



Utilizando Arrays

Introdução

Arrays são uma ferramenta poderosa e abrangente no Javascript.

São muito intuitivos de usar e permite fazer quase tudo com ele.

No entanto, existem diferenças importantes entre arrays em Javascript e outras linguagens convencionais. Conhecê-los ajudará você a liberar seu verdadeiro potencial.

Objetivos da aula

- Demonstrar o emprego, na prática, dos Arrays Javascript nas soluções de problemas

Resumo

Quando desenvolvemos uma aplicação, geralmente, temos que manipular os arrays para que possamos por exemplo filtrar seus dados. Dessa forma, o Javascript possui alguns métodos importantes que facilitam a manipulação de array. Nesse material iremos tratar sobre como podemos trabalhar com arrays de forma eficiente e eficaz.

Reduce

O método `reduce()` aplica uma função em um acumulador e cada valor do array (da esquerda para a direita) para reduzi-lo a um único valor. Este método pode ser usado para condensar todos os valores de um array em um único valor. Vejamos o exemplo abaixo:

[Ir para resumo](#)

```
[1, 2, 3, 4].reduce(function(a, b) {  
    return a + b;  
});  
// → 10
```

Fonte: Autoral

Map


Muitas vezes é necessário gerar um novo array, com base nos valores de um array existente. Isso é possível usando o map. Por exemplo, para gerar um array de comprimentos de strings a partir de um array de strings:

```
['one', 'two', 'three', 'four'].map(function(value, index, arr) {  
    return value.length;  
});  
// → [3, 3, 5, 4]
```

Neste exemplo, uma função anônima é fornecida à função map(), e a função map a chamará para cada elemento na matriz, fornecendo os seguintes parâmetros, nesta ordem:

- O próprio elemento
- O índice do elemento (0, 1...)
- A matriz inteira(Array)

Filter

O método `filter()` aceita uma função de teste e retorna um novo array contendo apenas os elementos do array original que passa no teste. 

fornecido.

```
var numbers = [5, 32, 43, 4];
```

```
var odd = numbers.filter(function(n) {  
    return n % 2 !== 0;  
});
```

A variável `odd` conteria o seguinte array: `[5,43]`.

For...in

Podemos iterar em um array de várias formas. Uma boa opção é utilizar o `for...in`.

```
for (i in myArray) {  
    console.log(myArray[i]);  
}
```

Fonte: Autoral

For...of

Uma outra forma de iterar é usando o loop `for...of`. Essa é a maneira recomendada de iterar sobre os valores de um array:

```
let myArray = [1, 2, 3, 4];  
for (let value of myArray) {  
    let twoValue = value * 2;  
    console.log("2 * value is: %d", twoValue);  
}
```



Fonte: Autoral

O exemplo a seguir mostra a diferença entre um loop for...of e um loop for...in:


```
let myArray = [3, 5, 7];  
myArray.foo = "hello";  
  
for (var i in myArray) {  
    console.log(i); // logs 0, 1, 2, "foo"  
}  
  
for (var i of myArray) {  
    console.log(i); // logs 3, 5, 7  
}
```

Fonte: Autoral

Conteúdo Bônus

Obra: “JavaScript Arrays: Aprofundando seu conhecimento e melhores práticas”

Autor: John Smith

Para aprofundar ainda mais seu conhecimento sobre arrays em JavaScript e explorar melhores práticas, recomenda-se a leitura  livro “JavaScript Arrays: Aprofundando seu conhecimento e melhores práticas” escrito por John Smith. Este livro aborda de forma detalhada os conceitos avançados relacionados a arrays, além de apresentar exemplos práticos e técnicas recomendadas para lidar com arrays de forma eficiente. A leitura desse material complementar ajudará você a expandir seu domínio sobre arrays em JavaScript.

Referência Bibliográfica

FLANAGAN, David. **JavaScript: O Guia Definitivo**. 6ª Ed. Porto Alegre: Bookman, 2013.

FREEMAN, Eric. **Use a cabeça!**: programação JavaScript. 1ª Ed. São Paulo: Alta Books, 2016.

ATIVIDADE PRÁTICA

Título da Prática: Usando métodos avançados sobre os arrays

Objetivos: Compreender a implementação de métodos do array

Materiais, Métodos e Ferramentas: Para realizar esta prática vamos utilizar o Visual Studio Code

Prática

Você está trabalhando em um projeto de programação e precisa manipular um array que contém tanto strings quanto números. Para isso, você decide

utilizar as funções 'map', 'filter' e 'reduce' em JavaScript.



O array dado possui uma combinação de strings e números, e sua tarefa é realizar algumas operações específicas nele. Primeiramente, utilizando a função 'map', você deve criar um novo array onde os números serão duplicados e as strings serão transformadas em letras maiúsculas. Caso existam elementos de outros tipos no array, eles devem ser mantidos sem alteração.

Em seguida, utilizando a função 'filter', você deve criar outro array contendo somente os números presentes no array original. Ou seja, todos os elementos que não são números devem ser excluídos.

Por fim, utilizando a função 'reduce', você deve somar todos os números do array original. O resultado dessa soma deve ser armazenado em uma variável.

Essas operações permitem que você manipule e transforme os elementos do array de acordo com suas necessidades específicas. Ao final do processo, você terá um novo array com elementos modificados, um array contendo apenas os números e a soma de todos eles.

Lembre-se de utilizar corretamente as funções 'map', 'filter' e 'reduce' em JavaScript para resolver esse problema.

Boa sorte!

Resolução

```
const array = [1, 'dois', 3, 'quatro', 5, 'seis'];
```

```
// Utilizando map para duplicar os números e transformar as strings em  
letras maiúsculas
```



```
const mappedArray = array.map((element) => {
```

```
  if (typeof element === 'number') {
```

```
    return element * 2;
```

```
  } else if (typeof element === 'string') {
```

```
    return element.toUpperCase();
```

```
  } else {
```

```
    return element;
```

```
  }
```

```
});
```

```
console.log('Array mapeado:', mappedArray);
```

```
// Utilizando filter para filtrar somente os números do array
```

```
const filteredArray = array.filter((element) => typeof element === 'number');
```

```
console.log('Array filtrado:', filteredArray);
```

```
// Utilizando reduce para somar os números do array
```

```
const sum = array.reduce((accumulator, element) => {
```

```
  if (typeof element === 'number') {
```

```
    return accumulator + element;
```

```
  } else {
```

```
    return accumulator;
```

```
}
```



```
}, 0);
```

```
console.log('Soma:', sum);
```

Ir para exercício