Functions

IT TALENTS
Training Camp

The scary term “function”

- We are going to see and talk for JS functions today.
- Think of the function as a black-boxed JS magic that do something for us
- You can put something on the one side of this box and get something different as a result from the other side

JavaScript Magic box

"slavozar"

toUpperCase()

"SLAVOZAR"

Why do we need functions

- **The holy grail of coding**
- It is all about code reuse
- We need to write less code
- We cry when we have to write same code more than once
- We also curse... a lot

Functions

- **The strict definition**
- A function is a block of code which is only invoked when the function is called.
- If the function is never invoked, the code associated with the function is never executed.
- Functions are very powerful as they allow us to break up our code into modular chunks

Declare a function

- We use the keyword function to define a custom function. A user defined functions have:
  - Name – a unique identifier for the function
  - Arguments – values that someone passes for the function to work with
  - Block of code – actual statements defined within the body

```
function name(arguments){
    //body
}
```

Simple Function

- Let's create our first function
- The function will output a sum of two numbers

```
function sum() {
  console.log(4 + 2);
}
```

- Calling user defined functions…is there something tricky?
  - No … just use the name and parentheses.

- You can call a function even before defining ig

```
sum();
```

IT TALENTS
Training Camp

Function Arguments

- We can define a function which accepts arguments
- We use variable syntax in the braces to define what arguments the function will have

```
function sum(a, b) {
    console.log(a+b);
}

sum(4, 5); //9
```

- **arguments** is an Array-like object accessible inside functions that contains the values of the arguments passed to that function

```
function sum() {
    console.log(arguments[0]); //4
    console.log(arguments[1]); //5
}

sum(4, 5);
```

Function Result

- We could define a result of a function by using the **return** keyword
- When used, an immediate result is retrieved based on the expression after the **return** keyword
- No other statements below the **return** expression are executed

```
function sum(a, b) {
    return a+b;
}

const res = sum(4, 5);
console.log(res); //logs 9
```

Functions and Scopes

- Check this out:

```
function sum() {
    var a = 4;
    var b = 2;
    console.log(a+b);
}

sum();
console.log(a); // a is not defined
```

- Scope of a variable is the “place” where a variable exists and can be used
- A variable can be used from when it is declared until its scope ends

Variables and Scopes

- **Variables in JavaScript have function scope.**
- All variables defined in a function are invisible for the other parts of the script (outside of the function)
- All the variables outside of a function can be accessed in the function – they do live in the global scope

```
function sum() {
    var a = 4; //local variable
    var b = 2;
    console.log(a+b);
    //its scope ends here
}
//here a does not exist
sum();
console.log(a); // a is not defined
```

```
let x = 5;//global variable

function sum() {
    //available here also
    console.log(x + 3);
}

sum();
console.log(x);
//its scope ends with the end of the script
```

IT TALENTS
Training Camp