cybercom creation

JavaScript Functions

By Pritey Mehta

Functions are one of the fundamental building blocks in JavaScript. A function in JavaScript is similar to a procedure—a set of statements that performs a task or calculates a value

Function declarations

A function definition (also called a function declaration, or function statement) consists of the function keyword,

followed by:

- The name of the function.
- A list of parameters to the function

```
function area(length, width) {
  return length * width;
}
```

Function expression

The function keyword can be used to define a function inside an expression.

Notes:

A function expression is very similar to and has almost the same syntax as a function declaration, he main difference between a function expression and a function declaration is the function name, which can be omitted in function expressions.

A function expression can be used as an IIFE (Immediately Invoked Function Expression) which runs as soon as it is defined.

Function expressions in JavaScript are not hoisted unlike function declarartions.

```
const getArea = function(width,
height) {
  return width * height;
};

console.log(getRectArea(1, 2));
//output will be 2
```

Named Function expression

The function can be also define as name inside the scope of function body as individual..

```
const math = {
sum: function _doSum(a,b) {
    console.log("sum is", a + b);
  }
};

math.sum(3,2); // 5
```

Notes:

The variable to which the function expression is assigned will have a name property. The name doesn't change if it's assigned to a different variable.

```
const varFuncName =
function originalName() {}

console.log(varFuncName.name)
```

Arrow Functions

An arrow function expression is a compact alternative to a traditional function expression, with some semantic differences and deliberate limitations in usage:

- Arrow functions don't have their own bindings to this, arguments, or super, and should not be used as methods.
- Arrow functions cannot be used as constructors. Calling them with new throws a TypeError. They also don't have access to the new.target keyword.
- Arrow functions cannot use yield within their body and cannot be created as generator functions.

Syntax.txt

```
param => expression

(param) => expression

(param1, paramN) => expression

param => {
  statements
}

(param1, paramN) => {
  statements
}
```

Example.txt

```
let myFunction = (a,b) => {
    console.log("sum", a+b)
}

myFunction(1,5); // sum 6
```

So which is the best?

If no. of arguments are not defined then go with normal function (as you can get the args object in normal function, while not in arrow function.

There is no prototype object for Arrow functions like regular functions

Unlike regular functions, arrow functions don’t have their own this binding. If we access this in the arrow function it will return the this of the closest non-arrow parent function.

cybercom creation

Your task is to find what's the difference between method & function?

cybercom creation

SCAN TO GET
ALL FUNCTION EXAMPLES