

JS Statements

JS Syntax

JS Comments

JS Variables

JS Operators

JS Arithmetic

JS Assignment

JS Data Types

JS Functions

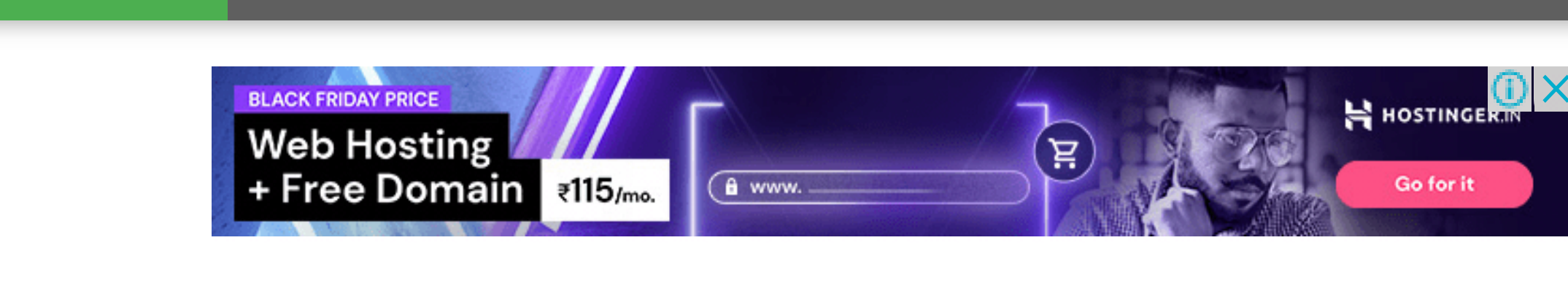
JS Objects

JS Events

JS Strings

JS String Methods

JS Numbers



JavaScript Functions

< Previous

Next >

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

Example

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

Try it Yourself >

JavaScript Function Syntax

A JavaScript function is defined with the `function` keyword, followed by a **name**, followed by parentheses `()`.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas: **(parameter1, parameter2, ...)**

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

```
function name(parameter1, parameter2, parameter3) {  
  // code to be executed  
}
```

Function **parameters** are listed inside the parentheses `()` in the function definition.

Function **arguments** are the **values** received by the function when it is invoked.

Inside the function, the arguments (the parameters) behave as local variables.

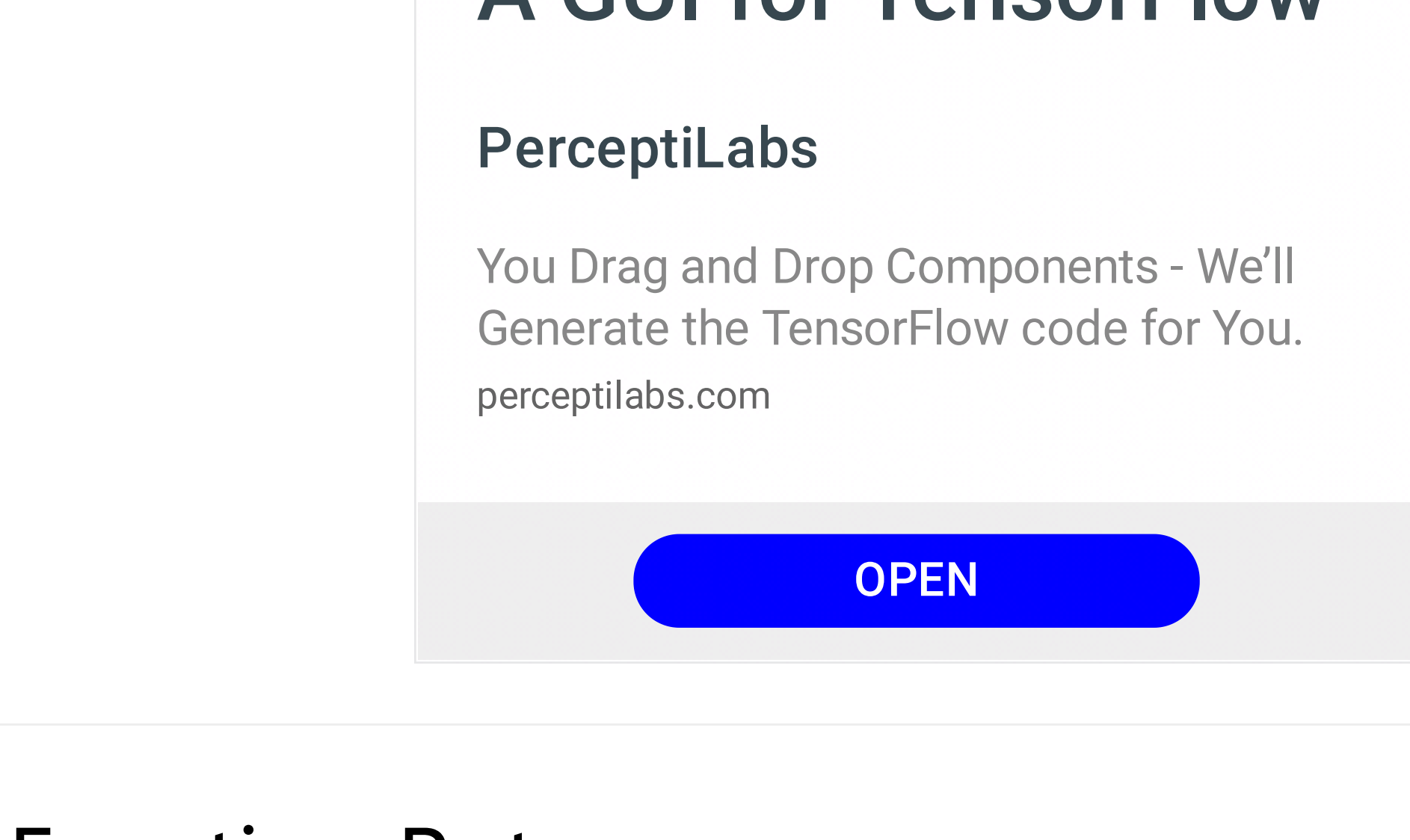
A Function is much the same as a Procedure or a Subroutine, in other programming languages.

Function Invocation

The code inside the function will execute when "something" **invokes** (calls) the function:

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

You will learn a lot more about function invocation later in this tutorial.



Function Return

When JavaScript reaches a `return` statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

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

Example

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

Try it Yourself >

Why Functions?

You can reuse code: Define the code once, and use it many times.

You can use the same code many times with different arguments, to produce different results.

Example

Convert Fahrenheit to Celsius:

```
function toCelsius(fahrenheit) {  
  return (5/9) * (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius(77);
```

Try it Yourself >

The () Operator Invokes the Function

Using the example above, `toCelsius` refers to the function object, and `toCelsius()` refers to the function result.

Accessing a function without `()` will return the function object instead of the function result.

Example

```
function toCelsius(fahrenheit) {  
  return (5/9) * (fahrenheit-32);  
}  
document.getElementById("demo").innerHTML = toCelsius;
```

Try it Yourself >

Functions Used as Variable Values

Functions can be used the same way as you use variables, in all types of formulas, assignments, and calculations.

Example

Instead of using a variable to store the return value of a function:

```
var x = toCelsius(77);  
var text = "The temperature is " + x + " Celsius";
```

You can use the function directly, as a variable value:

```
var text = "The temperature is " + toCelsius(77) + " Celsius";
```

Try it Yourself >

You will learn a lot more about functions later in this tutorial.

Local Variables

Variables declared within a JavaScript function, become **LOCAL** to the function.

Local variables can only be accessed from within the function.

Example

```
// code here can NOT use carName  
  
function myFunction() {  
  var carName = "Volvo";  
  // code here CAN use carName  
}  
  
// code here can NOT use carName
```

Try it Yourself >

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.

Test Yourself With Exercises

Exercise:

Execute the function named `myFunction`.

```
function myFunction() {  
  alert("Hello World!");  
}  
_____;
```

Submit Answer >

[Start the Exercise](#)

< Previous

Next >



REPORT ERROR

FORUM

ABOUT

SHOP

Top Tutorials

Top References

Top Examples

Web Certificates

HTML Tutorial

CSS Tutorial

JavaScript Tutorial

How To Tutorial

SQL Tutorial

Python Tutorial

W3.CSS Tutorial

Bootstrap Tutorial

PHP Tutorial

Java Tutorial

C++ Tutorial

jQuery Tutorial

HTML Reference

CSS Reference

JavaScript Reference

SQL Reference

Python Reference

W3.CSS Reference

Bootstrap Reference

PHP Reference

HTML Colors

Java Reference

Angular Reference

jQuery Reference

HTML Examples

CSS Examples

JavaScript Examples

How To Examples

SQL Examples

Python Examples

W3.CSS Examples

Bootstrap Examples

PHP Examples

Java Examples

XML Examples

jQuery Examples

HTML Certificate

CSS Certificate

JavaScript Certificate

SQL Certificate

Python Certificate

PHP Certificate

Bootstrap Certificate

XML Certificate

jQuery Certificate

Get Certified >

W3schools.com

Copyright 1999-2020 by Refnesnes Data. All Rights Reserved.

W3Schools is Powered by W3.CSS.

ADVERTISEMENT

Ads by Google

datacamp

Learn Data Science Online

Start Now

COLOR PICKER

SHARE

HOW TO

Buttons

Form

Navigation

Modal

Progress

Parallax

Login

Form

HTML

Includes

Google

Maps

Range

Sliders

Slideshow

Filter List

Sort List

Certificates

HTML

CSS

JavaScript

Python

SQL

PHP

And more

ADVERTISEMENT

Get Certified in HTML, CSS and JavaScript

Get Certified