Types, Values, Operators

Part 1

$$a = b + 3$$

Statements

A group of

- words
- numbers
- operators

that performs a specific task is a statement.

Expressions

Statements are made up of one or more expressions.

a = b + 3

3 is a number literal

= and **+** are operators

a and **b** are variables

Types

number

string

boolean

null

undefined

Types: number

```
7, -7

0, -0

0.19, -0.19
```

Types: number

```
7, -7

0, -0

0.19, -0.19

5.1E+7

Infinity(Number.POSITIVE_INFINITY), -
Infinity(Number.NEGATIVE_INFINITY)

Number.MAX_VALUE, Number.MIN_VALUE

NaN
```

Types: number - base Systems

```
Base-10
       0, 3, 100


Base-16 (hexadecimal)
       0x9, 0xA,0x12E


Base-8 (octal)
       00, 07, 0112
```

Types: String

```
"Moon"

'Jupiter'

'Saturn\'s moon'

`Milky
  Way`
```

Types: boolean

```
true

false
```

Type Conversion

number -> string

string -> number

number -> boolean

boolean -> number

string -> boolean

boolean -> string

Type Conversion: Falsy values

- `""`
- `0, -0, NaN`
- `null, undefined`
- `false`

Variables

Containers to store the data.

var

let

const

Types: Null & undefined

null is a language keyword that evaluates to a special value that is usually used to indicate the absence of a value.

Types: Null & undefined

null is a language keyword that evaluates to a special value that is usually used to indicate the absence of a value.

undefined is the value of variables that have not been initialized or of the property of an object which does not exist.

Operators: Unary Operators

`++`(increments)

`--`(decrements)

`-`(sign changes)

`+`(converts)

`typeof`

Operators: Binary Arithmetic Operators

+

−

*

/

%

Operators: Assignment Operators

=

+=

-=

*=

/=

%=

Operators: Binary Comparison Operators

`==`, `===` (equality)

`!=`, `!==` (inequality)

`>`, `>=` (greater than, greater than or equal)

`<`, `<=` (less than, less than or equal)

Operators: Logical Operators

Boolean Algebra

! (logical not)

&& (boolean and)

|| (boolean or)

Ternary operator

?:

Arithmetics: Math

```
Math.pow(n, m)
Math.sqrt(n)
Math.ceil(n)
Math.floor(n)
Math.log(n)
Math.PI
Math.E
```

Conditionals

```
if(expression)
        statement
```

```
if(expression)
        statement

else if(another expression)
        statement

else
        statement
```

Conditionals

```
switch(expression) {
        case value:
                statements
                break;

    default:
                statements

}
```

Thanks!