Introdução a JavaScript

〈2021〉

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI ACATE zap

Anteriormente

- Aula 1
  - Algumas observações
  - Introdução
    - Demonstração
    - História
  - Objeto window
  - Variáveis
  - Tipos de dados

Anteriormente

- Aula 2
    - Operadores
        - aritméticos
        - atribuição
        - relacionais
        - lógicos
        - ternário

Anteriormente

- Aula 3
  - Estruturas de controle de fluxo
    - Condições
    - Repetições

Agenda

- Aula 4
  - Onde adicionar o script no HTML
  - Funções
  - Arrays

Onde adicionar o script no HTML

- Onde adicionar nosso script JS no documento HTML?
- Precisa ser colocado entre <head> ou <body>
- Mas qual posição é a melhor?

Onde adicionar o script no HTML

- Primeiro vamos entender o que o navegador faz quando carrega nosso documento HTML:

1. Busca a página HTML no servidor (e.g. index.html).
2. Começa a interpretar e renderizar o HTML.
3. O interpretador encontra uma tag <script> referenciando um arquivo externo.
4. O navegador solicita o arquivo ao servidor. Enquanto isso, bloqueia a renderização do resto do HTML.
5. Depois de um tempo, o script termina de baixar e é executado imediatamente.
6. O navegador continua a interpretar o resto do documento HTML.

Onde adicionar o script no HTML

- Considerando isso, identificamos dois possíveis problemas de colocar a tag <script> no bloco <head> do nosso HTML:
    - Enquanto o script é baixado, o usuário fica olhando para uma página em branco, vazia, enquanto aguarda tudo ser carregado.
    - O script executa imediatamente após ser baixado, mesmo que o HTML ainda não tenha terminado de renderizar. Isso pode causar algum erro caso o script tente modificar um elemento que ainda não foi “lido” pelo navegador.

Onde adicionar o script no HTML

- Uma possível solução para ambos os problemas:
  - Referenciar o arquivo de script apenas ao final do <body>.
  - Dessa forma o navegador interpreta e já exibe para o usuário toda a página HTML, e só então baixa e executa o script.
  - Porém, pode ocorrer outro problema: a página já carregou e o usuário está visualizando todos os elementos, mas o script não terminou de baixar ainda. Nesse cenário, se o usuário tenta realizar alguma ação que depende de JS, não vai funcionar, e ele pode achar que existe um problema com o site/app.

Onde adicionar o script no HTML

- Outra solução, mais “moderna”:
  - Referenciar o script no <head>, utilizando uma das propriedades abaixo:
    - “async”
    - “defer”

Onde adicionar o script no HTML

- async: baixa os scripts enquanto a página está sendo carregada, ao mesmo tempo, sem bloquear o carregamento do HTML. Assim que o script termina de baixar (independente se o HTML acabou de ser renderizado), ele é executado. Se tiver mais de 1 script externo referenciado com a propriedade async, a ordem de declaração também não é respeitada, ou seja, o primeiro a baixar vai ser o primeiro a ser executado.

Onde adicionar o script no HTML

- defer: também baixa os scripts enquanto a página carrega, sem bloquear o navegador. Mas diferente da “async”, os scripts com essa propriedade são executados somente depois que a página HTML foi totalmente carregada, e na ordem em que foram declarados.

Onde adicionar o script no HTML

- Com essas 2 propriedades, podemos referenciar o script externo no <head>, pois não irá bloquear a renderização do HTML para o usuário.

- Porém, com “async”, ainda pode ocorrer o problema de o script baixar e executar antes do navegador “ler” todo o HTML, e se o script tentar utilizar um elemento que ainda não foi “descoberto”, pode causar erro.

Onde adicionar o script no HTML

- Você pode fazer da forma que achar melhor.

- Dependendo do conteúdo do seu script, uma dessas formas pode ser mais interessante que a outra.

- Porém, na maior parte das vezes, referenciar o script externo no <head>, utilizando a propriedade *defer*, parece ser a melhor alternativa.

Funções

- São tarefas, rotinas, ações que são executadas somente quando são chamadas ou quando ocorre algum evento.

- Podem receber parâmetros e retornar um resultado.

```
function nomeDaFuncao(parametros) {
    /*
    sequência de instruções
    a serem executadas
    quando a função for chamada
    */

    return valorResultado;
}
```

Funções

Exemplo:

```
<div id="divSaque">
    <label for="valorSaque">Digite o valor do saque:</label> <br>
    <input type="number" name="valorSaque" id="inputSaque" value="0.0">
    <button id="btnSaque">Sacar</button>
</div>
```

```
var cliente = {nome: "João", dataNasc: "25/02/1991", saldo: "100.00"}

var btnSaque = document.getElementById("btnSaque");
var inputSaque = document.getElementById("inputSaque");

btnSaque.addEventListener('click', function(){
    sacar(Number(inputSaque.value));
});
```

Funções

Exemplo:

```
function sacar(valor) {
    if (valor <= cliente.saldo) {
        cliente.saldo -= valor;
        alert("Saque de R$" + valor + " realizado com sucesso. \n" +
            "O novo saldo é de: R$" + cliente.saldo);
    } else {
        alert("Saldo atual insuficiente: R$" + cliente.saldo);
    }
}
```

Funções

Exemplo com “return”:

```
function parOuImpar(numero) {
    if (numero % 2 === 0) {
        return "Par";
    } else {
        return "Ímpar";
    }
}

alert("O saldo atual de R$" + cliente.saldo + " é um valor " + parOuImpar(cliente.saldo) + ".");
```

Funções

Também podemos guardar funções dentro de variáveis:

```
var fatorial = function(num) {
    var fat = 1;
    for (var i = num; i > 1; i--) {
        fat *= i;
    }
    return fat;
}

console.log(fatorial(5));
```

Arrays

- Arrays são variáveis compostas.

- Ao contrário de uma variável simples, que guarda só 1 valor (string, number, boolean), uma variável composta guarda uma lista de valores.

- Exemplo:

```
var listaDeCarros = ['uno', 'fusca', 'sandero', 'gol', 'hb20', 'gurgel'];
var anosDeCopa = [2022, 2018, 2014, 2010, 2006, 2002, 1998];
```

Arrays

- Operações com arrays:
  - Verificamos o tamanho (quantidade de itens) através da propriedade length:

```
var listaDeCarros = ['uno', 'fusca', 'sandero', 'gol', 'hb20', 'gurgel'];
var tamanho = listaDeCarros.length;
```

  - Acessamos um item específico através do índice:

```
var anosDeCopa = [2022, 2018, 2014, 2010, 2006, 2002, 1998];
var primeiro = anosDeCopa[0];
var último = anosDeCopa[anosDeCopa.length - 1];
```

Arrays

- Operações com arrays:
  - Removemos um item do final do array através da função pop:

```
var listaDeCarros = ['uno', 'fusca', 'sandero', 'gol', 'hb20', 'gurgel'];
listaDeCarros.pop();
```

  - Adicionamos um item ao final através da função push:

```
var anosDeCopa = [2022, 2018, 2014, 2010, 2006, 2002, 1998];
anosDeCopa.push(1994);
```

Arrays

- Operações com arrays:
  - Removemos um item do início através da função shift:

```
var listaDeCarros = ['uno', 'fusca', 'sandero', 'gol', 'hb20', 'gurgel'];
listaDeCarros.shift();
```

  - Adicionamos um item ao início através da função unshift:

```
var anosDeCopa = [2022, 2018, 2014, 2010, 2006, 2002, 1998];
anosDeCopa.unshift(2026);
```

Arrays

- Operações com arrays:
  - Descobrimos o índice de um item através da função indexOf:

```
var listaDeCarros = ['uno', 'fusca', 'sandero', 'gol', 'hb20', 'gurgel'];
listaDeCarros.indexOf('fusca');
```

  - Ordenamos por ordem crescente (ou alfabética) através da função sort:

```
var anosDeCopa = [2022, 2018, 2014, 2010, 2006, 2002, 1998];
anosDeCopa.sort();
```

Material complementar

- Onde colocar a tag <script>: https://stackoverflow.com/a/24070373/4326295 (inglês) ou um artigo do Google traduzido: https://developers.google.com/speed/docs/insights/BlockingJS

- Funções em JS (pt-br): Funções - JavaScript | MDN

- Array em JS (pt-br): Array - JavaScript | MDN

Obrigado

SENAI ACATE zap