JavaScript Let

- description : JavaScript Let
- author : 오션
- email : shlim@repia.com
- lastupdate : 2021-04-13

ECMAScript 2015

ES2015는 `let`과 `const`라는 두 가지 중요한 새 JavaScript 키워드를 도입했습니다.

이 두 키워드는 JavaScript에서 **블록 범위** 변수(Block Scope)와 상수(constant)를 제공합니다.

ES2015 이전에는, JavaScript에는 **전역 범위(Global Scope)**와 **함수 범위(Function Scope)**라는 두 가지 유형의 범위만 있었습니다.

Global Scope

모든 함수 외부에서 **전역으로(Globally)** 선언된 변수들은 **전역 범위(Global Scope)**를 가지고 있습니다.

```
<!DOCTYPE html>
<html>

<body>
    <h2>JavaScript Scope</h2>
    <p>A Global variable can be accessed from any script or function.</p>
    <p id="demo"></p>                          <!-- I can display All New
Rexton -->
    <script>
        var carName = "All New Rexton";
        myFunction();

        function myFunction() {
            document.getElementById("demo").innerHTML = "I can display " +
carName;
        }
    </script>
</body>

</html>
```

전역 변수(Global variables)는 JavaScript 프로그램의 어느 곳에서나 액세스 할 수 있습니다.

Function Scope

함수 내에서 **로컬로(Locally)** 선언된 변수는 **함수 범위(Function Scope)**를 가집니다.

```
<!DOCTYPE html>
<html>
<body>
    <h2>JavaScript Scope</h2>
    <p>Outside myFunction() carName is undefined.</p>
    <p id="demo1"></p>     // string Volvo
    <p id="demo2"></p>     // undefined
    <script>
        myFunction();
        function myFunction() {
            var carName = "Volvo";
            document.getElementById("demo1").innerHTML = typeof carName + "
" + carName;
        }
        document.getElementById("demo2").innerHTML = typeof carName;
    </script>
</body>
</html>
```

지역 변수(Local variables)는 선언된 함수 내에서만 액세스 할 수 있습니다.

JavaScript Block Scope

var 키워드로 선언된 변수는 **블록 범위(Block Scope)**를 가질 수 없습니다.

블록 {} 내부에서 선언된 변수는 블록 외부에서 액세스 할 수 있습니다.

```
{
  var x = 2;
}
// x CAN be used here
```

ES2015 이전에는 JavaScript에 블록 범위가 없었습니다.

let 키워드로 선언된 변수는 블록 범위를 가질 수 있습니다.

블록 {} 내부에서 선언된 변수는 블록 외부에서 액세스 할 수 없습니다.

```
{
  let x = 2;
}
```

```
// x can Not be used here.
```

Redeclaring Variables

var 키워드를 사용하여 변수를 다시 선언하면 문제가 발생할 수 있습니다.

블록 내부의 변수를 다시 선언하면 블록 외부의 변수도 다시 선언됩니다:

```
<!DOCTYPE html>
<html lang="en">
<body>

    <h2>Declaring a Variable Using var</h2>
    <p id="demo"></p>
    <script>
        var x = 10;  // Here x is 10
        {
            var x = 2;   // Here x is 2
        }
        document.getElementById("demo").innerHTML = x;   // Here x is 2
    </script>
</body>
</html>
```

let 키워드를 사용하여 변수를 다시 선언하면, 이 문제를 해결할 수 있습니다.

블록 내부의 변수를 다시 선언해도 블록 외부의 변수는 다시 선언되지 않습니다.

```
<!DOCTYPE html>
<html lang="en">
<body>
  <h2>Declaring a Variable Using let</h2>
  <p id="demo"></p>
  <script>
    var x = 10;
    {
      let x = 2;
    }
    document.getElementById("demo").innerHTML = x;  // Here x is 10
  </script>
</body>
</html>
```

let 키워드는 Internet Explorer 11 이하에서 완전히 지원되지 않습니다.

Loop Scope

loop에서 var를 사용하는 것.

```
<!DOCTYPE html>
<html>
<body>
    <h2>JavaScript var</h2>
    <p id="demo"></p> <!-- Here i is 10 -->
    <script>
        var i = 5;
        for (var i = 0; i < 10; i++) {
            // some statements
        }
        document.getElementById("demo").innerHTML = i;
    </script>
</body>
</html>
```

loop에서 let를 사용하는 것.

```
<!DOCTYPE html>
<html>
<body>
    <h2>JavaScript let</h2>
    <p id="demo"></p> <!-- Here i is 5 -->
    <script>
        let i = 5;
        for (let i = 0; i < 10; i++) {
            // some statements
        }
        document.getElementById("demo").innerHTML = i;
    </script>
</body>
</html>
```

첫 번째 예제에서는, var를 사용하여 루프에서 선언된 변수가 루프 외부의 변수를 다시 선언합니다.

두 번째 예제에서는, let을 사용하면 루프에서 선언된 변수가 루프 외부에서 변수를 다시 선언하지 않습니다.

let이 루프에서 i 변수를 선언하는 데 사용되면, i 변수는 루프 내부에서만 볼 수 있습니다.

Function Scope

var 및 let으로 선언된 변수들은 함수 내에서 선언될 때 매우 유사합니다.

둘 다 **함수 범위(Function Scope)**를 가집니다.

```
function myFunction() {
  var carName = "Volvo";  // Function Scope
```

```
function myFunction() {
  let carName = "Volvo";  // Function Scope
```

Global Scope

var 및 let으로 선언된 변수는 블록 외부에서 선언될 때 매우 유사합니다.

둘 다 **전역 범위(Global Scope)**를 갖습니다.

```
 var x = 2;   // Global scope
```

```
 let x = 2;   // Global scope
```

Global Variables in HTML

JavaScript에서 전역 범위는 JavaScript 환경입니다.

HTML에서, 전역 범위는 window 오브젝트입니다.

var 키워드로 정의된 전역 변수는 window 오브젝트에 속합니다.

```
<!DOCTYPE html>
<html>
<body>
    <h2>JavaScript Global Variables</h2>
    <p>In HTML, global variables defined with <b>var</b> will become window
variables.</p>
    <p id="demo"></p>
    <script>
        var carName = "Carnival";
        // code here can use window.carName
        document.getElementById("demo").innerHTML = "I can display " +
window.carName;
    </script>
</body>
</html>
```

let 키워드로 정의된 전역 변수는 window 오브젝트에 속하지 않습니다.

```
<!DOCTYPE html>
<html lang="en">
<body>
    <h2>JavaScript Global Variables</h2>
    <p>In HTML, global variables defined with <b>let</b> will not become
window variables.</p>
    <p id="demo"></p>
    <script>
        let carName = "Tusan";
        // code here can not use window.carName
        document.getElementById("demo").innerHTML = "I can not display " +
window.carName;
    </script>
</body>
</html>
```

오션,, Javascript, Let

From:
http://125.132.25.164/dokuwiki/ - **문제를 잘 정의하는 것은 문제를 절반 해결한 것이다. - 2023.12**

Permanent link:
http://125.132.25.164/dokuwiki/doku.php?id=wiki:javascript:javascript_note:js_let&rev=1618298349

Last update: **2022/03/10 19:52**