JavaScript – Alura

Parte 1 – Primeiros Passos Com a Linguagem

1. **Aula 1 – Introdução a Linguagem:**
   1. Podemos executar o programa no Power shell digitando “node + nome programa”.
   2. Console.log: Printa o que você escrever em *“ aqui ”*.
   3. SHIFT+ALT+↓ copia e cola uma linha no vscode.
   4. Salvamos variáveis utilizando *“const nome da variável: ;”*.
2. **Aula 2 – Tipos de Variáveis:**
   1. Colocar o comando console é diferente de colocar o comando Console. Quando tem letra maiúscula ele dá erro, pois o JS é Case Sensitive.
      1. Nesse caso, podemos colocar variáveis com cases diferentes, mas com o mesmo nome que o JS irá entender como coisas diferentes, porém, isso é uma má prática, portanto não faça.
   2. Colocar o *const* e não colocar não tem diferença visual na execução do código a princípio. O JS vai continuar interpretando como variável.
   3. Seguir a declaração das variáveis é uma boa prática.
   4. *“let”* é um outro jeito de declarar variáveis.
   5. Seja consistente: se colocar “;” em uma coisa, coloca em tudo, se não, não coloque em nada. Se as variáveis começarem com letras minúsculas, faça para todos, caso contrário, nenhuma.
   6. Colocar um sinal de “+” quando se tem uma string e um valor, ou duas strings faz com que o JS apenas concatene as duas coisas, mas, se colocar dois números, será feita uma soma.
   7. Quando se faz a mesma coisa com outros sinais de operação ele irá fazer a conta matemática corretamente.
   8. Se a string não conter um número, ele irá tentar fazer a conta e retornará um NaN (not a number) ao invés de dar erro.
   9. Para converter uma string para número use o comando *parseInt/parseFloat(“string”);*, sendo elas: números inteiros e flutuantes/reais, respectivamente.
   10. Quando trabalhar com número flutuante, colocar “.” ao invés de “,”, caso contrário ele irá colocar um espaço vazio no lugar dela ao invés de considerar o número como um número real.
   11. Quando printar um texto como nome e sobrenome, podemos simplesmente colocar uma vírgula para que ele separe os itens e fique correto, ao invés de concatenar o nome com um espaço e depois o sobrenome, que dá mais trabalho. *console.log(nome, sobrenome);/console.log(nome + “ “ + sobrenome);*.
   12. Conseguimos colocar interpolação de strings nas versões mais novas do JS utilizando crase para escrever o texto dentro e colocando **${}** ao invés de fazer a concatenação.
   13. Podemos reescrever uma variável com a informação que ela já continha somada a informação de outra variável.
   14. A declaração de variável “*const*” determina que aquela variável é uma constante, ou seja, se é uma constante não pode mudar. Já a declaração “*let*”, é uma variável mesmo e pode sim ser alterada.
   15. Uma boa prática é manter uma variável constante e não ficar alterando o tempo inteiro.
   16. Sempre crie uma variável nova, é bem melhor.
   17. Sempre dê o nome certo para as coisas e seja o mais específico possível para referenciar bem suas variáveis e não ter que ficar procurando para lembrar do que se trata.
3. **Aula 3 – Trabalhando com Muitos Elementos:**
   1. Conseguimos declarar e atribuir valores à variáveis em momentos e linhas diferentes, não preciso fazer isso no mesmo momento sempre.
   2. Podemos comentar várias linhas de uma vez utilizando o atalho CTRL+K+C e tirar o comentário de várias linhas selecionados usando o atalho CTRL+K+U.
   3. Para criar listas utilizamos o comando *const lista = new Array(`elemento1`, `elemento2`, );*, dessa forma criando ela dentro da variável constante *lista*, que poderia ser qualquer outro nome.
   4. Podemos adicionar outros elementos nas listas dinamicamente utilizando o comando *variável.****push(`****elemento novo****`);***.
   5. A vantagem das listas é não ter que criar uma variável para cada item, o que levaria muito tempo tornando inviável fazer isso com muitos elementos.
   6. E o mesmo vale para printar eles na tela, uma vez que teria que reescrever todas as variáveis dentro do console.log(); mais uma vez.
   7. Do mesmo modo que podemos adicionar itens dinamicamente na lista, também podemos retirá-los utilizando o comando *variável****.splice (****posição do**item****,*** *quantidade de elementos****);***. A posição começa no 0, ou seja, a posição 0 é a do primeiro item.
   8. Podemos printar somente 1 elemento colocando entre “[]” o número da posição do item que queremos ver dá nossa lista.
   9. Não podemos criar variáveis com nomes reservados.
4. **Aula 4 – Condicionas e Fluxo de execução:**
   1. É uma boa prática sempre tirar os comentários de comandos inutilizados.
   2. A condição é determinada pelo comando *if(condição){bloco de código para execução}*.
   3. Caso a condição colocada acima seja falsa, podemos colocar um *else{bloco de código}* depois dele para que isso seja executado no lugar.
   4. Podemos colocar vários ifs ou elses encadeados/aninhados dentro de outros para fazer um sistema complexo de escolhas.
   5. Apertando ALT+SHIFT+F o vscode irá formatar automaticamente o código para você.
   6. Muitas condicionais no código pode acabar gerando erros e dificultar na hora de achar.
   7. Ter código duplicado não é uma boa prática.
   8. Podemos criar condições onde uma OU outra operação tem que ser verdadeira para que ele execute o bloco. Do mesmo modo, podemos criar para se uma E a outra seja verdadeira, utilizando as operações **||** e **&&**, respectivamente para indicar isso.
5. **Aula 5 – Iterando Sobre Coleções:**
   1. Posso jogar operações de um if para uma variável.
   2. Podemos colocar um comando break em uma condição para que a partir do momento que ela seja satisfeita, o programa quebre direto para o final, sem ter que ficar passando várias e várias vezes pelo programa até chegar ao fim do contador.
   3. Podemos usar o While para poder repetir interações quantas vezes quiser a partir de um contador.
   4. Também podemos criar laços de repetição usando o *for*. Para colocar ele, precisamos escrever de forma diferente sendo: *for(declara contador; contador; contador++){ bloco de código como no while }*.
   5. Pessoal usa a variável “i” para contador.
6. **Extra – Funções e Classes:**