JavaScript Arrays

- description : JavaScript Arrays
- author : 오션
- email : shlim@repia.com
- lastupdate : 2021-04-26

The source of this article

[[https://www.w3schools.com/js/js_events.asphttps://www.w3schools.com/js/js_arrays.asp JavaScript 배열은 단일 변수에 여러 값을 저장하는 데 사용됩니다.

```
<script>
  let cars = ["Saab", "Volvo", "BMW"];
  document.getElementById("demo").innerHTML = cars;
</script>
```

What is an Array?

배열은 한 번에 하나 이상의 값을 보유할 수 있는 특별한 변수입니다.

아이템 목록 (예:자동차 이름 목록)이 있는 경우, 단일 변수에 자동차를 저장하면 다음과 같이 보일 수 있습니다.

```
let car1 = "Saab";
let car2 = "Volvo";
let car3 = "BMW";
```

그러나 자동차를 반복하고 특정 자동차를 찾으려면 어떻게 해야 합니까? 차가 3 대가 아니라 300 대라면 어떨까요?

해결책은 배열입니다!

배열은 단일 이름으로 많은 값을 보유할 수 있으며, 색인 번호를 참조하여 값에 액세스 할 수 있습니다.

Creating an Array

배열 리터럴을 사용하는 것이 JavaScript 배열을 만드는 가장 쉬운 방법입니다.

Syntax

```
var array_name = [item1, item2, ...];
```

```
var cars = ["Saab", "Volvo", "BMW"];
```

공백과 줄 바꿈(line breaks)은 중요하지 않습니다. 선언은 여러 줄에 걸쳐있을 수 있습니다.

```
let cars = [
  "Saab",
  "Volvo",
  "BMW"
];
```

Using the JavaScript Keyword new

다음의 예제는 배열을 생성하고, 배열에 값을 할당합니다.

```
let cars = new Arrays("Saab", "Volvo", "BMW");
```

위의 두 예제는 정확히 동일합니다. new Array()를 사용할 필요가 없습니다.
단순성, 가독성 및 실행 속도를 위해, 첫 번째 방법 (배열 리터럴 방법, the array literal method)을 사용하십시오.

Access the Elements of an Array

인덱스 번호(index number)를 참조하여 배열 요소에 액세스합니다.

이 스테이트먼트는 cars의 첫 번째 요소 값에 액세스 합니다.

```
let name = cars[0];
```

```
let cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars[1];  // Volvo
```

Note: 배열 인덱스는 0으로 시작합니다. [0]은 첫 번째 요소입니다. [1]은 두 번째 요소입니다.

Changing an Array Element

다음의 스테이트먼트는 cars의 첫 번째 요소의 값을 변경합니다.

```
cars[0] = "Opel";
```

```
let cars = ["Saab", "Volvo", "BMW"];
cars[2] = "Opel";
document.getElementById("demo").innerHTML = cars; // Saab,Volvo,Opel
```

Access the Full Array

JavaScript를 사용하면, 배열 이름을 참조하여 전체 배열에 액세스 할 수 있습니다.

```
let cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars;  // Saab,Volvo,BMW
```

Arrays are Objects

배열은 특별한 유형의 오브젝트 입니다. JavaScript의 typeof 연산자는 배열에 대해 “ 오브젝트 ” 를 반환합니다.

그러나, JavaScript 배열은 배열로 가장 잘 설명됩니다.

배열은 숫자를 사용하여 자신의 “ 요소 ” 에 액세스합니다. 다음 예제에서 person[0]은 John을 반환합니다:

Array:

```
let person = ["John", "Doe", 46];
document.getElementById("demo").innerHTML = person[0];  // John
```

오브젝트는 이름을 사용하여 오브젝트의 “ 구성원(members)“에 액세스합니다.
다음 예제에서 person.firstName은 John을 반환합니다.

Object

```
let person = { firstName: "John", lastName: "Doe", age: 46 };
```

```
document.getElementById("demo").innerHTML = person["firstName"];  // John
```

Array Elements Can Be Objects

JavaScript 변수는 오브젝트가 될 수 있습니다. 배열은 특별한 종류의 오브젝트 입니다.

이로 인해, 동일한 배열에 다른 유형의 변수를 가질 수 있습니다.

배열에 오브젝트를 가질 수 있습니다. 배열에 함수를 가질 수 있습니다. 배열에 배열을 가질 수 있습니다:

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

Array Properties and Methods

JavaScript 배열의 진정한 강점은 내장 배열 속성과 메서드 입니다.

```
let x = cars.length;  // The length property returns the number of elements
let y = cars.sort();  // The sort() method sorts arrays
```

배열 메서드는 다음 장에서 다룹니다.

The length Property

배열의 length 속성은 배열의 길이 (배열의 요소들의 개수)를 반환합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.length;
// the length of fruits is 4
```

length 속성은 항상 가장 높은 배열 인덱스보다 하나 더 많습니다.

Accessing the First Array Element

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
let first = fruits[0];
```

```
document.getElementById("demo").innerHTML = first;  // Banana
```

Accessing the Last Array Element

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
let last = fruits[fruits.length - 1];
// fruits[fruits.length-1] = fruits[4-1] = fruits[3]
document.getElementById("demo").innerHTML = last;  // Mango
```

Looping Array Elements

배열을 순환하는 가장 안전한 방법은 for loop를 사용하는 것 입니다.

```
let fruits, text, fLen, i;    // 변수 명 선언
fruits = ["Banana", "Orange", "Apple", "Mango"];  // fruits 배열 선언
fLen = fruits.length;   // 4

text = "<ul>";
for (i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";

document.getElementById("demo").innerHTML = text;
```

Array.forEach() 메서드를 사용할 수도 있습니다.

```
let fruits, text;
fruits = ["Banana", "Orange", "Apple", "Mango"];

text = "<ul>";
fruits.forEach(myFunction);
text += "</ul>";
document.getElementById("demo").innerHTML = text;

function myFunction(value) {
text += "<li>" + value + "</li>";
}
```

Adding Array Elements

배열에 새로운 요소를 추가하는 가장 쉬운 방법은 “push()” 메서드를 사용하는 것입니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;

function myFunction() {
  fruits.push("Lemon");
  document.getElementById("demo").innerHTML = fruits;
}   // Banana,Orange,Apple,Mango,Lemon
```

length 속성을 사용하여 배열에 새로운 요소를 추가할 수도 있습니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;

function myFunction() {
fruits[fruits.length] = "Lemon";
  document.getElementById("demo").innerHTML = fruits;
}
```

WARNING! 인덱스가 높은 요소를 추가하면 배열에 정의되지 않은(undefined) “구멍(holes)“이 생길 수 있습니다.

```
let fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
\fruits[6] = "Lemon";

fLen = fruits.length;
text = "";
for (i = 0; i < fLen; i++) {
  text += fruits[i] + "<br>";
}
  document.getElementById("demo").innerHTML = text;
// Banana Orange Apple Mango undefined undefined Lemon
```

Associative Arrays(연관 배열)

많은 프로그래밍 언어는 명명된 인덱스를 가지고 있는 배열을 지원합니다.

명명된 인덱스를 가진 배열을 연관 배열(또는 해시 hashes)이라고 합니다.

JavaScript는 명명된 인덱스를 가진 배열을 지원하지 않습니다.

JavaScript에서 배열은 항상 번호가 매겨진 인덱스를 사용합니다.

```
let person = [];
person[0] = "John";
person[1] = "Doe";
person[2] = 46;
let x = person.length;
let y = person[0];
console.log(x);   // person.length will return 3
console.log(y);   // porson[0] will return "John"
document.getElementById("demo").innerHTML = person[0] + " " + person.length;
// John 3
```

WARNING!!
명명된 인덱스를 사용하는 경우, JavaScript는 배열을 표준 오브젝트로 재정의합니다.
그 후 일부 배열 메서드 및 속성이 잘못된 결과를 생성합니다.

```
let person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
document.getElementById("demo").innerHTML = person[0] + "" + person.length;
// undefined()
```

The Difference Between Arrays and Objects

JavaScript에서 **배열(arrays)***은 **번호가 매겨진 인덱스(numbered indexes)**를 사용합니다.

JavaScript에서 **오브젝트(objects)**는 **명명된 인덱스(named indexes)**를 사용합니다.

배열은 번호가 매겨진 인덱스를 가진 특수한 종류의 객체입니다.

When to Use Arrays. When to use Objects.

- JavaScript는 연관 배열을 지원하지 않습니다.
- 요소 이름을 문자열 (텍스트)로 지정하려면 오브젝트를 사용해야 합니다.
- 요소 이름을 숫자로 만들려면 배열을 사용해야 합니다.

Avoid new Array()

JavaScript의 내장 배열 생성자 `new Array()`를 사용할 필요가 없습니다.

대신 `[]`를 사용하십시오.
다음의 두 가지 다른 스테이트먼트 모두 포인트라는 이름의 새로운 빈 배열을 만듭니다:

```
let points = new Array();    // Bad
let points = [];             // Good
```

다음의 2개의 다른 스테이트먼트는 6개의 숫자를 가진 새로운 배열을 생성합니다.

```
//
let points = new Array(40, 100, 1, 5, 25, 10);
let points = [40, 100, 1, 5, 25, 10];
document.getElementById("demo").innerHTML = points[0];
console.log(points);    // 40
```

new 키워드는 코드를 복잡하게 만듭니다. 또한 예상치 못한 결과가 발생할 수 있습니다.

```
let points = new Array(40, 100);  // Creates an array with two elements ( 40 and 100)
```

위의 요소 중에서 하나를 제거하면 아래와 같이 됩니다.

```
let points = new Array(40);
document.getElementById("demo").innerHTML = points[0];
// Creates an array with 40 undefined elements!!!
```

How to Recognize an Array

일반적인 질문: 변수가 배열인지 어떻게 알 수 있습니까?

문제는 JavaScript 연산자 `typeof`가 ”`object` “ 를 반환한다는 것입니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = typeof fruits;  // object
```

typeof 연산자는 JavaScript 배열이 오브젝트이기 때문에 오브젝트를 반환합니다.

Solution 1:

이 문제를 해결하기 위해, ECMAScript 5는 새로운 메서드 `Array.isArray()`를 정의합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
    document.getElementById("demo").innerHTML = Array.isArray(fruits);  //
true
```

이 솔루션의 문제점은 ECMAScript 5가 구번전의 브라우저에서 지원되지 않는다는 것입니다.

Solution 2:

이 문제를 해결하기 위해, 고유한 `isArray()` 함수를 만들 수 있습니다:

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = isArray(fruits);

function isArray(myArray) {
return myArray.constructor.toString().indexOf("Array") > -1;
} // true
```

위의 함수는 인수가 배열이면 항상 true를 반환합니다.

또는 더 정확하게는, 오브젝트 프로토 타입에 "Array"라는 단어가 포함되어 있으면, true를 반환합니다.

Solution 3:

`instanceof` 연산자는 주어진 생성자에 의해 오브젝트가 생성된 경우 true를 반환합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits instanceof Array;  //
true
```

오션, Javascript, Arrays