**What is JavaScript?**

Adds behaviour and interactivity to a website. Client side language, it runs ont he user’s PC.

JS enhances the audience’s experience only. JavaScript is not Java, it’s original name is ECMAScript.

**TypeScript**: superset of JavaScript, adds more functions, it has a built in checker (for debug). JavaScript = Typescript, Typescript =/= JavaScript.

**Hello World in JavaScript**

<body>

<script>

alert(”Hello World”);

</script>

</body>

**Basic JavaScript Syntax & Rules**

JS like CSS runs from top to bottom (and from left to right).

JS is case sensitive! In JS, we write statements, and each statement ends with a semicolon (;)

Not whitespace sensitive (spaces and line breaks are not counted).

Comments:

/\* …… \*/

for single line comments easier to use: // …….

**JavaScript Variables**

Variables are used to store informations. We don’t have to specify the exact type of the variable (waekly typed language).

Definition: var myVar (can’t be started with number, f.e.: var 1myVar)

var myVar = 10;

var myVar2 = ”Hello”; -> it’s a string

We can change the value of a var f.e.:

myVar = ˝Hello”;

myVar = ˝Hello again˝;

We can change the type of a value (though it’s not recommended), f.e.:

myVar = 5; so it becomes insted of Hello again to 5.

**let vs. var vs. const**

var x = 100; var uses function scope, hoistic. This means it lets variables be avaliable to a broader scope beyond they are actually declared. It can prone to errors.

let x = 100; let uses block scope. This means let variables are only avaliable within whatever curly brackets they’re declared

const x = 100; const is only an option to help the efficiency of how my program runs and manages memory. It’s used when we know that the variable is never going to have another value, because const variables cannot be changed.

**Basic mathematical operators**

Five common operators: =, +, -, /, \*.

=

var myVar = 5; -> 5 is assigned to myVar

+

myVar + 10; -> myVar’s value stays 5, because we haven’t assigned the new value

myVar = myVar + 5; -> myVar’s value changes to 10 because we’ve assigned the new value to it.

-. \*, / works exactly same.

5 + ˝hello”

˝5hello”

’’hello” + ’’david’’

’’hellodavid’’

5 \* ’’hello’’

NaN (not a number)

%: maradékos osztásnál a maradékot adja vissza

\*\*: hatványozás

**Math operator Short-Hand**

var myVar = 10

*myVar = myVar + 5* == *myVar += 5* -, /, \* works the same.

myVar

15

myVar++ -> adds +1, BUT first it prints out the original value and THEN adds +1.

++myVar -> adds +1, and prints out the modified value.

**Logging ot he Console**

var myVar = ’’Shaun’’;

document.write(myVar); ->prints out in the document

console.log(myVar); -> prints out in the console only

**Booleans in JS**

Boolean is a value that represents true or false.

**If statements**

var youLikeMeat = true;

if (youLikeMeat) {

document.write(’’here is the meaty menu’’);

}

In this case if var youLikeMeat is true, the here is the meaty menu string will be printed out. If it’s false, nothing will be printed out.

var myNum = 10;

if (myNum > 10) {

document.write(’’myNum is greater than 10’’);

}

In this case nothing will be printed because 10 is not greater than 10

var myNum = 8;

if (myNum == 10) {

document.write(’’myNum is ewual to 10’’);

} else {

document.write(’’myNum is not equal to 10”);

}

In this case the else branch will be printed out because 8 =/= 10.

**Else if statements**

var myAge = 29;

if (myAge > 30) {

document.write(’’you are over 30’’);

} else if {

document.write(’’you are over 20”);

} else if {

document.write(’’you are over 10”);

} else {

document.write(’’you are not over 10’’);

}

**Comparison operators**

>= grater than or equal to

<= smaller than or equal to

== equal to, cheks only the value

=== equal to, cheks the value AND the type

var x = ’’5”;

x == 5; TRUE, because the value is equal

x === 5; FALSE, because the value is equal BUT the type is not. (string vs number).

x === ’’5’’; TRUE, because the value and the type is equal.

!= not equal to, cheks only the value

!== not equal to, cheks only the value and the type.

var x = ’’5’’;

x !== 5; true, because the type is different (string vs number).

**Logical operators**

They are use to check multiple conditions.

if (myAge >= 18 && myAge <=30) && -> AND, both conditions must be true

if (myAge < 18 || myAge > 30) || -> OR, only 1 condition must be true

**While loops**

var age = 5; 🡪index variable

while (condition) {

code

}

So while condition is true, the code will be always runned, it stops only when the condition changes to false. (this is why it is called loop, because it’s looping).

Do while:

egyszer biztosan le fog futni, a kilépő feltételt utána vizsgálja csak.

var age = 5;

while (age < 10) {

console.log(’’your age is less than 10’’);

age++; //adds +1 to age

}

document.write(’’you are now over 10’’);

In this case the script will be runned 5 times, ont he 5th run age will be 10 and break out from the loop.

**For loops**

for (age = 5; age <10; age++) {

console.log(’’Your age is less than 10’’);

}

document.write(’’You are now over 10’’);

var links = document.getElementsByTagName(’’a’’); // in the html search for <a href> tags

for (i = 1; i <= links.length; i++) {

console.log(’’this is link number ’’ + i);

}

document.write(’’all links now looped’’);

**Break & continue**

for (i = 0, i < 10; i++) {

if (i === 5 || i === 3) {

continue;

}

console.log(i);

if(i === 7) {

break;

}

}

console.log(’’I have broken out of the loop’’);

In this case when i reaches 3 and 5 the rest of the code will not run (and 3 and 5 won’t be in the results -🡪 continue;) and when i = 7 it will break out from the loop and console.log will be printed.

So the result will be:   
1

2

4

6

7

I have broken out of the loop.

**Practical Example using loops**

**Numbers**

console.log(Math.round(7.8)); rounding, the result will be 8.

console.log(Math.floor(7.9)); flooring, result will be 7.

console.log(Math.ceil(7.1)); ceiling, result will be 8.

console.log(Math.max(7, 4, 9, 8)); chooses the biggest number, result will be 9 in this case.

**NaN**

var a =”7”;

var b = 5;

console.log(a \* b); 🡪result will be 35, in type of number.

var a =”7”;

var b = 5;

if (isNaN(a)) { //this cheks a is not a number, if it’s not number, it’s true.

console.log(’’That’s not a number’’);

} else {

console.log(’’both are numbers’’ + (a \* b));

}

if (!isNaN(a)) { // double negative, cheks a is a number, if it is, it’s true.

code

}

**Strings**

var myString = ’I\’m a „fun” string’; 🡪 the \ character makes the second ’ to not close the string tag.

console.log(myString.length); 🡪 counts the characters

console.log(myString.toUpperCase()); 🡪makes the string to uppercase

console.log(myString.toLowerCase()); 🡪makes the string to lowercase

console.log(myString.indexOf(’’string’’)); 🡪counts where starts the ’’string’’ word (res: 12)

var myString = ’I\’m a ’’fun” string’;

if (myString.indexof(’’ninja’’) === 1) {

console.log(’’the word ninja is not in the string’’);

} else {

console.log(’’the word ninja starts at position ’’ + myString.indexOf(’’ninja’’));

}

var string1 =’’abc’’;

var string2 =’’ABC’’;

console.log(string1.toLowerCase() === string2.toLowerCase()); 🡪true.

**Material Review**

* Node.js
  + node: parancssorban futtatja a JS fájlt
* TypeScript
  + tsc: átkonvertálja a TSC-t JSC-re
* TypeScript Node
  + ts-node: átkonvertálja és futtatja is a TSC-t JSC-ben
* console
* variables
  + var
  + let
  + const
* JavaScript types
  + number
  + string
  + boolean
  + null: szándékosan üres értéket adtunk a változónak
  + undefined: még sosem használt változó, lehet a típus is és az érték is undefined,
* TypeScript types
  + any
* operators
  + arithmetic (+, -, /, \*, %, \*\*, ++, --)
  + logical (&&, ||, !)
  + comparison (==, !=, ===, !==, <, <=, >, >=)
  + assignment (=)
  + typeof
* control structures
  + if
  + for
  + while
  + do while
  + (switch, break, continue)