Netcracker

Front-end development course

JavaScript

Lecture #4

Functions

Functions

Functions in JavaScript is just special kind of objects that are “callable”, we can create functions using Function constructor (but it is insecure and slow as “eval”) https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function

Also you can add properties to the function the same way you can do it with objects.

In JavaScript there is no such thing as 'function overloading' and functions can take arbitrary number of arguments. We can access that arguments inside function through “arguments” pseudo-array.

Arguments passed in is passed by value. But the item that is passed by value is itself a reference. This is called call-by-sharing. This means that if you change the parameter itself, that won't affect the item that was fed into the parameter. But if you change the INTERNALS of the parameter, that will propagate back up.

Scope chain

Functions and curly braces for “let” and “const” can produce scope chain - inside scope chain interpreter start checking if some variables exist in scope, starting from most nested scope.

eval

eval() - is a global function that evaluates arbitrary JS code represented as a string. It is may be insecure if you are evaluating third party code and it is also generally slower than the alternatives, since it has to invoke the JS interpreter, while many other constructs are optimized by modern JS engines.

More on that topic here.

Function Declarations and Expressions

Function Declaration:

```
function foo() {}
```

Function Expression:

```
var foo = function() {};
```

The main difference between those two is - declaration in fully “hoisted” and will be available anywhere inside that scope.

Named Function Expression - is a subtype of Function Expression, it’s main use case is address a function in recursion.

Good article on that topic https://shamansir.github.io/JavaScript-Garden/#function.general

Immediately-Invoked Function Expression

IIFE - special kind of function that is invoked immediately after declaration.

By default browser operates in statement context - he assumes that everything he is facing is a statement, that is why we should use parentheses to indicate that IIFE is expression.

Good article on that topic https://getinstance.info/articles/javascript/immediately-invoked-function-expressions/

Recursion

Recursion - in general means that some function invokes itself. Recursion can be used for some computation (in that case it can be converted to loop), for data structures like Trees and Graphs or for recursive data structures like Linked List.

Recursion stack is limited in browsers around 10000.

A function is **tail-recursive** if the main recursive calls it makes are in tail positions.

Interesting article on that topic http://2ality.com/2015/06/tail-call-optimization.html

Closures

There is such thing called “LexicalEnvironment”, it is created during global environment initialization and every time function is invoked. It contains all variables and functions that are declared inside current scope.

Arrow functions

ES6 introduces new syntax for functions called arrow functions.

Arrow functions is simply:

Identifier => Expression

Arrow function itself is expression.

Arrow functions doesn’t create their own “this” context, they uses “this” from surrounding context.

Arrow functions doesn’t have their own “arguments” array.

More on arrow functions https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Functions/Arrow_functions

Homework

- Create a github repository a send a link to your lecturer. It can be named like “nc-fe-course-assignment”

Q&A

Thank You

Netcracker