Vscode

Segurando ALT e clicando nos valores, ele entenderá como um só e alterará todos os marcados de uma única vez;

Outra alternativa é utilizando o ctrl+f; ele vai abrir o campo com o item a ser modificado. Clica na setinha e em REPLACEe você insere o novo valor. Ao dar enter ele mudará um por um.

Ctrl + P: busca de arquivos

Ctrl+shift+p: mostra comandos do vscode

Emmet: função que consiste em vários atalhos para criar estrutura de códigos.

Mkdir: cria novas pastas

git.init: inicializa o projeto

Git branch: mostra a branch que se está

 o asterisco indica que teve alteração no arquivo que não foi enviado para o repositório no git

Path: são comandos digitáveis na linha de comando. Ele busca-os na pasta bin

Code: utilizado para rodar o comando

Code + nome da pasta abre a pasta em nova janela

INTRODUÇÃO AO JAVASCRIPT

Aula 1: O que é Javascript?

Introdução ao Javascript

Objetivos

1. Entender a história da linguagem

2. Familiarizar-se com a sintaxe

3. Utilização de recursos básicos

4. Criar seu primeiro projeto

5. Conhecer possibilidades no mercado de trabalho

História

Interpretada: linguagem interpretada, ou seja, ela está sendo rodada em tempo real. O código está sendo disponibilizado instantaneamente. Ele não precisa ser transformado em outra coisa antes de ser enviado para o browser, ao contrário de uma linguagem compilada, como C++, que precisa ser transformado. Se estivermos usando algum framework de compilação no JVS, pode se dizer que essa compilação também é em tempo real enquanto o código está sendo executado.

Baseada em Protótipos: base para maioria das estruturas de dados do JVS. Conjunto de funções e funcionalidades em comum.

Multiparadigma: peremite escolher se você quer escolher com orientação a objetos, programação funcional ou programação estruturada. Atende à sua necessidade não limitando a um paradigma. É uma linguagem flexível.

Comumente utilizada em aplicações ***web client-side*** (parte da programação que vai ter comunicação com o usuário: JVS é a linguagem da internet, da web. É a linguagem que os browsers entendem.

Segue o padrão ECMAScript: padrão para várias linguagens. Conjunto de normas que vai dizer quais funcionalidades oficialmente estão sendo lançadas naquela linguagem. Significa que o JVS está seguindo as novas atualizações.

Evolução

Linha do tempo

Descrição gerada automaticamente

Foi criado em 1995 oficialmente. Em 1997 começou a seguir o padrão ECMASprit.

ECMASprit 3 em 1999: permite o uso de expressões regulares, permitindo, por exemplo, validação de e-mails.

Try/catch: maneira de tratar erros

ECMASprit 5 em 2009: suporte ao JSON (quando quer consumir um API). Algumas funções para alguns array.

ECMASprit 6 em 2015: trouxe muita atualização para o JVS

Gráfico, Gráfico de mapa de árvore

Descrição gerada automaticamente

Muito do que se consegue fazer hoje no JVS deve a essa atualização. Permite, por exemplo, o paradigma da orientação de objetos, declare classes, que o programador consiga se familiarizar, permite formas novas de declarar variáveis e concatená-las em streaming.

Há mais evoluções, mas não tão significativas quanto a 6.

Aplicações

Diagrama

Descrição gerada automaticamente

É uma linguagem muito flexível, não limitando ao mundo web.

RECURSOS BÁSICOS:

Objetivos:

1. Se familiarizar com a sintaxe da linguagem

2. Entender a diferença entre variável e constante

3. Ser capaz de comentar seu código

Manipulando um arquivo JS

Para inserir comentários em uma linha começar com //

Para inserir comentários em várias linhas, iniciar com /\* e fechar com \*/

Ctrl + / insere comentário de linha

Variável: valor dinâmico, permitindo alteração.

CONSTANTE: valor não pode ser manipulado. A boa prática é declará-las em letras maiúsculas, facilitando diferenciação com a variável

FUNÇÕES:

Para declará-la, utiliza-se a palavra function e dá um nome a ela, abrindo parenteses para você declarar atributos

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Para obter o resultado chama o return a+b

Para ver no console, utiliza-se o console.log(a + b)

Para chamar a função, repeti-la atribuindo valores:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

CONSOLE: acessando através da tela de inspeção na página WEB, aba CONSOLE, é possível editar e inserir dados. Utilizado para mostrar ao usuário. Outra forma de acessar e debugar é utilizando o nodeJS, ao lado do desenvolvedor.

CTRL + SHIFT + I: ABRE TELA DE INSPEÇÃO

NODEJS: maneira de enxergar o console diretamente dentro do VScode

returnEvenValues: retorna números pares

Array: lista de dados

JS NUMA PÁGINA DA WEB:

ESTRUTURA DE PROJETO

 <meta name="viewport" content="width=device-width, initial-scale=1.0">

Responsivo para todas mídias

Projetos que utiliza diversos tipos de arquivos devem ser separados por pastas (arquitetura) desenhando toda a estrutura do projeto.

Assets: ferramentas que comporão a página

Link: importa o arquivo

Rel= atributo que relaciona o arquivo que está chamando e o chamado

Stylesheet: folha de estilo

Href= referência de onde está o arquivo

Quando não se coloca nada no meio da tag, não precisa ser fechada, bastando colocar />

Se o script for muito grande, o ideal é colocar o estilo antes do encerramento da tag html

Script: tag do JS. Ele tem que ser fechado por completo, pois é possível colocar informações adicionais.

Src= origem do arquivo script

INTERAGINDO COM ELEMENTOS DO DOM

DOM: Document Object Model= Documentos que existem na página. Estrutura dos elementos dentro da janela. Árvore de elementos DOM.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Criando um computador:

Tagname: uma h1, um p, um html

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

É possível selecionar o elemento específico através da tela de inspeção:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

É possível enviar o comando direto da janela de inspeção para este elemento:

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

CRIANDO UM CONTADOR:

Eventos: propriedades do JS

Pela regra do HTML só é possível uma única ID para cada elemento

innerHTML é o html que está dentro do elemento

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="styles.css">

    <title>Contador</title>

</head>

<body>

    <h1>Contador</h1>

    <p>Este é meu primeiro projeto JavaScript pela DIO</p>

    <div id="counter">

        <button name="subtrair" onclick="decrement()">-</button>

        <span id="currentNumber">0</span>

       <button name="adicionar" onclick="increment()">+</button>

    </div>

    <script src="scripts.js"></script>

</body>

</html>

Script

var currentNumberWrapper = document.getElementById('currentNumber');

var currentNumber = 0;

function increment() {

    currentNumber = currentNumber + 1;

    currentNumberWrapper.innerHTML = currentNumber;

}

function decrement() {

    currentNumber = currentNumber - 1;

    currentNumberWrapper.innerHTML = currentNumber;

}

Style

@import url('https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;600&display=swap');

\* {

    font-family: 'Open Sans', sans-serif;

}

body {

    display: flex;

    align-items: center;

    justify-content: center;

    flex-direction: column;

    background-color: rgb(177, 225, 255);

}

button {

    height: 40px;

    width: 40px;

    border-radius: 50%;

    border: 2px solid lightcoral;

    background-color: rgb(241, 214, 214);

    color: lightcoral;

    font-size: 16pt;

    font-weight: 600;

    text-align: center;

}

#counter {

    display: flex;

}

#currentNumber {

    display: flex;

    align-items: center;

    margin: 0 20px;

    font-size: 16pt;

}

PROJETO TO-DO-LIST

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="styles.css">

    <title>To-do-list</title>

</head>

<body>

    <h1>To-do-list</h1>

    <!-- Começando a criar a página -->

    <div id="list"> <!-- Aqui é para criar os itens em lista -->

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

<form id="task-form"> <!-- Inserindo o formulário para usuário preencher -->

            <input id="task-input" name="tarefa" type="text" placeholder="Digite aqui sua tarefa" /> <!-- código pedindo para usuário inserir uma tarefa no formato texto -->

            <button type="submit">Adicionar</button> <!-- Cria o botão para adicionar a informação digitada em lista -->

        </form>

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

HTML completo do projeto:

<!DOCTYPE html> <!—dar comando emmit e clicar em html5 já vai abrir o head completo -->

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="styles.css">

    <title>To-do-list</title>

</head>

<body>

    <h1>To-do-list</h1>

    <!-- Começando a criar a página -->

    <div id="list"> <!-- Aqui é para criar os itens em lista -->

        <form id="task-form"> <!-- Inserindo o formulário para usuário preencher -->

            <input id="task-input" name="tarefa" type="text" placeholder="Digite aqui sua tarefa" /> <!-- código pedindo para usuário inserir uma tarefa no formato texto -->

            <button type="submit">Adicionar</button> <!-- Cria o botão para adicionar a informação digitada em lista -->

        </form>

        <div id="tasks"></div>

    </div>

    <script src="scripts.js"></script>

</body>

</html>

Analisando o Script

// os código abaixo vão criar o formulário e o id para receber as tarefas do usuário //

const form = document.getElementById('task-form');

const taskList = document.getElementById('tasks');

form.onsubmit = function (e) {

    e.preventDefault();

    const inputField = document.getElementById('task-input');

    addTask(inputField.value);

    form.reset();

};

function addTask(description) {

    const taskContainer = document.createElement('div');

    const newTask = document.createElement('input');

    const taskLabel = document.createElement('label');

    const taskDescriptionNode = document.createTextNode(description);

// comandos para criar a lista das tarefas digitadas com o checkbox //

    newTask.setAttribute('type', 'checkbox');

    newTask.setAttribute('name', description);

    newTask.setAttribute('id', description);

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

// Código para que o texto digitado seja inserido na página //

    taskLabel.setAttribute('for', description);

    taskLabel.appendChild(taskDescriptionNode);

    // Criação do container que vai armazenar a lista de tarefas //

    taskContainer.classList.add('task-item');

    taskContainer.appendChild(newTask);

    taskContainer.appendChild(taskLabel);

    // o comando abaixo é quem vai criar a lista das tarefas digitadas //

    taskList.appendChild(taskContainer);

}