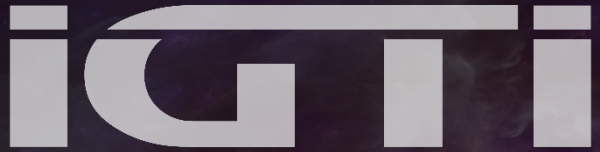

Fundamentos de Desenvolvimento Full Stack

Capítulo 2. JavaScript

Prof. Raphael Gomide

Aula 2.1. Introdução ao JavaScript

Nesta aula

- JavaScript.
  - Introdução.
  - Integração com HTML e CSS.

JavaScript – Introdução

- Linguagem de programação.
- A relação com Java é apenas no nome comercial.
- Nome formal da linguagem – ECMAScript.
- Principal utilização – Front End Web.
- Também pode ser utilizada em:
  - Backend, com Node.js.
  - Aplicações desktop, com Electron.
  - Dispositivos embarcados.
  - Internet das Coisas (IoT).
  - Machine learning.

JavaScript – Integração com HTML e CSS

- A maneira mais comum é a criação de um arquivo externo com extensão .js.
- Inclusão da tag <script src=“arquivo.js”></script> antes do encerramento de <body>
- Acompanhe o professor.

Conclusão

- ☑ JavaScript é uma das principais linguagens de programação atualmente.
- ☑ É utilizada principalmente para interação do usuário em Front End Web.
- ☑ Possui diversas outras aplicações, tais como:
  - Backend com Node.js.
  - Desktop com Electron.
  - Machine Learning.
- ☑ Facilmente integrada ao HTML e CSS.

Próxima aula

- JavaScript básico.

Aula 2.2. JavaScript básico

Nesta aula

- JavaScript básico:
  - Principais tipos e valores.
  - Variáveis.
  - Operadores.
  - Console.
  - Comentários.

JavaScript – principais tipos e valores

- Existem atualmente **8** tipos de dados em JavaScript.
- Vamos focar apenas nos principais:
  - Number → 1, -3, 8.56
  - String → "Teste", "3.14", 'Aspas simples'
  - Boolean → true, false
  - Null → null (explicitamente definido pelo programador)
  - Undefined → undefined (ausência de valor)
  - Object → [1, 3, 5], [6, 'sete', true], {id: 2, nome: 'Raphael'}
- Apesar dos tipos, JavaScript é considerada uma linguagem de programação com **tipagem fraca**.

JavaScript – Variáveis

- Criamos variáveis para guardar valores que serão reutilizados posteriormente.
- Esses valores são guardados em memória.
- Palavra-chave: var**.**
- Operador de atribuição de valores: = (igualdade).
- Exemplo de comando: `var pi = 3.14;`
- Acompanhe o professor na demonstração de:
  - Tipos, valores e variáveis.

JavaScript – Operadores aritméticos

- Soma → +
- Subtração → -
- Multiplicação → *
- Divisão → /
- Exponenciação → **
- Resto de divisão → %
- Incremento → ++
- Decremento → --

JavaScript – Operadores de atribuição

- Atribuição padrão → =
- Atribuição com operação → += -= *= /= %=

JavaScript – Operadores de comparação

- Igualdade → ===
- Diferença → !==
- Maior → >
- Menor → <
- Maior ou igual → >=
- Menor ou igual → <=

JavaScript – Operadores lógicos

- E → &&
- OU → ||
- Negação → !

JavaScript – Console

- Muito útil para testes simples e depuração de código (debug).
- Principal comando: console.log().

```
> var x = 10;
<· undefined
> console.log(x);
  10
<· undefined
> console.log("O valor de x + 10 é " + (x + 10));
  O valor de x + 10 é 20
<· undefined
```

JavaScript – Comentários

- Muito útil para descrever operações complexas.
- Algumas formas:
  - Comentários de linha: // Comentário
  - Comentários de bloco: /* Comentário */
- Acompanhe o professor na demonstração de:
  - Operadores.
  - Console.
  - Comentários.

Conclusão

☑ Principais tipos e valores:

- Number, String, Boolean, Object, Null e Undefined.

☑ Variáveis:

- Palavra-chave var.

☑ Operadores:

- Aritméticos, lógicos, de comparação, de atribuição.

☑ Console:

- O comando console.log().

☑ Comentários:

- Comentários de linha e de bloco.

Próxima aula

- JavaScript – comandos de bloco.

Aula 2.3. JavaScript – Comandos de bloco

Nesta aula

- JavaScript – comandos de bloco:
  - Proposições lógicas.
  - Estruturas de decisão.
  - Estruturas de repetição.
  - Funções.

JavaScript – Proposições lógicas

- Afirmativas que podem ser:
  - **Ou** verdadeiras.
  - **Ou** falsas.
  - **Jamais** serão **verdadeiras e falsas** ao mesmo tempo.
  - Podem ser aninhadas com operadores lógicos (&& / || / !)
- Exemplos:
  - 7 > 5 → true
  - 2 === 2 && 4 > 3 → true
  - 3 === '3' || 2 > 2 → false
  - 9 < 11 && !(9 < 11) → false

JavaScript – Estruturas de decisão

- O comando **if/else:**
  - Principal estrutura de decisão.
  - Testa uma proposição lógica.
  - O bloco **if** é executado se a proposição lógica for **true**.
  - O bloco **else** é executado se a proposição lógica for **false**.
  - O bloco **else** é opcional.
  - Os comandos podem ser aninhados.
  - O comando executa **somente um dos blocos**.
- Acompanhe o professor.

JavaScript – Estruturas de decisão

- O comando **switch:**
  - Útil para testes de **igualdade** com **vários valores possíveis**.
  - Sintaxe mais elegante e legível que if/else.
  - O **else** é implementado com a cláusula **default**.
- Acompanhe o professor.

JavaScript – Estruturas de decisão

- **Operador ternário:**
  - Útil para substituir if/else em comandos simples.
  - Sintaxe um pouco confusa.
  - Utiliza os símbolos ? e :
- Acompanhe o professor.

JavaScript – Estruturas de repetição

- O comando **while**:
  - Executa o bloco **enquanto** a proposição lógica for **true**.
  - É necessário que tornemos a proposição **false** em algum momento.
  - Caso isso não ocorra, acontecerá o **loop infinito**.
  - Geralmente utilizamos uma variável de controle para garantir a correta execução do comando.
- Acompanhe o professor.

JavaScript – Estruturas de repetição

- O comando **do... while**:
  - Executa o bloco **enquanto** a proposição lógica for **true**.
  - É necessário que tornemos a proposição **false** em algum momento.
  - Caso isso não ocorra, acontecerá o **loop infinito**.
  - Geralmente utilizamos uma variável de controle para garantir a correta execução do comando.
  - A diferença para o comando **while** é que, no do... while, a proposição é **testada ao final do bloco.**
  - Com isso, garante-se que o bloco seja **executado pelo menos uma vez.**
- Acompanhe o professor.

JavaScript – Estruturas de repetição

- O comando **for**:
  - Comando semelhante ao **while** e **do... while.**
  - Entretanto, possui uma **sintaxe mais elegante** e **menos propensa a erros.**
  - Inicialização da variável de controle, teste da proposição e incremento/decremento são feitos na mesma linha.
- Acompanhe o professor.

JavaScript – Funções

- Bloco de código que executa algum tipo de algoritmo.
- **Não** é **executado imediatamente**.
- Deve ser **invocado posteriormente**.
- Permite a reutilização de lógica.
- Pode aceitar **parâmetros**, também conhecidos como **argumentos**.
- Pode retornar **zero** ou **apenas um valor**.
- Acompanhe o professor.

Conclusão

☑ Proposição lógica:

- Afirmativa que podem ser ou verdadeira ou falsa.

☑ Estruturas de decisão:

- Comandos if/else, switch e operador ternário (? :).

☑ Estruturas de repetição:

- Comandos while, do... while e for.

☑ Funções

- Blocos de código reutilizáveis que implementam algoritmos.

Próxima aula

- Capítulo 3 – DOM, formulários e eventos.