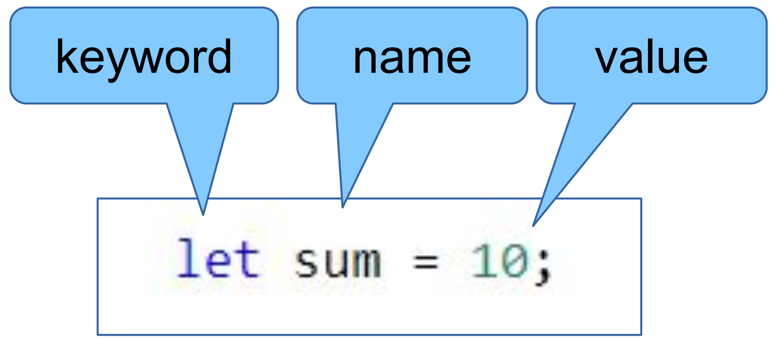
Primitive types, variables. Working with console. If-else statement

IT TALENTS
Training Camp

Variables

- It's purpose is to hold information
- Have an unique name
- Have a type
- Have a value (can be changed)



```
let sum = 10;
```

- Declare a variable
- Initialize a variable
- Initialize a variable that cannot be changed.

```
const name = 'Slavi';
```

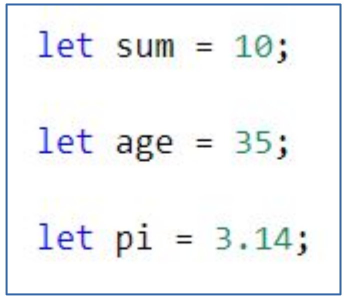
Data types in JavaScript

- Each variable holds information from a specific data type
- Types in JavaScript
- ● number
- ● boolean
- ● string
- ● undefined
- ● object

IT TALENTS Training Camp

Number data type

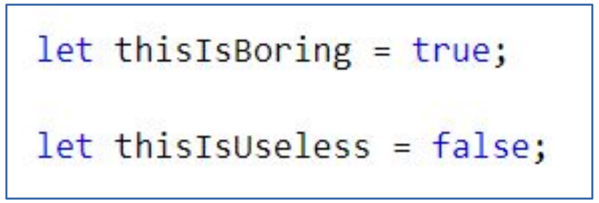
- Number data types in JavaScript are floating-point numbers, but they may or may not have a fractional component.
- If they don’t have a decimal point or fractional component, they’re treated as integers — base-10 whole numbers in a range of -2^{53} to 2^{53}
- Infinity
- NaN



```
let sum = 10;

let age = 35;

let pi = 3.14;
```

Boolean data type

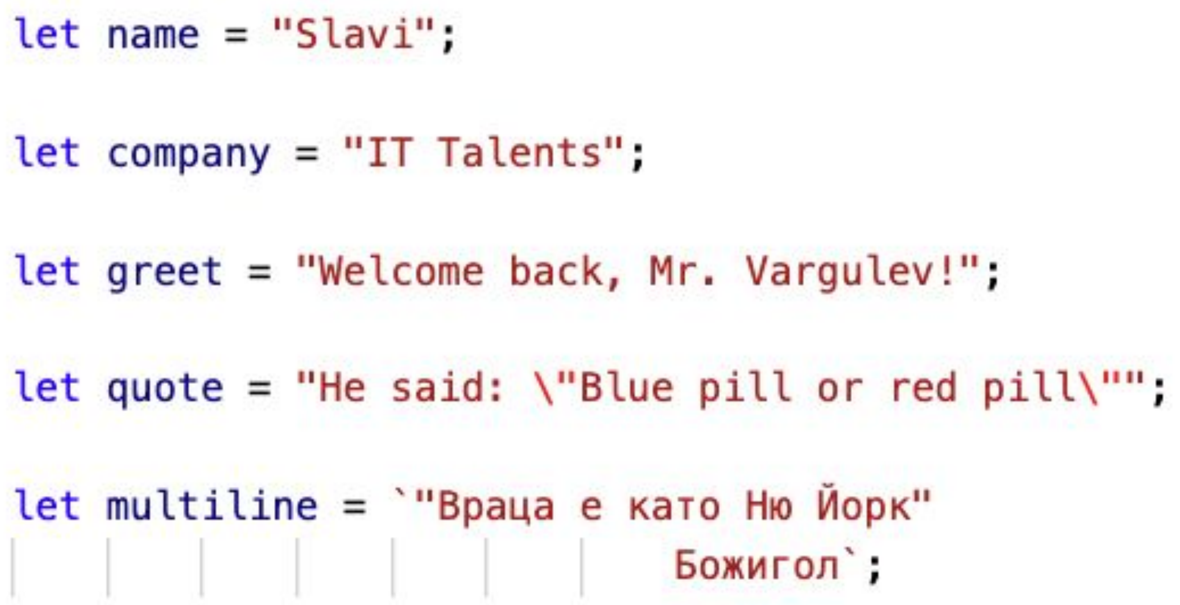
- Can hold 2 values – **true** and **false**
- Used to hold the validity of a statement



```
let thisIsBoring = true;

let thisIsUseless = false;
```

String data type

- Used to represent and manipulate a sequence of characters (hold some text)
- String literals can be specified using single, double quotes or back ticks, which are treated identically
- Special characters can be encoded using the escape notation: \



```
let name = "Slavi";

let company = "IT Talents";

let greet = "Welcome back, Mr. Vargulev!";

let quote = "He said: \"Blue pill or red pill\"";

let multiline = `"Враца е като Ню Йорк"
                             Божигол`;
```

Null and undefined data types

- When a variable is only declared, but not given a value – then its type is **undefined**
  - **Undefined** means someone did not set a value to the variable (either on purpose or not)
- When a variable is given a value of “null” - then its type is **null**
  - **Null** means someone did set a value on purpose and the value means that the variable holds nothing

```
let x; // undefined

let y = null; // on purpose
```

Dynamic Typing

- JavaScript is a loosely typed and dynamic language.
- Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types:

```
let x = 42;           // x is now a number
x = "Mr. Anderson"; // x is now a string
x = true;           // x is now a boolean
```

IT TALENTS Training Camp

Operators

- Javascript offers many operators for manipulating data
  - Unary – takes one operand
  - Binary – takes two operands
  - Ternary – takes three operands
- Operands are the elements that the operator performs an operation on
- Example: 2 + 3
  - + is the operator.
  - 2 and 3 are the operands

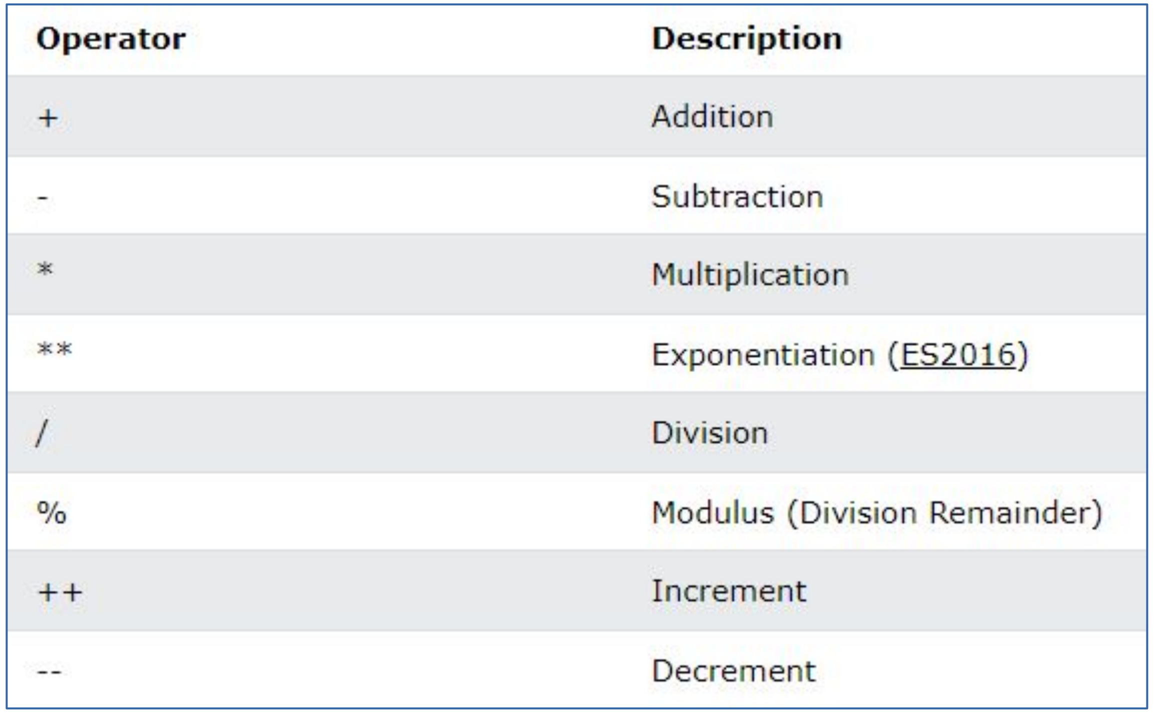
Operators

- Arithmetic +, -, *, /, %, **
- Differences between / and %
- Assignment =, +=,-=, *=, /=
- Equality ==, !=, ===, !==
- Comparison >, < , >= , <= (greater than, less than, greater than or equal, less than or equal).
- Logical &&, ||, !
- Try using some of them and print the result in console

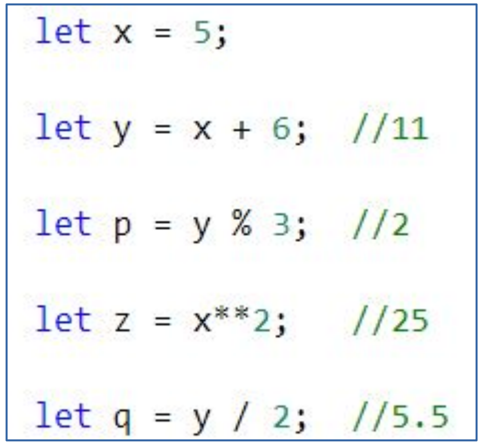
IT TALENTS
Training Camp

Operators

- Arithmetic



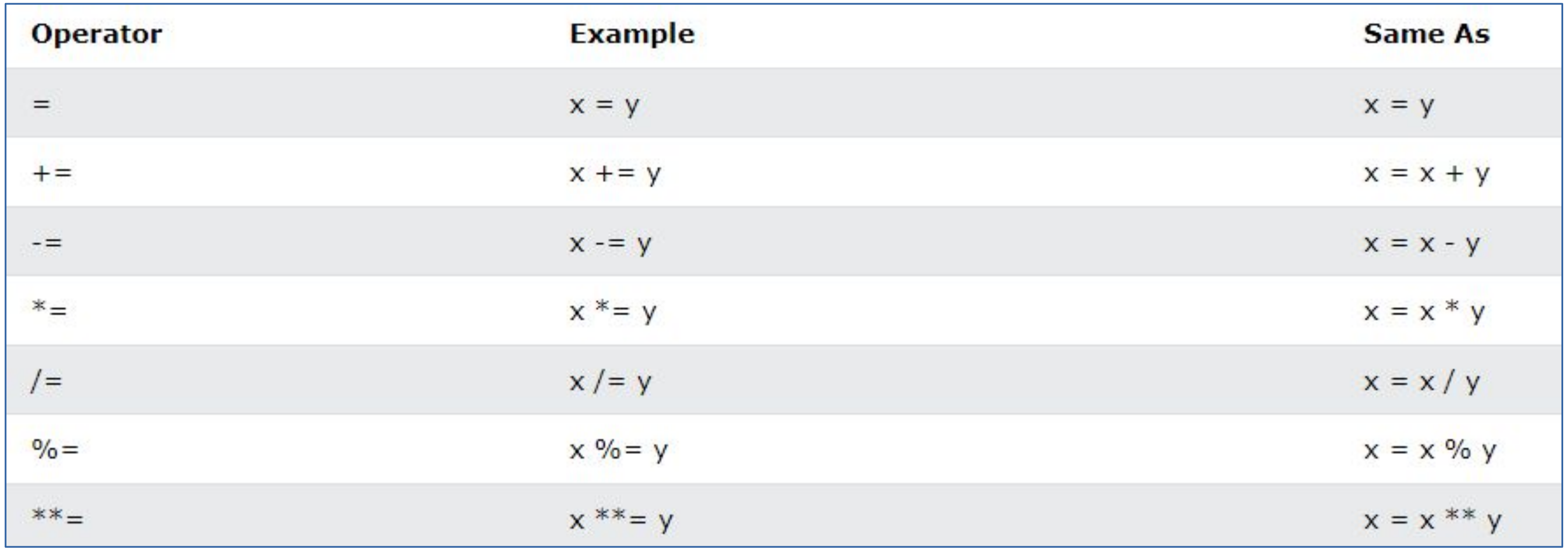
Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement



```
let x = 5;

let y = x + 6;  //11

let p = y % 3;  //2

let z = x**2;   //25

let q = y / 2;  //5.5
```

Operators

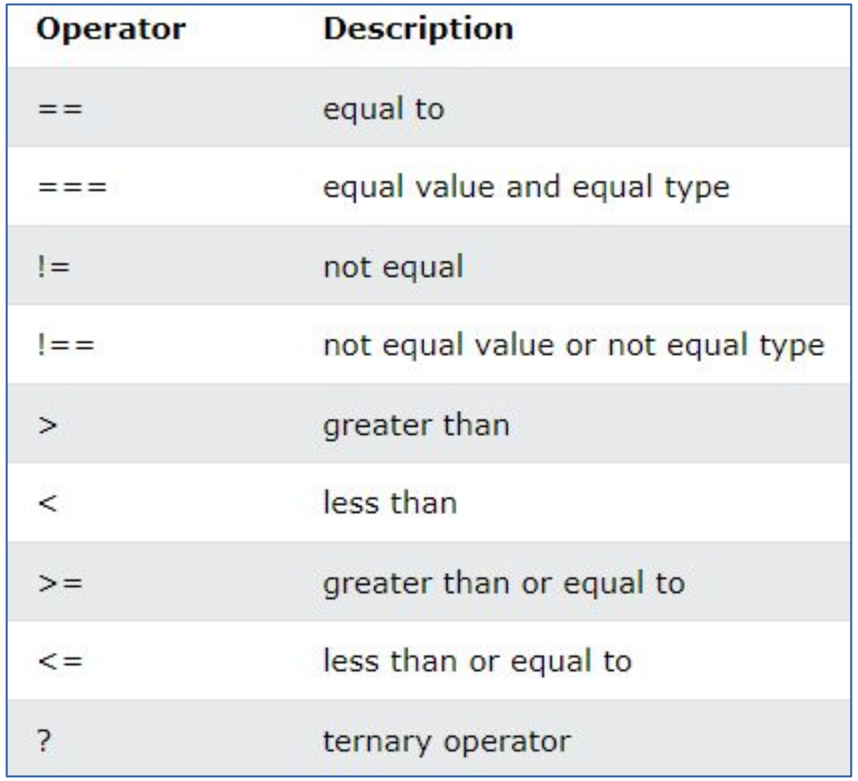
- Assignment



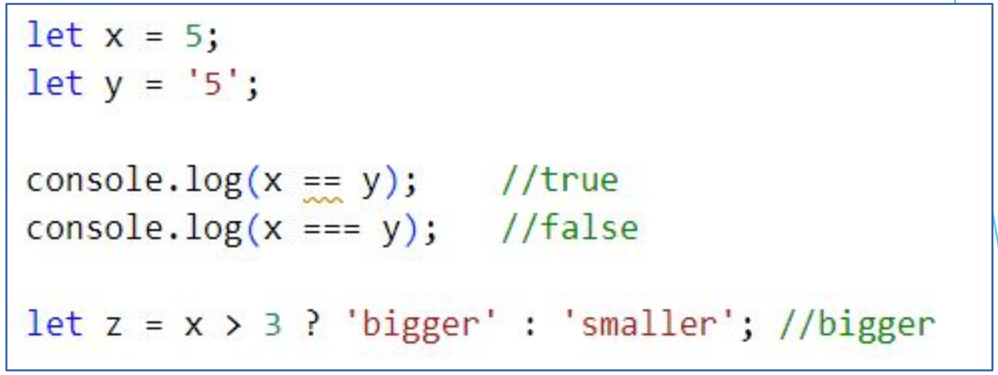
Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

Operators

- Comparison



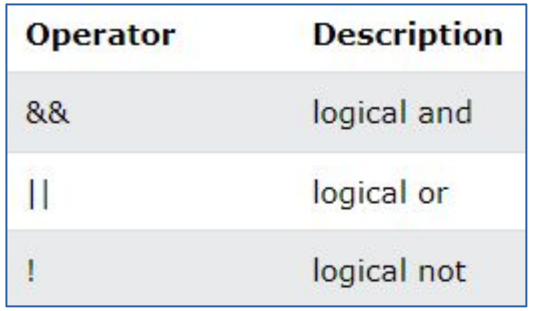
Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator



```
let x = 5;
let y = '5';

console.log(x == y);     //true
console.log(x === y);    //false

let z = x > 3 ? 'bigger' : 'smaller'; //bigger
```

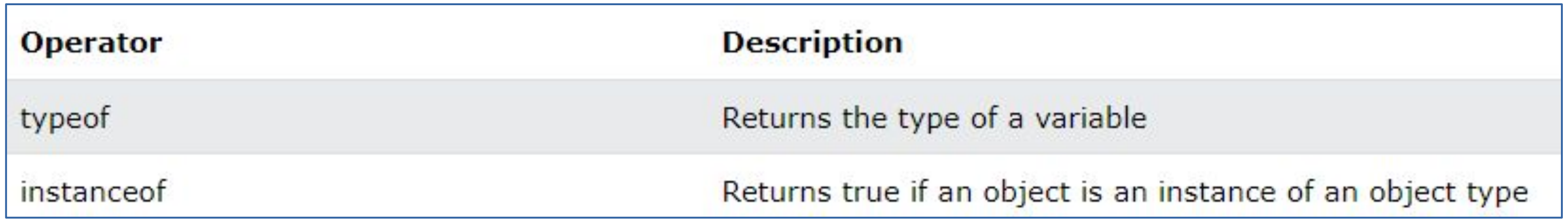
Operators

- Logical



Operator	Description
&&	logical and
||	logical or
!	logical not

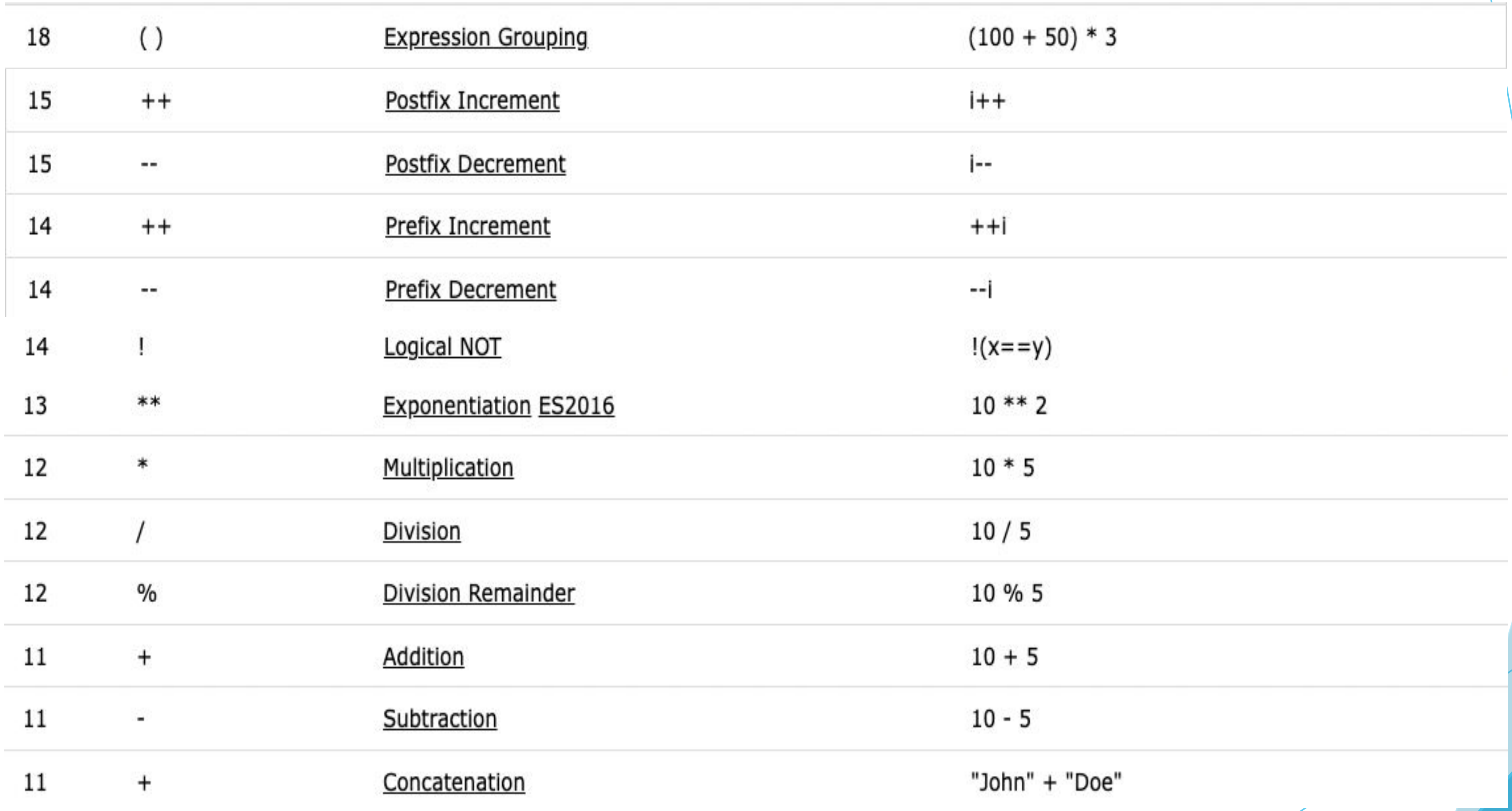
- Type



Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

Operator precedence

- When we have multiple operators, the rules that define what operator is evaluated first are called operator precedence



18	( )	Expression Grouping	(100 + 50) * 3
15	++	Postfix Increment	i++
15	--	Postfix Decrement	i--
14	++	Prefix Increment	++i
14	--	Prefix Decrement	--i
14	!	Logical NOT	!(x==y)
13	**	Exponentiation ES2016	10 ** 2
12	*	Multiplication	10 * 5
12	/	Division	10 / 5
12	%	Division Remainder	10 % 5
11	+	Addition	10 + 5
11	-	Subtraction	10 - 5
11	+	Concatenation	"John" + "Doe"

IT TALENTS
Training Camp

Operator precedence

9	<	Less than	x < y
9	<=	Less than or equal	x <= y
9	>	Greater than	x > y
9	>=	Greater than or equal	x >= Array
8	==	Equal	x == y
8	===	Strict equal	x === y
8	!=	Unequal	x != y
8	!==	Strict unequal	x !== y
4	&&	Logical AND	x && y
3	||	Logical OR	x || y

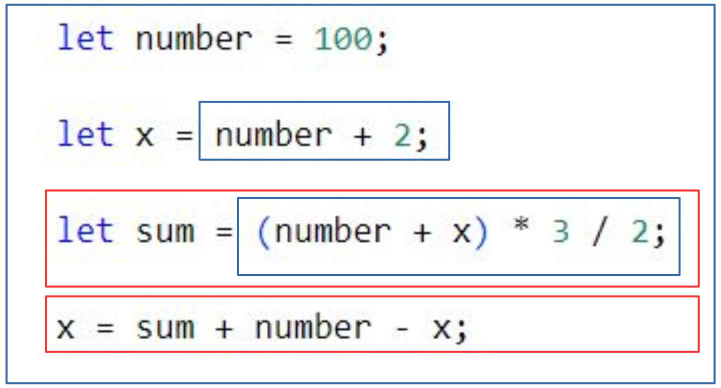
IT TALENTS
Training Camp

Expressions and statements

- Expression is a construct, made up of variables, operators and method invocations, that evaluates to a single value

- Statement is a complete unit of execution. Terminate with ;

- Example expressions ▭
- Example statements ▭

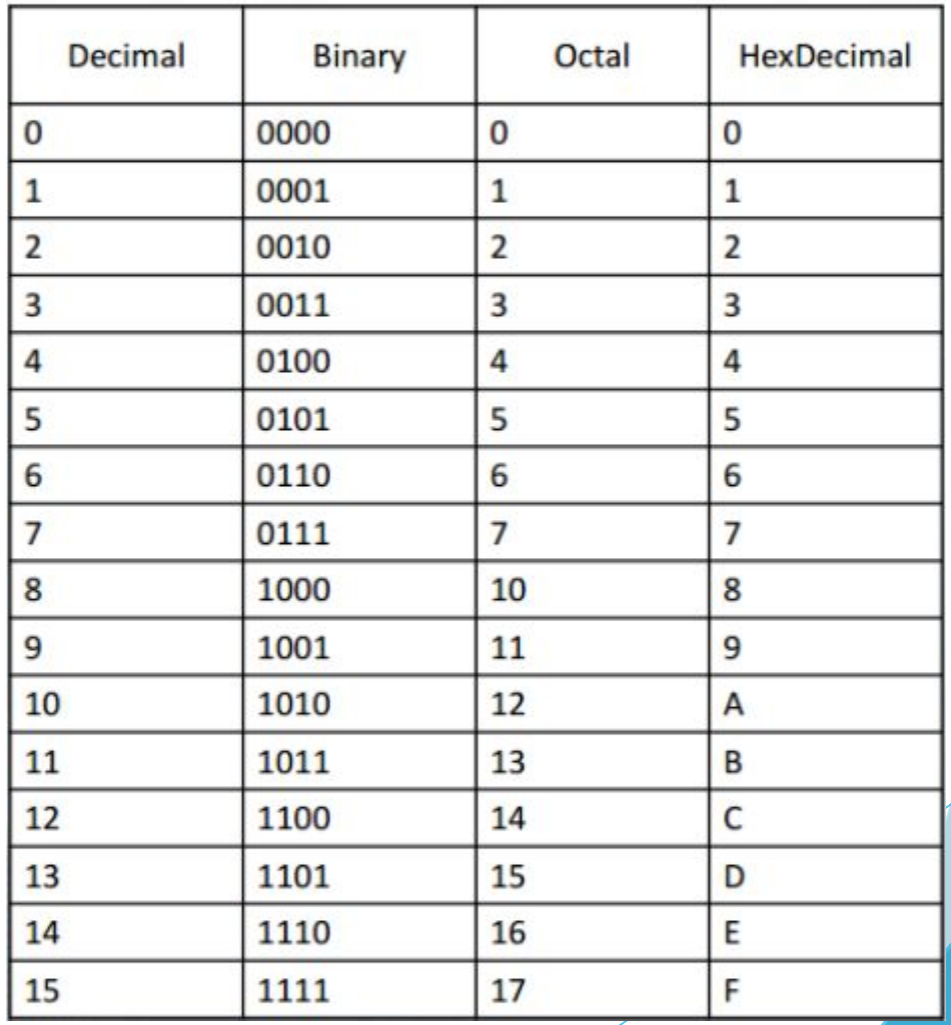


```
let number = 100;

let x = number + 2;

let sum = (number + x) * 3 / 2;

x = sum + number - x;
```

Numeral Systems

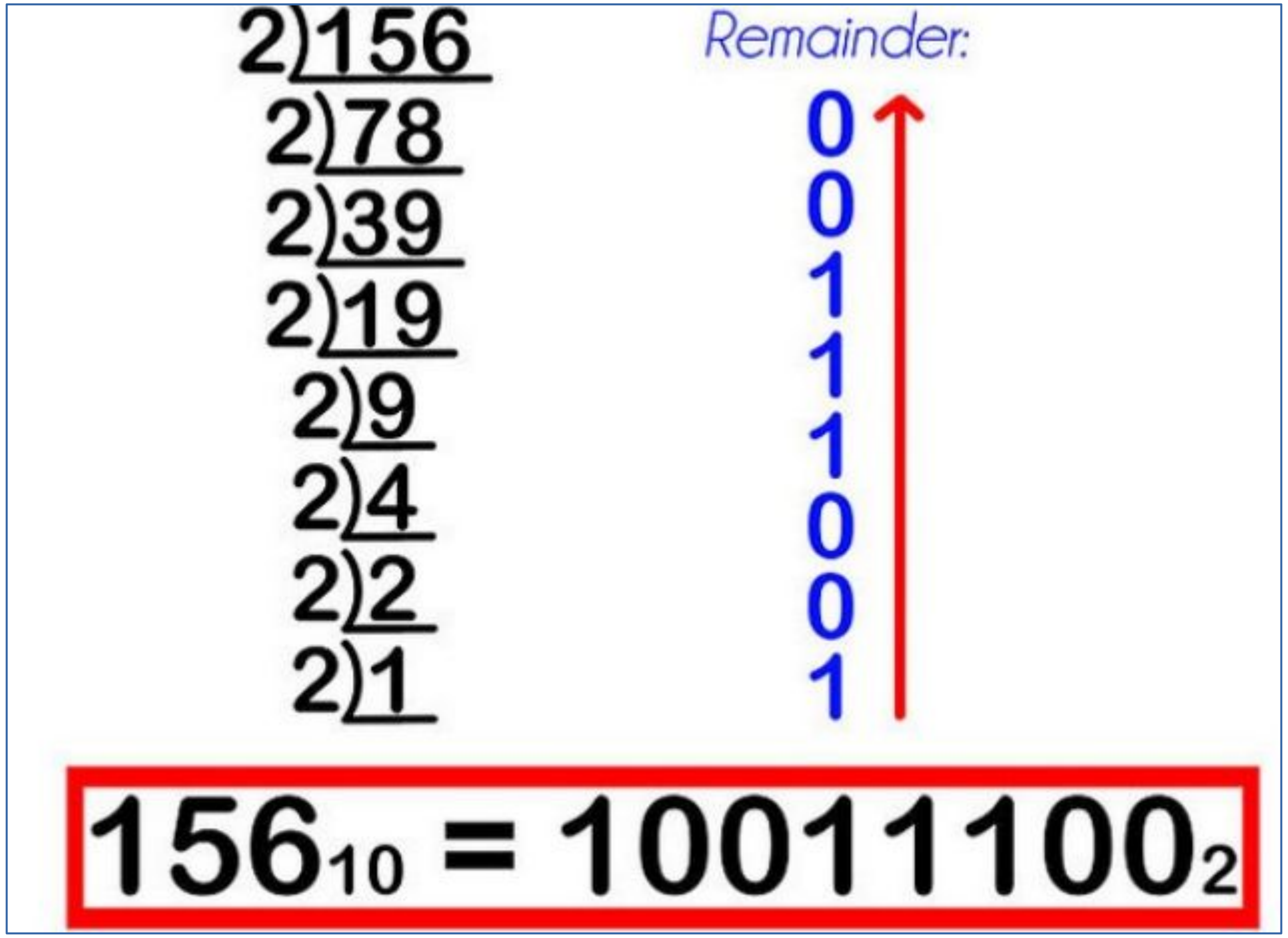
- A numeral system is a writing system for expressing numbers, that is, a mathematical notation for representing numbers of a given set, using digits or other symbols in a consistent manner.

- Different Numeral Systems



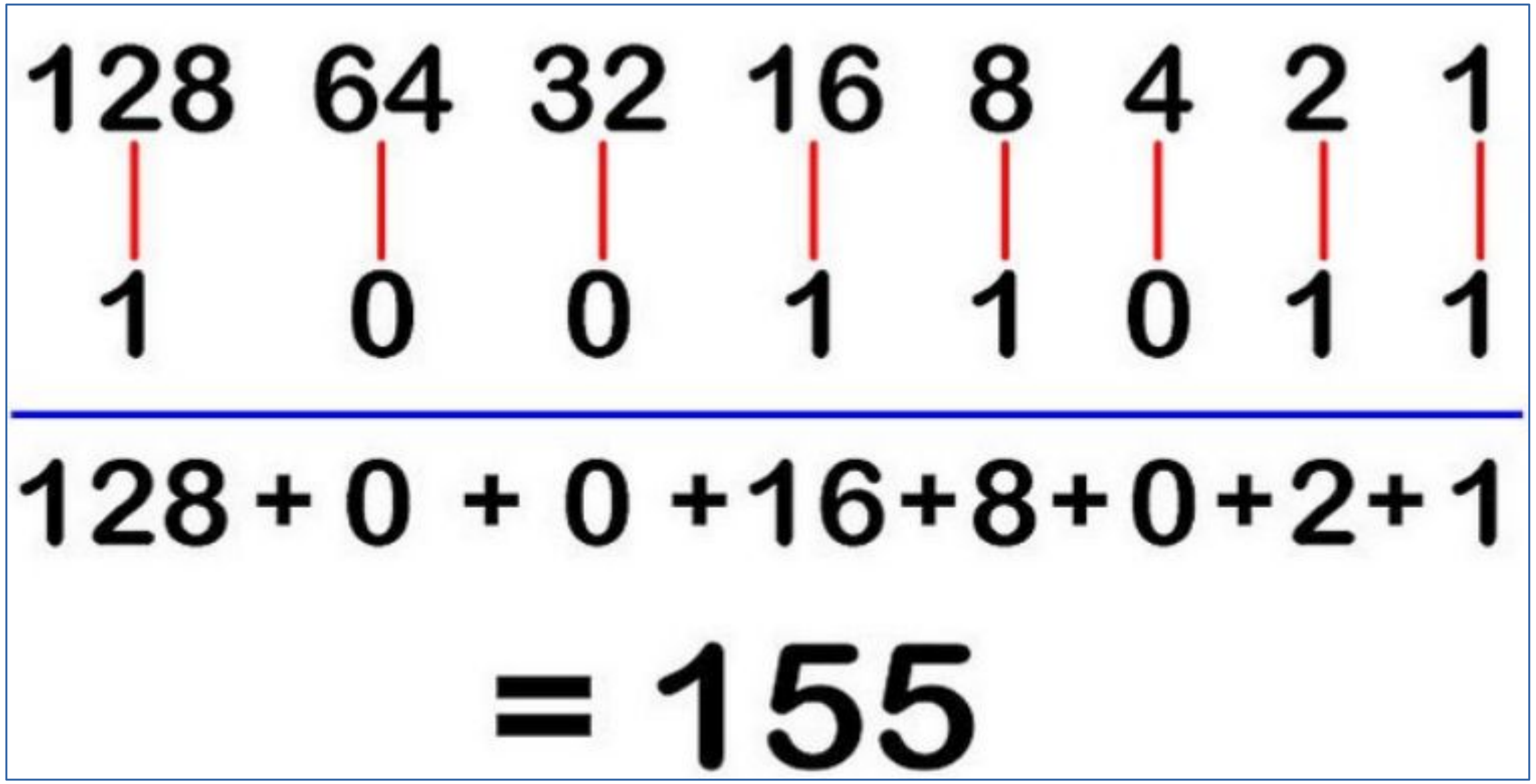
Decimal	Binary	Octal	HexDecimal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Converting From Decimal to Binary



$156_{10} = 10011100_2$

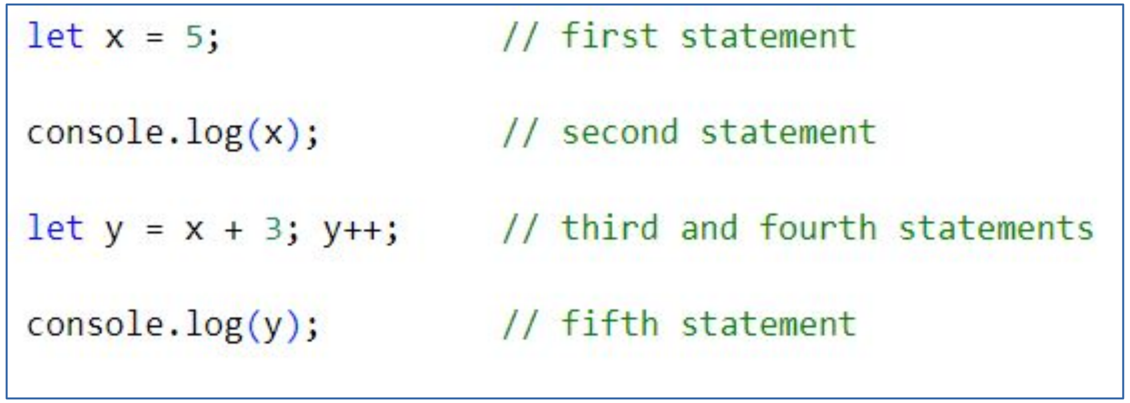
Converting From Binary to Decimal



Control Flow

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

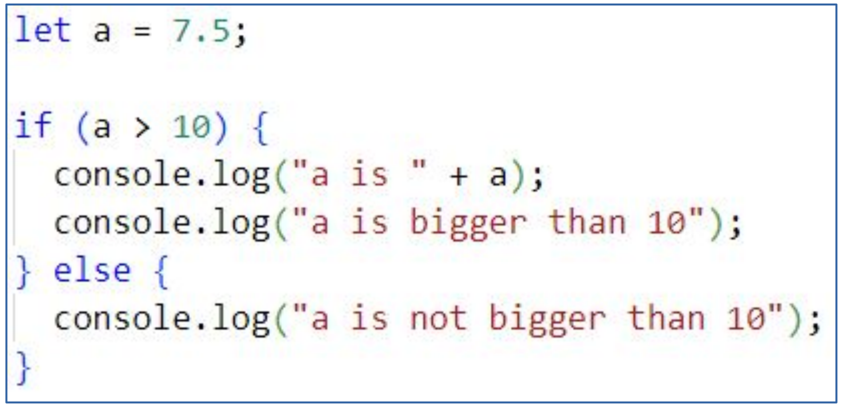
- **Usually** the control flow is from left to right, from up to down



```
let x = 5;               // first statement

console.log(x);          // second statement

let y = x + 3; y++;      // third and fourth statements

console.log(y);          // fifth statement
```

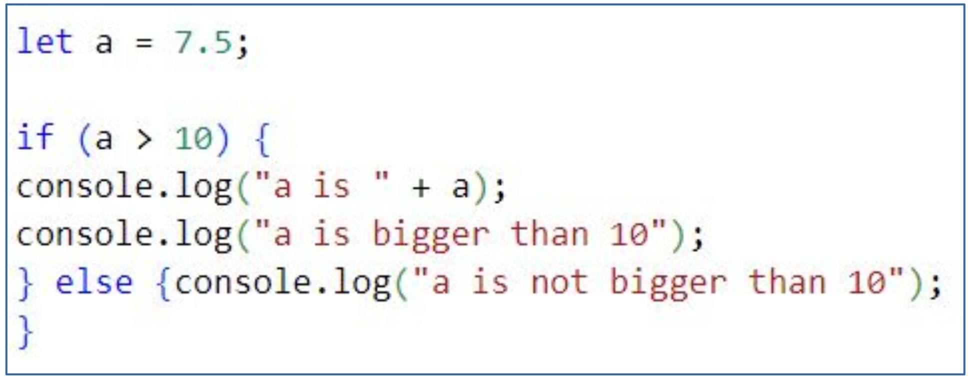
Blocks

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



```
let a = 7.5;

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:



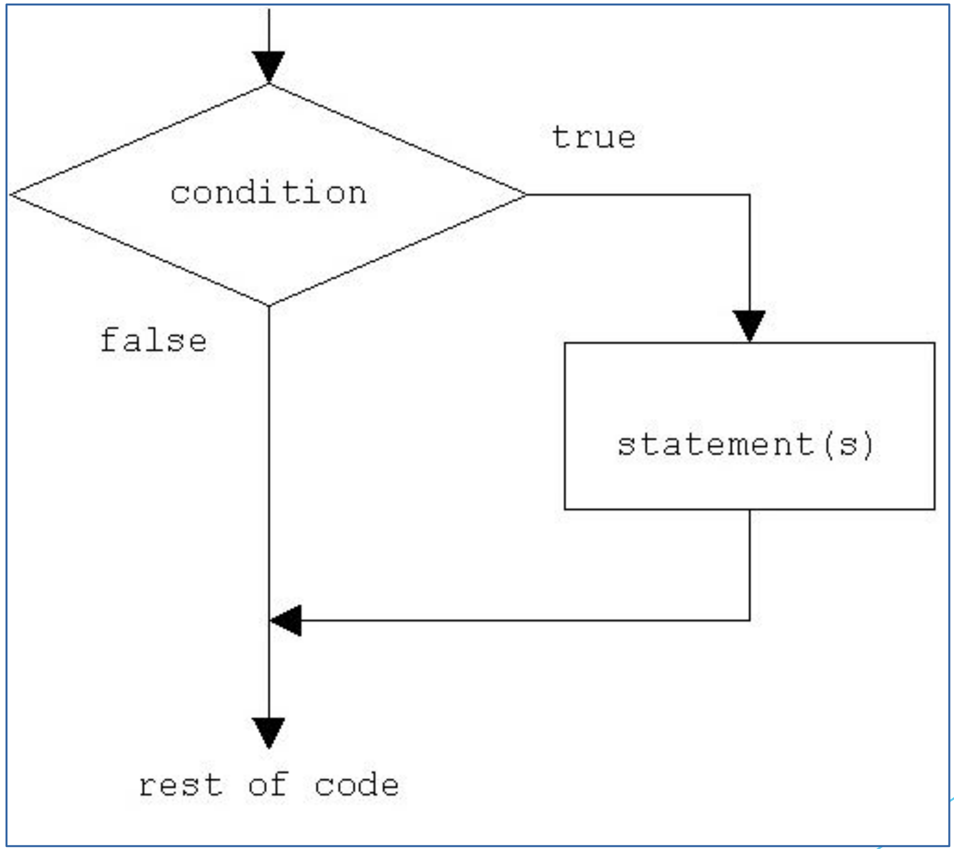
```
let a = 7.5;

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

If-else statement

- A statement that gives us the opportunity to execute a piece of code only if a certain condition is met

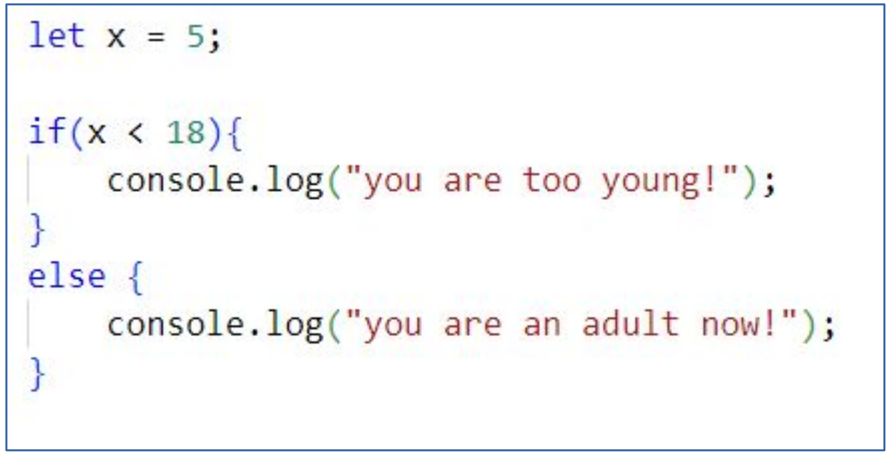
```
if ( condition ) {

    statement

}
```

```
if ( condition ) {

    statement A

}
else {

    statement B

}
```



If-else statement

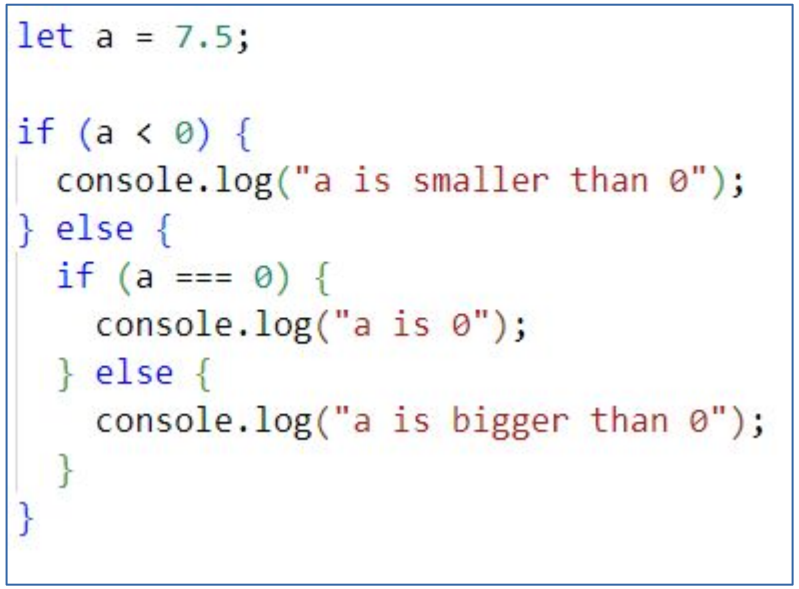
- There is an option to execute one piece of code if the condition is met and another one if the condition is not met
- Both pieces of code will never be executed together



```
let x = 5;

if(x < 18){
    console.log("you are too young!");
}
else {
    console.log("you are an adult now!");
}
```

If-else statement

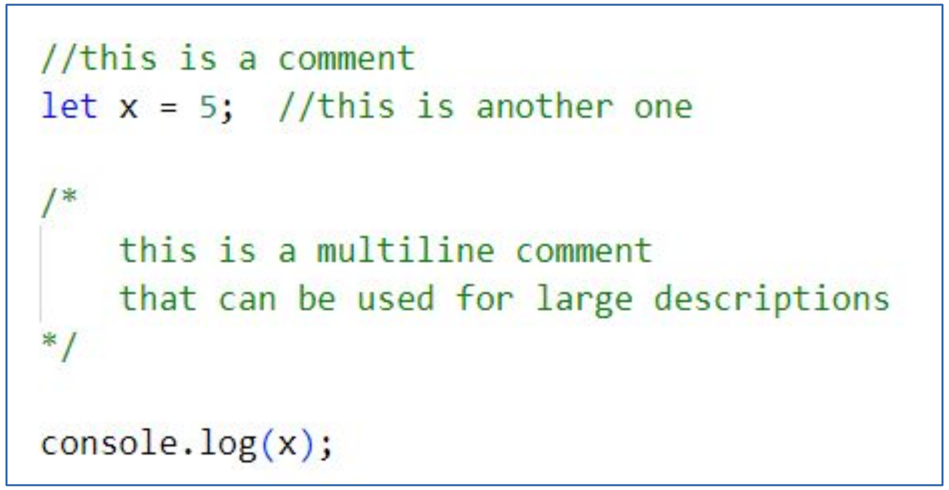
- **If** can exist without **else**
- But **else** can't exist without if
- Nested if-else statement



```
let 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");
  }
}
```

Comments

- If you want to leave a note in the code you can use comments.
- Comments are not executed from the JS Engine
- The general purpose of the comments is to make code more readable and easier to understand
- Examples:
  - Single line comment with: //
  - Multi line comment with: /* .... */



```
//this is a comment
let x = 5;  //this is another one

/*
    this is a multiline comment
    that can be used for large descriptions
*/

console.log(x);
```

Using the Math Object

- Math is a built in object that you can use to do some Math stuff.
- Find the absolute value of a number: **Math.abs(n)**
- Round the number: **Math.round(num)**
- Random number generator: **Math.random()**
- Find the square root of number: **Math.sqrt(num)**
- and many more…

IT TALENTS
Training Camp