TÓPICO 9

ÚLTIMAS PALAVRAS –

ALÉM DA LÓGICA DE

PROGRAMAÇÃO.

OBJETOS / PROPRIEDADES / VALORES:

10

TRATAMENTO DE EXCEÇÃO:

10

INTRODUÇÃO:

Para começar, se você chegou até aqui, meus mais sinceros parabéns. Como o título já diz, neste tópico trataremos de assuntos genéricos, porém de suma importância na vida de todo e qualquer desenvolvedor.

Vamos começar passando algumas dicas sobre teclas de atalho no VSCode, conceito de objetos, após isso partiremos para <tags>, tipos de comentários, ferramentas de compartilhamento, boas práticas, debug tanto no navegador (Chrome) como no VSCode, Cal back, bibliotecas, Devtools e tratamento de exceção.

Se você achou que este curso acabaria sem esse tópico e seria hora do “tchau”, você se enganou. Alongue suas falanges e mão na massa.

Para padronizar e facilitar o seu entendimento, usaremos durante os exemplos deste tópico, o Google Chrome como nosso navegador e o VSCode como nosso editor de código.

Caso queira ter uma melhor paridade visual, recomendamos utilizar as mesmas ferramentas.

TECLAS DE ATALHO NO VSCODE:

1: Comando “code” no cmd. Este comando fará o VSCode ser executado em sua máquina.

1.1: Caso digite no cmd do seu computador o comando “code” + nome do arquivo + .html ou .js ou .css, ele automaticamente executa o VSCode já com o nome do arquivo digitado e a estrutura do arquivo criada, necessitando apenas salvar;
1.1.2: Também é possível abrir repositórios pelo cmd tendo acesso as pastas direto no VSCode com apenas um comando. Basta abrir o cmd e digitar: “code + (nome do repositório);
2: “! + Enter” Ao criar seu arquivo .html, digite “!”, e pressione o enter. Este atalho criará toda a base que necessita para começar seu projeto.

3: “Ctrl + Shift + P” Ativa a paleta de comandos. Através dela podemos acessar comando internos do VSCode e também comando externos instalados como do Git ou Flutter; **4: “Ctrl + P”** Abre uma aba para pesquisa de pastas. Esse atalho facilita muito a vida do programador em caso de projetos enormes, onde há inúmeras pastas e arquivos; **5: “Ctrl + P + *@*”** Mostra a quantidade de elementos que o arquivo tem. Ex: Funções, métodos... Exibe na ordem que eles foram gerados;

5.1: **“Ctrl + P + *@* + :”** Faz a subdivisão da quantidade de cada elemento há no arquivo: **6: “Ctrl +**

Shift + L” caso necessite trocar o nome de algum elemento, este atalho irá facilitar e muito a sua vida. Ao trocar apenas o primeiro elemento, o atalho executará o resto do trabalho, alterando o código por completo. Para utilizar este atalho, basta selecionar o elemento e com ele selecionado, pressionar “Ctrl + Shift + L”, pronto! Agora é só fazer a alteração; **7: “Ctrl +]”** Abrirá uma segunda tela já particionado a imagem do seu monitor para que consiga visualizar ambos os projetos simultaneamente; **8: “Alt + UP_ARROW” ou “DOWN_ARROW”** Moverá um

bloco selecionado sem a necessidade de copiar e colar a região desejada;

9: “Alt + Shift + UP_ARROW” ou “DOWN_ARROW” Clona o código selecionado; **10: “Ctrl** + ‘ “ Abre o terminal integrado. Terminal, Debug Console, OUTPUT, PROBLEMS; **11: “Ctrl + Shift + D”** Abre O Debugger;

12: “editor.wordWrap” : “on” Ao ativar o “editor.wordWrap”, ele fará com que textos ou linhas de códigos muitos extensas, fiquem visíveis sem ter que usar o scrol Horizontal. Para ativar, basta ir em “File” -> “Preferences” -> “Settings” e digitar

“editor.wordWrap” na barra de pesquisa.

Após isso basta alterar para “on”.

13: “Alt + Shift + F” Faz a identação do código;

```
<html>
    exemplo<br>
    exemplo2<br>
</html>
```

```
<script>
    var i = 'Hello \n World'
    console.log(i)
</script>
```

```
<html>
<!--exemplo<br>-->
```

```
<html>
<!--exemplo<br>
exemplo<br>
exemplo<br>
exemplo<br>
exemplo<br>
exemplo<br>
exemplo<br>-->
</html>
```

```
<script>
  //var i = 'Hello \n World'
```

DIFERENÇA DE BREAKLINES:

**1: “
”** Faz a quebra de linha em um texto no html; **Exemplo:**

2: “\n“ Faz a quebra de linha no JavaScript;

Exemplo:

COMENTÁRIOS:

1: “<!-- -->” Esta tag é utilizada no Html para realizar comentários em linha única, como também em bloco;

Exemplo Linha Única:

Exemplo Bloco:

2: “//” Utilizada DoubleBars para realizar comentários em linha única no JavaScript; **Exemplo:**

```
<script>
  /*var i = 'Hello \n World'
  console.log(i)*/
</script>
```

2.1 “/* */” Para fazer comentário em blocos, usa-se “/*” para iniciar o comentário e “*/” para finalizar o comentário.

Exemplo:

BOAS PRÁTICAS:

1 - Nomes de variáveis, métodos e const-: Escolher o nome de maneira clara e direta sobre o que se trata (Curtos e Objetivos)

ps: As abreviações vêm de uma limitação de antigamente onde as variáveis só poderiam conter 8 caracteres;

2 - Indentação: Facilita na leitura e entendimento do código; **2.1 -** Deve seguir a convenção da equipe. Ex: Se a equipe faz as validações sempre no começo das funções, isso seria bom ser seguido, pois assim, sempre que o dev

foi procurar por validações, ele saberá que fica no início das funções.

3 - Condição de negação no IF: Usar o IF para avaliar se ele é TRUE, e deixe o ELSE para o FALSE;

4- Comentários: Não servem para explicar o algoritmo passo a passo e sim para explicar em linhas gerais a utilização do código e para que utilizar ele; **4.1 -** Usar o comentário para descrever uma solução que pode não ser a ideal, porém devido a alguns fatores (prazos, limitações), foi a única que você e a equipe chegaram. Este tipo de comentário serve para dar

notoriedade para futuramente pensar-se em alguma solução melhor; **4.2 -** Caso não tenha muita experiência ou trabalhe em um projeto com muitos colaboradores é interessante (Não Obrigatório), comentar o código para facilitar o entendimento de todos.

Utilizar para trechos importantes;

4.3 - Se não está comentando o código para o deixar mais auto explicativo ou não está entendendo seu código, é melhor começar a refazer!

5 - Evitar a Duplicidade do código: Diminui o volume de código e torna a

manutenção mais fácil; **6 -** BeckUp: Fazer o BackUp na nuvem.;

7 - CammelCase: declarar variáveis, funções, constantes, com o início de cada nova palavra com a primeira letra MAIÚSCULA, seguido do resto minúsculo; **7.1 -** UpperCammelCase: definição de classes e métodos; **Exemplo:** void ExemploUpperCammelCase()

7.2 - lowerCammelCase: variáveis, atributos e métodos; Exemplo: var exemploLowerCammelCase = xxxx;

8 - CleanCode : Livro escrito por Uncle Bob(Roben C Martin). Estudo

que mostra que gastamos muito mais tempo lendo e dando manutenção um código do que codando e criando algo novo; **8.1 -** Fazer Validações de Funções em funções separadas. Evitar ter uma única função com muitas linhas;

```
<body>
<link rel="stylesheet" href="style.css">
<script src=script.js></script>
</body>
```

8.4 - Formatação: Uma linha de código deveria caber exatamente no tamanho da tela, assim facilita a leitura do código;

9 - Chamar .css e .js no .html para simplificar o código

:

10 - Declarar variáveis fora de funções e manter o tipo (var x = “string”, tentar manter sempre string.)

11 - Utilizar arrays sempre que possível para minimizar o código.

12 - Atribuir “nul ” a uma variável que irá guardar um objeto no futuro;

EDITORES DE CÓDIGO ONLINE:

Os editores de código online são ferramentas muito utilizadas pelos desenvolvedores, principalmente os iniciantes, tendo a função de um interpretador de uma linguagem de

programação, só que como o nome já diz, ONLINE.

Estas opções, facilitam muito a aprendizagem do programador, pois lá você consegue criar seu projeto, compartilhar com outros desenvolvedores, buscar códigos e projetos já prontos ou em desenvolvimento, encontrar bibliotecas... Enfim, uma infinidade de funções, e o melhor de tudo isso é: Não há a necessidade baixar e configurar nenhum programa em sua máquina, basta acessar o site do editor e pronto.

Alguns exemplos de editores de código online são: Codepen e o P5.

Codepen: Além de um editor de código online que executa muito bem a sua função, ele é também uma espécie de rede social/portfólio do desenvolvedor. Nele você consegue compartilhar seus códigos, comentar, dar like, compartilhar projetos de terceiros.

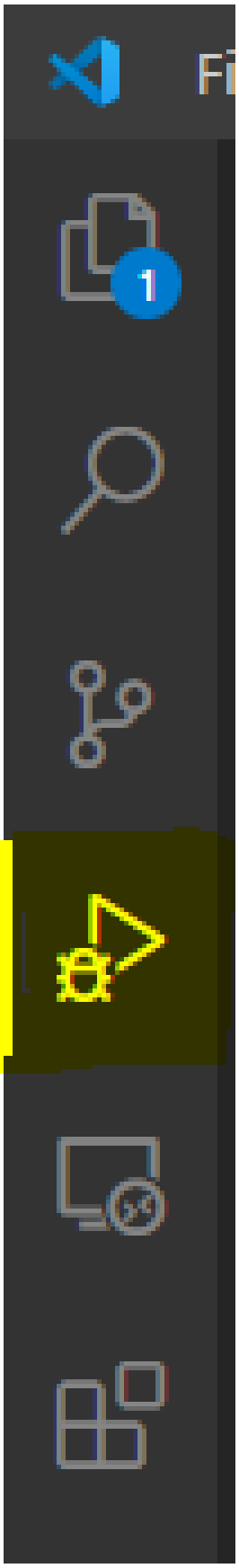
Codepen é um ambiente voltado mais para o frontend, permitindo que você trabalhe com Html, Css e JavaScript. Tendo opções pagas e gratuitas e

permitindo o usuário trabalhar com ou sem log in.

Acesso ao editor: Codepen

P5.js: Análogo ao codepen, o P5.js é mais um exemplo muito utilizado por professores e desenvolvedores iniciantes, para criação de seus primeiros códigos. Também voltado para o frontend, o P5.js permite ao usuário a busca de bibliotecas, compartilhamento dos seus códigos e muitas outras possibilidades, todas de maneira gratuita e online.

Acesso ao editor: P5.js



DEVTOOLS:

DevTools é uma ferramenta que auxilia o desenvolvedor em aplicações frontend. A sua função é indicar de forma bem visual e direta, o que está acontecendo na execução do seu código dentro do navegador. Nele é possível inspecionar blocos, layout, conteúdo, performance, conectividade, fazer alterações sem afetar o código fonte e muito mais.

Composto por nove abas (Elements, Console, Sources, Network, Performance, Memory, Application, Security, Lighthouse. Cada uma dessas

abas tem uma função específica e saber utilizá-las é essencial na vida de todo programador.

Para acessar a área de Devtools, basta buscar por ferramentas do desenvolvedor no seu navegador ou pressionar a tecla “F12”.

Recomendamos também, caso queira se aprofundar mais e conhecer a fundo todas as funcionalidades do Devtools, uma playlist do youtube no canal da Google Developers.

Segue o link para acesso.

DEBUG:

Debug é um termo utilizado para descrever o processo de encontrar e remover erros em um software ou hardware.

Erros esses que podem ser desde algo simples, como o de digitação, sintaxe ou até mesmo de lógica de programação.

Através do debug, o desenvolvedor consegue acompanhar a execução do seu código linha por linha, colocar breakpoints e até mesmo acompanhar uma única ou mais variáveis durante a execução, assim facilitando o entendimento e fixação do erro.

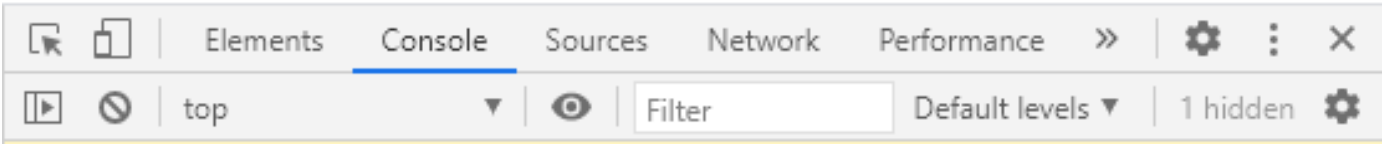
FERRAMENTAS DE DEBUG:

Existe a possibilidade de debugar seu código diretamente no seu editor de código ou navegador no momento da execução do mesmo. No caso do editor de código, mais especificamente VSCode, o debug pode ser acessado através do ícone na lateral esquerda, que se chama “Run” ou então pressionando as teclas “CTRL + SHIFT + D”.

Já para debugar seu código diretamente no navegador, será necessário executar o arquivo .html e

após isso, buscar por ferramentas do desenvolvedor ou pressionar a tecla

“F12”.



Cada navegador disponibiliza uma ferramenta diferente com possibilidades diferentes.

Neste curso, daremos exemplo do debug no navegador Google Chrome.

Alguns exemplos desta ferramenta em navegadores são:

-Chrome Dev Tools (Google Chrome);

-Web Inspector (Safari);

-FireFox Developer Tools (Mozil a FireFox).

Há uma certa vantagem em debugar o código dentro do editor como o VSCode. Um exemplo simples: Caso você execute seu código no navegador e por algum erro de sintaxe ou incremento de variável, seu código entra em um loop infinito. O console não mostrará erro algum na maioria dos casos e isso dificulta a sua vida para conseguir identificar o que está acontecendo. Já se decidir debugar no editor, você poderá colocar breakpoint

e até mesmo selecionar algumas variáveis ou funções, colocando-as em evidência. Assim, toda vez que ocorrer um incremento ou decremento, será fácil a sua visualização.

LOG DE ERRO:

São arquivos de texto com informações que mostram o que aconteceu na execução do seu código. Pode-se encontrar esses arquivos em diversos ambientes, como: Sistema operacional, banco de dados, servidor web...

BIBLIOTECAS:

Ferramenta utilizada pelo desenvolvedor para executar ações específicas, pré-definidas.

Existem milhares de bibliotecas já desenvolvidas, onde cada uma é composta por inúmeras funções. Quando necessário, pode-se baixar, chamar para seu projeto e executar o trecho onde achar pertinente.

Um exemplo simples, mas de fácil entendimento e analogia, é o caso de fórmulas matemáticas. Quando se está na escola, você não precisa desenvolver toda a fórmula de Bhaskara, para que só depois

solucionar seu problema. Basta recorrer à biblioteca e estará lá, algo que já foi pensado e desenvolvido por outra pessoa e que facilitará seu trabalho.

OBJETOS / PROPRIEDADES / VALORES:

No mundo da computação, objeto todo e qualquer elemento que pode ser alocado em memória é usado posteriormente quando chamado. Todo objeto tem propriedades e valores.

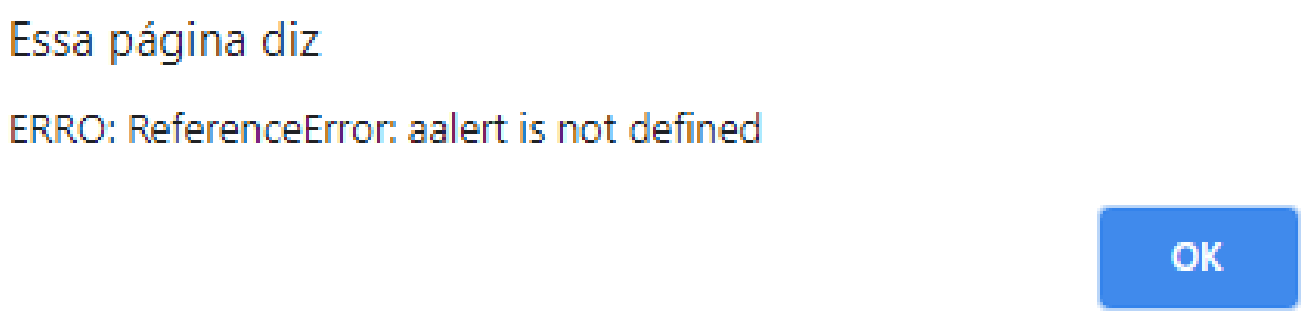
Já as propriedades por sua vez são consideradas um atributo do HTML onde eventualmente será aplicada

alguma regra. Como por exemplo: background color, font, text align.

Por último, mas não menos importante, os valores são as características que as propriedades recebem. Por exemplo: letter type, letter color, style.background = “red”.

```
<script>
 try{
      aalert("eai");
 } catch (error){
      alert("ERRO: "+ error);
 }

</script>
```



TRATAMENTO DE EXCEÇÃO:

O tratamento de exceção pode ser entendido como “tratamento de erros”, erros esses que acontecem durante o fluxo de execução do código. Normalmente os erros não são exibidos no JavaScript, este problema pode acarretar uma série de óbices, no qual o usuário comum (sem conhecimento prévio de linguagem de programação) não fica ciente e justamente por não ter noção do que

está acontecendo, seu código não será utilizado como planejado.

Caso você tenha digitado, por exemplo, um **“alert”** de maneira errada, seu código não exibirá nada ao usuário. O tratamento de exceção trabalha com **“try{}”,** onde o que estiver especificado no bloco do **“try”** será executado e caso apresente algum erro, o **“catch(erro){}”**

entra em ação! Ele pegará o problema, armazenará os dados do erro na variável “erro” e irá exibi-la de acordo com o que foi declarado em seu bloco consequentemente **(alert(erro), <p>).**

Desta maneira, o usuário recebe um retorno do código (é melhor receber uma mensagem de erro, do que nada ser executado) e o desenvolvedor recebe a faca e o queijo na mão, para que possa realizar a manutenção o mais brevemente possível.

Irei disponibilizar um **link** de uma aula do canal “RBtech”, onde é tratado com exemplos este assunto.