01

Repetir enquanto...

Transcrição

Começando deste ponto? Você pode fazer o DOWNLOAD (https://github.com/alura-cursos/logica-de-programacao-I/archive/aula-7.zip) do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Desde 1930 é disputada a Copa do Mundo de Futebol e a cada edição ela possui uma sede diferente. A periodicidade do evento é de 4 em 4 anos, então, será que 2030 é ano de copa do mundo?

Uma maneira de saber isso é tomar por base um ano em que ocorre copa, por exemplo, 2014, e somar 4 anos. Logo sabemos que 2018 também é ano de copa e se continuarmos nesse processo o próximo ano será 2022, em seguida 2026 e... Pronto! Chegamos ao ano de 2030 e descobrimos que sim, haverá copa do mundo! **Podemos continuar esse processo infinitamente, enquanto não nos cansarmos**.

Nessa aula o objetivo é escrever um programa que mostre os anos em que ocorrem copa, desde 1930. Faremos isso de uma forma extremamente simples. Basta criar uma variável cujo valor seja 1930 e que sempre tenha o valor aumentado de 4 em 4. Por exemplo:

```
var anoCopa = 1930;
alert(anoCopa + " tem copa!");

anoCopa = anoCopa + 4;
alert(anoCopa + " tem copa!");

anoCopa = anoCopa + 4;
alert(anoCopa + " tem copa!");

anoCopa = anoCopa + 4;
alert(anoCopa + " tem copa!");
```

Poderíamos continuar repetindo esse código até cansarmos. Mas repare o trabalho que teríamos. E se quisermos calcular um valor mair de anos de copas? Nesse caso, imagine a quantidade de código que deve ser escrito. É muito trabalho repetitivo! Vamos tentar uma abordagem um pouco diferente.

Queremos imprimir os anos da copa, de 4 em 4, para saber todas as copas. Será possível? A ideia é criar um programa que some `4` na variável `anoCopa`, nos alerte esse valor e volte a executar o mesmo procedimento.

Seguindo esse raciocínio, vamos criar um novo programa, no arquivo `anos_de_copa.html`, que mostrará os anos em que teve e ainda terá copa do mundo. Vamos fazer o cálculo com 1930 como ano inicial, teremos:

```
<script>
var anoCopa = 1930;

while(true) {
    alert(anoCopa + " tem copa!");
    anoCopa = anoCopa + 4;
}
</script>
```

Utilizamos a variável `anoCopa` iniciando no ano de 1930, que foi o primeiro ano do evento. Em seguida, usamos o comando `while` (*enquanto*), que significa que os comandos dentro das `{ }` serão realizados quantas vezes quisermos, como uma **repetição**. O fenômeno da repetição também é bastante conhecido como **iteração**.

Nesse caso, o que fizemos foi mudar o conteúdo da variável `anoCopa` para ir 4 anos adiante, isto é, simplesmente somamos 4 no `anoCopa` e guardamos o resultado na mesma variável.

Quando executamos o programa, vemos uma mensagem de alerta aparecendo na tela. Se seguirmos clicando em *OK*, veremos que para cada ano de copa, um novo alerta é mostrado. Se tivermos paciência para clicar várias vezes, vamos até descobrir que não haverá copa no ano de 2100! Repare que podemos clicar infinitamente em OK que uma mensagem sempre será exibida. Ou seja, ficamos em uma repetição infinita, também conhecida como **loop infinito**.

Cuidados com alerts e loops infinitos em browsers

Para cada novo ano de Copa do Mundo ser exibido, uma mensagem de alerta é mostrada. Isso faz com que muitas mensagens sejam exibidas sem parar.

Alguns navegadores, como o Firefox e o Chrome, permitem a exibição de mensagens em sequência, quando uma primeira mensagem abre logo após a segunda aparece. No entanto, o Internet Explorer não faz o mesmo, ou seja, seu navegador exibirá as mensagens sem parar até que o fechamento da janela seja forçado.

Apesar de interessante, estamos mostrando mais informações do que o necessário. Principalmente, se nosso objetivo for mostrar **apenas** os anos em que o evento aconteceu no intervalo de 1930 **até** 2014, ano da Copa no Brasil.

Em português é frequente aparecer o termo **laço** ao em vez de loop. Também é comum afirmar que o laço **itera** (repete) determinado trecho de código que queremos. Lembre-se que já havíamos nomeado as repetições como iterações.

O `while()`, de forma similar ao `if()`, aceita a indicação de uma condição, em nosso caso, para cada repetição uma avaliação deve ser feita para saber se o conteúdo da `loop` (repetição), isto é, o bloco de código dentro dele, deve ser executado ou não. Na seção anterior utilizamos apenas o `while(true)`, indicando que o bloco de código deve sempre repetir.

Agora queremos melhorar e evitar o `loop` infinito. Desejamos que o bloco seja executado somente **enquanto `anoCopa` for menor ou igual a 2014**.

```
<script>
var anoCopa = 1930;

while(anoCopa <= 2014) {
    alert(anoCopa + " tem copa!");
    anoCopa = anoCopa + 4;
}
</script>
```

Dessa forma indicamos para o `while` que ele deve, a cada repetição, verificar se o valor da variável `anoCopa` ainda é menor que 2014. Quando o valor `anoCopa` for maior que 2014, a condição do `while` não será satisfeita e o bloco não será executado pulando para o próximo comando.

Abra em seu navegador o arquivo `anos_de_copa.html` e perceba que ele mostrará um primeiro alerta para 1930. Quando clicamos em *Ok* o programa nos mostra 1934 e assim até chegar em 2014. Porém, mostrar `alerts` deve estar começando a

ficar desagradável para você, certo? Se ainda não ficou, logo ficará! Então vamos começar a usar a função `mostra`, a mesma que utilizamos em aulas anteriores, no capítulo * (#comunicandose).

Primeiro, vamos reescrever o código da função `mostra`. Antes criávamos a função `pulaLinha`, por uma questão didática, dessa vez colocaremos as duas ações na mesma função: chamar o `document.write` e pular linhas por meio do `"<br/>"`. Teremos:

```
function mostra(frase) {
    document.write(frase + "<br/>");
}
```

Agora troque a linha onde você estava usando o `alert` para usar o `mostra`.

```
mostra(anoCopa + " tem copa!");
```

Podemos inclusive mostrar uma mensagem indicando que acabou:

```
<script>
var anoCopa = 1930;

while(anoCopa <= 2014) {
    mostra(anoCopa + " tem copa!");
    anoCopa = anoCopa + 4;
}

mostra("Ufa! Esses foram os anos de copa até 2014.");
</script>
```

Você pode pensar no `while()` como o elemento que a cada repetição pergunta: *Posso continuar*? A condição dentro dos parênteses (no caso o `anoCopa <= 2014`) será utilizada para decidir se o `while()` continua ou não. Se a resposta for `true`, significa que pode repetir o bloco mais uma vez, caso contrário, ele deve parar as repetições e ir para o próximo comando.

Vamos relembrar o que fizemos no arquivo `anos_de_copa.html`. Nosso objetivo era criar um programa que facilitasse ao usuário descobrir quais são os anos em que haverá ou houve copa do mundo. Fizemos a definição da função `mostra`, que exibe os anos no navegador, e inserimos a inicialização da variável `anoCopa` em 1930, pois foi quando ocorreu o evento pela primeira vez. O código está da seguinte maneira:

```
<script>
function mostra(frase) {
    document.write(frase + "<br/>");
}

var anoCopa = 1930;
```

Também imprimimos os anos de copa até **2014**. Lembrando que as copas são realizadas de 4 em 4 anos.

```
while(anoCopa <= 2014) {
    mostra(anoCopa + " tem copa!");
    anoCopa = anoCopa + 4;
}
</script>
```

No final de tudo, pudemos mostrar que chegamos ao fim da exibição dos anos de copa. Quando a condição no `while` não for mais verdadeira o programa executa o código logo após o bloco do `while` , ou seja, vai para depois do fechamento das chaves.

Ainda podemos adicionar uma linha para que o programa mostre uma mensagem, logo após o bloco do `while` teremos.

```
mostra("Ufa! Esses foram os anos de copa até 2014.");
```

Nesse momento, nosso programa pergunta qual deve ser a data limite para mostrar os anos de copa do mundo. Temos o seguinte código:

```
<script>
function mostra(frase) {
    document.write(frase + "<br/>");
}

var anoCopa = 1930;
var limite = prompt("Qual é o ano limite?");

while(anoCopa <= limite) {
    mostra(anoCopa + " tem copa!");
    anoCopa = anoCopa + 4;
}
mostra("Ufa! Esses foram os anos de copa até " + limite);
</script>
```

Nosso programa mostra os anos a partir de 1930. Podemos ir além e deixá-lo flexível para que também pergunte qual deve ser o ano em que começa a amostragem. Por exemplo, podemos mostrar apenas as copas do século XXI - entre os anos 2001 e 2100. Assim, vamos parar de iniciar a variável em 1930 e passar a perguntar qual o ano inicial para o cálculo. Teremos:

```
var anoCopa = prompt("Informe o ano inicial");
```

Quando executamos o programa, com ano inicial em 1930 e ano limite em 2014, algo errado ocorre! O que é mostrado no navegador é apenas o primeiro ano, os demais não são exibidos. O programa ficou maluco? Não! Vamos explicar o que ocorre com muita calma!

Primeiro, precisamos entender o que está acontecendo! Será o primeiro momento no qual entraremos mais a fundo em como o JavaScript funciona. É normal que cada linguagem de programação tenha características não tão óbvias a primeira vista e faz parte do aprendizado encará-las.

Quero saber porque não saiu como eu esperava!

Para começarmos a entender o motivo da falha, precisamos analisar o que o programa fez. Repare que na primeira vez ele conseguiu entrar no `while`, tanto que mostrou a mensagem desejada: `1930 é ano de copa!`. Com isso, já sabemos que a condição do `while` foi verdadeira logo no começo, o que faz sentido, pois, 1930 é menor que 2014, ano limite que informamos.

Porém ainda não entendemos o motivo de não ter continuado, já que estamos somando 4 no ano inicial e com isso ele deveria ter mostrado o 1934.

Vamos mostrar na tela qual é o valor do ano após a soma, dessa forma, poderemos visualizar o que está sendo comparado. Para isso, adicione como última instrução do bloco `while` a linha que mostra o conteúdo da variável `anoCopa`, assim, saberemos o seu conteúdo após a soma.

```
while(anoCopa <= limite) {
    mostra(anoCopa + " tem copa!");
    anoCopa = anoCopa + 4;
    mostra(anoCopa + " esse é o valor após a soma!");
}
```

Abrindo esse arquivo novamente no navegador teremos uma grande surpresa!

O número resultante da soma não é `1934` e sim `19304`? Como assim?! `1930 + 4 = 19304` ??? Nosso programa não somou 4. Ele juntou o texto do começo, ou seja, ele realizou a **concatenação**. Aquele procedimento que aprendemos no passado.

Pelo menos encontramos o problema! Agora precisamos entender o porquê disso acontecer.

Porque a concatenação e não a soma?

A explicação é simples. Ao utilizar o `prompt`, o JavaScript considera que digitamos na caixa um texto, ou seja, uma `string`.

Lembre-se que aprendemos que para somar um texto, o conteúdo acabava sendo concatenado? Por isso, se fizermos `var teste = "1930" + 4;` e mostrarmos a variável `teste` veremos o resultado `19304`. E como o número `19304` é certamente maior que `2014`, então, o `while` para de executar as repetições. O `loop` para logo depois da primeira execução. Faz sentido! Mas não era o que queríamos, certo?

Como resolvo? Não queria concatenar!

Percebemos que o problema acontece por tentar somar um número com um texto (uma `string`). Se queremos fazer uma soma, precisamos que as duas variáveis contenham valores que sejam números, assim, na linguagem de programação falamos que elas devem ser do **tipo** número.

Não somos os primeiros a passar por esse problema. Justamente por isso o JavaScript possui uma função própria chamada `parseInt()`, cujo objetivo é "transformar" texto em número. Dessa forma podemos fazer as operações matemáticas corretamente.

Para fazer um teste, vamos experimentar construir uma `string` que contenha apenas números. Somando-a com um número, ocorrerá a concatenação. Passando essa variável para a função `parseInt` o resultado será um valor numérico, permitindo a correta manipulação matemática. Repare:

```
var anoComoTexto = "1930";
alert(anoComoTexto + 4);

var ano = parseInt(anoComoTexto);
alert(ano + 4);
```

Agora podemos aplicar a função para resolver o problema do código. Primeiramente, vamos continuar perguntando para o usuário qual o ano limite que ele está usando, lembrando que isso é devolvido como uma `string` e precisamos transformá-la em um número:

```
var anoComoTexto = prompt("Informe o ano inicial");
var anoCopa = parseInt(anoComoTexto);
```

Com essa simples alteração, nosso programa volta a estar pronto para funcionar no navegador.

O programa `anos_de_copa.html` estava funcionando corretamente até o momento em que decidimos perguntar para o usuário qual é o ano em que o programa deve começar a mostrar os anos da copa. Observe como está o código:

```
<script>
function mostra(frase) {
    document.write(frase + "<br/>");
}

var AnoCopa = prompt("Informe o ano inicial");
var limite = prompt("Qual é o ano limite?");

while(AnoCopa <= limite) {
    mostra(AnoCopa + " tem copa!");
    AnoCopa = 4 + AnoCopa;
}
mostra("Ufa! Esses foram os anos de copa até " + limite);
</script>
```

Mas, quando abrimos o arquivo no navegador e informamos 1930 como ano inicial e 2014 como ano limite tivemos uma surpresa, ele não funcionou como o esperado. Isso ocorreu, pois os valores foram concatenados em vez de somados.

Como precisávamos somar número com número e não número com texto resolvemos o problema por meio da função `parseInt()`. Ela transforma o texto que recuperamos em número:

```
var anoComoTexto = prompt("Informe o ano inicial");
var AnoCopa = parseInt(anoComoTexto);
```

Nesse momento, o código completo do programa é o seguinte:

```
<script>
```

```
function mostra(frase) {
    document.write(frase + "<br/>");
}

var anoComoTexto = prompt("Informe o ano inicial");
var AnoCopa = parseInt(anoComoTexto);
var limite = prompt("Qual o ano limite?");

while(AnoCopa <= limite) {
    mostra(AnoCopa + " é ano de copa!");
    AnoCopa = 4 + AnoCopa;
}
mostra("Ufa! Esses foram os anos de copa até " + limite);
</script>
```

Agora podemos abrir o programa `anos_de_copa.html` no navegador, que ele nos perguntará o ano inicial e o limite e fará todo o processo corretamente.