A closer look at functions

Abdelrahman Shaheen

OUTCOMES

- First-Class and Higher-Order Functions
- Functions accepting callback functions
- Functions returning functions
- **call** , **apply** and **bind** methods
- Closures

Complete “this” (with event handlers)

- See code 1

First-Class and Higher-Order Functions

- First class function ? → functions in js is a value.

- First class functions leads to → higher order function .

- Higher order function ? → functions that accepts functions as a parameter or functions
Return other functions.

- See code 2

- In summery : first class functions is a concept or feature that the language has and higher order functions is not possible without first class function feature in js .

Why we need to pass function or to return it ?

- This is another paradiagm called functional programming
- To know more read this article : Functional programming

Call and apply methods

- We can change the value that “this” points to.
- See code 3
- They(call and apply) are the same but call take list of arguments and apply take an array of arguments.

Bind method

- It’s more important than call and apply.
    - bind also allows us to manually set this keywords for any function call. The difference is that bind does not immediately call the function. Instead it returns a new function where this keyword is bound. So it's set to whatever value we pass into bind.
- See code 4.
- The important use of it when we need to just pass the function as a call back ,not to call it(With ***addEventHandler()***).
- Look at my project.

closures

- What is actually the closure ?

- We’ve learned (EC , scope chain and the variable environment), closure brings all of this in one thing.
- A closure is not a feature that we explicitly use. So a closure simply happens automatically in certain situations, we just need to recognize those situations.

- See code 5.

- So we can say that a closure makes a function remember all the variables that existed at the function's birthplace.

closures

- Any function always has access to the variable environment of the execution context in which the function was created even after this EC is gone. This means that even though the execution context has actually been destroyed, the variable environment somehow keeps living somewhere in the engine.

- This **connection** that we call closure.

- Closure has priority over the scope chain ?

- Closure is an internal property of a function.

Resources

- Closure
- call, apply and bind

Tasks

- Solve as possible as you can : http://csbin.io/closures ,you’ve to solve at least 10 problems