e completar a funcionalidade do nosso exemplo.

## Finalizando a funcionalidade do jogo

Vamos adicionar a função `configFimDeJogo()` ao final do nosso código e então explorá-lo. Adicione agora isso, abaixo do restante do seu JavaScript:

```
1
2       function configFimDeJogo() {
3         campoPalpite.disabled = true;
4         envioPalpite.disabled = true;
5         botaoReinicio = document.createElement('button');
6         botaoReinicio.textContent = 'Iniciar novo jogo';
7         document.body.appendChild(botaoReinicio);
```

```
        botaoReinicio.addEventListener('click', reiniciarJogo);
    }


```

- As primeiras duas linhas desabilitam a entrada de texto do formulário e o clique do botão, definindo a propriedade disabled (desabilitado) de cada um como true (verdadeiro). Isso é necessário, pois se não o fizermos, o usuário poderia submeter mais palpites depois do jogo ter terminado, o que iria bagunçar as coisas.
- As próximas três linhas geram um novo elemento `<button>|`, define o texto de seu rótulo como "Iniciar novo jogo", e o adiciona ao final do nosso HTML existente.
- A linha final define um monitor de evento (event listener) em nosso botão, para que quando seja clicado, uma função chamada `reiniciarJogo()` seja executada.

Agora precisamos definir essa função também! Adicione o seguinte código, novamente ao final do nosso JavaScript:

```

    function reiniciarJogo() {
        contagemPalpites = 1;

        var reiniciarParas = document.querySelectorAll('.resultadoParas p');
        for (var i = 0 ; i < reiniciarParas.length ; i++) {
          reiniciarParas[i].textContent = '';
        }

        botaoReinicio.parentNode.removeChild(botaoReinicio);

        campoPalpite.disabled = false;
        envioPalpite.disabled = false;
        campoPalpite.value = '';
        campoPalpite.focus();

        ultimoResultado.style.backgroundColor = 'white';

        numeroAleatorio = Math.floor(Math.random() * 100) + 1;
    }


```

Esse longo bloco de código redefine completamente tudo do modo como era no início do jogo, para que o jogador possa jogá-lo novamente. Ele:

- Coloca o valor da variável `contagemPalpites` novamente igual a 1.

- Limpa todos os parágrafos de informativos.
- Remove o botão resete do nosso código.
- Habilita os elementos do formulários, esvazia e direciona o foco ao campo de texto, pronto para que um novo palpite seja inserido.
- Remove a cor de fundo do parágrafo `ultimoResultado`.
- Gera um novo número aleatório para que o jogador não esteja tentando adivinhar o mesmo número novamente

**Neste ponto você deve ter um jogo (simples) completamente funcional — parabéns!**

Tudo o que temos que fazer agora neste artigo é falar sobre alguns outros recursos importantes que você já viu, mesmo que não os tenha notado ainda.

## Loops

Uma parte do código acima que precisamos olhar mais detalhadamente é o loop for. Loop é um conceito muito importante em programação, que permite a você continuar executando um pedaço do código repetidamente, até que determinada condição seja satisfeita.

Para começar, vá novamente até o console JavaScript do seu navegador, e insira o seguinte:

```
1
2          for (var i = 1 ; i < 21 ; i++) { console.log(i) }
3
4
```

O que aconteceu? Os números de 1 a 20 foram exibidos no seu console. Isso acontece por causa do loop. Um loop for utiliza a inserção de três valores (argumentos):

1. **Um valor inicial**: Nesse caso estamos iniciando a contagem em 1, mas poderia ser qualquer outro número que quisesse utilizar. Você pode substituir i por qualquer número que quiser também, mas i é utilizado por convenção porque é curto e fácil de lembrar.
2. **Uma condição de saída**: Aqui nós especificamos `i < 21` — o loop irá continuar rodando até que i não seja mais menor que 21. Quando i alcançar 21, o loop não será mais executado.

3. Incremento: Nós especificamos `i++`, que siginifica "adicione 1 à i". O loop irá rodar uma vez para cada valor de i, até que i alcance o valor de 21 (como abordado acima). Nesse caso, nós estamos simplesmente imprimindo o valor de i no console em cada iteração usando console.log().

Agora vamos olhar o loop em nosso jogo de adivinhar o número — o código seguinte pode ser encontrado dentro da função `reiniciarJogo()`

```
1
2         var reiniciarParas = document.querySelectorAll('.
   resultadoParas p');
3     for (var i = 0 ; i < reiniciarParas.length ; i++) {
4             reiniciarParas[i].textContent = '';
5     }
6
7
```

Esse código cria uma variável contendo uma lista de todos os parágrafos dentro de `<div class="resultadoParas">` usando o método `querySelectorAll()`, e então faz o loop em cada um, removendo o conteúdo de texto dos mesmos.

## Uma pequena discussão sobre objetos

Vamos adicionar uma melhoria final antes de chegarmos a essa discussão. Adicione a linha seguinte logo abaixo da linha `var botaoReinicio;` próximo ao topo do seu JavaScript, em seguida salve nosso arquivo:

```
1
2         campoPalpite.focus();
3
4
```

Essa linha usa o método `focus()` para automaticamente colocar o cursor dentro campo de texto do ¡input¿ assim que a página carrega, significando que o usuário já pode começar a digitar o primeiro palpite, e não precisa clicar no campo do formulário primeiro. É apenas uma pequena adição, mas melhora a usabilidade — dando ao usuário uma boa dica visual do que ele deve fazer para jogar o jogo.

Vamos analisar o que está acontencedo aqui com um pouco mais de detalhes. Em JavaScript, tudo é um objeto. Um objeto é uma coleção de funcionalidades relacionadas armazenadas em um único agrupamento. Você pode criar seus próprios objetos, mas isso é bastante avançado e nós não iremos abordar até mais tarde no curso. Por agora, vamos apenas discutir brevemente os objetos pré-construídos presentes em seu navegador, que lhe permite fazer várias coisas úteis.

Neste caso particular, nós primeiro criamos a variável `campoPalpite` que armazena uma referência ao campo de inserção de texto do formulário em nosso HTML — a linha seguinte pode ser achada entre nossas declarações de variáveis próximas ao topo:

```
1
2       var campoPalpite = document.querySelector('.campoPalpite');
3
4
```

Para pegar essa referência, usamos o método `querySelector()` do objeto document. `querySelector()` pega um pedaço de informação — um seletor CSS que seleciona o elemento ao qual você quer referenciar.

Como agora `campoPalpite` contém referência ao elemento `<input>`, ele terá agora acesso a um número de propriedades (basicamente variáveis armazenadas dentro de objetos, sendo que alguns não podem ter seus valores alterados) e métodos (basicamente, funções armazenadas dentro de objetos). Um método disponível para elementos de inserção `<input>`, é o `focus()`, então agora podemos usar essa linha para focar o campo de inserção de texto:

```
1
2       campoPalpite.focus();
3
4
```

Variáveis que não contém referências a elementos de formulário não terão `focus()` disponível para elas. Por exemplo, a variável palpites contém referência de um elemento `<p>`, e contagemPalpites contém um número.