**AULA 108**

Aula foi sobre promises, que é utilizado para executar elementos assíncronos (quando você não sabe quando exatamente o elemento vai ser executado),

**Exercício**

Criar uma função que execute e mostre 3 frases em ordem numérica utilizando promises

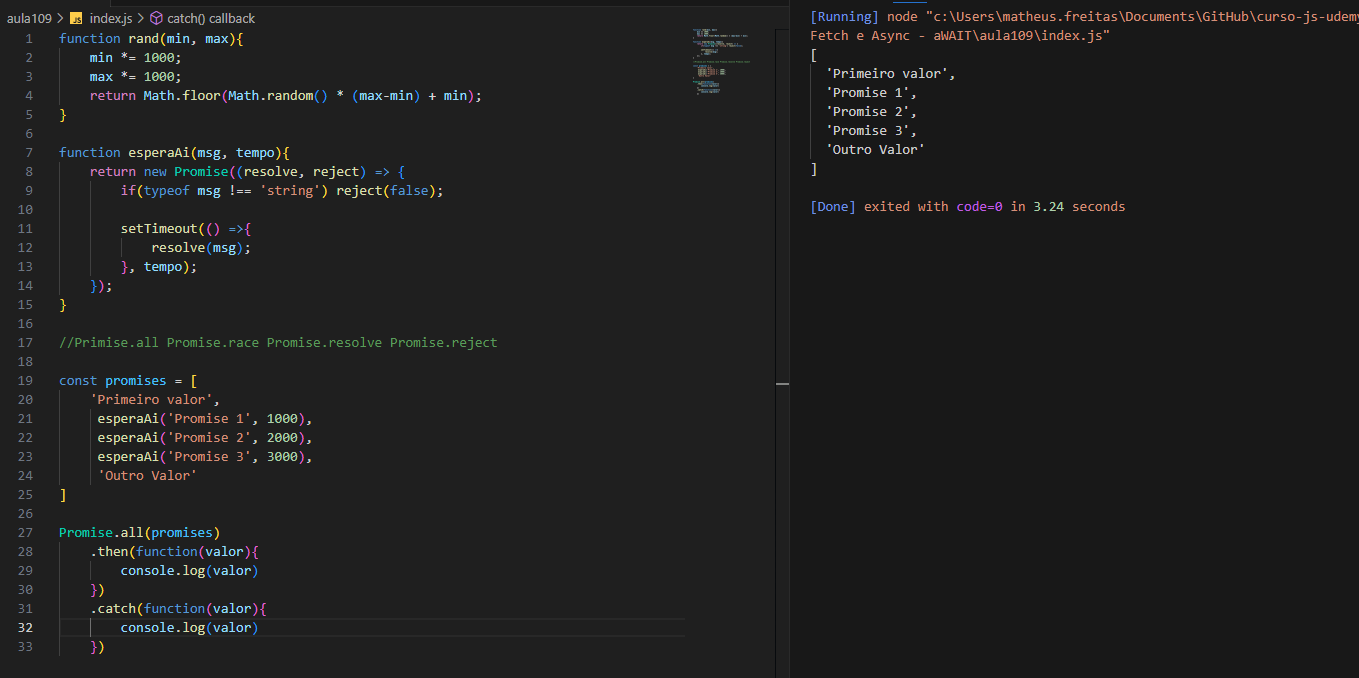
**Resolução**

* Criamos duas funções, uma que gera um número aleatório e outra que apresenta uma mensagem de acordo com o tempo recebido pela primeira função
* Na segunda função, retornarmos um new Promise(), uma função que recebe dois parâmetros: resolve e reject
* Acionamos a função com promise utilizando o comando .then (uma função que utiliza o método resolve, que retornara os parâmetros da função (frase e tempo) e .catch (utilizado para passar o resultado caso a função seja negativa

**AULA 109**

A aula apresenta alguns métodos que podem ser utilizamos com promises, facilitando a execução e utilização do elemento

* Promise.all => Retorna um array com promises. Exemplo: Temos uma array cujo entre seus valores possuímos o acionamento de uma promises. Quando utilizamos o comando .all, na função .then conseguimos retornar esses valores em forma de array:



* Promise.race => Retorna o valor da primeira promise que ele identificar
* Promise.reject => “Aponta’’ a promise direto para o .cache
* Promise.resolve => “Aponta’’ a promise direto para o .then

**AULA 110**

Async e await foram implementados a partir de 2017, são sintaxes que simplificam a programação assíncrona, facilitando o fluxo de escrita e leitura do código. Quando definimos uma função com o comando async antes da função, podemos utilizar o comando await antes de alguma expressão que retorne uma promise. Desa forma, a execução da função externa (async) será pausada até que a promise seja resolvida.

Quando utilizamos await, o JavaScript vai aguardar até que a Promise finalize. Se for finalizada com sucesso (o termo utilizado é *fulfilled*), o valor obtido é retornado. Se a Promise for rejeitada (o termo utilizado é *rejected*), é retornado o erro lançado pela exceção.

Na aula, vimos o exercício da aula anterior convertido para Async/Await:

* + Criamos uma função utilizando o comando async
  + Declaramos variáveis onde seu valor era estabelecido pela promise “esperAI”
  + Utilizamos o comando “console.log” para visualizar os valores das variáveis
  + Utilizamos os comandos “try” e “catch” para que possamos capturar o erro quando a promessa for rejeitada
  + Executamos a função



**AULA 111**

* Aula sobre ajax
* AJAX Web não é apenas uma tecnologia, mas um conjunto de técnicas envolvendo para desenvolvimento web
* Geralmente sistema utiliza as tecnologias:
  1. HTML /XHTML – Linguagem principal e CSS para apresentação
  2. DOM – Exibição dinâmica dos dados e interação
  3. XML – Troca de dados
  4. XSLT – Manipulação de dados
  5. Objeto XHMLHttpRequest – Para comunicação assíncrona
  6. JavaScript – Junção das técnicas acima
* O XML tem a capacidade de armazenar e transmitir dados
* Trabalhando com o AJAX, é possível realizar requisições de dados sem recarregar a página
* Trabalham de forma assíncrona
* Como funciona:
  1. O navegador do usuário gera uma chamada do JavaScript que ativa o objeto XMLHttpRequest
  2. Em segundo plano, o navegador cria uma requisição HTTP para o servidor
  3. O servidor recebe a requisição, busca os dados e envia para o navegador
  4. O navegador recebe os dados requisitados que irão aparecer imediatamente na página. Não é necessário recarregar a página

**AULA 112**

* Aula sobre Fetch API
* Uma melhoria sobre o XHLHtppRequest
* Usado para requisições HTTP
* Recebe uma URL na sua declaração
* Retorna uma promises, então para vermos os resultados utilizamos os métodos .then e .catch

**AULA 113**

* Aula sobre Axios
* Vimos um exemplo de manipulação de dados utilizando Facth API e JSON
* É outra forma de requisição HTTP
* Em sua sintaxe, declaramos axios em seguido como parâmetros passamos o local da requisição HTTP.
* Exemplo:

