JavaScript Function Definitions

- description : JavaScript Function Definitions
- author : 오션
- email : shlim@repia.com
- lastupdate : 2021-04-14

Source of the article

- JavaScript Function Definitions

JavaScript 함수는 `function` 키워드는 정의합니다.

함수 선언(function declaration) 또는 함수 표현식(function expression)을 사용할 수 있습니다.

Function Declarations

함수는 다음 구문으로 선언됩니다:

```
function functionName(parameter) {
  // code to be executed
}
```

선언된 함수는 즉시 실행되지 않습니다. “ 나중 사용을 위해 저장 ” 되고,
나중에 적용(호출)될 때 실행됩니다.

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

  <h2>JavaScript Functions</h2>

  <p>
    This exmple calls a function which performs a calculation and returns
the result:
  </p>

  <p id="demo"></p>

  <script>
    var x = myFunction(4, 3);
    document.getElementById("demo").innerHTML = x;
```

```
    function myFunction(a, b) {
      return a * b;
    }
  </script>

</body>
</html>
```

세미콜론은 실행 가능한 JavaScript 스테이트먼트를 구분하는 데 사용됩니다.
함수 선언은 실행 가능한 스테이트먼트가 아니기 때문에, 세미콜론으로 끝내는 것은 일반적이지 않습니다.

Function Expressions

JavaScript 함수는 표현식(expression)을 사용하여 정의할 수도 있습니다.

함수 표현식은 변수에 저장할 수 있습니다:

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

  <p>A function can be stored in a variable:</p>
  <p id="demo"></p>

  <script>
    var x = function (a, b) { return a * b };
    document.getElementById("demo").innerHTML = x;
  </script>

</body>
</html>
```

함수 표현식이 변수에 저장되면, 변수를 함수로 사용할 수 있습니다:

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

  <p>After a function has been stored in a variable, the variable can be
used as a function:</p>

  <p id="demo"></p>

  <script>
    var x = function (a, b) { return a * b };
```

```
    document.getElementById("demo").innerHTML = x(4, 3);
  </script>

</body>
</html>
```

위의 함수는 실제로 **익명 함수(anonymous function)** (이름이 없는 함수)입니다.

변수에 저장된 함수에는 함수 이름이 필요하지 않습니다. 항상 변수 이름을 사용하여 적용 (호출)됩니다.

위의 함수는 실행 가능한 스테이트먼트의 일부이기 때문에 세미콜론으로 끝납니다.

The Function() Constructor

이전 예제에서 본 것처럼, JavaScript 함수는 `function` 키워드로 정의됩니다.

`Function()`이라는 내장 JavaScript 함수 생성자(constructor)를 사용하여 함수를 정의할 수도 있습니다.

```
   var myFunction = new Function("a", "b", "return a * b");

   var x = myFunction(4, 3);
```

실제로 함수 생성자를 사용할 필요가 없습니다. 위의 예제는 아래의 예제와 동일한 결과입니다.

```
    var myFunction = function (a, b) { return a * b }

    var x = myfunction(4, 3)
```

대부분의 경우 JavaScript에서 new 키워드를 사용하지 않아도 됩니다.

Function Hoisting

Hoisting은 선언을 현재 범위의 맨 위로 이동하는 JavaScript의 기본 동작입니다.

호이스팅은 변수 선언과 함수 선언에 적용됩니다.

이 때문에, JavaScript 함수는 선언하기 전에 먼저 함수를 호출할 수 있습니다:

```
myFunction(5);

function myFunction(y) {
  return y * y;
}
```

표현식을 사용하여 정의된 함수는 호이스팅되지 않습니다.

Self-Invoking functions

함수 표현식은 “ 자체 호출(self-invoking)“로 만들 수 있습니다.

자체 호출 표현식은 호출되지 않고 자동으로 호출(시작)됩니다.

표현식 뒤에 ()가 오면, 함수 표현식이 자동으로 실행됩니다.

함수 선언을 자체 호출할 수 없습니다.

함수 표현식이라는 것을 나타내려면 함수 주위에 괄호를 추가해야 합니다.

```
<!DOCTYPE html>
<html>
<body>

  <p>Functions can be invoked automatically without being called:</p>

  <p id="demo"></p>

  <script>
    (function () {
      document.getElementById("demo").innerHTML = "Hello!!! I called
myself";
    })();
  </script>

</body>
</html>
```

위의 함수는 실제로 **익명의 자체 호출 함수(anonymous self-invoking function)**입니다 (이름없는 함수).

Functions Can Be Used as Values

JavaScript 함수는 값으로 사용될 수 있습니다.

```
function myFunction(a, b) {
  return a * b;
}

var x = myFunction(4, 3);
```

JavaScript 함수는 표현식으로 사용될 수 있습니다.

```
function myFunction(a, b) {
```

```
    return a * b;
}

var x = myFunction(4, 3) * 2;
```

Functions are Object

JavaScript의 `typeof` 연산자는 함수에 대해 “ 함수 ” 를 반환합니다.

그러나 JavaScript 함수는 오브젝트로 가장 잘 설명 할 수 있습니다.

JavaScript 함수에는 **속성(properties)**과 **메서드(methods)**가 모두 있습니다.

`arguments.length` 속성은 함수가 호출될 때 수신된 인수들의 수를 반환합니다.

```
```

`toString()` 메서드는 함수를 문자열로 반환합니다.

```
```

오브젝트의 속성으로 정의된 함수를 오브젝트에 대한 메서드라고 합니다.
새로운 오브젝트를 생성하도록 설계된 함수를 객체 생성자(object constructor)라고합니다.

오션,, Javascript, Function, Definitions

From:
http://rwiki.repia.com/ - **문제를 잘 정의하는 것은 문제를 절반 해결한 것이다.** - 2023.12

Permanent link:
http://rwiki.repia.com/doku.php?id=wiki:javascript:javascript_note:function_definition&rev=1618379362

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