Functions

A function declaration looks like this:

```
function name(parameters, delimited, by, comma) {
  /* code */
let userName = ""
return xxxxx
}
```

- Values passed to a function as parameters are copied to its local variables.
- A function may access outer variables. But it works only from inside out. The code outside of the function doesn't see its local variables.
- A function can return a value. If it doesn't, then its result is `undefined`.

To make the code clean and easy to understand, it's recommended to use mainly local variables and parameters in the function, not outer variables.

It is always easier to understand a function which gets parameters, works with them and returns a result than a function which gets no parameters, but modifies outer variables as a side-effect.

Function naming:

- A name should clearly describe what the function does. When we see a function call in the code, a good name instantly gives us an understanding what it does and returns.
- A function is an action, so function names are usually verbal.
- There exist many well-known function prefixes like `create…`, `show…`, `get…`, `check…` and so on. Use them to hint what a function does.

Functions are the main building blocks of scripts. Now we've covered the basics, so we actually can start creating and using them. But that's only the beginning of the path. We are going to return to them many times, going more deeply into their advanced features.

Types of functions in javascript?

1. **Named** function
2. **Anonymous** function
3. Expression Function
4. Self Invoking Functions
5. **Arrow Function**

Named function

Named function is the function that we ***define*** it in the code and then call it whenever we need it by referencing its *name and passing some arguments* to it. Named functions are useful if we need to call a function **many times** to pass **different values** to it or run it several times.

```
function areaOfCircle(r) {
  let area = Math.PI * r * r
  return area
}
```

Anonymous Function

Anonymous function or without name

```
const anonymousFun = function() {
  console.log(
    'I am an anonymous function and my value is stored in anonymousFun'
  )
}
```

Expression Function

Expression functions are anonymous functions. After we create a function without a name and we assign it to a variable. To return a value from the function we should call the variable. Look at the example below.

```
// Function expression
const square = function(n) {
  return n * n
}

console.log(square(2)) // -> 4
```

Self Invoking Functions

Self invoking functions are anonymous functions which do not need to be called to return a value.

```
(function(n) {
  console.log(n * n)
})(2) // 4, but instead of just printing if we want to return and
//store the data, we do as shown below

let squaredNum = (function(n) {
  return n * n
})(10)

console.log(squaredNum)
```

Arrow Function

Arrow function is an alternative to write a function, however function declaration and arrow function have some minor differences.

Arrow function uses arrow instead of the keyword *function* to declare a function. Let us see both function declaration and arrow function.

```
// This is how we write normal or declaration function
// Let us change this declaration function to an arrow function
function square(n) {
  return n * n
}

console.log(square(2)) // 4

const square = n => {
  return n * n
}

console.log(square(2))  // -> 4
```

```
// if we have only one line in the code block, it can be written as follows, explicit return
const square = n => n * n  // -> 4
```

```
const printFullName = (firstName, lastName) => {
  return `${firstName} ${lastName}`
}

console.log(printFullName('Afaf', 'kelai'))
```

Exo