UNIPAC
Barbacena

Centro Universitário Presidente Antônio Carlos
Programação para Internet

Arrays JavaScript

Felipe Roncalli de Paula Carneiro
felipecarneiro@unipac.br

O que vamos aprender nessa aula

- Introdução;
- Importância dos arrays no desenvolvimento web;
- Características e Uso Comum de Arrays na Programação;
- Características dos Arrays;
- Propriedades do Array;
- Percorrendo Arrays;
- Arrays Multidimensionais;

O que é um array?

Um array é uma estrutura de dados em programação que representa uma coleção ordenada de elementos do mesmo tipo. Cada elemento é identificado por um índice, que começa a partir de 0 e aumenta sequencialmente. Os arrays são usados para armazenar e organizar conjuntos de dados de maneira eficiente, permitindo acesso rápido e manipulação de elementos individuais.

Importância dos arrays no desenvolvimento web

Arrays são estruturas de dados fundamentais em programação e desempenham um papel crucial no desenvolvimento web. Eles permitem armazenar e gerenciar coleções de elementos de forma organizada, o que é essencial para lidar com informações em aplicações web.

Importância dos arrays no desenvolvimento web

Armazenamento de Dados Estruturados:

Em muitos casos, você precisa lidar com conjuntos de dados relacionados. Arrays fornecem uma maneira conveniente de armazenar esses dados de forma estruturada. Por exemplo, em uma lista de usuários de um site, você pode usar um Array para armazenar informações como nome, email e idade de cada usuário.

Importância dos arrays no desenvolvimento web

Manipulação de Listas e Itens:

Páginas web frequentemente exibem listas de itens, como posts de blog, produtos de uma loja online ou mensagens em uma rede social. Arrays permitem que você armazene esses itens e os manipule de maneira eficiente, como adicionar, remover, reordenar ou modificar itens.

Importância dos arrays no desenvolvimento web

Renderização Dinâmica:

Em frameworks de desenvolvimento front-end, como React, Angular e Vue, Arrays são frequentemente usados para representar listas de elementos que precisam ser renderizados dinamicamente na interface do usuário. Isso permite que você crie interfaces interativas e atualizadas conforme os dados mudam.

Importância dos arrays no desenvolvimento web

Trabalho com Dados de Formulários:

Em formulários web, os dados frequentemente são coletados em forma de Arrays, como seleções múltiplas ou campos repetidos. A capacidade de acessar e processar esses dados como Arrays facilita a validação e o tratamento dos dados do usuário.

Importância dos arrays no desenvolvimento web

Gerenciamento de Eventos e Interações:

Ao lidar com eventos em páginas web, é comum armazenar informações sobre os eventos em Arrays. Isso permite o rastreamento de interações do usuário, como cliques, envios de formulários e movimentos do mouse, para análises e feedback.

Importância dos arrays no desenvolvimento web

Armazenamento de Dados em Cache:

Em algumas situações, você pode armazenar temporariamente dados em um Array para melhorar o desempenho, evitando solicitações excessivas ao servidor. Isso é especialmente útil em casos de uso em que você precisa exibir os mesmos dados várias vezes.

Importância dos arrays no desenvolvimento web

Estruturas de Dados Complexas:

Arrays podem ser usados para criar estruturas de dados mais complexas, como filas, pilhas e matrizes multidimensionais, que são úteis em várias situações, como gerenciamento de tarefas, histórico de navegação e representação de dados tabulares.

Importância dos arrays no desenvolvimento web

Algoritmos e Lógica de Programação:

Muitos algoritmos e problemas de lógica de programação envolvem a manipulação e o processamento de conjuntos de dados. Arrays fornecem uma base sólida para resolver esses problemas de maneira eficiente.

Características e Uso Comum de Arrays na Programação

Arrays são uma das estruturas de dados fundamentais e amplamente usadas na programação. Eles possuem várias características distintas e são aplicados de maneira frequente em diversos cenários.

Características dos Arrays

- **Coleção Ordenad**a: Arrays armazenam elementos em uma sequência ordenada, o que significa que a posição de cada elemento é determinada pelo seu índice.
- **Acesso por Índice**: Cada elemento em um Array é acessado por meio de um índice inteiro não negativo. O primeiro elemento tem índice 0, o segundo índice 1 e assim por diante.
- **Homogeneidade**: Um Array contém elementos do mesmo tipo de dados. Por exemplo, um Array pode conter apenas números inteiros, strings, objetos, etc.

Características dos Arrays

- **Tamanho Fixo ou Dinâmico:** Alguns Arrays têm tamanho fixo, o que significa que o número de elementos não pode ser alterado após a criação. Outros Arrays têm tamanho dinâmico, permitindo a adição e remoção de elementos.
- **Alocação Contígua de Memória:** Em muitas linguagens, os elementos de um Array são armazenados em locais de memória contíguos, o que facilita o acesso eficiente aos elementos.

Usos Comuns de Arrays na Programação

- **Armazenamento de Dados:** Arrays são usados para armazenar coleções de dados relacionados, como listas de números, nomes de usuários, registros de produtos, etc.
- **Iteração e Processamento:** Arrays são frequentemente percorridos usando loops, como for e for...of, para realizar operações em cada elemento, como cálculos, validações e transformações.
- **Representação de Matrizes e Tabelas:** Arrays multidimensionais, também conhecidos como matrizes, são usados para representar estruturas de dados bidimensionais, como grades e tabelas.

Usos Comuns de Arrays na Programação

- **Manipulação de Strings:** Em algumas linguagens, como JavaScript, strings podem ser tratadas como Arrays de caracteres. Isso permite acessar e manipular caracteres individuais da string.
- **Implementação de Filas e Pilhas:** Arrays podem ser usados para implementar estruturas de dados como filas (FIFO) e pilhas (LIFO), que são utilizadas para gerenciar sequências de elementos.

Usos Comuns de Arrays na Programação

- **Armazenamento de Resultados Intermediários:** Em algoritmos complexos, Arrays podem ser usados para armazenar resultados intermediários ou estados durante a execução.
- **Organização de Dados em Aplicações Web:** Arrays são usados para gerenciar dados em aplicações web, como listas de comentários, feeds de notícias e listas de reprodução de mídia.
- **Manipulação de Dados de Formulários:** Dados coletados de formulários web, como seleções múltiplas, podem ser armazenados em Arrays para posterior processamento.

Aplicações de Arrays na Web

- **Redes Sociais**
- **Catálogos de Produtos**
- **Sistema de Reservas**
- **Aplicativos de Mídia**
- **Aplicações de Calendário**

Esses são apenas alguns exemplos, mas a verdade é que os Arrays são amplamente aplicáveis em quase todos os tipos de aplicativos e sistemas de software, onde a organização e manipulação de dados são essenciais. Eles fornecem uma maneira eficiente e estruturada de lidar com conjuntos de informações relacionadas.

Sintaxe Básica Array JavaScript

```
1  var nomeDoArray = [elemento1, elemento2, elemento3, ...];
2
```

Onde:

nomeDoArray: é o nome que você dá ao seu Array.

elemento1, elemento2, elemento3, ...: são os elementos individuais que você deseja armazenar no Array.

Eles podem ser números, strings, objetos, outros Arrays, ou qualquer outro tipo de dado suportado por JavaScript.

Exemplos de Arrays - JavaScript

```
1  // Um Array de números inteiros
2  var numeros = [1, 2, 3, 4, 5];
3
4  // Um Array de strings
5  var frutas = ["maçã", "banana", "laranja"];
6
7  // Um Array de objetos
8  var pessoas = [
9    { nome: "João", idade: 25 },
10   { nome: "Maria", idade: 30 },
11   { nome: "Carlos", idade: 22 }
12 ];
13
14 // Um Array misto
15 var misto = [1, "texto", true, { chave: "valor" }];
16
```

Exemplos de Arrays - JavaScript

```
1  var meuArrayVazio = [];
2  meuArrayVazio.push("primeiro elemento");
3  meuArrayVazio.push("segundo elemento");
```

Acessar elementos através de índice

Acessar elementos por **índices** em um Array em JavaScript é uma operação fundamental. Você pode usar a notação de **colchetes** ([]) para acessar elementos específicos com base nos seus índices. Lembre-se de que os índices dos Arrays começam a partir de **0**

```
1  var frutas = ["maçã", "banana", "laranja"];
2  console.log(frutas[0]); // Saída: "maçã"
3  console.log(frutas[1]); // Saída: "banana"
4  console.log(frutas[2]); // Saída: "laranja"
5
```

Acessar elementos através de índice

Pode-se usar variáveis ou expressões para indicar o índice:

```
1  var indice = 1;
2  console.log(frutas[indice]); // Saída: "banana"
3
4  console.log(frutas[1 + 1]); // Saída: "laranja"
5
```

Se o índice que você passar estiver fora do intervalo válido, o resultado será undefined:

```
1  console.log(frutas[3]); // Saída: undefined (índice fora do intervalo)
2
```

Acessar elementos através de índice

Pode-se usar variáveis ou expressões para indicar o índice:

```
1  var valores = ["A", "B", "C"];
2
3  console.log(valores[1 + 1]); // Saída: "C"
4  console.log(valores[Math.floor(1.5)]); // Saída: "B"
5
```

Acessar elementos através de índice

Embora seja mais comum usar números inteiros como índices para acessar elementos em um Array, é possível usar valores não inteiros, strings ou até mesmo objetos como índices em JavaScript. No entanto, essa prática é menos comum e pode levar a resultados confusos se não for utilizada com cuidado.

```
1  var dados = {
2      "chave1": "valor1",
3      "chave2": "valor2"
4  };
5
6  console.log(dados["chave1"]); // Saída: "valor1"
7  console.log(dados["chave2"]); // Saída: "valor2"
8
```

Tipos de Arrays

Podemos fdefinir Arrays de acordos com tipos primitivos ou objetos

- **Array de Números Inteiros:**

```
1  var numeros = [1, 2, 3, 4, 5];
2
```

- **Array de Strings**

```
1  var frutas = ["maçã", "banana", "laranja"];
2
```

Tipos de Arrays

Podemos fdefinir Arrays de acordos com tipos primitivos ou objetos

- **Array de Números Inteiros:**

```
1  var numeros = [1, 2, 3, 4, 5];
2
```

- **Array de Strings**

```
1  var frutas = ["maçã", "banana", "laranja"];
2
```

- **Array de Booleanos**

```
1  var estados = [true, false, true, true, false];
2
```

Tipos de Arrays

```
1   //Array de Objetos
2   var pessoas = [
3       { nome: "João", idade: 25 },
4       { nome: "Maria", idade: 30 },
5       { nome: "Carlos", idade: 22 }
6   ];
7
8   //Array Multidimensional
9   var matriz = [
10      [1, 2, 3],
11      [4, 5, 6],
12      [7, 8, 9]
13  ];
14
15  //Array de Arrays
16  var listas = [
17      ["a", "b", "c"],
18      [1, 2, 3, 4],
19      [true, false]
20  ];
21
```

Tipos de Arrays

```
22    //Array de Dados Mistos
23    var misto = [
24      "texto", 42, true,
25      { chave: "valor" },
26      ["a", "b", "c"]
27    ];
28
29    //Array de datas
30    var datas = [
31      new Date("2023-08-22"),
32      new Date("2023-09-15"),
33      new Date("2023-10-10")
34    ];
35
```

Tipos de Arrays

```
36  //Array de Registros (json)
37  var livros = [
38      {
39          titulo: "A Guerra dos Tronos",
40          autor: "George R. R. Martin",
41          ano: 1996
42      },
43      {
44          titulo: "1984",
45          autor: "George Orwell",
46          ano: 1949
47      }
48  ];
49
```

Principais propriedades dos Arrays

length: Essa propriedade retorna o número de elementos em um Array. É uma maneira rápida de descobrir o tamanho atual do Array.

```
1   var frutas = ["maçã", "banana", "laranja"];
2   console.log(frutas.length); // Saída: 3
3
```

Principais propriedades dos Arrays

push: O método push() é usado para adicionar um ou mais elementos ao final de um Array. Ele retorna o novo tamanho do Array após a adição.

```
6
7   var frutas = ["maçã", "banana"];
8   frutas.push("laranja");
9   console.log(frutas); // Saída: ["maçã", "banana", "laranja"]
10
11
12
```

Principais propriedades dos Arrays

pop: O método pop() é usado para remover o último elemento de um Array. Ele retorna o elemento removido.

```
var frutas = ["maçã", "banana", "laranja"];
var ultimaFruta = frutas.pop();
console.log(ultimaFruta); // Saída: "laranja"
console.log(frutas); // Saída: ["maçã", "banana"]
```

Principais propriedades dos Arrays

shift: O método shift() é usado para remover o primeiro elemento de um Array. Ele retorna o elemento removido.

```
0
1   var frutas = ["maçã", "banana", "laranja"];
2   var primeiraFruta = frutas.shift();
3   console.log(primeiraFruta); // Saída: "maçã"
4   console.log(frutas); // Saída: ["banana", "laranja"]
5
6
```

Principais propriedades dos Arrays

unshift: O método unshift() é usado para adicionar um ou mais elementos no início de um Array. Ele retorna o novo tamanho do Array após a adição.

```
7
8   var frutas = ["banana", "laranja"];
9   frutas.unshift("maçã");
0   console.log(frutas); // Saída: ["maçã", "banana", "laranja"]
1
2
3
```

Principais propriedades dos Arrays

indexOf: O método indexOf() é usado para encontrar o índice do primeiro elemento correspondente em um Array. Se não encontrar, retorna -1.

```
4   var frutas = ["maçã", "banana", "laranja"];
5   var indiceBanana = frutas.indexOf("banana");
6   console.log(indiceBanana); // Saída: 1
7
```

Principais propriedades dos Arrays

splice: O método splice() é usado para alterar o conteúdo de um Array, adicionando ou removendo elementos. Ele pode ser usado para remover elementos em um índice específico e/ou adicionar novos elementos no mesmo índice.

```
39
40  var numeros = [1, 2, 3, 4, 5];
41  numeros.splice(2, 1); // Remove 1 elemento no índice 2
42  console.log(numeros); // Saída: [1, 2, 4, 5]
43
44  numeros.splice(1, 0, 6, 7); // Adiciona 6 e 7 no índice 1
45  console.log(numeros); // Saída: [1, 6, 7, 2, 4, 5]
46
47
```

Principais propriedades dos Arrays

Essas propriedades e métodos são fundamentais para manipulação de Arrays em JavaScript. Eles permitem adicionar, remover e modificar elementos em Arrays, o que é crucial para muitas tarefas de programação.

Percorrendo Arrays

Geralmente, percorrer Arrays é uma tarefa fundamental para qualquer linguagem de programação, pois permite lidar com coleções de dados de maneira eficiente e aplicar lógica e operações em massa a esses dados. Em JavaScript, existem várias formas de percorrer Arrays, como loops for, for...of, while, e métodos como forEach(), map(), filter(), entre outros. Cada abordagem tem suas vantagens e é escolhida com base no contexto e nas necessidades específicas do problema.

Percorrendo Arrays

Geralmente, percorrer Arrays é uma tarefa fundamental para qualquer linguagem de programação, pois permite lidar com coleções de dados de maneira eficiente e aplicar lógica e operações em massa a esses dados. Em JavaScript, existem várias formas de percorrer Arrays, como loops for, for...of, while, e métodos como forEach(), map(), filter(), entre outros. Cada abordagem tem suas vantagens e é escolhida com base no contexto e nas necessidades específicas do problema.

Percorrendo Arrays

Usando um Loop **for**: Você pode usar um loop for para iterar através dos elementos de um Array usando seus índices.

```
1  var frutas = ["maçã", "banana", "laranja"];
2
3  console.log("Usando loop for:");
4  for (var i = 0; i < frutas.length; i++) {
5    console.log(frutas[i]);
6  }
```

Percorrendo Arrays

Usando um Loop **for...of:** O loop **for...of** é uma maneira mais simples de percorrer um Array, pois ele itera diretamente sobre os elementos do Array.

```
1  var frutas = ["maçã", "banana", "laranja"];
2
3  console.log("\nUsando loop for...of:");
4  for (var fruta of frutas) {
5    console.log(fruta);
6  }
7
```

Percorrendo Arrays

Usando o Método **forEach()**: O método **forEach()** é uma alternativa mais legível e eficiente para percorrer Arrays. Ele executa uma função fornecida para cada elemento do Array.

```
1  var frutas = ["maçã", "banana", "laranja"];
2
3  console.log("\nUsando método forEach():");
4  frutas.forEach(function(fruta) {
5    console.log(fruta);
6  });
7
```

Percorrendo Arrays

Break e **continue** são palavras-chave usadas em loops em várias linguagens de programação, incluindo JavaScript, para controlar o fluxo de execução do loop.
Eles são especialmente úteis quando você deseja interromper o loop ou pular para a próxima iteração com base em certas condições.

Percorrendo Arrays

```
1   for (var i = 0; i < 5; i++) {
2     if (i === 3) {
3       break; // Interrompe o loop quando i for igual a 3
4     }
5     console.log(i);
6   }
7
8   for (var i = 0; i < 5; i++) {
9     if (i === 2) {
10      continue; // Pula a iteração quando i for igual a 2
11    }
12    console.log(i);
13  }
14
```

Arrays Multidimensionais

Arrays multidimensionais, também conhecidos como matrizes, são uma extensão dos Arrays simples em que cada elemento é um Array por si só. Isso permite representar estruturas de dados bidimensionais ou até mesmo tridimensionais e superiores, onde os dados são organizados em linhas e colunas, formando uma espécie de "grade" ou tabela.

Arrays Multidimensionais

Uma matriz multidimensional é essencialmente um Array de Arrays, em que cada Array interno representa uma "sub-array" ou uma linha da matriz. Cada sub-array pode ter um número diferente de elementos, mas todas as sub-arrays devem ter a mesma quantidade de elementos na mesma dimensão.

Arrays Multidimensionais

```
1  var matriz2D = [
2      [1, 2, 3],
3      [4, 5, 6],
4      [7, 8, 9]
5  ];
6
7  var matriz3D = [
8      [
9          [1, 2],
10         [3, 4]
11     ],
12     [
13         [5, 6],
14         [7, 8]
15     ]
16 ];
17
```

Arrays Multidimensionais

```
1  var elemento = matriz2D[1][2];
2  // Acessa o elemento na segunda linha
3  // e terceira coluna (valor 6)
4
5  var elemento = matriz3D[1][0][0];
6  // Acessa o elemento no segundo registro,
7  //primeira linha e primeira coluna (valor 5)
8
9
```

Exercício

Dúvidas??