



# FEWD - FUNCTIONS

**NICO CASTRO**

Web Developer at Red Badger

# AGENDA

- HW Review
- Functions
- Temperature Converter

# **HW REVIEW**

# FUNCTIONS

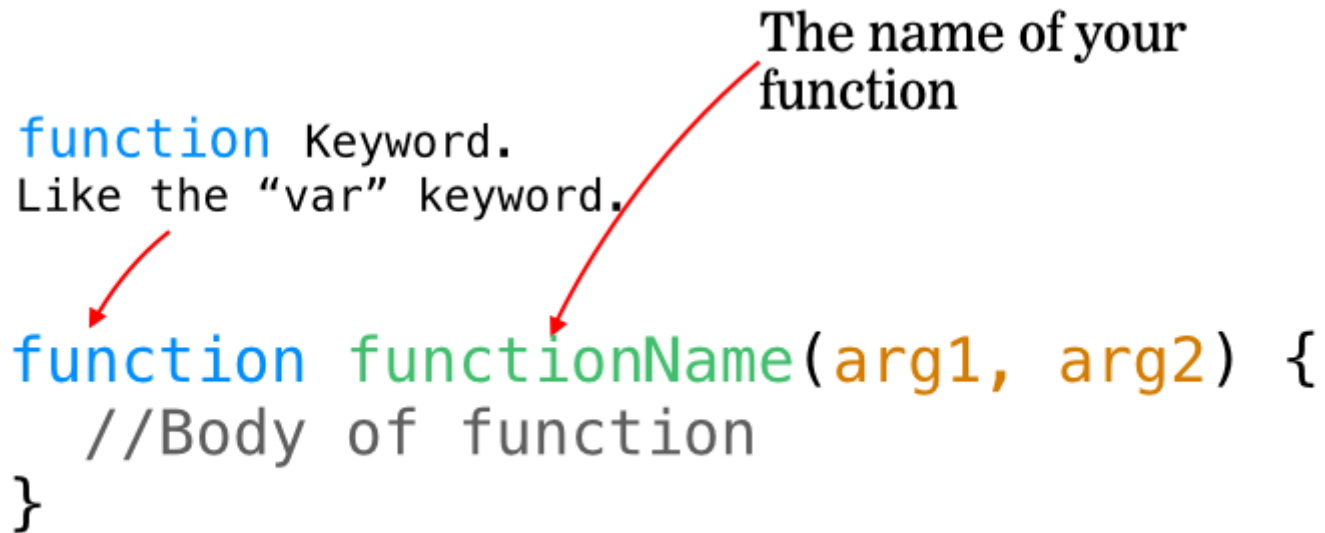
A function is a block of code (i.e.: a mini-program) assigned to a value and designed to perform a particular task. The code inside the function will only be executed when the function is *called* by external code.

# FUNCTIONS SYNTAX

`function` Keyword.  
Like the “var” keyword.

The name of your function


```
function functionName(arg1, arg2) {  
    //Body of function  
}
```

The diagram illustrates the syntax of a function declaration. It shows the keyword 'function' in blue, followed by the function name 'functionName' in green, and arguments 'arg1' and 'arg2' in orange, all enclosed in curly braces. A red arrow points from the text 'The name of your function' to the function name. Another red arrow points from the text 'function Keyword. Like the “var” keyword.' to the keyword 'function'. A third red arrow points from the text 'functionName' to the opening curly brace of the function body.

# FUNCTION CALLS

```
function helloWorld() {  
  console.log("Hello Functions");  
}
```

```
helloWorld(); //Prints "Hello Functions to the  
console.
```



The brackets execute the function.  
Try calling the function without  
them to see what happens.

# FUNCTION ARGUMENTS

Arguments let you pass data into the function

```
function functionName(arg1, arg2) {  
    //Body of function  
}
```

A diagram illustrating the components of a function definition. A red arrow points from the text 'Arguments let you pass data into the function' to the arguments 'arg1, arg2' in the function signature. Another red arrow points from the text 'The functions executed code goes between the { } brackets. Much like an “if” statement.' to the body of the function, which is the code between the curly braces.

The functions executed code goes between the { } brackets. Much like an “if” statement.

# FUNCTION ARGUMENTS

```
function addAndPrint(num1, num2) {  
    var sum = num1 + num2;  
    console.log(sum);  
}
```

```
addAndPrint(1, 2); // Result is 3
```

```
addAndPrint(8, 2); // Result is 10
```



# RETURN FUNCTIONS

Functions that explicitly `return` a value

# RETURN FUNCTIONS

```
var firstNumber = document.querySelector("#first-number").value;
var secondNumber = document.querySelector("#second-number").value;

function pythagorean(num1, num2) {
    return (num1 * num1) + (num2 * num2);
}

function updateResult(num1, num2) {
    var result = pythagorean(num1, num2);
    document.querySelector("#result").innerHTML = result;
}

document.querySelector("#addButton").onclick = updateResult(firstNumk
```

# ORGANISING FUNCTIONS

Where do you put functions?

Remember the three essential steps of working with JS:

1. Get the *data* you need and store it on a variable(s)
2. Specify *how* the data is going to be *modified* via a function
3. Hook it up: connect the function to the moment in *time* (a click **event**, a hover, upon typing into the input) in which it's meant to occur



# CASH REGISTER



## TEMPERATURE CONVERTER

- if the temperature is below or equal to  $0^{\circ}\text{C}$ , turn the background blue
- if the temperature is above  $0^{\circ}\text{C}$  and below or equal to  $18^{\circ}\text{C}$ , turn the background grey
- if the temperature is above  $18^{\circ}\text{C}$  and below or equal to  $30^{\circ}\text{C}$ , turn the background orange
- if the temperature is above  $30^{\circ}\text{C}$ , turn the background red



## TEMPERATURE CONVERTER

Other considerations:

- what happens when the "convert" button is clicked before either input has been filled out?
- what happens when the "convert" button is clicked and both inputs are filled out?

# **HOMEWORK**

Complete Temperature Converter