Compilado - Aula 05

- Quando precisamos desenvolver uma aplicação, muitas vezes o comportamento das funcionalidades muda dependendo de alguma condição. Por exemplo: se um site é proibido pra menores de idade, dependendo da idade da pessoa devemos deixar ou não ela entrar
- O algoritmo que queremos então é fazer algo como:

```
se alguma_condicao fazer tal coisa senão fazer outra coisa
```

- Pra implementar esse algoritmo em JS é bem parecido, basta usar a sintaxe do `if/else`. Por exemplo, se quisermos responder se alguém é maior ou menor de idade poderíamos fazer:

```
function verificar(idade) { if(idade < 18) { return "É menor de idade"; } else { return "É maior de idade"; } }
```

- O `if` ("se" em inglês) serve pra, dada alguma condição, executar ou não o código dentro dele:

```
if(...) { // esse código só será executado caso a condição seja VERDADEIRA }
```

- Já o `else` ("senão em inglês") pode ser usado logo após o `if` pra executar algum outro código caso a condição dele não seja verdadeira.

```
if(...) { ... } else { // esse código só será executado caso a condição do if seja FALSA }
```

- Nem todo if precisa ter um else. Caso você não queira executar nada no caso contrário, basta utilizar somente o `if`, sem o `else { ... }` depois
- Caso você queira checar várias condições de uma vez, você pode encadear vários ifs dessa forma:

```
function verificarMedia(media) { if(media >= 7) { return "Aprovado"; } else if(media >= 5) { return "Prova final"; } else { return "Reprovado"; } }
```

- Repare que a construção do meio emenda um `else` e um `if`. Isso significa que ele só vai verificar a condição do meio (maior ou igual a 5) se a primeira tiver sido falsa (a nota não foi maior ou igual a 7).
- Falando agora das condições que você pode verificar dentro de um if, as mais comuns são:

```
media > 7 // Maior que 7 media >= 7 // Maior ou igual a 7 media < 7 // Menor que 7 media <= 7 // Menor ou igual a 7 media == 7 // Igual a 7 (ATENÇÃO, SÃO DOIS IGUAIS!) media != 7 // Diferente de 7
```

- Quando fazemos comparações em geral isso nos dá um valor que pode ser **verdadeiro** ou **falso**. Por exemplo, se a nota for 8, a comparação `nota > 7` nos dará **verdadeiro** e o código de dentro do if será executado. Porém se nota for 5, a comparação `nota > 7` dará **falso** e o código do if não será executado.
- Como na computação precisamos o tempo todo fazer comparações e tomar decisões, esses valores de verdadeiro ou falso tem até um nome especial: são chamados de **booleanos**. E você pode representá-los em JS usando as palavras `true` (verdadeiro) ou `false` (falso).

- Por exemplo, se passarmos `true` pro if, ele sempre executará (ou se passar `false`, nunca executará):

```
if(true) { // sempre executará, pois true é sempre verdadeiro }
if(false) { // nunca executará, pois false é sempre falso }
```

- Você também pode armazenar esses valores em variáveis, por exemplo:

```
let minhaVariavel = true; if(minhaVariavel) { ... }
```

- Claro que não faz muito sentido armazenar um booleano dessa forma, afinal se você já sabe que ele vai ser `true`, não tem nem porque colocar um if 🙂

- Mas passa a ficar interessante quando armazenamos o resultado de uma comparação em uma variável, por exemplo:

```
function verificarAprovacao(nota1, nota2, nota3) { let media = (nota1
+ nota2 + nota3) / 3; let passou = (media > 7); // Vai armazenar true
ou false na variável "passou", dependendo do resultado da comparação
if(passou) { return "Aprovado"; } else { return "Reprovado"; } }
```