How To “Function()”

What it is, what it does, how to use them

Agenda

TIME	CONTENT
5 min	Intro: assumptions, definitions, and function basics.
2 min	Look at live code (codepen)
5 min	Whiteboard activity (pseudocode - how to make coffee)
2 min	Review Best Practices

Assumptions:

Today we are going to talk about functions

We'll be using JAVASCRIPT

- You have made a cup of coffee
- You are familiar with concept of 'variables'
- You may have completed *some* prep work or online-code examples (freecode academy, GA workshop etc).

Functions - What are they?

Key Ideas

A function is a type of object *that does something*

Functions are about action.

Key Steps:

INPUT: **arguments** go in

ACTIONS :
Instructions about what to do with **arguments** are interpreted.

OUTPUT :
A **value** is returned

Functions - What are they?

BIG PICTURE DEFINITION

Functions are groups of instructions the computer/browser can understand

PARTS: Made up of keywords, statements and operations.

TYPE: A Function is an **object** *that is built to do something with an input.*

GOAL: Functions allow you to change and use data in efficient and orderly steps.

Functions - What are they?

DEFINITION:
Name the function

INPUT:

arguments go in:

Variables are declared and will be used when the code is run.

ACTIONS & PROCESSES :

a block of instructions inside { } tells the computer to act on your arguments.

Operations
Logic Statements
Loops

The code is interpreted line by line and data is read, evaluated, and changed.

OUTPUT:
a new value is "returned"

Note: The declared function is can only run when you "call" the function.

Declaring a function: Function Definition

```
function theNameOfYourFunction ( string1, string2, …) {

    const aNewString = string1 + string2;

    // Step 1. assigning the arguments to a new variable
    // Step n.  the process of the code may continue...

    return aNewString;

    // Step 2. a new value is returned that “concatenates” the two
    argument strings into a new string whenever the function is called.

}
```

__ = keyword
a special word designated by JS to “construct” a function.

__ = variable / name
a string of characters that substitute for a set of values

__ = arguments
placeholders for values inside the function

__ = operators
how the function acts on the parameters values.

Declaring a function: “Function Expression”

```
const theNameOfYourFunction = function ( string1, string2, …) {
        const aNewString = string1 + string2;
        return aNewString
}
```

A function *expression* is a way to *store* the values returned by a function in a variable. This variable can then be ‘called’ by another function.

Calling a function

```
function theNameOfYourFunction (arg1, arg2){
        return console.log(arg1+” “+arg2);
}

const aNewWord = theNameOfYourFunction ( );
```

OUTPUT >> “undefined undefined”

// the functions value (undefined) is stored in a variable newWord

```
aNewWord("a duck,"  , "dill pickle");
```

OUTPUT >> “a duck, dill pickle”

// we are calling, or invoking by the function, by using the variable name and passing in values as “parameters”

When declaring a function:

1. **Choose a good name**: functions do something so try to use words that describe the process of the function. (get, set, find, add, decrease, etc)
2. Arguments should also be topical and concrete.
3. Try to keep your functions as simple as possible. **A good function does one thing and returns a predictable result.**

SAMPLE CODE USED IN PREPARATION: codepenIO

When calling a function:

More advanced things to remember:

1. **Code runs synchronously** - code runs line by line, one statement at a time. A later function cannot execute until the previous function has completed its process.
2. **Function definitions** may be defined anywhere in the code, even after the code is called (!)
3. **Function Expressions** should only be declared before the code is called. Otherwise an undefined variable will be called instead.

SAMPLE CODE USED IN EXERCISE : codepenIO