Lógica de Programação com JavaScript

Termos Técnicos e Definições Gerais

1. **O que é um programa?**
   Um programa, dentro do contexto computacional, é aquilo que nos permite realizar algum tipo de tarefa.
   Ex:
   Word = Escrever textos, criar documentos, currículos, etc.
   Excel = Criar planilhas financeiras.
   Chrome = Navegar em sites da internet.
   Paint = Criar desenhos simples.
   Photoshop = Editar fotografias de forma profissional.

2. **O que é programação?**
   Se o *programa* é o que nos permite realizar tarefas, *programação* é o ato de *escrever as tarefas que o programa vai realizar*.
   Em outras palavras, *programação* é como se fosse a *receita* da tarefa que o programa irá executar.

   Exemplo simples:
   Receita/Programa para fazer "Pão com manteiga"
   Passo 1: Abra o armário
   Passo 2: Pegue um pão
   Passo 3: Abra a gaveta do armário
   Passo 4: Pegue uma faca
   Passo 5: Abra o pão com a faca
   Passo 6: Vá até a geladeira
   Passo 7: Abra a geladeira
   Passo 8: Pegue a manteiga
   Passo 9: Abra o pote de manteiga
   Passo 10: Com a faca, pegue um pouco de manteiga
   Passo 11: Ainda com a faca, passe a manteiga no pão
   Passo 12: Feche o pão
   FIM

Quando escrevemos receitas, precisamos ser específicos não é mesmo? (Ok, talvez não tão específicos como no exemplo dizendo: "Vá até a geladeira" porque nós, humanos, sabemos mais ou menos onde estão as coisas que precisamos).

Mas quando escrevemos uma receita para um programa (ou seja, quando programamos) precisamos dizer ao computador, em detalhes, o que precisará ser feito.

O passo a passo que fizemos no exemplo anterior já é detalhado, mas para um robô ele precisaria ainda mais detalhado. Veja: ao dizer "Abra o armário", estou supondo que o robô que vai fazer a tarefa já está próximo do armário e que ele sabe o que é um armário, certo? Se nenhuma dessas situações for verdadeira, ainda precisamos especificar onde está o armário, o que o armário é e como ele deve chegar lá. *Ex: ande cinco passos, vire a esquerda, estique o braço, etc*.

Lembre-se sempre: **o computador é meio TONTO; se você não for específico ele talvez não saiba o que fazer**.

Sendo assim, quando formos programar, primeiro precisamos pensar na **tarefa** que vamos fazer (*ex: mostrar uma mensagem na tela*) e em seguida escrever a **receita** (*ou seja, um passo a passo com várias instruções*) de como a tarefa deverá ser feita pelo computador, blz?

3. **O que faz o programador?**
   Bom, *programador* é quem faz *programação*. Ou seja, é o profissional que escreve as **receitas** (isto é, as instruções, o passo a passo, o código, etc.) dos programas que deverão realizar alguma tarefa.

4. **E o que é Lógica de Programação?**
   Concordamos que *programação* são as instruções (a sequência de passos) que são necessárias(os) para realizar uma tarefa, certo?

Usando o exemplo citado do “Pão com manteiga” o que aconteceria se, o primeiro passo fosse “Com a faca, pegue um pouco de manteiga” e o segundo fosse “Abra a geladeira”?

Antes mesmo de eu ter o pão, a manteiga e a faca em mãos, a receita já está pedindo para eu pegar um pouco de manteiga com a faca que nem está na minha mão ainda. Em seguida, pede para eu abrir a geladeira... por quê? O que isso tem a ver com o passo anterior?

Sendo assim, perceba que para a receita funcionar você precisa seguir os passos numa ordem que faça sentido, não é? Se não as coisas podem não sair do jeito que você imaginava.

Desse modo, o *programador (aquele que escreve a receita)* deve organizar a receita de maneira que os passos façam sentido, para que assim, o resultado desejado seja obtido. E é justamente para isso que a *lógica de programação* existe.

A **Lógica de Programação** é o que nos ajuda a organizar os passos da nossa receita de uma maneira que faça sentido para chegar ao resultado que desejamos.

5. **O que é uma Linguagem de Programação?**
   Ainda usando como exemplo nossa receita de “Pão com manteiga”, quando escrevemos esse manual de como executar essa tarefa, alguns requisitos são necessários para quem irá executá-la, certo?

   Vejamos alguns deles:
   - Quem irá ler as instruções precisa ser... um ser humano (porque um cachorro não vai conseguir fazer um pão com manteiga... rs);
   - Essa pessoa precisa saber ler;
   - Essa pessoa precisa saber o idioma “Português” (porque, ainda que um americano possa fazer um pão com manteiga, as instruções precisam estar num idioma que ele conheça, se não, ele nunca entenderá as instruções);

   Da mesma forma que um americano que não entenda português

não conseguiria ler e entender a receita que escrevemos, o computador não consegue entender a programação se ela não estiver traduzida para o idioma dele.

O idioma dos computadores é o binário (Zeros e Ums), mas para nós, humanos, seria muito difícil escrever um monte de código usando só 0s e 1s.

Assim é necessário um idioma que tanto a máquina (computador) quando o ser humano (nós programadores) consigamos entender. É pra isso que foram criadas as **Linguagens de Programação**.

Sendo assim, quando você escreve uma receita de um programa em língua humana (como fizemos no nosso exemplo), precisamos TRADUZI-LA para uma linguagem que o computador entenda.

Isso que a Linguagem de Programação faz: **traduzir o que foi pensado em idioma humano para o idioma da máquina (computador)**.

A *linguagem de programação* que vamos usar no curso é o **JavaScript**, mas existem várias outras como:

- Python;
- PHP;
- Java;
- Ruby;
- C# (lê-se C-sharp);
- entre outras;

6. **Onde podemos escrever códigos (receitas) para o computador executar?**
   Quando vamos escrever uma receita para alguém geralmente fazemos isso onde? Num papel, às vezes num documento no Word ou coisa do tipo, certo?

   Para escrevermos comandos para o computador, usamos editores de textos simples (Bloco de Notas, por exemplo). Porém, atualmente, existem editores específicos para escrever código. Eles

são:

- Notepad++;
- Visual Studio Code (esse é o que vamos usar no curso);
- Sublime Text;
- e muitos outros;

Isso porque esses editores possuem recursos que nos ajudam a escrever código de uma forma mais rápida e com menos erros, já que muitas vezes esses editores nos avisam quando há algo errado (um código escrito de forma errada, por exemplo).

Por isso, usar editores profissionais para escrever nossos códigos é sempre recomendado.

Editores como o Word, que também permite escrever textos, **NÃO** servem para escrever código, já que quando o arquivo é salvo, ele é "contaminado" com vários outros códigos gerados automaticamente pelo Word, o qual o programador não consegue remover.

Sendo assim, novamente, SEMPRE USE UM EDITOR PROFISSIONAL DE CÓDIGO ou um editor simples como o BLOCO DE NOTAS, blz?!

7. **Linguagens: HTML, CSS e JavaScript.**
   Vimos anteriormente que a linguagem de programação é o mecanismo que nos permite traduzir as instruções da nossa receita para uma forma que o computador entenda, ou seja, para o idioma da máquina.

   Como o ambiente onde vamos testar nossos programas é o Navegador (Chrome, Firefox, Microsoft Edge, etc), vamos usar linguagens que são usadas para desenvolver sites/páginas de internet.

   A tríade que é usada para isso é composta por: HTML, CSS e JavaScript.

   Um programa é como se fosse o corpo humano. Isso fica mais

evidente quando nosso programa é um site.

O corpo humano possui uma estrutura interna (ossos e órgãos) que são definidos numa ordem e posição específica. Sem essa estrutura interna, o corpo não tem condições de existir e funcionar corretamente. Do mesmo jeito, um programa (site, aplicativo, etc) possui uma estrutura interna. No caso dos sites, quem permite criar essa estrutura é o **HTML**.

O **HTML** (HyperText Markup Language – *Linguagem de Marcação de Hipertexto*) é uma **linguagem estática** que permite definir a estrutura de um site. Ela define por exemplo: o que será um título, um parágrafo, onde haverá quebra de linha, onde será colocada uma imagem, etc.

Como ela é uma **linguagem estática**, uma vez que a estrutura é definida ela **NÃO MUDARÁ** até que o programador vá lá no código e altere alguma coisa OU até que uma **linguagem dinâmica** faça isso.

Ainda falando de corpo humano, uma vez que a estrutura do corpo está definida ela precisa ter algumas características melhor configuradas. Os órgãos tem um formato, uma cor, um tamanho. A pele pode ser negra, branca, parda. O cabelo pode ser crespo ou liso. As roupas que cobrem esse corpo podem ser as mais variadas.

Aqui, falamos de estética; de como o corpo será apresentado para as pessoas; isto é, falamos da aparência do corpo (programa).

No caso de sites, a linguagem usada para definir estilos (aparência) é o **CSS** – Cascading Style Sheets ou *Folhas de Estilo em Cascata*. Ela também é uma **linguagem estática**, como o HTML, mas a função dela é definir como os elementos do HTML (a estrutura) serão apresentados para o usuário (quais cores, quais fontes, quais bordas, etc).

Finalmente, definida a estrutura e a aparência o corpo precisa funcionar senão não tem muita serventia, concorda? Pode ser o corpo com os órgãos todos no lugar e muito bonito também.

Entretanto, se ele não se mexe, não pensa, não age de forma alguma, é apenas um “corpo morto” ainda que bem estruturado.

É aí que entra o **JavaScript**. O **JavaScript** é uma **linguagem dinâmica** que permite dar vida e lógica para o funcionamento do corpo (programa). Ou seja, da mesma forma que o nosso cérebro é o responsável por coordenar o funcionamento de tudo no nosso corpo, o **JavaScript** é o motor responsável por fazer toda essa estrutura servir para alguma coisa.

Por isso, diferente das outras duas linguagens que citamos ela nos permite escrever códigos mais complexos e que façam tarefas mais sofisticadas como: cálculos, animações, validações de dados, etc. Por isso, neste curso, o foco é o JS. 😊

8. **Extensões**
   Quando criamos um documento no Word e salvamos na área de trabalho do Windows por exemplo, você já percebeu que o ícone do nosso documento indica que ele é um arquivo do Word?

   O mesmo ocorre se você criar uma planilha no Excel. Quando você salvar o arquivo, o ícone indicará que ele é um arquivo que será aberto no Excel. Como o computador sabe qual programa usar para abri-lo?

   Toda vez que você salva um arquivo (seja ele qual for), um “sinalizador” é adicionado a ele. Esse sinalizador se chama **extensão**. No caso do Word, quando crio um arquivo chamado “Meu Currículo” e o salvo, ele é guardado no computador assim: Meu Currículo**.doc**

   Esse **.doc** (ou **.docx**) é o que chamamos de extensão e ele sinaliza que aquele arquivo é um arquivo do Word.

   No caso das planilhas (arquivos do Excel), a extensão pode ser **.xls** ou **.xlsx** (dependendo da versão do Office que é utilizada).

Ao longo do curso precisaremos criar arquivos que executem nossos códigos de exemplo. Como nossos códigos serão uma mistura de HTML e JavaScript e precisarão funcionar num navegador para testarmos (Chrome, Firefox, Edge, Opera, etc.) precisamos criar um arquivo que não apenas aceite os códigos que vamos escrever como também sejam executados apenas por navegadores isto é, devemos ajudar o computador a entender que toda vez que abrirmos nosso arquivo para testar nosso código, ele deve abri-lo num navegador qualquer.

Como vimos, sinalizamos isso usando extensões e a extensão que vamos utilizar é **.html**

Então, para testarmos todos os nossos códigos, vamos precisar sempre criar um arquivo **.html** e nele adicionar nossa lógica para que quando formos testar, o computador saber que ele deve abrir aquele arquivo no navegador que o usuário tiver configurado no sistema operacional dele.

9. **Variáveis**
   Muitas vezes, ao longo do nosso programa, vamos precisar armazenar informações em memória. Para realizar isso, criamos variáveis.

   Variável é uma espécie de gaveta onde guardamos alguma informação e a consultamos sempre que precisamos.

   Em JavaScript, criamos variáveis assim:

   ```
   var nome_da_variavel = "valor da variável";
   var minha_idade = 27;
   ```

Comandos HTML Básicos

- <meta charset="utf-8">
  Determina o conjunto de caracteres que a página usará. UTF-8

corrige as palavras com acento usadas no site.

- <h1>Meu texto</h1>
  Determina um texto como título da página

-
  Comando para quebra de linha no HTML

- Meu texto
  Comando para criação de parágrafos inteiros no HTML

-
  Comando para adicionar uma imagem no HTML.

- <script> ... Meu código JavaScript vai aqui ... </script>
  Comando que cria um bloco onde pode ser inserido código JavaScript.

- <!-- Comentário no HTML -->
  Comando para adicionar comentários no HTML. Lembre-se que comentários de código em programação *são textos que não são exibidos para o usuário do nosso programa, mas que ajudam o programador a entender o que o código faz e a lógica que está sendo aplicada ali quando lido muito tempo depois pelo próprio programador ou por outro programador que vá dar manutenção no código desenvolvido*.

Comandos JavaScript Básicos

- alert("Minha mensagem no Pop-up");
  Mostra uma janela/pop-up para o usuário com uma mensagem qualquer.

- prompt("Digite sua idade:");
  Mostra uma janela/pop-up para que o usuário digite alguma informação. O texto que vai nos parênteses é a mensagem que vai

ser exibida para o usuário na janela/pop-up.

- document.write(“ Meu texto num parágrafo ”);
  Permite escrever textos e código HTML diretamente na área branca do navegador.

- console.log(“Minha mensagem no console do navegador...”);
  Permite escrever mensagens no painel “Console” do navegador. Este painel só é acessível para programadores e geralmente é usado para realizar testes de código. Para ver o console basta apertar o botão **F12** no Chrome ou no Firefox.

- Math.round(7.5673);
  Arredonda um número qualquer para o inteiro mais próximo (para cima ou para baixo). Ex: o número 7.563 será arredondado para 8; enquanto 7.423 será arredondado para 7.

- Math.ceil(7.122);
  Arredonda um número qualquer para o inteiro mais próximo sempre para CIMA. Ex: o número 7.923 será arredondado para 8; assim como 7.122 também será.

- Math.floor(7.9999);
  Arredonda um número qualquer para o inteiro mais próximo sempre para BAIXO. Ex: o número 7.0233 será arredondado para 7; assim como 7.999 também será.

- parseInt(7.122);
  Remove as casas decimais e retorna o número inteiro. No caso do exemplo acima, será 7.

- // Comentário simples no JavaScript (em apenas 1 linha)
  /* Comentário longo no JavaScript (quando é necessário mais de 1 linha para explicar o código */
  Como no exemplo do HTML, os comentários existem para explicar como determinado código funciona visando ajudar o programador a

entender a lógica aplicada naquele lugar.