



Arrays


Para iniciar nossa aula sobre Arrays, vamos começar a entender o que são e como funcionam essas estruturas de dados fundamentais em JavaScript.

Arrays são utilizados para armazenar e manipular coleções de elementos em uma única variável, oferecendo uma maneira eficiente de gerenciar múltiplos valores simultaneamente, sejam eles números, textos, objetos ou outros tipos de dados. Ao longo desta aula, exploraremos desde a definição básica dos arrays até sua criação e manipulação, incluindo o uso de métodos específicos que facilitam operações como adição, remoção e combinação de elementos.

Definição de Arrays

Nesta primeira parte da nossa aula, vamos explorar o conceito fundamental dos arrays, um tipo de dado essencial na programação em JavaScript. Arrays são estruturas de dados que nos permitem armazenar e manipular uma coleção de elementos em uma única variável. Diferentemente de uma variável comum, que pode armazenar apenas um único valor, seja ele um texto, um número ou outro tipo de dado, um array nos oferece a capacidade de armazenar múltiplos valores simultaneamente. Esses valores podem ser de diferentes tipos, como textos, números, objetos ou decimais, o que torna os arrays extremamente versáteis.

Em termos práticos, ao declarar um array em JavaScript, podemos inserir uma sequência de elementos que serão organizados em uma lista. Essa lista pode ser ordenada ou não, dependendo da nossa necessidade. Por exemplo, imagine que desejamos armazenar uma lista de números como [1, 2, 3, 4, 5]. Utilizando um array, podemos manter todos esses valores dentro de uma única estrutura de dados, o que facilita o gerenciamento e a manipulação dessas informações.

Além de armazenar diversos tipos de dados, os arrays em JavaScript são acessados através de índices numéricos, começando do zero. Isso significa que, para acessar o primeiro elemento de um array, utilizamos o índice 0; para o segundo, o índice 1; e assim por diante. Por exemplo, se tivermos um array chamado `meuArray` com os valores `[10, 20, 30]`, `'meuArray[0]'` retornará 10, `'meuArray[1]'` retornará 20, e `'meuArray[2]'` retornará 30. Essa forma de acesso é extremamente útil quando precisamos iterar sobre os elementos de um array ou acessar elementos específicos de forma eficiente. 

Uma das principais vantagens dos arrays é sua capacidade de lidar com listas ordenadas, permitindo-nos controlar a ordem de inserção e remoção dos elementos. Essa característica é particularmente útil quando precisamos manter uma sequência lógica de dados, como uma fila de tarefas a serem executadas. Contudo, os arrays também são capazes de lidar com listas não ordenadas, oferecendo flexibilidade no gerenciamento dos dados.

Por fim, os arrays em JavaScript vêm acompanhados de uma série de métodos nativos que facilitam sua manipulação. Métodos como `'push()'`, `'pop()'`, `'shift()'`, e `'unshift()'` nos permitem adicionar ou remover elementos dos arrays de maneira prática e eficiente. Além disso, métodos mais avançados, como `'map()'`, `'filter()'`, e `'reduce()'`, nos oferecem ferramentas poderosas para transformar e analisar os dados armazenados nos arrays.

Criando e manipulando arrays

Nesta segunda parte da nossa aula, vamos focar na criação e manipulação dessas estruturas em JavaScript. Para declarar um array, a maneira mais básica e comum é utilizar colchetes. Diferente das chaves, que usamos para definir blocos de código, ou dos parênteses, utilizados para definir parâmetros de funções, os colchetes são a estrutura que identifica um array em JavaScript. Para começar, declaramos uma variável usando `'let'`, seguido do nome da variável e, em seguida, atribuímos os valores desejados dentro de colchetes.

Por exemplo, ao criar um array de frutas, podemos declarar a variável 'frutas' da seguinte forma: `let frutas = ['maçã', 'banana', 'laranja'];`. Esse array agora contém 3 elementos, que podem ser acessados individualmente através de seus índices. Lembre-se de que o primeiro elemento sempre tem o índice zero, o segundo índice um, e assim por diante. Se quisermos acessar a "banana", que é o segundo elemento da lista, utilizamos `frutas[1]`, e para acessar a "maçã", usamos `frutas[0]`.

Manipular arrays em JavaScript também é bastante simples. Para modificar um elemento específico, basta acessar o índice correspondente e atribuir um novo valor. Por exemplo, se quisermos substituir "laranja" por "morango", usamos `frutas[2] = 'morango';`. Dessa forma, o terceiro elemento do array, que antes era "laranja", agora será "morango". Essa modificação será refletida em todo o código subsequente.

Além de modificar elementos individuais, também podemos substituir todo o conteúdo de um array. Por exemplo, se quisermos redefinir o array 'frutas', podemos atribuir novos valores da seguinte forma: `frutas = ['mamão', 'pêssego', 'banana'];`. Agora, o array 'frutas' conterá esses novos elementos, com "mamão" no índice zero, "pêssego" no índice um, e "banana" no índice dois. Se quisermos visualizar todo o array no console, basta usar `console.log(frutas);`, que exibirá todos os elementos armazenados.

Outro aspecto importante ao trabalhar com arrays é entender como o JavaScript lida com a exibição dos dados. Quando imprimimos um array no console, o JavaScript apresenta o conteúdo em uma estrutura que inclui tanto os valores quanto seus índices correspondentes. Isso nos permite verificar facilmente a posição de cada elemento dentro do array e confirmar se as manipulações realizadas foram bem-sucedidas.

Praticando uso de Arrays

Agora, vamos focar na prática de criação e manipulação dessas estruturas em JavaScript, consolidando os conceitos apresentados anteriormente. Iniciaremos

criando novos arrays e manipulando seus elementos, utilizando operações comuns que são essenciais no desenvolvimento de software.



Para começar, a maneira mais básica de declarar um array em JavaScript é utilizando colchetes. Diferente das chaves, que delimitam blocos de código, e dos parênteses, que definem parâmetros de funções, os colchetes indicam que estamos trabalhando com um array. Por exemplo, podemos criar um array de frutas com a seguinte declaração: `let frutas = ['maçã', 'banana', 'laranja'];`. Esse array agora contém três elementos que podem ser acessados individualmente pelos seus índices. Lembre-se de que o primeiro elemento sempre tem o índice zero, o segundo índice um, e assim por diante.

Uma vez declarado, podemos manipular os elementos do array de diversas maneiras. Para acessar um elemento específico, usamos seu índice entre colchetes. Por exemplo, `frutas[1]` retorna “banana”, que é o segundo elemento do array. Além de acessar, também podemos modificar elementos. Se quisermos alterar “laranja” para “morango”, podemos fazer isso com `frutas[2] = 'morango';`. Essa alteração será refletida imediatamente no array, substituindo o antigo valor pelo novo.

Outra operação útil é a substituição completa do array. Podemos redefinir o conteúdo de `frutas` atribuindo um novo conjunto de valores, como `frutas = ['mamão', 'pêssego', 'banana'];`. Dessa forma, o array passa a conter esses novos elementos, com “mamão” no índice zero, “pêssego” no índice um, e “banana” no índice dois. Para visualizar o array completo no console, utilizamos `console.log(frutas);`, que exibirá todos os elementos armazenados.

Durante a manipulação de arrays, é importante observar como o JavaScript exibe os dados no console. Ele apresenta o array com seus elementos e respectivos índices, facilitando a verificação e o debug do código. Por exemplo, ao imprimir `console.log(frutas)`, o console mostrará `['mamão', 'pêssego', 'banana']`, com o índice zero correspondente a “mamão”, o índice um a “pêssego”, e o índice dois a “banana”. Isso ajuda a garantir que as operações realizadas estão corretas.

Métodos de Arrays


Na última parte da nossa aula sobre Arrays, vamos explorar os principais métodos disponíveis para manipulação dessas estruturas em JavaScript. Entender como utilizar métodos como 'push', 'pop', 'shift', 'unshift' e 'concat' é essencial para trabalhar eficientemente com arrays, facilitando a adição, remoção e combinação de elementos.

Começamos pelos métodos 'push' e 'pop'. O método 'push' é utilizado para adicionar um novo elemento ao final de um array. Por exemplo, se temos um array de frutas (`let frutas = ['mamão', 'pêssego', 'banana']`), podemos adicionar "morango" ao final do array com `frutas.push('morango')`. Ao exibir o array no console, veremos que "morango" foi corretamente inserido na última posição. Já o método 'pop' faz o oposto: ele remove o último elemento do array. Assim, se executarmos `frutas.pop()`, o elemento "morango" será removido, e o array retornará ao seu estado anterior.

Os métodos 'shift' e 'unshift' funcionam de maneira semelhante, mas atuam no início do array. O método 'unshift' adiciona um novo elemento na primeira posição, deslocando os outros elementos para a direita. Se quisermos adicionar "laranja" ao início do array de frutas, utilizamos `frutas.unshift('laranja')`, e "laranja" se tornará o primeiro elemento. Já o método 'shift' remove o primeiro elemento do array. Se executarmos `frutas.shift()`, o elemento "laranja" será removido, e os elementos restantes serão reposicionados.

Outro método importante é o 'concat', que nos permite combinar dois ou mais arrays em um novo array. Suponha que temos dois arrays: um de frutas e outro de objetos (`let objetos = ['geladeira', 'mesa', 'cadeira']`). Podemos combinar esses arrays com `let novoArray = frutas.concat(objetos)`. O resultado será um novo array que contém todos os elementos de ambos os arrays originais, com as frutas seguidas pelos objetos.

Esses métodos nativos de array são fundamentais para garantir uma manipulação de dados eficiente e organizada em JavaScript. Eles nos permitem adicionar, remover e combinar elementos de forma prática, sem deixar espaços vazios no

array, o que poderia ocorrer ao manipular os índices diretamente. Portanto, é sempre recomendável utilizar esses métodos para manter a integridade e a ordem dos elementos no array. 

Concluimos aqui nossa aula sobre Arrays, uma estrutura de dados indispensável no desenvolvimento de software. Com o conhecimento adquirido, você agora pode manipular arrays com confiança, utilizando os métodos mais adequados para cada situação.

Conteúdo Bônus

Para auxiliar nos estudos sobre operadores, recomendo assistir ao vídeo “O que todo iniciante em Front End deveria aprender | Como manipular arrays e objetos em JavaScript”, que está disponível no canal Mario Souto - Dev Soutinho no YouTube.

Referência Bibliográfica

ASCENCIO, A. F. G.; CAMPOS, E. A. V. de. Fundamentos da programação de computadores. 3.ed. Pearson: 2012.

BRAGA, P. H. Teste de software. Pearson: 2016.

GALLOTTI, G. M. A. (Org.). Arquitetura de software. Pearson: 2017.

GALLOTTI, G. M. A. Qualidade de software. Pearson: 2015.

MEDEIROS, E. Desenvolvendo software com UML 2.0 definitivo. Pearson: 2004.

PFLIEGER, S. L. Engenharia de software: teoria e prática. 2.ed. Pearson: 2003.

SOMMERVILLE, I. Engenharia de software. 10.ed. Pearson: 2019.



Atividade Prática 11 - Trabalhando com Arrays em JavaScript

Objetivos:

- Compreender e aplicar a criação e manipulação de arrays em JavaScript.
- Utilizar métodos nativos de arrays para adicionar, remover e combinar elementos.
- Desenvolver habilidades práticas na manipulação de coleções de dados utilizando arrays.

Materiais, Métodos e Ferramentas:

- Computador com um editor de código (Visual Studio Code, Sublime Text, etc.)
- Navegador web (para executar e testar o código JavaScript)
- Acesso à documentação do JavaScript (MDN Web Docs ou similar)

Atividade Prática

Primeiramente, leia atentamente o texto a seguir:

Nesta atividade, você irá criar e manipular arrays em JavaScript, aplicando conceitos básicos e avançados para gerenciar coleções de dados. A atividade envolve a criação de arrays, a modificação de seus elementos e a utilização de métodos nativos para realizar operações comuns.

Agora, vamos praticar!

PASSO A PASSO DETALHADO DA ATIVIDADE:

- Criação e Manipulação de Arrays:



- Crie um arquivo HTML e inclua um arquivo JavaScript.
 - No arquivo JavaScript, declare um array chamado `numeros` com os valores `[5, 10, 15, 20, 25]`.
 - Acesse e exiba o terceiro elemento do array no console.
 - Substitua o quarto elemento por 30 e exiba o array atualizado no console.
 - Modifique o array para conter os valores `[1, 2, 3, 4, 5]` e exiba o array modificado no console.
- Uso de Métodos de Arrays:
 - Declare um array chamado `frutas` com os valores `['maçã', 'banana', 'laranja']`.
 - Utilize o método `push` para adicionar o valor `'manga'` ao final do array e exiba o array atualizado no console.
 - Utilize o método `pop` para remover o último elemento do array e exiba o array atualizado no console.
 - Utilize o método `unshift` para adicionar o valor `'abacaxi'` ao início do array e exiba o array atualizado no console.
 - Utilize o método `shift` para remover o primeiro elemento do array e exiba o array atualizado no console.

- Combinação de Arrays:



- Crie dois arrays: `numeros1` com os valores `[1, 2, 3]` e `numeros2` com os valores `[4, 5, 6]`.
- Utilize o método `concat` para combinar os dois arrays em um novo array chamado `todosNumeros`.
- Exiba o array `todosNumeros` no console.

Gabarito Esperado

- Criação e Manipulação de Arrays: o array `numeros` deve ser criado e manipulado corretamente, com o acesso e modificação dos elementos funcionando como esperado.
- Uso de Métodos de Arrays: os métodos `push`, `pop`, `unshift` e `shift` devem ser aplicados corretamente, com os resultados atualizados exibidos conforme o esperado.
- Combinação de Arrays: o método `concat` deve combinar os arrays `numeros1` e `numeros2` corretamente, resultando em um novo array `todosNumeros` com todos os valores combinados.

Certifique-se de que todos os resultados estejam corretos e que as operações sejam executadas conforme especificado.

Ir para exercício