**Nathalie Cristina Soto Rojas**

**ACADUCR Módulo 2: JavaScript**

**Tarea #2**

**(Este texto incluye material escrito en inglés y español, dado que utilicé también notas personales y resúmenes de clases que ya tenía guardados anteriormente).**

JavaScript es un lenguaje de programación utilizado para crear páginas web interactivas. Es uno de los lenguajes de programación más populares en el mundo y es empleado por millones de desarrolladores para crear sitios web, aplicaciones web y aplicaciones móviles.

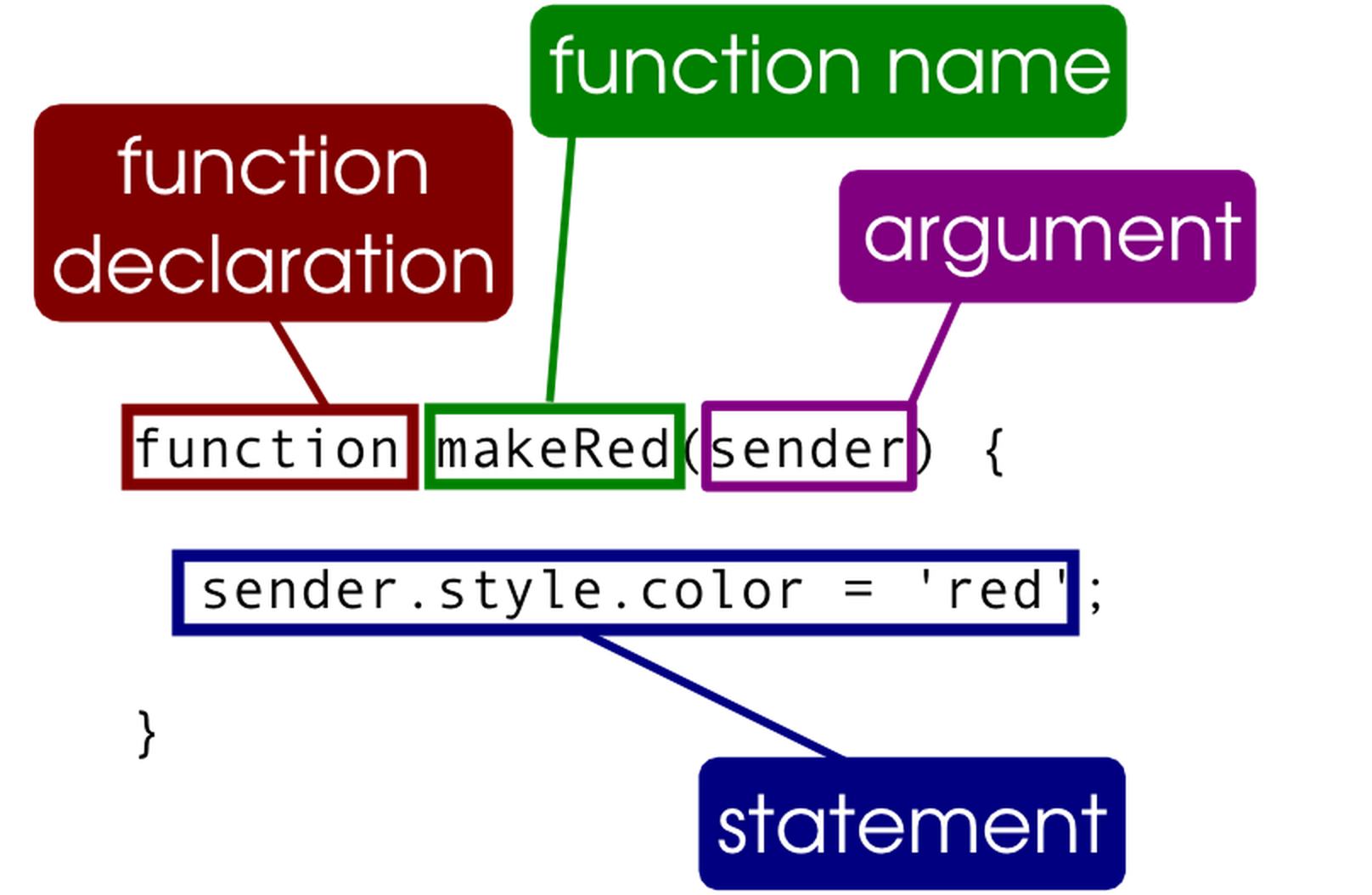
Este resumen ofrece una visión general de JavaScript, abarcando su historia, sintaxis y características. También incluye algunos de los temas importantes, como variables, funciones, arreglos y objetos.

**Aprendizaje de JavaScript (Mozilla Developer Network):**

En el extenso recurso proporcionado por Mozilla Developer Network (MDN), se ofrece una guía integral para aprender JavaScript desde sus conceptos más fundamentales hasta su aplicación práctica en el desarrollo web.

Iniciando por los elementos básicos, abarcando temas como variables, tipos de datos y operadores.

## **Syntax**

****

// Single-line comments in JavaScript.

/\* Multiple line comments in JavaScript \*/

### **Testing the HTML link**

1. Create an index.html document and write a basic boilerplate.
2. Inside of the <body> element, include <script src=”./script.js”></script>.
3. Create the script.js documento and write console.log(”Hello world!”);, then go to the Console. If the message appears there, the document was linked correctly.

**Using the Console in the Developers Tool**

1. Open Google Chrome.
2. Double click on the webpage and select “Inspect”.
3. Click the Console tab or press ESC to toggle on and off from the console.
4. Type any JavaScript command into the DevTools console.

console.log(”Hello world!”);

1. Tap Enter to run it.

# **Data Types**

JavaScript has 8 Data Types:

* + String
* JavaScript text data, including spaces, symbols, numbers, etc, enclosed in single or doubled quotes.  
    
   console.log(”string");  
  + Number
* JavaScript numerical data, including decimals.  
    
   console.log(1 + 1);  
  + BigInt
* A data type that accommodates a greater range of numbers than a Number data type.  
  + Boolean
* Has only two values: “True” and “False”.  
  + Undefined
* Only has the value “Undefined” and it means that a given value does not exist.  
  + Null
* Only has the value “No” and represents the intentional absence of a value.  
  + Symbol
* Unique identifiers, useful in more complex coding.  
  + Object
* Collections of related data.
* *Concatenation* Appending one or more strings to the end of another string.  
    
   console.log(1 + "1");

There are four methods or ways by which we can concatenate strings in JavaScript:

* using the concat() method
* using the '+' operator
* using the array join() method
* using template literals

### **The Object Data Type**

The object data type can contain:

1. An object
2. An array
3. A date

# **Variables**

ℹ️ Containers with aliases to store data that allow us to make the exact same code produce different results, making our programs versatile by re-using code. It’s keyword is var.

let myName = “Nathalie”;

console.log(myName);

JS Input: Nathalie

✏️ The = symbol is called the assignment operator in JavaScript, and it works to declare the value of a variable.

### **4 Ways to Declare a JS Variable:**

1. Using var
2. Using let
3. Using const
4. Using nothing

# **Operators**

An [operator](https://www.codecademy.com/resources/docs/javascript/operators?page_ref=catalog) is a character that performs a task in our code.

CONDITIONAL FLOW/IF ELSE STATEMENTS

For the computer to think by itself

### **Logical Operators**

AND, OR, IF, ELSE IF, ELSE

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Meaning** | **Example** |
| && | And | Checks for both conditions to be true | a>5 && a<10 |
|  |  |  | Or |
| ! | Not | Returns false if the result is true | !(a>5) |

### **Arithmetic Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Meaning** | **Example** |
| + |  | Addition | 2 + 3 |
| - |  | Substraction | 3 - 2 |
| / |  | Division | 35 / 5 |
| \* |  | Multiplication | 7 \* 4 |
| % |  | Reminder |  |

### **Comparison Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Meaning** | **Example** |
| > |  | Greater than | 3 > 2 |
| < |  | Less than | 2 < 3 |
| == | Equality operator | Equal to | 5 == 5 |
| != | Inequality operator | Not equal to | 5 != 6 |

### **Additional Operators**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Meaning** | **Example** |
| % | Modulus operator |  |  |
| === | Strict equality operator |  |  |
| !== | Strict inequality operator |  |  |
| += | Addition assignment |  |  |
| += | Concatenation assignment |  |  |

# **Functions**

A way for us to execute code at a specific moment.

function functionName(parameter){

\*code to execute\*

}

Modulo operator provides the remainder when the first number is divded by the second number.

function oddOrEven(num){

console.log(num % 2);

}

## **Arrays**

A way to group together related data, made of “elements”, referring to each individual piece of information.

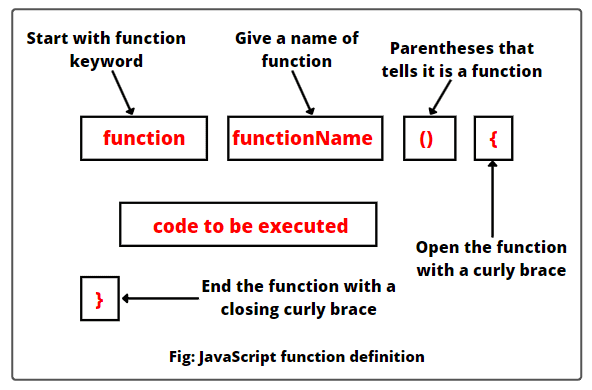
let foods = ["Apple", "Chicken", "Chocolate"];

let numbers = [123, 321, 112, 221];

In order to reference a specific element inside of an array, we can use that element's INDEX. Every array has elements that have a specific index number starting from 0 and incrementing by 1 as more elements added.

console.log(foods[2]);

JS output: Chocolate



A medida que se avanza, se exploran estructuras de control de flujo, funciones y objetos, proporcionando una base sólida para el desarrollo de scripts interactivos y dinámicos.

El enfoque didáctico del recurso MDN se destaca por su énfasis en la práctica. A través de ejemplos prácticos y proyectos interactivos, los estudiantes pueden aplicar directamente los conceptos aprendidos. Esto contribuye a una comprensión más profunda y a la capacidad de utilizar JavaScript de manera efectiva en situaciones del mundo real.

**Introducción a JSON en JavaScript (w3schools):**

JavaScript Object Notation (JSON) es un formato versátil para intercambiar datos entre aplicaciones. Profundizando en la manipulación de datos JSON en el contexto de JavaScript. JSON, siendo un formato ligero y legible, se destaca como una opción popular para estructurar datos.\*

Según la sintaxis básica de JSON, ilustrando cómo representar objetos y arreglos, se pueden establecer las siguientes reglas:

* La información se presenta en nombre/valores
* La información se separa en comas
* Se utilizan los paréntesis curvos {} como contenedores para encerrar objetos y separarlos
* Se utilizan los paréntesis cuadrados [] para contener arrays.

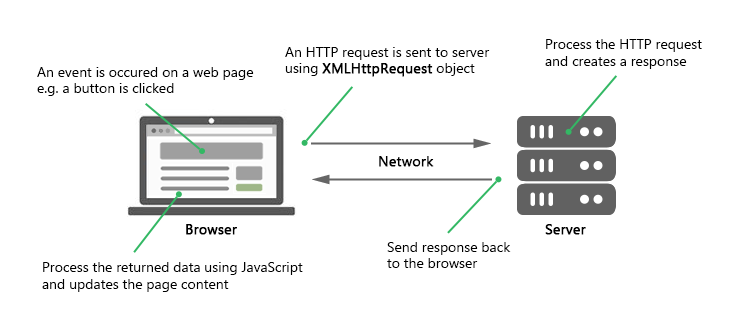
A través de ejemplos prácticos, se demuestra cómo crear y manipular objetos JSON en JavaScript. Los objetos JSON, al ser una colección de pares clave-valor, permiten organizar y transmitir datos de manera eficiente.

Además, se aborda el proceso de enviar y recibir datos JSON entre el cliente y el servidor. El uso de XMLHttpRequest para realizar solicitudes asíncronas al servidor es un enfoque destacado. Este método permite actualizar partes específicas de una página web sin la necesidad de recargar por completo, mejorando la experiencia del usuario y la eficiencia del intercambio de datos.

**Introducción a AJAX en JavaScript (w3schools):**

Asynchronous JavaScript and XML (AJAX) es una tecnología fundamental para la creación de aplicaciones web dinámicas y eficientes. A diferencia de los métodos de solicitud tradicionales, AJAX permite realizar solicitudes asíncronas al servidor sin recargar toda la página.

Se puede utilizar el objeto XMLHttpRequest para implementar solicitudes AJAX. Este objeto facilita la comunicación entre el cliente y el servidor, posibilitando la actualización de datos de manera selectiva. La capacidad de cargar datos en segundo plano sin interrumpir la experiencia del usuario es esencial para la construcción de aplicaciones web modernas y receptivas.



Además, se abordan estrategias para manejar las respuestas del servidor, lo que incluye la manipulación de datos JSON para una integración efectiva. La sección concluye destacando la importancia de AJAX en la mejora de la velocidad y la eficiencia de las aplicaciones web, proporcionando una experiencia de usuario más fluida.