

```
<script>
  /*
  CSCI-111 Web Programming and Problem Solving
  */
  function week_7_lecture(session)
  {
    console.log("JavaScript Functions")
    const instructors = ["Irina Dolzhikova", "Talgat Manglayev", "Aygerim Yessenbayeva"]
    console.log(instructors[session - 1])
  }

  week_7_lecture(2)

</script>
```

Content

- Comparison
- Logical operators
- Functions
 - Function declaration and call
 - Arguments and parameters
 - Scope

Comparison

Greater than	$a > b$
Lower than	$a < b$
Greater than or equal to	$a \geq b$
Lower than or equal to	$a \leq b$

Comparison

Equal value	<code>a == b</code>
Value and data type are equal	<code>a === b</code>
Not equal value	<code>a != b</code>
Not equal value or data type	<code>a !== b</code>

Comparison

```
if (condition)  
    statement
```

*if condition in brackets is true
execute the statement*

```
if (hiddenNumber > enteredNumber)  
    console.log("The hidden number is larger")
```

Comparison

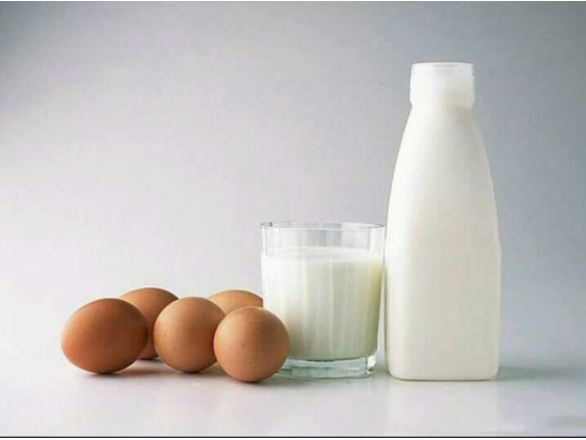
```
if (condition)
{
    statement 1
    statement 2
}
```

*if condition in brackets is true
execute the statements 1 and 2*

*{ } are mandatory for more than
one statement*

```
if (hiddenNumber > enteredNumber)
{
    console.log("The hidden number is larger")
    console.log("Try again")
}
```

Comparison



Mom: *"Honey, please go to the market and buy 1 bottle of milk. If they have eggs, bring 6".*

I came back with 6 bottles of milk.

Mom: *"Why did you buy 6 bottles of milk?"*

Me: *"Because they had eggs"*

Comparison



```
if(eggs == true)
{
    buy 6 bottles of milk
}
else
{
    buy 1 bottle of milk
}
```

Mom: *"Honey, please go to the market and buy 1 bottle of milk. If they have eggs, bring 6".*

I came back with 6 bottles of milk.

Mom: *"Why did you buy 6 bottles of milk?"*

Me: *"Because they had eggs"*

Comparison

```
if (condition)
{
    statement 1
}
    statement 2
```

logic error

*if condition in brackets is true
execute the statement 1*

*then (even if condition is false)
execute the statement 2*

```
if (hiddenNumber > enteredNumber)
{
    console.log("The hidden number is larger")
}
    console.log("Try again")
```

Comparison

```
if (condition)
{
    statement 1
}
else
{
    statement 2
}
```

*if condition in brackets is true
execute the statement 1*

*otherwise
execute the statement 2*

{ } are optional for one statement

Comparison

```
if(hiddenNumber == enteredNumber)
{
    console.log("Congratulations!")
    console.log("You have found the hidden number!")
}
else
{
    console.log("It is not the hidden number!")
    console.log("Try Again!")
}
```

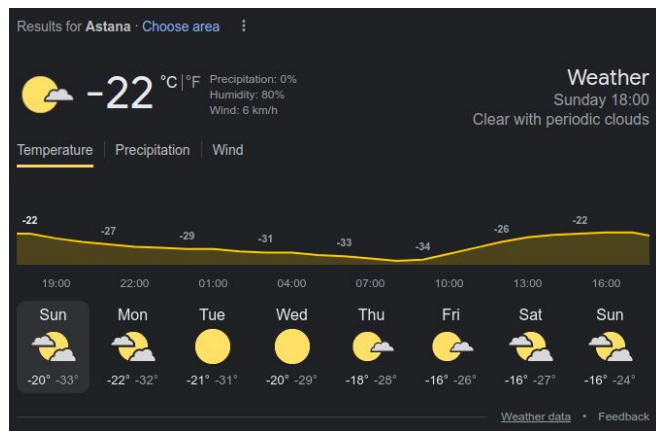
Logical Operators

and	<code>(a > 0) && (a < 10)</code>
or	<code>(a > 0) (a < 10)</code>
not	<code>! (a == 10)</code>

Functions

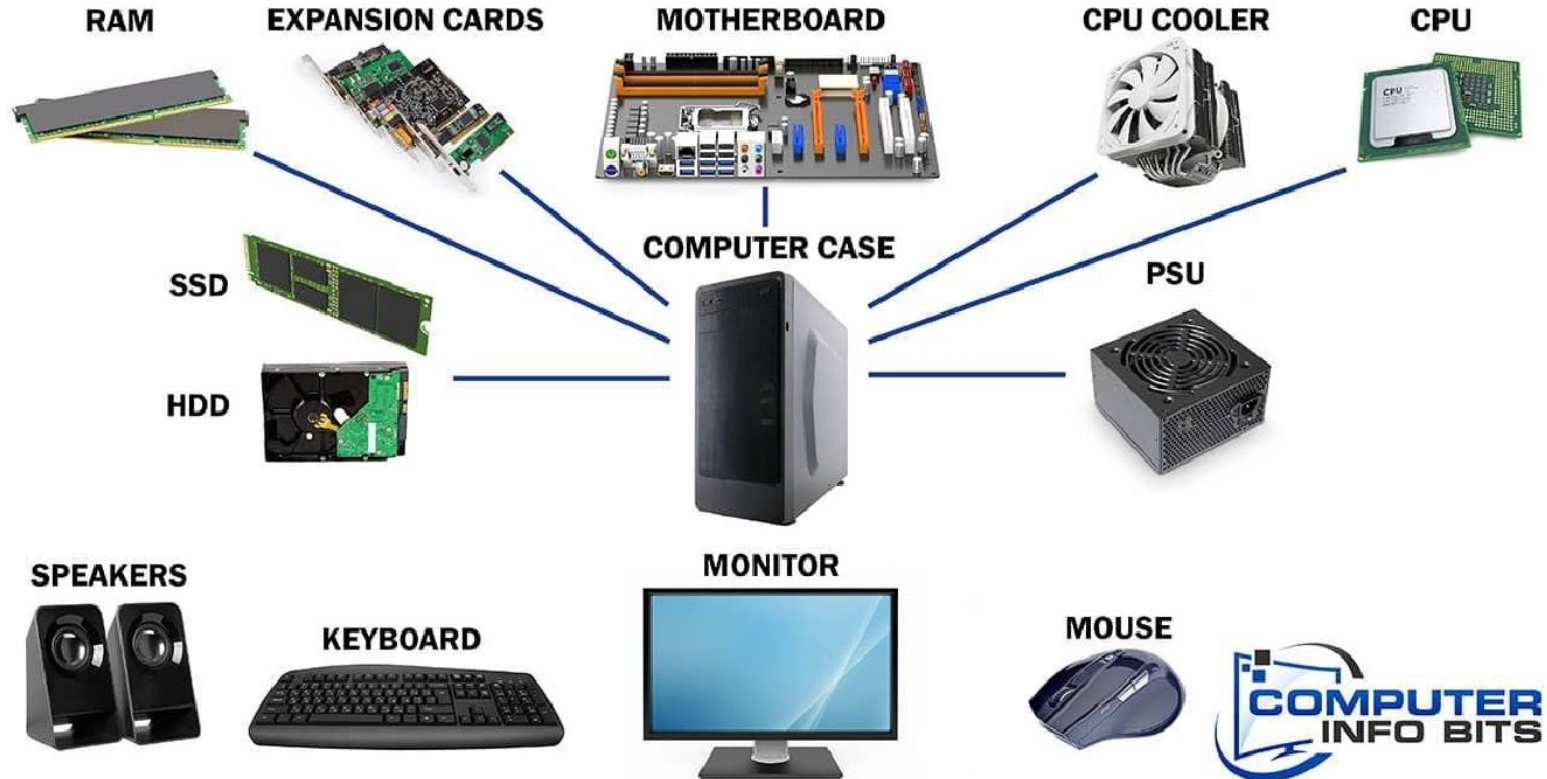
When we consume service:

- we are not interested in the way it is done;
- we can call it multiple times;



Introduction

PARTS OF A COMPUTER



Example 1: Parts of a Computer that perform some specific task

Introduction



Example 2: ATM machine

What is JavaScript function?

A **function** is a block of a program code designed to perform a specific task.

The functions are used to provide:

- *Reusability*: to use the same code multiple times in the program
- *Abstraction*: to hide the internals of the code for end user
- *Modularity*: to organize and divide the program into sub tasks

Function usage

The usage of a function is a two phase process:

- 1 – Function declaration (or definition)
- 2 – Function invocation (or execution, or call)

Function declaration

```
function multiplyByTwo(a)
{
    return a * 2;
}
```

Function call

```
function multiplyByTwo(a)
{
    return a * 2;
}

let a = 3;

let b = multiplyByTwo(a);

console.log(b);
```

Function declaration

```
const multiplyByThree = function(a)
{
    return a * 3;
}
```

Function call

```
const multiplyByThree = function(a)
{
    return a * 3;
}

let a = 3;

let b = multiplyByThree(a);

console.log(b);
```

Function declaration and call comparison

```
function multiplyByTwo(a)
{
    return a * 2;
}
```

```
let a = 3;
```

```
let b = multiplyByTwo(a);
```

```
console.log(b);
```

```
const multiplyByThree = function(a)
{
    return a * 3;
}
```

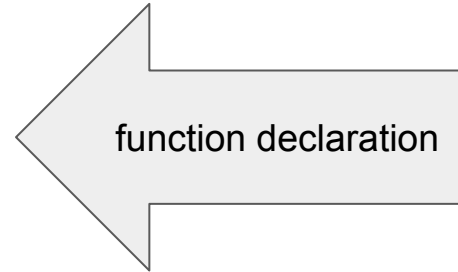
```
let a = 3;
```

```
let b = multiplyByThree(a);
```

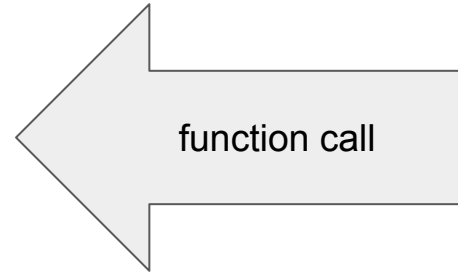
```
console.log(b);
```

Function parameters and arguments

```
function functionName(parameter)  
{  
    //do something  
    return something;  
}
```



```
let a = 3;  
  
let b = functionName(argument);
```



Function arguments

- *The arguments are passed by values and not visible to the outside of the function*
- *If a function changes an object property, it changes the original value*
- *The functions can be passed as the arguments*

Scope

Scope determines the accessibility (visibility) of variables.

JavaScript has 3 types of scope:

- Global scope – visible everywhere
- Block scope – visible within a block “{}”
- Function scope – visible within a function

Scope

Global Scope. A variable declared outside a function, have Global Scope. Global variables can be accessed from anywhere in a JavaScript program.

Block Scope. Let and Const variables declared inside a { } block cannot be accessed from outside the block.

Function Scope. Variables declared within a JavaScript function, can only be accessed from within the function.

*variables assigned without var, let and const automatically have a global scope;

https://www.w3schools.com/js/js_scope.asp

13.html, 15.html

Summary

Key takeaways:

- A function is block of code to perform specific task and used for:

 reusability, abstraction and modularity
- Remember 2-step function usage: declare and call
- The arguments are passed by value, though object and array arguments can be changed inside a function
- JavaScript has 3 types of scope: block, function, global
- Visibility of variables differs depending on scope