Lecture 1

Primitive types, variables.

Working with console.

If-else statement

IT TALENTS
Training Camp

Contents

- The JavaScript language
- Setting up working environment
- First JavaScript program
- Primitives and variables
- Basic operations
- Statements
- Working with the console
- If-else statement and blocks

IT TALENTS
Training Camp

JavaScript language

- What is javascript as language
  - Developed in 10 days in May 1995 by Brendan Eich for Netscape
  - Also known as ECMA Script, LiveScript, JScript
  - Very widely used programming language
  - Suitable for desktop, mobile, web, office applications, machine automation
  - Object Oriented language
  - Uses C-like syntax and Lisp like lambdas
  - JavaScript is executed by JavaScript engines

JavaScript engine

- JavaScript source code is human readable code in .js (or other) files
- Interpretation
- Execution

First steps in JavaScript

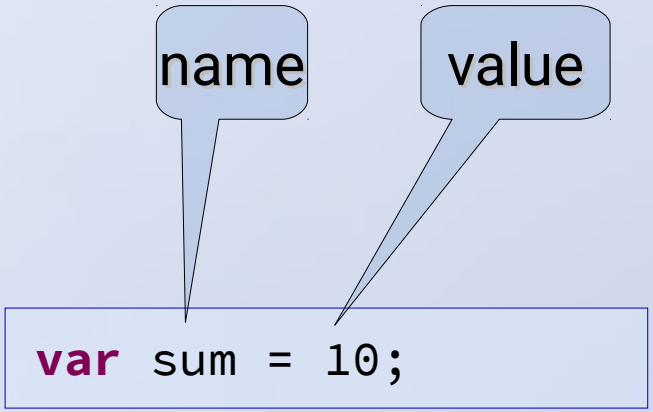
- Installing IDE
- Choose your favorite browser
- Press F12 and open the console

My first program

```
console.log('Hello world!');
```

Variables

- Variables in JavaScript
  - It's purpose is to hold information
  - Have an unique name
  - Have a type
  - Have a value (can be changed)
  - Can be global or local



- Declaring variable

Data types in JavaScript

- Variables can be used with basic operations
- Types in JavaScript
  - string
  - number
  - Boolean
  - Object

Number data type

- Number data types in JavaScript are **floating-point numbers**, but they may or may not have a fractional component. If they don't have a decimal point or fractional component, they're treated as integers—base-10 whole numbers in a range of $\mathbf{-2^{53}}$ to $\mathbf{2^{53}}$.

Null and undefined

var x; // variable is only declared, not initialized - undefined

var x = null ; //variable is initialized in null

Booleans

Can hold 2 values - true and false;

```
var isTrue = true;
var isNotTrue = false;
```

Strings

Because JavaScript is a loosely typed language, nothing differentiates a string variable from a variable that's a number or a boolean, other than the literal value assigned to the string variable when it's initialized and the context of its use. Example initialization:

```
var thisIsAString = "Some String";

var thisIsAnotherString ='Some Other String';

var someOtherString = new String(someVar);
```

Escaping strings

```
var hiMyNameIs = 'Hi, my name is \'Pesho\''
```

Operators

- Arithmetic +, -, *, /, %
- Logical &&, ||
- Assignment =, +=,-=, *=, /=
- Equality ==, !=, ===, !==
- Comparison >, < , >= , <= (greater than, less than, greater than or equal, less than or equal).
- Differences between / and %
- Bitwise operators - |, & , ^, ~, <<, >>, >>>

Try using some of them and print the result in console

Conditional Statement

- Logical NOT !
- Logical AND &&
- Logical OR | |

A	B	A || B	A && B	! A
false	false	false	false	true
true	false	True	false	false
false	true	true	false	true
true	true	true	true	false

Assignment Operators

Name	Shorthand operator	Meaning
Assignment	x = y	x = y
Addition assignment	x += y	x = x + y
Subtraction assignment	x -= y	x = x - y
Multiplication assignment	x *= y	x = x * y
Division assignment	x /= y	x = x / y
Remainder assignment	x %= y	x = x % y
Left shift assignment	x <<= y	x = x << y
Right shift assignment	x >>= y	x = x >> y
Unsigned right shift assignment	x >>>= y	x = x >>> y
Bitwise AND assignment	x &= y	x = x & y
Bitwise XOR assignment	x ^= y	x = x ^ y
Bitwise OR assignment	x |= y	x = x | y

IT TALENTS
Training Camp

Bitwise Operators

Operator	Usage	Description
Bitwise AND	a & b	Returns a one in each bit position for which the corresponding bits of both operands are ones.
Bitwise OR	a | b	Returns a one in each bit position for which the corresponding bits of either or both operands are ones.
Bitwise XOR	a ^ b	Returns a one in each bit position for which the corresponding bits of either but not both operands are ones.
Bitwise NOT	~ a	Inverts the bits of its operand.
Left shift	a << b	Shifts a in binary representation b (< 32) bits to the left, shifting in zeroes from the right.
Sign-propagating right shift	a >> b	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off.
Zero-fill right shift	a >>> b	Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off, and shifting in zeroes from the left.

Numeral Systems

IT TALENTS
Training Camp

Definition

A numeral system is a writing system for expressing numbers, that is, a mathematical notation for representing numbers of a given set, using digits or other symbols in a consistent manner.

IT TALENTS
Training Camp

Different Numeral Systems

Decimal	Binary	Octal	HexDecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

IT TALENTS
Training Camp

Converting From Binary to Decimal

128	64	32	16	8	4	2	1
1	0	0	1	1	0	1	1

$$128 + 0 + 0 + 16 + 8 + 0 + 2 + 1$$

$$= 155$$

IT TALENTS
Training Camp

Converting From Decimal to Binary

	Remainder:
2)156	0
2)78	0
2)39	1
2)19	1
2)9	1
2)4	0
2)2	0
2)1	1

$$156_{10} = 10011100_2$$

IT TALENTS
Training Camp

Control flow

- Control flow is the way a program goes – execution of predifined statements
- Control flow may differ each time in dependance of conditions – either input data, or predifined conditions by the programer(i.e – time and so on)
- During the program execution decisions are being met – the program flow branches

if-else statement

If (*condition*) {

statement

}

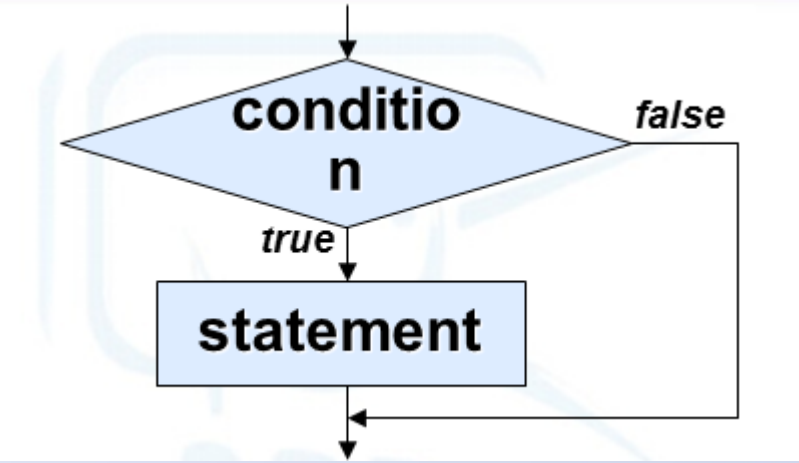
if (*condition*) {

executionA

} else {

executionB

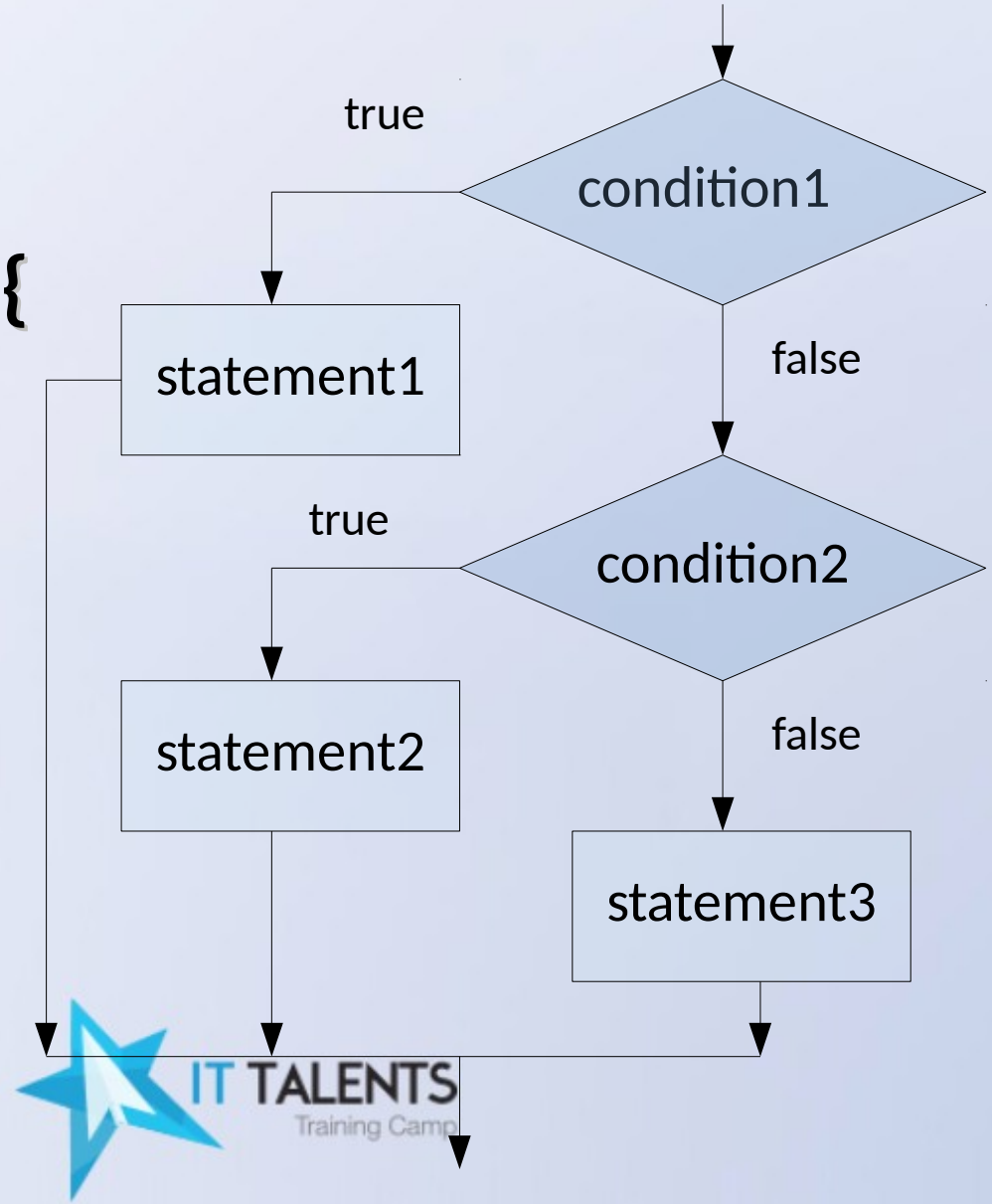
}



IT TALENTS
Training Camp

if-else statement

```
If (condition1) {
    statement1
} else if (condition2) {
    statement2
} else {
    statement3
}
```



if-else statement

If (*condition*) {

statement

}

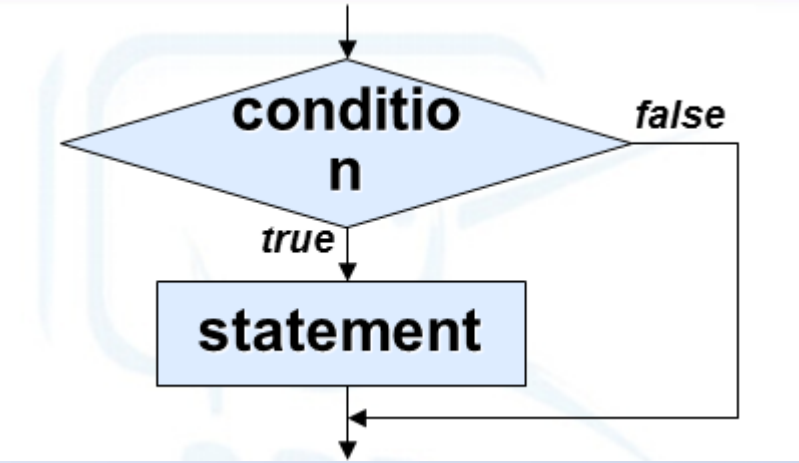
if (*condition*) {

executionA

} else {

executionB

}



IT TALENTS
Training Camp

if-else statement

- If can exist without else

  But

- Else can't exist without if
- Nested if-else statement

```
var a = 7.5;

if (a < 0) {
    console.log("a is smaller than 0");
} else {
    if (a == 0) {
        console.log("a is 0");
    } else {
        console.log("a is bigger than 0");
    }
}
```

Blocks

A block is a group of zero or more statements between balanced braces and can be used anywhere a single statement is allowed

```
if (a > 10) {
   Console.log("a is " + a);
   Console.log("a is bigger than 10");
} else {
   Console.log("a is not bigger than 10");
}
```

Always format your code! Do not write code like this:

```
if (a > 10) {
console.log("a is " + a);
console.log("a is bigger than 10");}
else {console.log("a is not bigger than 10");
}
```

Mistake

```
var a = 7;

if (a > 10); {
    console.log("a is " + a);
    console.log("a is bigger than 10");
}
```

In this case println statements will be executed no matter the condition!

```
var a = 7;

if (a > 10);

{
    console.log("a is " + a);
    console.log("a is bigger than 10");
}
```

Summary

- Startup
- Variables
- Primitive types
- Operators
- Working with the console
- If-else statement and blocks