# Módulo 02 - Javascript

## Aula 09 – Modularização

- Servidor Web
- Defer
- Módulos
- Import e Export

**JavaScript Hospedado em um Servidor Web**

- Quando programamos um JavaScript em uma pasta da nossa máquina junto com nossos arquivos existem vários mecanismos bloqueados que fazem com que a gente não possa usar a linguagem em sua totalidade.
- Uma vez que hospedamos nossos arquivos em um servidor, novas portas se abrem como a possibilidade de carregar um script após o carregamento da página e a partição de nossos scripts em diversos arquivos.

# O Atributo DEFER

# *1. Defer*

## O Atributo Defer da Tag Script

- O primeiro grande artifício que iremos falar é sobre o atributo defer, que é ***REESTRITO*** a scripts hospedados em servidor web.

  < script src = ”arquivo.js” defer >  < / script >

- Basta adicionar a palavra-chave DEFER dentro da Tag de Abertura do seu script (assumindo que você irá usar essa tag para invocar um arquivo js externo) fazendo com que seu script seja carregado SOMENTE após o carregamento do HTML.

# *1. Defer*

## O Atributo Defer da Tag Script

- Carregar seus scripts ***APÓS*** o carregamento do documento HTML é fundamental para qualquer um que queira usar JS para manipular tags, como os comandos da família do *document.getElement()* que só conseguem buscar uma Tag no documento quando o mesmo já foi carregado.

# ***Módulos***

# *2. Módulos*

**Modularização**

- Um outro artifício crucial para qualquer programador em qualquer linguagem, é o poder de dividir seu software em módulos.

**O que são módulos ?**

- Módulo é o nome que se dá a fração de um todo, um pedaço genérico de código que pode ser encaixado em outros softwares(seria como uma peça do brinquedo LEGO).

# *2. Módulos*

**Modularização**

- Na prática, “modularizar”(dividir em módulos) nada mais é do que pegar um sistema robusto escrito em um arquivo e dividir ele em vários arquivos diferentes, onde cada arquivo será encarregado de uma tarefa especifica, cada arquivo será um módulo.
- (Seria como pegar um brinquedo grande de LEGO e dividir ele em todas as pecinhas, cada arquivo é uma pecinha(um módulo) e quando se junta essas peças você tem o brinquedo de fato(software de fato).

# *2. Módulos*

**Sistema Robusto** → Poucos arquivos grandes(manutenção difícil)

index.html

crud.js

**Sistema Modularizado** → Muitos arquivos pequenos(manutenção fácil)

index.html

cadastraRegistro.js

deletaRegistro.js

atualizaRegistro.js

lerRegistro.js

# 2. Módulos

**Modularização**

- Agora você deve estar se perguntando, como é possível criar um programa dividido em arquivos separados e todos eles fazerem parte do mesmo software ?
- Existem alguns comandos especiais para isso, onde você pode mover variáveis e funções entre arquivos diferentes (semelhante ao processo de amarrar um arquivo de imagem a uma tag A, ou puxar todo um código CSS para dentro de uma tag link)

# Import e Export

# 3. Import e Export

## Migração de dados

- Agora veremos comandos que apesar de simples, pode ser um pouco difíceis de entender seu real funcionamento, vamos imaginar a situação hipotética onde eu tenha uma variável em um arquivo1.js e eu queira acessar seu valor em um arquivo2.js.
- Como os arquivos funcionam de forma independente, o arquivo2.js não consegue “enxergar” que aquela variável foi criada em outro lugar, ele só assume que a variável não existe e gera um erro

# *3. Import e Export*

**Arquivo1.js**

Let x = 20

*Esse programa não faz nada, apenas declara uma variável.*

***Arquivo2.js***

*Console.log(x)*

*Esse programa dá erro, em tese a variável x nunca foi declara.*

# 3. Import e Export

**Migração de dados**

- Para que essa operação seja possível precisamos fazer duas operações :

  **EXPORTAR a variável do ARQUIVO1.js**

  - Exportar é o ato de “chutar” a variável de um arquivo para outros

  **IMPORTAR a variável do ARQUIVO2.js**

  - Importar é o ato de tentar “pegar” a variável que foi chutada por um outro arquivo

# *3. Import e Export*

**EXPORT**

- Para exportar variáveis ou funções basta adicionar a palavra-chave “export” antes da declaração.

  Exemplo:

  **export let x = 10**

  **export function exemplo() { ... }**

- Agora tanto essa váriavel quanto essa função estão prontas para serem “invocadas” por outros arquivos.

# *3. Import e Export*

**IMPORT**

- Uma vez que um dado foi exportado por um arquivo, quaisquer outros arquivos podem agora acessá-los fazendo sua importação

  Exemplo:

  **import { x } from ‘./arquivo.js’**

- Import → inicia o comando de importação
- { x } → define o que vai ser importado do outro arquivo
- From “caminho” → mostra ao código de onde esses dados estão vindo

# *3. Import e Export*

**Arquivo1.js**

Export let x = 20

*Esse programa prepara uma variável para ser acessada por outros módulos*

***Arquivo2.js***

*Import { x } from ‘./arquivo1.js’*

*Console.log(x)*

*Esse programa puxa para dentro de si uma variável de um arquivo externo*

# *3. Import e Export*

**Migração de dados**

- É possível realizar essa operação VÁRIAS vezes, exportando vários dados diferentes e especificando o nome de cada um deles nas chaves da importação.

  Atenção a escrita do import

  **import { valor , valor , valor } from “. / caminho do arquivo ”**

# *3. Import e Export*

**Arquivo1.js**

```
Export let x = 20
Export let y = 30
Export function msg(){
    // Código
}
```

*Esse programa exporta variáveis e funções*

***Arquivo2.js***

```
Import { x , y , msg } from ‘./arquivo1.js’
Console.log(x)
```

*Esse programa puxa para dentro de si vários dados de um arquivo externo*

# 3. Import e Export

***ATENÇÃO***

- Algumas coisas podem dar errado nesse meio tempo, então certifique-se de :

  1 → colocar o atributo type=”module” na tag script

  2 → usar export em toda var/func que você deseja exportar

  3 → usar o ./ no começo do caminho de arquivo de um modulo

  4 → adicionar apenas NOMES nas chaves do import (não use parênteses para chamar funções, basta escrever o nome delas)

# *3. Import e Export*

**Migração de dados**

- Se você parar para pensar um pouco, já deve imaginar que em uma aplicação grande seria muito trabalhoso ficar escrevendo o nome de todos os seus dados no import certo ?
- Import { var1 , var2, var3, var4, var5, var6 } from ‘./modulo.js’
- É aqui que entra a **GIGANTESCA IMPORTANCIA** dos arrays e objetos.

# *3. Import e Export*

**Arquivo1.js**

Export let x = 20

Export let y = 30

Export let z = 40

Export let w = 50

Export let a = 60

Export let b = 70

Export let c = 80

***Arquivo2.js***

*Import { x , y , z , w, a , b , c } from ‘./arquivo1.js’*

# *3. Import e Export*

**Arquivo1.js**

Export let container =

[ 20, 30, 40, 50, 60, 70, 80]

***Arquivo2.js***

*Import { container } from ‘./arquivo1.js’*

let x = container[0]

let y = container[1]

let z = container[2]

let w = container[3]

*// etc ...*