JavaScript

Functions

Hackbright Academy

Functions

A block of code designed to perform a particular task.

Functions

A block of code designed to perform a particular task.

And is executed when "something" invokes it (calls it).

Function Example

```
function myFunction(p1, p2) {
    return p1 * p2; // the function returns the product of p1 and p2
}
```

Function Syntax

Defined with the function keyword

Hackbright Academy

Function Syntax

Defined with the function keyword, followed by a name

Hackbright Academy

Function Syntax

Defined with the function keyword,
followed by a name,
followed by parentheses ().

Function Syntax

The parentheses may include a list of parameter names:

```
(parameter1, parameter2, .....)
```

Hackbright Academy

Function Syntax

The code to be executed by the function is placed inside curly brackets: {}

Hackbright Academy

Function Syntax

```
function functionName(parameters) {
    code to be executed
}
```

Invoking a Function

- When an event occurs (when a user clicks a button)

Hackbright Academy

Invoking a Function

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code

Invoking a Function

- When an event occurs (when a user clicks a button)
- When it is invoked (called) from JavaScript code
- Automatically (self invoked)

Function Parameters and Arguments

You can pass values to functions.

These values are called arguments or parameters.

Function Parameters and Arguments

Multiple parameters are separated by commas:

```
function myFunction(parameter1, parameter2) {
    code to be executed
}
```

Hackbright Academy

Function Parameters and Arguments

The parameters and the arguments must be in the same order:

```
var x = myFunction(argument1, argument2);
```

Hackbright Academy

Function Parameters and Arguments

Inside the function, the arguments can be used as local variables. (More on this in a moment.)

The Return Statement

The function stops executing when a return statement is reached.

The Return Statement

If invoked from a statement, the code will continue after the function returns.

The Return Statement

Functions often computes a return value. The return value is "returned" back to the "caller":

```
//Calculate the product of two numbers, and return the result:
var x = myFunction(4, 3); // Function is called, return value will end up in x

function myFunction(a, b) {
    return a * b;         // Function returns the product of a and b
}

//The result in x will be:
//12
```

Hackbright Academy

Why Functions?

Reusable code. (Write once, use many times.)

Hackbright Academy

Why Functions?

Reusable code. (Write once, use many times.)

Use the same code with different arguments to get different results.

Hackbright Academy

Why Functions?

```
function toCelsius(farenheit) {
    return (5/9) * (farenheit-32);
}
```

Hackbright Academy

Local JavaScript Variables

A variable declared (using var) within a JavaScript function becomes **LOCAL** to the function.

Local JavaScript Variables

The variable gets a **local scope**: It can only be accessed from within that function.

Local JavaScript Variables

Variables can have the same name in different functions.

Local JavaScript Variables

Parameters work as local variables in functions.

Local JavaScript Variables

Local variables are created when a function starts.

Local JavaScript Variables

Local variables are created when a function starts.

And deleted when a function ends.

Hackbright Academy

Global JavaScript Variables

Variables declared outside of a function become global variables.

Global JavaScript Variables

Variables declared outside of a function become global variables.

The variable gets a **global scope**: All scripts and functions on the web page can access it.

Hackbright Academy

Assigning Undeclared Variables

Assign values to variables that have not yet been declared, will automatically be declared as a **GLOBAL** variable.

Assigning Undeclared Variables

```
carName = "Volvo";
```

This will declare the variable carName as a global variable, even if it is executed inside a function.

Hackbright Academy

Conclusion

Functions are used for holding blocks of logic.

Great for reusing code.

Variables are local inside of functions.

Functions accept parameters (arguments).