Scope & Closures

phuong.vu@htklabs.com

Agenda

- Scope
- Closures

Scope

- Refers to where variables and functions are accessible, and in what context it is being executed.
- A variable or function can be defined in a global or local scope.
- Variables have so-called function scope, and functions have the same scope as variables.

Global Scope

- It is accessible from anywhere in your code.
- Ex:

```
// Global scope
var monkey = "Gorilla";
```

- In a web page, global variables belong to the window object.

Local Scope

- As opposed to the global scope, the local scope is when something is just defined and accessible in a certain part of the code, like a function.
- Ex:

```
// Scope A: Global scope out here
var myFunction = function () {
    // Scope B: Local scope in here
};
```

Function Scope

- All scopes in JavaScript are created with *Function Scope* only.
- Rule: new functions = new scope
- Ex:

```
// Scope A
var myFunction = function () {
    // Scope B
    var myOtherFunction = function () {
        // Scope C
    };
};
```

Lexical Scope

- The inner function has access to the scope in the outer function, this is called *Lexical Scope* or *Closure* - also referred to as *Static Scope.*
- Ex:

```
// Scope A
var myFunction = function () {
    // Scope B
    var name = 'Todd'; // defined in Scope B
    var myOtherFunction = function () {
        // Scope C: `name` is accessible here!
    };
};
```

Closures

- Closures are expressions, usually functions, which can work with variables set within a certain context
- Try and make it easier: inner functions referring to local variables of its outer function create closures.
- A closure is a function having access to the parent scope, even after the parent function has closed.

1. Closures have access to the outer function's variable even after the outer function returns

```
function celebrityName (firstName) {
      var nameIntro = "This celebrity is ";
      // this inner function has access to the outer function's variables, including the parameter
      function lastName (theLastName) {
            return nameIntro + firstName + " " + theLastName;
      }
      return lastName;
}

var mjName = celebrityName ("Michael"); // At this juncture, the celebrityName outer function has
returned.

// The closure (lastName) is called here after the outer function has returned above
// Yet, the closure still has access to the outer function's variables and parameter
mjName ("Jackson"); // This celebrity is Michael Jackson
```

2. Closures store references to the outer function's variables

```
function celebrityID () {
      var celebrityID = 999;
      // We are returning an object with some inner functions
      // All the inner functions have access to the outer function's variables
      return {
            getID: function () {
                  // This inner function will return the UPDATED celebrityID variable
                  // It will return the current value of celebrityID, even after the changeTheID
            function changes it
                  return celebrityID;
            },
            setID: function (theNewID) {
                  // This inner function will change the outer function's variable anytime
                  celebrityID = theNewID;
            }
      }
}

var mjID = celebrityID (); // At this juncture, the celebrityID outer function has returned.
mjID.getID(); // 999
mjID.setID(567); // Changes the outer function's variable
mjID.getID(); // 567: It returns the updated celebrityId variable
```

3. Closures Gone Awry

```
// This example is explained in detail below (just after this code box).
function celebrityIDCreator (theCelebrities) {
      var i;
      var uniqueID = 100;
      for (i = 0; i < theCelebrities.length; i++) {
            theCelebrities[i]["id"] = function () {
                  return uniqueID + i;
            }
      }
      return theCelebrities;
}

var actionCelebs = [{name:"Stallone", id:0}, {name:"Cruise", id:0}, {name:"Willis", id:0}];

var createIdForActionCelebs = celebrityIDCreator (actionCelebs);

var stalloneID = createIdForActionCelebs [0];
console.log(stalloneID.id()); // 103
```

Assignment

1. Change code in previous assignments to closure

Q&A

Thanks for your listening!