Js voltado para o back-end

Trabalhando com dados

Para armazenar dados, usamos o const

Const guarda um dado absoluto, não tem como mudar ele depois

Não usamos mais var no back-end

Usamos let, para usar precisamos declarar primeiro e depois aplicar

Aula 2

* var tem um escopo global, então funciona em qualquer parte do código, o que pode gerar possíveis problemas em relação à lógica da aplicação.
* let e const têm escopo local, sendo assim, não podem ser acessadas em outros escopos, e const não pode ter seu valor alterado depois de definido.
* O JavaScript considera alguns valores como *false* durante comparações, como com null, undefined, 0, NaN e Strings vazias.
* null é usado quando não queremos dar um valor para uma variável, mas também não queremos deixá-lo como *undefined* (não definido).
* null é um objeto no JavaScript, diferente de *undefined*, que tem seu próprio tipo, e isso pode causar problemas se não tomarmos cuidado ao utilizá-lo.
* O JavaScript converte automaticamente entre tipos de valores, como números para texto e para booleanos, porém não é interessante deixá-lo fazer isso automaticamente, pois podem ocorrer erros.
* Temos como forçar uma mudança de tipo utilizando os métodos Number() e String(), onde Number() converte as variáveis para números ou NaN caso tenha algum erro na conversão, e String() converte as variáveis para texto.

Aula 3

As características principais do JavaScript, como ser uma linguagem interpretada e dinamicamente tipada.

Que o NodeJS é um interpretador de JavaScript para backend, ou seja, não executa no navegador, mas sim no servidor.

Como ler os erros que aparecem no console da aplicação, podendo usar um tradutor, como o Google Tradutor, para entender o que o erro está avisando.

O que é stacktrace, também chamado de pilha, que nos auxilia a saber o que está acontecendo quando temos um erro, e onde ele se localiza.

O que é o Console API e como podemos utilizá-lo usando o console.log() e console.error().

Onde as mensagem do console devem aparecer dependendo do interpretador que estamos usando, sendo que para o NodeJS as mensagens são mostradas no terminal

Aula 4

* A diferença entre fazer comparações com ==, em que o JavaScript faz conversão entre os tipos de variáveis antes de fazer a comparação, e ===, em que tanto o valor quanto o tipo da variável deve ser o mesmo.
* Como é escrito um operador ternário, com o qual fazemos uma comparação entre valores digitando um ?, seguido da possibilidade *true*, um : e a possibilidade *false*, ou seja, comparação ? true : false.
* O porquê do nome “operador ternário”, que se deve ao fato de termos 3 operadores juntos em uma única linha para desempenhar uma tarefa e devolver um resultado.
* O uso da template literal, que facilita a construção de strings que demandam concatenação (a operação em que se junta 2 strings), e foi introduzido no ES6.
* Como usar a template literal, escrevendo o texto entre acentos graves (`) e colocando as variáveis com ${variavel} para que o valor deles fique nessa posição.

Aula 5

As funções ajudam muito no desenvolvimento de um código, pois colaboram para a separação de trechos de código com funções específicas, tornando-o menor e mais legível, o JavaScript nos oferece algumas funções prontas, como é o caso de funções matemáticas (Math em inglês), alguns exemplos são:

* Math.round(): Faz o arredondamento (round em inglês) de um número de ponto flutuante para o inteiro mais próximo.
  + Math.round(4.3) retorna 4
  + Math.round(3.85) retorna 4
  + Math.round(2.5) retorna 3, quando o número termina com 0.5 a função arredonda para cima
* Math.ceil(): Faz o arredondamento para o valor mais alto, também conhecido como teto (ceil em inglês).
  + Math.ceil(5.2) retorna 6
* Math.floor(): Faz o arredondamento para o valor mais baixo, também conhecido como piso (floor em inglês).
  + Math.floor(5.2) retorna 5
* Math.trunc() : Desconsidera os números decimais, o que é conhecido como truncamento.
  + Math.trunc(4.3) retorna 4
  + Math.trunc(4.8) retorna 4
* Math.pow() : Faz a exponenciação de 2 números, quando for simples, como ao quadrado(2) ou cubo(3). Recomenda-se usar a multiplicação por ser mais rápido.
  + Math.pow(4 , 2) retorna 4^2 = 16
  + Math.pow(2.5 , 1.5) retorna 2.5^(3/2) = 3.9528 ...
* Math.sqrt() : Retorna a raiz quadrada de um número.
  + Math.sqrt(64) retorna 8
* Math.min(): Retorna o menor valor entre os argumentos.
  + Math.min(0, 150, 30, 20, -8, -200) retorna -200
* Math.max(): Retorna o maior valor entre os argumentos.
  + Math.max(0, 150, 30, 20, -8, -200) retorna 150
* Math.random(): Retorna um valor randômico (random em inglês) entre 0 e 1, então não teremos um valor esperado para receber.
  + Math.random() retorna 0.7456916270759015
  + Math.random() retorna 0.15449802102729304
  + Math.random() retorna 0.4214269561951203

O que são funções, pequenos trechos de código que podem ser executados uma ou mais vezes.

As vantagens de se usar uma função, pois com elas conseguimos deixar o código mais curto e mais legível, facilitando a manutenção.

Como retornar informações da função, utilizando o return, lembrando que o console.log() apenas mostra a informação no terminal e não para outras partes do código.

A utilidade dos argumentos, já que com eles podemos passar variáveis para as funções poderem usar os valores.

Que com o hoisting o JavaScript analisa todo o código procurando por variáveis declaradas com var e funções para trazer tais declarações para o início do código.

Expressões de função, uma maneira diferente de montar funções usando variáveis do tipo const e chamando-as pelo nome. Lembrando que é necessário que o programa passe pela variável antes de podermos chamá-la, já que não há suporte à hoisting.

Arrow function, uma função declarada de maneira mais compacta usando uma const. A arrow function também não tem suporte à hoisting.