Programação Estruturada

Aula 05 - Comandos de iteração

Videoaula 01 - Entendendo o seu Código: depuração

Videoaula 01 - Entendendo o seu Código: depuração

Conteúdo interativo, acesse o Material Didático.

Olá, caro aluno!

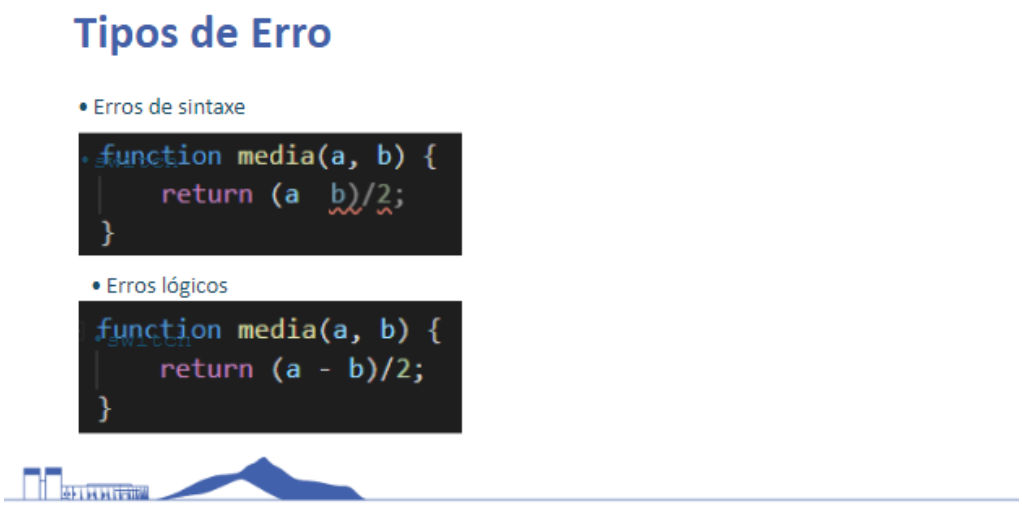
Nas aulas anteriores você viu que é normal cometermos erros durante a escrita de um programa. Até me arrisco a dizer que você já cometeu alguns erros enquanto realizava as atividades propostas desta disciplina.

Na aula de hoje, vamos falar sobre como identificar esses erros para resolvê-los de forma eficaz. De maneira resumida, os erros mais comuns são os **erros de sintaxe** e os **erros lógicos**.

A sintaxe de uma linguagem define como os seus elementos podem ser compostos. Por exemplo, JavaScript define que, para somar dois valores, devemos inserir o símbolo “+” entre eles. Outras linguagens poderiam ter uma sintaxe que define a soma de maneira diferente. Portanto, os erros de sintaxe acontecem quando esquecemos algum elemento da sintaxe do programa ou os colocamos em lugares não permitidos por ela.

A maioria dos erros de sintaxe são identificados automaticamente pelos editores de texto atuais, como é o caso do Visual Studio Code, o qual utilizamos nesta disciplina. No slide, você pode notar que a função `media` tem um erro de sintaxe marcado com um sublinhado vermelho. Se analisarmos o código, podemos observar que o símbolo “+” entre os argumentos `a` e `b` não foi inserido, o que motivou o erro no código.

Figura 1



Os erros lógicos de um programa são aqueles que causam um comportamento incorreto ou inesperado. Ou seja, esse tipo de erro acontece em tempo de execução e pode simplesmente produzir a saída incorreta ou fazer com que um programa trave durante a sua execução. Observe no slide que temos um exemplo de um erro lógico em que, ao invés de somarmos `a` e `b` e dividir o resultado por dois, dividimos a subtração entre eles por dois. Claramente, isso está errado. Assim, o resultado gerado por essa função não corresponderá ao que esperamos dela.

Existem diversos tipos de erro de programação que podem causar erros lógicos. Infelizmente, muitos desses erros são difíceis de encontrar. Pior ainda, muitas vezes em JavaScript, quando o código contém erros, nada acontece, não há mensagens de erro e você não obterá indicações de onde procurá-los.

No entanto, é claro que precisamos ter uma maneira de encontrar erros como estes que vemos no slide. Esse processo de testar, localizar e remover os erros em programas de computador chama-se **depuração**.

Apenas como uma curiosidade. Você sabe por que erro de programa é chamado de *bug* em inglês? Bem, *bug* significa inseto, e este realmente foi o primeiro erro conhecido. Naquela época os computadores eram enormes e havia um inseto preso nos componentes eletrônicos. Ou melhor... o que sobrou dele, né?! Acho que ele devia ter torrado lá dentro... Em homenagem ao finado inseto, o termo *bug* tornou-se popular para fazer referência aos erros em programas de computadores. E à palavra **depuração**, em inglês, deu-se o nome de ***debugging***.

Para que esse processo obtenha sucesso, é preciso haver uma forma de saber o que está acontecendo com o programa a cada passo. Isso pode ser feito de diversas maneiras, é o que você vai estudar nesta aula.

Para sabermos o que está acontecendo com o programa, podemos usar os métodos de saída da linguagem de programação ou uma ferramenta de depuração. Em JavaScript, temos pelo menos três formas de escrever saídas do programa:

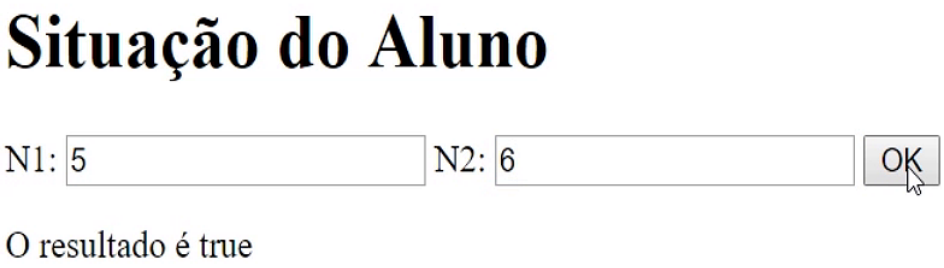
- Escrever em um elemento HTML usando o `innerHTML`
- Escrever na saída do HTML usando o método `document.write()`
- Escrever em uma caixa de alerta usando o método `window.alert()`

Note que esses métodos não precisam ser usados apenas para depuração, eles são métodos de saída dos programas JavaScript que podem ser usados normalmente para dar as saídas dos programas. Nesta disciplina, temos usado bastante o primeiro método, ou seja, temos alterado o valor do `innerHTML` de um campo texto.

Veja um exemplo de como usar esses métodos de saída para identificar erros. Vamos agora a nossa página HTML, onde temos dois campos para a primeira e para a segunda nota, vamos colocar aqui 5 e 6 e, ao clicar em OK, a intenção é que imprima na tela: “o resultado é true”, se a média for maior ou igual a 7, ou “o resultado é false”, se a média for menor que 7. Entre 5 e 6 a média é 5,5 e o que a gente espera que seja impresso na tela é: “o resultado é false”.

Então, clicando no ok, temos “o resultado é true”, claramente um erro na nossa página HTML. Vamos agora fazer uma depuração para tentar identificar onde é que está esse erro.

Figura 2



Utilizando o método mais básico, que é criar na página HTML um campo, um parágrafo, que eu vou chamar de depuração (linha 16), e a ideia é escrever neste campo os valores das variáveis que eu esteja querendo investigar.

Código 1 - Media.html

```
1  <html>
2    <head>
3      <meta charset="UTF-8" />
4      <title>Programação Estruturada - Aula 04</title>
5    </head>
6    <body>
7      <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
8
9      <h1>Situação do Aluno</h1>
10
11     N1: <input type="number" id="N1" value="">
12     N2: <input type="number" id="N2" value="">
13     <button onclick='situacao()'>OK</button>
14     <p id="resultado"></p>
15
16     <p id="depuracao"></p>
17
18     <script src="script.js"></script>
19   </body>
20 </html>
21
```

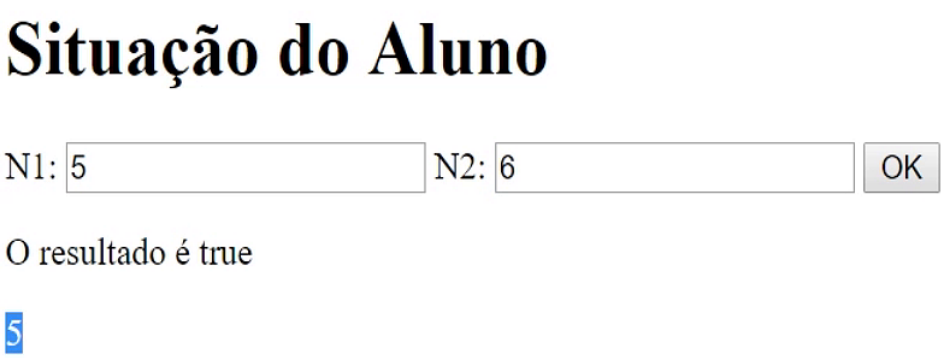
Vamos começar aqui escrevendo na linha 8, vou escrever no `innerHTML` desse campo depuração o valor de x, pra ver se eu estou lendo corretamente o valor de x.

Código 2 - Media.js

```
function situacao() {
    var saida = resultado();
    document.getElementById('resultado').innerHTML = "O resultado é " + saida;
}

function resultado() {
    var x = Number(document.getElementById('N1').value);
    document.getElementById('depuracao').innerHTML = x; // (1)
    //document.write(x);                                // (2)
    //window.alert(x);                                  // (3)

    var y = Number(document.getElementById('N2').value);
    var m = media(x,y);
    var r = m >= 7;

    return r;
}

function media(a, b) {
    var c = a + b/2;
    return c;
}

```

Vamos recarregar a tela, inserindo os valores que a gente estava testando (5 e 6), note que ele está lendo o valor de x (5) e escreveu aqui no campo de depuração o valor 5, então o valor de x está sendo lido corretamente.

Figura 3

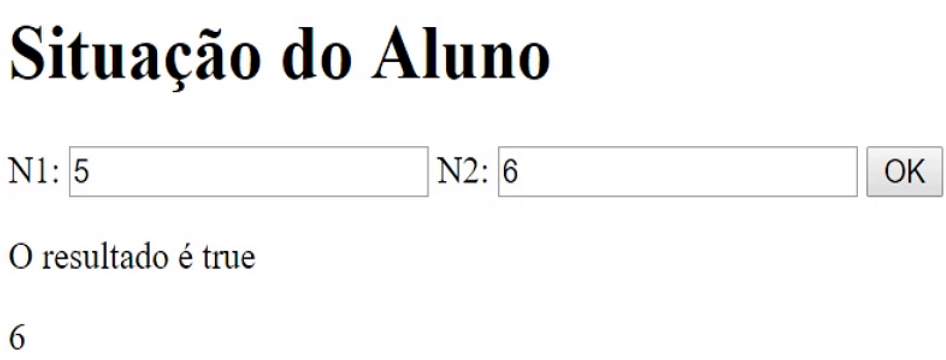


Vamos fazer a mesma coisa depois disso com o valor de y. Vou escrever agora o valor de y e ver se o valor de y está sendo lido corretamente, vamos colocar 6, e a gente vê que o valor de y está sendo lido corretamente, porque a gente está imprimindo o valor 6 na tela.

Código 3 - Media.js

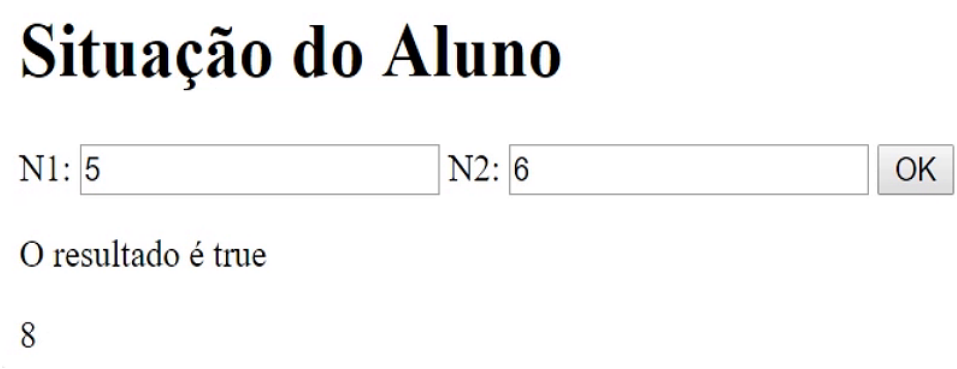
```
1  function situacao() {
2     var saida = resultado();
3     document.getElementById('resultado').innerHTML = "O resultado é " + saida;
4  }
5
6  function resultado() {
7     var x = Number(document.getElementById('N1').value);
8     //document.write(x);                                          // (2)
9     //window.alert(x);                                           // (3)
10
11    var y = Number(document.getElementById('N2').value);
12    document.getElementById('depuracao').innerHTML = y; // (1)
13
14    var m = media(x,y);
15    var r = m >= 7;
16
17    return r;
18 }
19
20 function media(a, b) {
21    var c = a + b/2;
22    return c;
23 }
24
```

Figura 4



Bom, o nosso próximo passo seria ver se a média está sendo calculada corretamente. Então vou escrever agora na tela o valor de `m` aqui na linha 15. Vou atribuir a `m` o valor do resultado da chamada da função de média com os valores que li, que eu sei que li corretamente (5 e 6), e vou escrever o resultado da chamada dessa função na tela usando aqui a variável m. Voltamos a nossa tela, vamos colocar o 5 e o 6, e note que a gente tem o resultado 8.

Figura 5



Opa, peraí! Essa média está errada, então claramente o erro está dentro da função `media`, porque ela está retornando para nós o resultado errado, está retornando 8 ao invés de 5,5. Então, vamos à página HTML colocar esse trecho de código, depois da atribuição da variável `c`, vamos escrever aqui a variável `c`, que é a variável dada como retorno, tá certo?

Voltando a nossa página HTML, vemos aqui que `c`, de fato, está com o valor 8, então, claramente, temos um erro aqui na linha 21, que está fazendo a atribuição da média para a variável `c`.

Código 4 - Media.js

```
function situacao() {
    var saida = resultado();
    document.getElementById('resultado').innerHTML = "O resultado é " + saida;
}

function resultado() {
    var x = Number(document.getElementById('N1').value);
    //document.write(x);                                    // (2)
    //window.alert(x);                                      // (3)

    var y = Number(document.getElementById('N2').value);

    var m = media(x,y);

    var r = m >= 7;

    return r;
}

function media(a, b) {
    var c = a + b/2;

    document.getElementById('depuracao').innerHTML = c; // (1)

    return c;
}

```

Bom, se a gente parar para analisar, o que está acontecendo aqui é um problema de precedência, onde a divisão tem uma precedência maior do que a soma, ou seja, ele está fazendo primeiro `b` dividido por 2 - 6 dividido por 2 dá 3 - para depois somar com o `a`, que é o 5 - 3 mais 5 = 8, tá? Então o erro está aí, o que a gente pode fazer para resolver esse problema é colocar os parênteses ao redor do `a + b`.

Código 5 - Media.js

```
1  function situacao() {
2      var saida = resultado();
3      document.getElementById('resultado').innerHTML = "O resultado é " + saida;
4  }
5
6  function resultado() {
7      var x = Number(document.getElementById('N1').value);
8      //document.write(x);                                    // (2)
9      //window.alert(x);                                     // (3)
10
11     var y = Number(document.getElementById('N2').value);
12
13     var m = media(x,y);
14
15     var r = m >= 7;
16
17     return r;
18 }
19
20 function media(a, b) {
21     var c = (a + b)/2;
22
23     document.getElementById('depuracao').innerHTML = c; // (1)
24
25     return c;
26 }
27
```

Voltando à página, vemos aqui o 5 e o 6, e agora o resultado ficou como “false”, e a média foi, de fato, corrigida. Tudo que falta fazer para voltar ao comportamento da página é comentar, ou simplesmente remover a linha 23, que é essa parte que escrevia no campo depuração, e aí nós temos o comportamento da nossa página conforme esperado, tá certo?

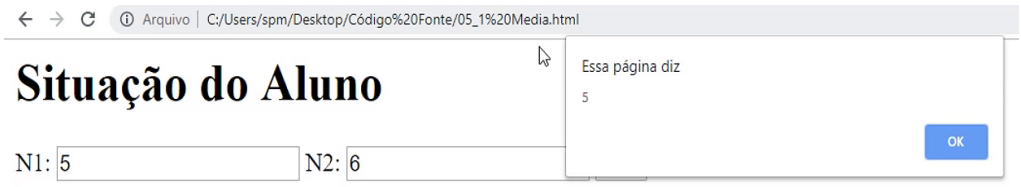
Mas existem outras maneiras de visualizarmos os valores das variáveis, diferentes do que fizemos agora nesse exemplo. Uma outra maneira de fazer isso é usar o método `document.writer`, que está aqui na linha 8, então vamos fazer o seguinte: vamos voltar para o começo do nosso exemplo. Vou inserir o erro novamente, na linha 21 tiro os parênteses, e vou descomentar aqui a linha 8, onde estamos usando o `document.writer`, para visualizar o valor de x. Vamos ver qual vai ser o comportamento disso aqui?

Eu vou atualizar minha página e usei `document.writer` para visualizar o valor de x, e o que acontece é que ele limpa toda tela e escreve nela apenas o que a gente quer ver, que é o valor de x que está sendo lido (5), e eu vou poder usar esse método para visualizar os valores de x, de y da média, da variável c, da mesma maneira que a gente fez usando a escrita desses valores na página HTML, ok?

Figura 6

Por fim, uma outra maneira que eu tenho de fazer isso é usando o `window.alert`, então estou deixando aqui na linha 8 agora o `window.alert(x)`, tá certo? Vamos recarregar a nossa página HTML, e vamos ver como é que a gente vai visualizar o valor de x ao ser executada aquela linha do `window.alert`. Vou dar OK, e o que acontece é a abertura de uma janela pelo Chrome dizendo “essa página diz: 5”, que é o valor de x.

Figura 7



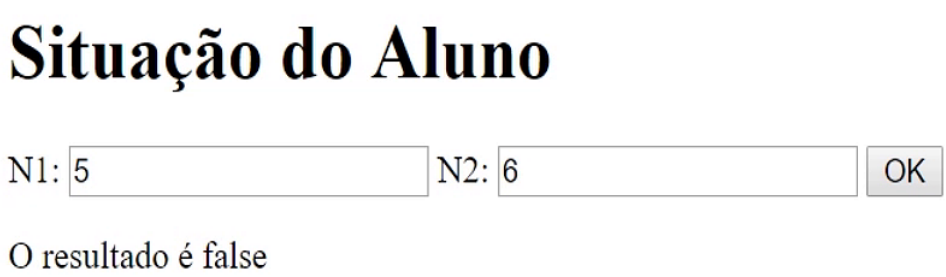
E aí, novamente, podemos dar o OK aqui, depois do OK a próxima linha, ou seja, o resto do programa é executado, e a gente escreve na tela “o resultado é true”. E aí poderemos fazer a mesma maneira de depuração até chegar dentro da função `media`, vai dá um `window.alert`, o valor c lá na linha 21, logo depois da atribuição a `c` do que a gente acha que seria a expressão para fazer a média.

Código 6 - Media.js

```
1  function situacao() {
2     var saida = resultado();
3     document.getElementById('resultado').innerHTML = "O resultado é " + saida;
4  }
5
6  function resultado() {
7     var x = Number(document.getElementById('N1').value);
8
9     var y = Number(document.getElementById('N2').value);
10
11    var m = media(x,y);
12
13    var r = m >= 7;
14
15    return r;
16 }
17
18 function media(a, b) {
19    var c = a + b/2;
20
21    window.alert(c);
22
23    return c;
24 }
25
```

Executamos essa página, a informação emitida será “Essa pagina diz 8” elogo depois vai escrever o true, e aí a gente teria da mesma maneira identificado que o problema estaria na linha 19. Poderia ser corrigido da mesma forma e a página voltaria a ter um comportamento correto.

Figura 8



Desse modo, a gente tem três maneiras de visualizar as saídas dos nossos programas:seja escrevendo em algum campo que a gente vê especificamente para isso no HTML, seja usando o método `document.writer`, seja usando o método `window.alert`, ok?