CS472 WAP

Functions

JavaScript for Modern Web Apps

Maharishi International University Fairfield, Iowa © 2020

Function Declaration

```
function name() {
 statement ;
 statement ;
 ...
 statement ;
}

function square(number) {
 return number*number;
}
```

- declarations are "hoisted" (vs function expressions) – see Scope lesson
  - They can be declared anywhere in a file, and used before the declaration.

Function Expressions

- Can be Anonymous function

```
const square = function(number) { return number * number };
let x = square(4) // x gets the value 16
```

- Widely used in JS with event handlers (tomorrow's lesson)
- Widely used in higher order functions and functional programming
  - [1, 2, 3].map(function(element) {return element * 2;});

Can also have a name to be used inside the function to refer to itself `//NFE (Named Function Expression)`

```
const factorial = function fac(n)
{ return n < 2 ? 1 : n * fac(n - 1) };

console.log(factorial(3));
```

- Basically, a function expression is same syntax as a declaration, just used where an expression is expected ('lhs' vs 'rhs')

Function Declaration or Function Expression?

- Function declarations have two advantages over function expressions:
  - They are hoisted, so you can call them before they appear in the source code. (some consider this poor style)
  - They have a name. - the name of a function is useful for debugging; especially anonymous function. (ES6 infers name for function expressions)
- Conclusion
  - Neither of the above are significant
  - Can use functions declarations if desired, but can always use function expressions to accomplish same thing (except hoisting)
  - Expression if want anonymous function for event handler, etc
  - Declaration if want a reusable function

Functions are objects and are created with a name property

- `function f1(){}; f1.name // 'f1'`
- `const f2 = function(){}; f2.name //'` or 'f2' in ES6
- `const f3 = function f3(){}; f3.name //'f3'`
  - Named Function Expression (NFE)

Anonymous functions

- JavaScript allows you to declare anonymous functions
- Can be stored as a variable, attached as an event handler, etc.
- Keeping unnecessary names out of namespace for performance and safety

```
window.onload = function(){
 alert("Hello World!");
}
```

- Function declaration or expression?

Semicolon ;

- Semicolons are (technically) ‘optional’
  - JS implicitly adds them to our code if parser is expecting
  - “semicolon insertion”
  - “bad part” of JavaScript
  - ‘seemed like a good idea at the time’ …
  - in certain cases that can cause problems
    - return, var, break, throw, …
  - Best practice to explicitly include them
  - Include brackets at the end of line versus new line
    - K&R (Kernighan and Ritchie) versus OTBS (one true brace style)

Semicolon ;

```
function a() {//K&R style
  return {
      a: 1 ;
   }
}
function b()
{  //OTBS not good practice (Crockford, …)
 return  //semicolon gets inserted here
   {
   a: 1 ;
   }
}
console.log(a()); //object
console.log(b());  //undefined  -- oops
```

Main Point

- JavaScript is a loosely typed language. It has types but does no compile time type checking. Programmers must be cautious of automatic type conversions, including conversions to Boolean types. It has a flexible and powerful array type as well as distinct types of null and undefined.

- *To be an effective JavaScript programmer one needs to understand the principles and details of the language. If our awareness is established in the source of all the laws of nature, then our actions will spontaneously be in accord with the laws of nature for a particular environment.*


MAHARISHI
HIGHER EDUCATION
FOR HIGHER CONSCIOUSNESS
INTERNATIONAL UNIVERSITY