Weird Parts 1-29

Udemy:
https://www.udemy.com/understand-javascript/learn/v4/overview

Repository:
C:\Users\Lenovo\Desktop\Isti\Programozás-\Kurzusok\Javascript\Udemy-Understanding-weird-Parts

Github:

Lesson_6 Syntax Parsers, Execution Contexts, and Lexical Environments
Section 2
Lesson_7 Name/Value pairs and Objects
Lesson_9_ Global Ojbect/Enviroment
Lesson_10 Hoisting
Lesson_11 Undefined
Lesson 12-14 Execution
Lesson_18 Asynchronous
Section 3
Lesson 19 Dynamic Typing
Lesson_20 Types
Lesson_22 Operator precedence and Associativity
Lesson_24_Coersion
Lesson_25_ Logical Operators
Lesson_26_Default Value with ||
Lesson_27_FrameWork

Lesson_6 Syntax Parsers, Execution Contexts, and Lexical Environments

Syntax Parsers:
A **program** the **reads your code** and **determines** if it is okay, and if its grammer valid to tranlsate it to the computer.

Lexical Environments:
Where you write something stis physically in the code you write.
For example:
Function context, written code in a function

Execution Contexts:
A wrapper to help manage the code that is running.

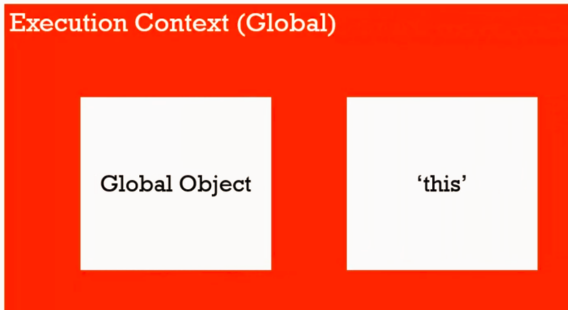
Lesson_7 Name/Value pairs and Objects

Name/Value pair:
A name which is mapped to 1 value.

Object:
Collection of name value Pairs.

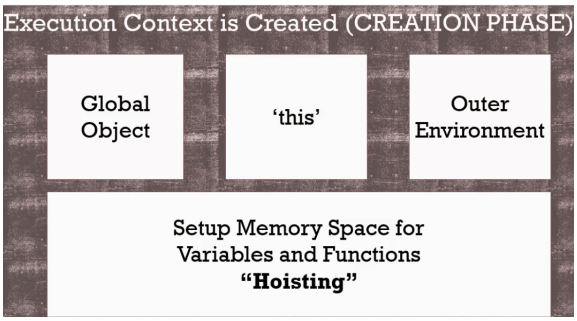
Lesson_9_ Global Ojbect/Enviroment

It is where the execution context, if it is global, it means the window. object in web browsers.



Lesson_10 Hoisting

So the V8 Javascript engine, prior the line by line execution, it creates space for the declared functions, variables in the memory. So tha'ts why it has been declared below, you can access the variable prior.



Lesson_11 Undefined

It is a special value, that means the **variable has not been set**.

(The variable has been never set.)

Lesson 12-14 Execution

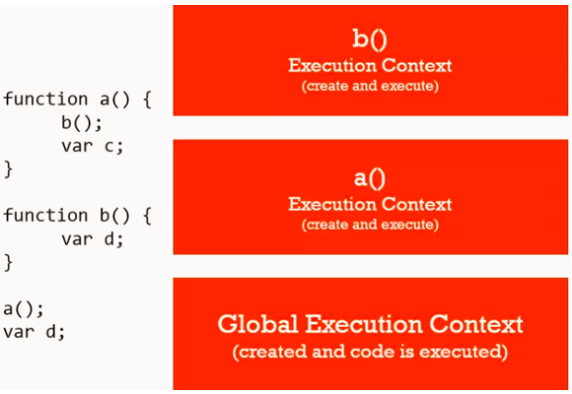
Execute the code, line by line.

```
function b() {
```

```
    console.log('Called b!');
}

b();
//The variable 'a' has not been set, but memory allocated, and basic "undefined type has been allocated"
console.log(a);

var a = 'Hello World!';

console.log(a);
```

Call Stack
The order in which the functions will run, "stacked".
Since javascript is scynhronous, it will run 1 line at a time.

So it well create and execute the execution context and if every line has been executed, it will stop.



Variable exectuion context

```
function b() {
    var myVar;              //With this we are not overwrtigin the global varible, we are just creating a new one in "b" execution context
    console.log(myVar);
}

function a() {
```

```
    var myVar = 2;          //This is a local variable defined in the 'a' execution
context
    console.log(myVar);
    b();
}

var myVar = 1;    //So this is the global variable
console.log(myVar);
a();
console.log(myVar);
```

Lesson_18 Asynchronous

- First the **call stack** is executed, then it will check the **queue stack,** created by the event checking loop

```
// long running function
function waitThreeSeconds() {
    var ms = 3000 + new Date().getTime();
    while (new Date() < ms){}
    console.log('finished function');
}

function clickHandler() {
    console.log('click event!');
}

// listen for the click event
document.addEventListener('click', clickHandler);



waitThreeSeconds();
console.log('finished execution');
```

```
/*
*****What if we click while the call stack is being executed?*****
*/
// 'finished funciton'
// 'finished execution'
// 'click evenet'

/*First the call stack is executed, then it will check the queue stack, created by the event checking loop*/
/The events will be processed in the order they happened/
```

Lesson 19 Dynamic Typing

While the code is running, you can change the variable type and the javascript engine will figures out the types on the fly.

Lesson_20 Types

Primitive Type: is just a single value

1. **Undefined** (lack of existence) this is for the engine
2. **Null** (lack of existence) this is for us
3. **Boolean** (true, false)
4. **Number** (1 floating type numerical value)
5. **String** (sequence of characters)
6. **Symbol** (ES6 -)

Lesson_22 Operator precedence and Associativity

precedence: Which operator function gets called first
Associativity: If they have to same presedence, this will tel if we read from left to right, or from right-to-left

```
//allocation is executed from righ-to-left
```

```
var a = 2, b = 3, c = 4;


a = b = c;


console.log();
console.log();
console.log();


//4
//4
//4
```

Lesson_24_Coersion

Coercion: Coverting a value from one type to another. (Due to dynamic typing)

```
console.log(1+3+"7");


//result is 47, since 1,3 is number they weill be added together,
```

Lesson_25_ Logical Operators

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Equality_comparisons_and_sameness

Lesson_26_Default Value with | |

```
function greet(name) {
    name = name || '<Your name here>';
    console.log('Hello ' + name);
}


greet('Tony');
```

```
greet();
```

Lesson_27_FrameWork

Will check if the library has been alread y defined or not.

```
window.libraryName = window.libraryName || "Lib2"
```