DESIGN DE INTERFACE PARA WEB

AULA 9 – TEMPLATE STRING

WALTER TRAVASSOS SARINHO

@WALTEROPROFESSOR

WALTER.TRAVASSOS@UNIPE.EDU.BR

Aula de Hoje

- Template String.
- Estrutura de repetição for in.
- Operador typeof.
- Vamos testar nosso conhecimento no https://www.jschallenger.com/

Template String ou Template Literals

- São uma forma de criar strings em JavaScript que permite a incorporação de expressões dentro delas. Isso torna mais fácil a criação de strings complexas e dinâmicas em comparação com as strings tradicionais em JavaScript, que são delimitadas por aspas simples ou duplas.
- Para criar uma template string em JavaScript, você usa a sintaxe de crase (`) em vez de aspas simples ou duplas.

Exemplo

```
const nome = "João";
const idade = 30;

const mensagem = `Olá, meu nome é ${nome} e eu tenho ${idade} anos.`;
console.log(mensagem);
```

```
Olá, meu nome é João e eu tenho 30 anos.
```

Template String vazia

- Para criar uma template string vazia, você pode usar crases sem conteúdo entre elas.

```
const vazia = ``;
```

- Às vezes, você pode precisar criar uma template string com espaços reservados para **preencher posteriormente com conteúdo dinâmico**.

Template String vazia

Você também pode começar com uma template string vazia e, em seguida, adicionar partes à medida que o código se desenrola. Isso pode ser especialmente útil quando você está construindo strings em um loop ou em uma função que gera saída gradualmente.

```
let mensagem = ``;
mensagem += "Isso é a primeira parte. ";
mensagem += "E aqui está a segunda parte.";
console.log(mensagem);
```

```
Isso é a primeira parte. E aqui está a segunda parte.
```

Comparando com o operador de concatenação

script.js

```
1  const umastring = "programador";
2  const outrastring = "psicólogo";
3  const umaoutrastring = "professor";
4
5  console.log('Você sabia que Walter é ' + umastring);
6  console.log("Você sabia que Walter é " + outrastring);
7  console.log(`Você sabia que Walter é ${umaoutrastring}`);
```

```
Você sabia que Walter é programador
Você sabia que Walter é psicólogo
Você sabia que Walter é professor
```

Você pode usar quebras de linha

As template strings suportam quebras de linha, então você pode criar strings multilinha de forma mais legível.

script.js

```
1  const paragrafo = `
2    Isso é um exemplo de uma
3    string multilinha usando
4    template strings em JavaScript.
5  `;
6  console.log(paragrafo);
```

```
Isso é um exemplo de uma
string multilinha usando
template strings em JavaScript.
```

Você pode usar aspas a vontade

Você pode usar aspas simples ou duplas dentro de uma template string sem escapar delas, o que torna a escrita de strings com citações mais simples.

script.js

```
1  const citacao = `Ela disse: "Isso é incrível!"`;
2
3  console.log(citacao);
```

```
Ela disse: "Isso é incrível!"
```

Você pode usar com objetos

- Você pode usar template strings com objetos em JavaScript para criar strings dinâmicas que **incorporam propriedades dos objetos**.
- Isso é útil quando você deseja criar mensagens ou saídas de texto personalizadas com base nas informações contidas nos objetos.

Template String com Objetos

script.js

```js
const pessoa = {
  nome: "Maria",
  idade: 25,
  cidade: "São Paulo"
};

const mensagem = `Olá, meu nome é ${pessoa.nome}, tenho ${pessoa.idade} anos e moro em ${pessoa.cidade}.`;

console.log(mensagem);
```

```
Olá, meu nome é Maria, tenho 25 anos e moro em São Paulo.
```

Podemos usar Template Strings com métodos e funções mais complexas

script.js

```
1  const pessoa = {
2    nome: "Maria",
3    idade: 25,
4    getNome: function() {return this.nome; },
5    getIdade: function() {return this.idade; }
6  };
7
8  function cumprimento(pessoa) {
9    return `Olá, meu nome é ${pessoa.getNome()} e tenho
   ${pessoa.getIdade()}.`;
10 }
11
12 const mensagem = cumprimento(pessoa);
13 console.log(mensagem);
```

```
Olá, meu nome é Maria e tenho 25.
```

for in

- É uma estrutura de controle em JavaScript que é usada para percorrer as propriedades de um objeto.
- É frequentemente usado para iterar sobre as chaves (nomes de propriedades) de um objeto, permitindo que você acesse os valores associados a essas chaves.

for in

script.js

```
1  const pessoa = {
2      nome: "Maria",
3      idade: 25,
4      cidade: "João Pessoa"
5  };
6
7  for (let chave in pessoa) {
8      console.log(chave + ": " + pessoa[chave]);
9  }
```

```
nome: Maria
idade: 25
cidade: João Pessoa
```

Exercício 1

- Crie uma função que gere uma saudação personalizada com base no período do dia (manhã, tarde, noite) usando template strings.
- Dependendo da hora fornecida como argumento, a função gerará uma saudação apropriada.
- Se a hora for 14 (2 da tarde), a saída será "Bom tarde, tenha um ótimo dia!".

Considere:

- Entre 5h e 12h, manhã.
- Entre 12h e 18h, tarde.
- Entre 18h e 5h noite.

```
console.log(saudacaoDoDia(14));
```

```
Bom tarde, tenha um ótimo dia!
```

Exercício 1 - Resposta

script.js

```js
function saudacaoDoDia(hora) {
  let periodo;
  if (hora < 12) {
    periodo = "manhã";
  } else if (hora < 18) {
    periodo = "tarde";
  } else {
    periodo = "noite";
  }

  return `Bom ${periodo}, tenha um ótimo dia!`;
}

console.log(saudacaoDoDia(14));
```

Questão 1 Resposta: C

A respeito do JavaScript, considere as seguintes afirmativas:

I. As palavras reservadas var, let, const e globals são utilizadas para declaração de variáveis.

II. É uma linguagem de script multiparadigma, baseada em protótipo, e suporta estilos de programação orientada a objetos, imperativo e funcional.

III. O operador === (três símbolos de igual) retorna verdadeiro caso os operandos sejam iguais e do mesmo tipo.

IV. O uso da sintaxe ${expressão} em literais string é denominada Template Strings. A interpolação ocorre em textos delimitados por aspas duplas (“).

Assinale a opção somente com alternativas corretas.

a) I. b) I e IV. c) II e III. d) II, III e IV. e) I, II, III e IV.

Sobre a questão

- **Globals** não é um comando, mas refere-se em geral ao escopo global.
- JavaScript é considerada uma linguagem de programação baseada em **protótipos de objetos**. Isso significa que, em vez de seguir o modelo de classes típico encontrado em linguagens orientadas a objetos tradicionais, como Java ou C++, JavaScript utiliza um sistema de protótipos para a criação de objetos e herança.
- JavaScript permite a programação **imperativa**, que é um estilo de programação que se concentra em declarar os passos a serem executados para atingir um determinado objetivo. Você pode usar estruturas de controle de fluxo, como loops e condicionais, para controlar o comportamento do seu programa.
- JavaScript também suporta a programação funcional. Isso significa que **você pode tratar funções como cidadãos de primeira classe**, passá-las como argumentos para outras funções, retorná-las como valores de outras funções e usar funções de ordem superior, como map, reduce e filter. A introdução de funções de seta (=>) no ES6 tornou a programação funcional mais concisa e expressiva em JavaScript.

Descobrindo o tipo de dado

- O operador **typeof** permite determinar o tipo de um valor ou expressão. Ele retorna uma string que descreve o tipo de dado do valor passado como argumento.
- O **typeof** é amplamente utilizado para realizar verificações de tipo em JavaScript e é especialmente útil quando você precisa tomar decisões com base no tipo de dado.

```
const numero = 3;
console.log(typeof numero);  //retorna number
```

Exemplos

```
typeof 42;     // Retorna "number"
typeof "Olá, Mundo";        // Retorna "string"
typeof true;  // Retorna "boolean"
typeof undefined;   // Retorna "undefined"
typeof null;   // Retorna "object" (isso é considerado um erro de design na linguagem)
typeof [1, 2, 3];        // Retorna "object" (um array é considerado um objeto)
typeof { chave: "valor" };    // Retorna "object" (um objeto literal)
typeof function() {}          // Retorna "function"
```

Retorno do typeof

- Ele **retorna uma string em minúsculas** que representa o tipo de dado. Por exemplo, "number", "string", "boolean", "undefined", "object", "function", entre outros.
- Quando aplicado a objetos (incluindo arrays e objetos literais), o typeof retorna "**object**". Isso ocorre porque o JavaScript não faz uma distinção detalhada entre diferentes tipos de objetos.
- Observe que **typeof null** retorna "object", o que é considerado um erro histórico na linguagem. Isso se deve a razões técnicas históricas e não é considerado um comportamento ideal.
- Ao aplicar o typeof a uma função, ele retorna "function", o que é útil para distinguir funções de outros tipos de objetos.

Podemos usar o console.log com 2 argumentos

Para mostrar também o valor do número após o seu tipo.

numero = 2e5;

console.log(typeof numero, numero);

```
numero = 2e5;
console.log(typeof numero, numero);
```

```
number 200000
```

https://www.jschallenger.com

Acessem!

Exercício 2

a) Cartão de Identificação de Pessoa:

- Crie um objeto que represente uma pessoa com informações como nome, idade e cidade. Use template strings para gerar um cartão de identificação da pessoa.

b) Lista de Compras:

- Crie um objeto que represente uma lista de compras. Use template strings para gerar uma lista legível de itens de compra.

c) Informações do Livro:

- Crie um objeto que contenha informações sobre um livro, como título, autor e ano de publicação. Use template strings para exibir as informações formatadas.

DESIGN DE INTERFACE PARA WEB

AULA 9 – TEMPLATE STRING

WALTER TRAVASSOS SARINHO

@WALTEROPROFESSOR

WALTER.TRAVASSOS@UNIPE.EDU.BR