Programação Estruturada

Aula 09 - Comandos de Seleção

Videoaula 04: Switch

Videoaula 04: Switch

Anteriormente, apresentei o comando `if-else`, o qual pode ser utilizado para resolver problemas de seleção. Porém, em alguns casos, o uso desse comando, quando encadeado, torna a escrita e a compreensão do código mais trabalhosa. Por exemplo, considere a função apresentada no slide (Figura 1) que retorna o nome do dia da semana passado como número. Lembre-se que em JavaScript os dias da semana são representados como números no intervalo que vai de 0 (para o domingo) até 6 (para o sábado). Nessa função, comparamos o valor recebido primeiramente com 0. Em caso afirmativo, atribuímos o texto `"Domingo"` à `resposta`. Caso contrário, comparamos o valor recebido com 1. Em caso afirmativo, atribuímos o texto `"Segunda-feira"` à `resposta`. Caso o valor recebido não seja 1, continuamos encadeando `if`'s até chegarmos na comparação com o 6.

Figura 1 - Comando de Seleção `if` encadeado

```
function dia_da_semana(dia) {
    var resposta = "Dia Inválido";
    if (dia == 0) {
        resposta = "Domingo";
    } else if (dia == 1) {
        resposta = "Segunda-feira";
    } else if (dia == 2) {
        resposta = "Terça-feira";
    } else if (dia == 3) {
        resposta = "Quarta-feira";
    } else if (dia == 4) {
        resposta = "Quinta-feira";
    } else if (dia == 5) {
        resposta = "Sexta-feira";
    } else if (dia == 6) {
        resposta = "Sábado";
    }
    return resposta;
```

Código 1 - 09_10 Dia da Semana com if.html e 09_10 Dia da Semana com if.js

```
<html>
  <head>
```

```
    <meta charset="UTF-8" />
    <title>Programação Estruturada - Aula 09</title>
  </head>
  <body>
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Dia da Semana Usando if</h1>

    <p id="resultado"></p>

    <script src="script.js"></script>
  </body>
</html>

```

```
var data = new Date(1969, 06, 20);
var dia = data.getDay();

document.getElementById("resultado").innerHTML = dia_da_semana(dia);

function dia_da_semana(dia) {
  var resposta = "Dia Inválido";
  if (dia == 0) {
    resposta = "Domingo";
  } else if (dia == 1) {
    resposta = "Segunda-feira";
  } else if (dia == 2) {
    resposta = "Terça-feira";
  } else if (dia == 3) {
    resposta = "Quarta-feira";
  } else if (dia == 4) {
    resposta = "Quinta-feira";
  } else if (dia == 5) {
    resposta = "Sexta-feira";
  } else if (dia == 6) {
    resposta = "Sábado";
  }
  return resposta;
}

```

O comando `switch` facilita a escrita de trechos de programa como este em que a seleção deve ser feita entre várias alternativas. O `switch` é chamado de comando interno de seleção múltipla, ele compara o valor de uma expressão com uma lista de valores. No slide (Figura 2), podemos ver a estrutura geral desse comando, o qual funciona da seguinte maneira: a expressão passada para o comando é avaliada uma

vez. Esse valor é comparado, em ordem, com os valores das constantes especificadas nos comandos `case`. Quando ocorrer uma condição em que a expressão é igual, a sequência de comandos associada ao `case` em questão será executada até chegar ao comando `break`, que para a execução do `case` em questão e salta para a próxima linha de código após o bloco do `switch`. O comando `default` será apenas executado se nenhum valor for igual ao valor da expressão passada para o comando. O comando `default` é opcional. Se ele não existir, nenhuma ação será realizada caso todos os testes falhem.

Figura 2 - Comando `switch`

```
switch(expressão) {
     case valor1:
       // bloco de código
       break;
     case valor2:
       // bloco de código
       break;
     …
     default:
       // bloco de código
 }
// resto do programa
```

Vamos ver como fica a mesma função usando o `switch`?

Nesse exemplo (ver Código 2), usaremos novamente uma página muito simples em que temos apenas um parágrafo cujo identificador é `"resultado"` e escreveremos nesse resultado, ok? Você aprendeu na aula sobre datas que a primeira vez que o homem pisou na lua foi no dia 20 de julho de 1969, era um domingo. Usaremos esse dia histórico no nosso exemplo.

Código 2 - 09_11 Dia da Semana com switch.html

```
1 <html>
2   <head>
3     <meta charset="UTF-8" />
4     <title>Programação Estruturada - Aula 09</title>
5   </head>
6   <body>
```

```
    <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>

    <h1>Dia da Semana Usando switch</h1>

    <p id="resultado"></p>

    <script src="script.js"></script>
  </body>
</html>

```

```
var data = new Date(1969, 06, 20);
var dia = data.getDay();

document.getElementById("resultado").innerHTML = dia_da_semana(dia);

function dia_da_semana(dia) {
  var resposta = "";
  switch (dia) {
    case 0:
      resposta = "Domingo";
      break;
    case 1:
      resposta = "Segunda-feira";
      break;
    case 2:
      resposta = "Terça-feira";
      break;
    case 3:
      resposta = "Quarta-feira";
      break;
    case 4:
      resposta = "Quinta-feira";
      break;
    case 5:
      resposta = "Sexta-feira";
      break;
    case 6:
      resposta = "Sábado";
      break;
    default:
      resposta = "Dia Inválido";
  }
  /*
  switch (dia) {
    case 0:
    case 6:
      resposta = resposta + ". Fim de Semana.";

```

```
        break;
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        resposta = resposta + ". Dia Útil.";
        break;
  }
  */
  return resposta;
}

```

Temos a criação de uma variável `data` com os campos do ano de 1969. O mês de julho é o mês 06, lembre-se que em JavaScript a numeração dos meses começa em 0, então 0 é janeiro, logo julho vai ser o mês 6 e o dia 20. Pegamos o dia dessa data, usando o método `getDay`, em seguida passamos esse dia, ou melhor, esse número que corresponde ao dia da semana dessa data. Passaremos para a função `dia_da_semana`, que vai retornar o texto em português, com o nome em português daquele dia da semana. E essa função, que é o que nos interessa, foi declarada usando o `switch`. Veja a Figura 3.

Figura 3 - Comando `switch` no Exemplo

Ela recebe o número associado ao argumento `dia` e faz um `switch` no valor dessa variável chamada `dia`. Temos o `case` se o valor for 0, diz que a resposta é `"Domingo"` e temos que dar o `break` para sair do bloco do `switch`. Caso seja 1, é `"Segunda-feira"`, caso seja 2, é `"Terça-feira"` e assim por diante até chegar no 6, que é o `"Sábado"`, na linha 28, depois temos o `break`, e, por fim, nas linhas 30 e 31, um caso `default`. Então se o valor passado não for 0, 1, 2, 3, 4, 5 ou 6, cairemos no valor `default` e a resposta que será retornada é `"Dia Inválido"`. Veja a Figura 4.

Figura 4 - Caso `default`

```
9         case 0:
10            resposta = "Domingo";
11            break;
12        case 1:
13            resposta = "Segunda-feira";
14            break;
15        case 2:
16            resposta = "Terça-feira";
17            break;
18        case 3:
19            resposta = "Quarta-feira";
20            break;
21        case 4:
22            resposta = "Quinta-feira";
23            break;
24        case 5:
25            resposta = "Sexta-feira";
26            break;
27        case 6:
28            resposta = "Sábado";
29            break;
30        default:
31            resposta = "Dia Inválido";
```

Como já foi dito, esse dia histórico aconteceu no Domingo e, ao carregar a página HTML, de fato, o texto que aparece é "Domingo", conforme esperado, como exposto na Figura 5.

Figura 5 - Exibição da Página HTML

Dia da Semana Usando switch

Domingo

Agora, iremos à linha 4 e passaremos o valor 9, por exemplo. Mas, no nosso exemplo, o `switch` só faz `case` de 0 a 6, então o que é que acontece ao passar esse valor? O resultado será "Dia Inválido", como exposto na Figura 6, porque ele comparou esse valor com 0, 1, 2, 3, 4, 5, 6, e ele não é igual a nenhum desses valores, então ele caiu no valor `default`.

Figura 6 - Exibição da Página HTML do Caso `default`

Por fim, veremos o que acontece se retirarmos o caso `default`. Apagaremos as linhas 30 e 31 e passaremos o valor 9, como foi feito na linha 4. A função vai comparar esse valor com 0, 1, 2 3, 4, 5, 6, e será diferente em todos os casos. Não teremos um caso `default`, e o comando `switch` não fará absolutamente nada.

Como a resposta começou com o valor vazio na linha 7, vai ser retornado à *string* vazia. Se formos para a página HTML, nada será exibido, porque a função retorna a *string* vazia, como apresentado na Figura 7.

Figura 7 - Exibição da Página HTML sem um Caso `default`

Mais um detalhe que eu queria mostrar nesse exemplo é o caso em que mais de um `case` compartilha o mesmo trecho de código. Manteremos a versão original colocando o `dia` na chamada da função `dia_da_semana` e teremos o `switch` que dirá qual foi o dia da semana em português, se aquele valor corresponde, e criaremos um novo `switch` que faz o seguinte: para os `case 0` e `case 6`, ele executa a linha de código: `resposta = resposta + ". Fim de Semana"`. Então 0 é o `"Domingo"` e 6 é

o "Sábado". Ele vai colocar ao final do texto concatenado à resposta que nós já começamos a criar no switch anterior o texto "Fim de Semana" Caso contrário, se for case 1, 2, 3, 4, 5, veremos que será acrescentada à resposta o texto "Dia Útil".

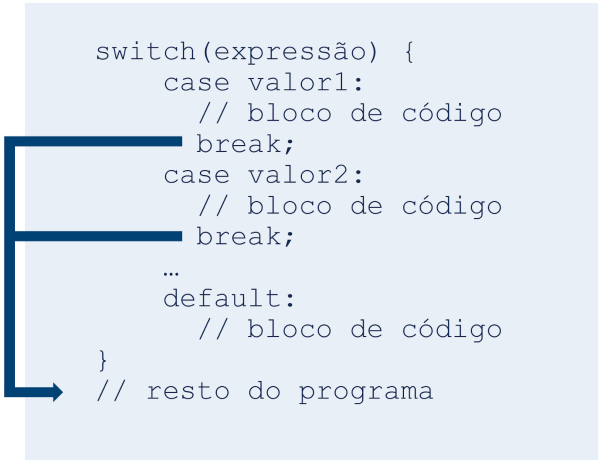
Faça o teste e verifique se o Domingo aparece como fim de semana. Você pode também voltar e colocar um dia da semana, coloque o 5, por exemplo, que é uma sexta-feira. O resultado deverá ser: "Sexta-feira. Dia Útil". Nesse exemplo, tivemos um switch normal, mas também temos um switch em que vários cases compartilham do mesmo código, como o que acontece nas linhas 34 e 35, exposto na Figura 8.

Figura 8 - Comando switch com casos compartilhados

```
33    switch (dia) {
34        case 0:
35        case 6:
36            resposta = resposta + ". Fim de Semana.";
37            break;
38        case 1:
39        case 2:
40        case 3:
41        case 4:
42        case 5:
43            resposta = resposta + ". Dia Útil.";
```

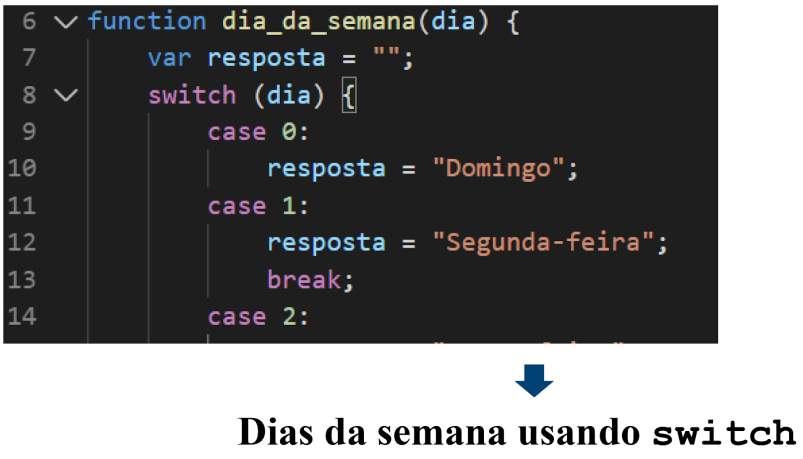
O comando switch possui algumas peculiaridades que merecem a nossa atenção. Primeiro, quando o JavaScript alcança o comando break, ele sai do bloco do switch. Isso interromperá a execução dentro do bloco. Note que não é necessário usar o break no último caso do bloco, pois ele termina lá de qualquer maneira. Além disso, se você omitir a instrução break, o próximo caso será executado mesmo que a avaliação não seja correspondente. Veja a Figura 9.

Figura 9 - Comando `break`



No nosso exemplo, se esquecermos de colocar o `break` no caso 0, como vemos no slide (Figura 10), a página exibirá o texto "Segunda-feira". Isso porque, como esquecemos o `break`, a atribuição da linha 12 será executada e apenas o `break` da linha 13 fará com que a execução saia do bloco `switch`.

Figura 10 - Esquecendo o `break`



Outro ponto interessante é que o caso `default` não precisa ser o último. No entanto, caso ele não seja o último, como podemos ver no slide (Figura 11), você deverá usar o `break` da mesma maneira que precisa usar nos outros casos.

Figura 11 - Localização do Comando `default`

```
 8       switch (dia) {
 9           default:
10               resposta = "Dia Inválido";
11               break;
12           case 0:
13               resposta = "Domingo";
14               break;
15           case 1:
16               resposta = "Segunda-feira";
17               break;
```

Além disso, casos diferentes podem compartilhar o mesmo trecho de código. No exemplo (Figura 12), os casos 0 e 6 compartilham o mesmo bloco de código, ou seja, a atribuição da linha 37. Por outro lado, os casos 1, 2, 3, 4 e 5 compartilham outro bloco de código, ou seja, a atribuição da linha 44.

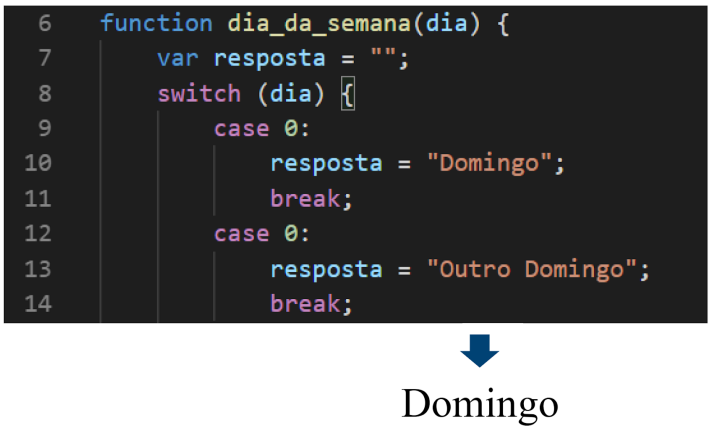
Figura 12 - Compartilhando Comandos

```
34  switch (dia) {
35      case 0:
36      case 6:
37          resposta = resposta + ". Fim de Semana.";
38          break;
39      case 1:
40      case 2:
41      case 3:
42      case 4:
43      case 5:
44          resposta = resposta + ". Dia Útil.";
45          break;
46  }
```

Você deve estar se perguntando o que acontece se o valor for igual em mais de um caso. Nessa situação, apenas o primeiro caso será selecionado. Veja no slide (Figura 13) que, se tivermos dois casos com o valor 0, apenas o primeiro caso é selecionado. Dessa forma, apenas a atribuição da linha 10 foi executada e o texto "Domingo" foi exibido na

página. Note que a atribuição da linha 13 NÃO foi executada. E lembre-se, se nenhum caso tiver o valor igual ao da expressão passada, o caso `default` será executado. Se ele não existir, o comando `switch` terminará sem fazer nada.

Figura 13 - Casos com o mesmo valor



Por fim, é muito importante saber que JavaScript usa a comparação estrita, ou seja `===`, entre o valor da expressão e os valores dos casos. Você deve lembrar que essa comparação apenas é verdade quando o valor e o tipo das expressões são iguais. Por exemplo, a comparação do texto `"0"` com o número 0 usando a comparação normal, ou seja, `"0" == 0`, retorna `true`. No entanto, a função apresentada no slide (Código 3) faz a declaração de uma variável chamada `dia`, cujo valor é o texto `"0"` e retorna o texto `"Outro dia"`. Note que esse é o valor atribuído à saída no caso `default` e não no caso `0`, pois a comparação estrita `"0" === 0` retorna `false` porque as expressões têm tipos diferentes, *String* e Número.

Código 3 - 09_12 Comparação Estrita do Switch.html e 09_12 Comparação Estrita do Switch.js

```
1 <html>
2   <head>
3     <meta charset="UTF-8" />
4     <title>Programação Estruturada - Aula 09</title>
5   </head>
6   <body>
7     <noscript>Seu navegador não suporta JavaScript ou ele está desabilitado.</noscript>
8
```

```
    <h1>Comparação Estrita do Switch</h1>

    <p id="resultado"></p>

    <script src="09_12 Comparação Estrita do Switch.js"></script>
  </body>
</html>

```

```
document.getElementById("resultado").innerHTML = dia_da_semana();

function dia_da_semana(dia) {
  var dia = "0";
  var resposta = "";
  switch (dia) {
    case 0:
      resposta = "Domingo";
      break;
    default:
      resposta = "Outro dia";
  }
  return resposta;
}

```

Encerramos esta aula por aqui. Nela, você conheceu os comandos de seleção `if` e `switch`. Você deve ter notado que eles são ligeiramente diferentes: o `switch` testa igualdade, enquanto o `if` testa uma determinada condição lógica, ou seja, uma expressão que pode assumir valor verdadeiro ou falso.

Você deve ter observado também que o comando `if` pode vir associado ao `else`, no caso de duas situações complementares (quando uma for verdadeira, a outra é falsa e vice-versa). Além disso, comandos `if-else` podem ser encadeados para melhorar a legibilidade. No entanto, note que em uma série de comandos `if` todas as condições serão verificadas mesmo se uma correspondência tiver sido encontrada. Dessa forma, para algumas situações como a que apresentamos no início da aula, onde tínhamos dois blocos `if` em sequência comparando a nota `m` com o valor 7, o desempenho desse método é mais lento que o do `switch`.

Mais uma vez, deixaremos uma lista de atividades. É muito importante praticar bastante o uso dos comandos de seleção, pois eles são fundamentais na programação. Não deixe de fazê-los. Até a próxima aula, tchau!