Presented by: AzurePython

JavaScript Fundamentals: Why Study JavaScript?

JavaScript is one of the three essential languages that all web developers should learn:

HTML: This language deals with the structure of a web page.

CSS: It's used for styling and controlling the visual presentation of a webpage.

JavaScript: This language is focused on scripting and adding interactivity and behavior to a webpage.

An Example of JavaScript

You can start by understanding the simplicity of JavaScript with an example like this:

const js = "free for everybody";

This line of code assigns the value "free for everybody" to the variable js.

NUMBERS AND MATH

Arithmetic Operations

Math in JavaScript involves various arithmetic operations, each performed with specific operators. Here are the essential arithmetic operators along with their descriptions:

+ Addition

- Subtraction

* Multiplication

/ Division

% Modulus (Remainder)

** Exponentiation

To perform these operations, you adhere to the following syntax:

operand1 operator operand2

For instance, you can carry out calculations like:

2 + 3

5 - 2

10 * 6

20 / 4

10 % 3

5 ** 2

Order of Operations

To ensure precise calculations, JavaScript follows a specific order of operations:

Parentheses (...)

Exponentiation **

Multiplication and Division * /

Addition and Subtraction + -

Increment and Decrement

In JavaScript, you can conveniently increase or decrease the value of a variable by 1. This is achieved using the increment (++) and decrement (--) operators, respectively. For example:

Increment: Increases the variable's value by 1 using ++.

Decrement: Decreases the variable's value by 1 using --.

Weird Behavior of Math in JavaScript

While JavaScript is a versatile language for numerical computations, it exhibits some peculiar behaviors. Notably, calculations involving floating-point numbers can sometimes yield inaccurate results. This is because computers store numbers in binary, making certain decimal numbers challenging to represent exactly.

Rounding numbers is a common practice in JavaScript to manage precision. The Math.round() function can help round off numbers to the nearest integer. For instance:

Math.round(2.65) returns 3.

VARIABLES

Variables serve as containers for storing data.

JavaScript variables can be declared using four different approaches, including:

a.

b. Using var

c. Using let

d. Using const

It is considered a best practice to declare variables before using them.

The var keyword was used for all JavaScript code from 1995 to 2015.

The let and const keywords were introduced to JavaScript in 2015.

The var keyword is now primarily reserved for code targeting older browsers.

IDENTIFIERS

In JavaScript, all variables must have unique names, which are referred to as identifiers.

Identifiers can be brief (e.g., x and y) or more descriptive (e.g., age, sum, totalVolume).

There are general rules for constructing variable names (identifiers):

Names can include letters, digits, underscores, and dollar signs.

Names must start with a letter.

Names are case-sensitive (e.g., y and Y are distinct variables).

Reserved words, such as JavaScript keywords, cannot be used as identifiers.

DATA TYPES

JavaScript variables can hold both numbers (e.g., 100) and text values (e.g., "Mary Ann").

In programming, text values are typically referred to as text strings.

JavaScript can handle various data types, but for simplicity, consider numbers and strings.

Strings are enclosed within double or single quotes, while numbers are written without quotes.

ARITHMETIC

JavaScript allows you to perform arithmetic operations with variables, using operators like =, +, and others.

STRINGS

Strings in JavaScript have a length property to determine their length.

To include quotes within a string, use the backslash escape character to avoid issues.

STRING METHODS

JavaScript provides various string methods to manipulate and work with text. These include functions like trim(), slice(), substring(), toUpperCase(), and more. These methods offer powerful tools for string manipulation and processing.

By organizing the content in this manner, you maintain a logical flow, making it easier for readers to follow the concepts related to variables, identifiers, data types, arithmetic, and strings in JavaScript.

BOOLEANS

A Boolean represents a logical value, either 'true' or 'false'.

CREATING BOOLEAN VALUES

You can create Boolean values using literals 'true' and 'false'.

VALUES AND TRUTHINESS/FALSINESS

Values like 0, an empty string (" "), null, and undefined are considered falsy in JavaScript. Everything else is considered truthy.

COMPARISON OPERATORS

Comparison operators (e.g., ==) compare two values and return a Boolean result. They are used to assess equality and relative values.

BOOLEAN USING LOGICAL OPERATORS

Logical Operators (AND, OR, NOT) combine and modify Boolean values.

IF STATEMENT

The if statement allows you to execute a block of code if a given condition is true.

The if...else statement extends the basic if by providing an alternative block of code to execute when the condition is false.

ELSE IF STATEMENT

The else if statement allows for multiple possible outcomes. In an else if statement, the first condition that evaluates to true from top to bottom is the block that gets executed.

SWITCH

The switch statement is used for multi-way branching based on the value of an expression. It's an alternative to multiple if...else if... statements.

TERNARY OPERATOR

The ternary operator, also known as the conditional operator, provides a concise way to write conditional statements.

FUNCTIONS

Functions are reusable blocks of code that perform specific tasks, keeping your code organized and facilitating reuse.

PARAMETERS AND ARGUMENTS

Parameters are placeholders in a function definition, while arguments are the actual values you pass when calling the function.

DEFAULT PARAMETERS

You can set default values for function parameters, making them optional when calling the function.

RETURN

Functions can return values using the return statement, which can be used elsewhere in your code.

FUNCTION SCOPE

Function scope refers to the visibility and accessibility of variables defined within a function.

FUNCTION EXPRESSION

A function expression is a way of defining a function that involves creating it as part of an expression. You can have an anonymous function (one without a name) or provide a name (a named function expression).

WHAT ARE EVENTS?

Events in JavaScript are occurrences that happen to a webpage element.

TYPES OF EVENTS:

Common HTML events include a variety of actions that can take place in a web page.

COMMON HTML EVENTS:

Some of the common HTML events and their descriptions include:

HANDLING EVENTS IN JAVASCRIPT

Events can be handled in JavaScript using various methods.

INLINE EVENT HANDLER

One way is by adding an attribute directly to the HTML element you want to handle events for.

You can assign any JavaScript expression to the attribute, and it will be executed when the event occurs.

For example:

bash

Copy code

```
<button onclick="this.innerHTML = Date()">Click this to show the date</button>
```

EVENT HANDLER PROPERTIES

Another approach is to add a property to a DOM element for which you want to handle an event.

You can assign a JavaScript expression to this property, which will execute when the event takes place.

USING addEventListener() METHOD

A more flexible and common method for handling events in JavaScript is by using the addEventListener() method.

You can listen for various events using this method. Some commonly used events and their descriptions are:

Event: Occurs when: Belongs to:

USING addEventListener() METHOD

JavaScript Objects

In JavaScript, objects are used to model real-life objects, storing properties and methods.

Properties of an object include attributes, such as phone.brand, phone.displaySize, phone.displayType, and phone.color.

Methods of an object are functions that define behaviors, like phone.on() and phone.off().

JavaScript Variables VS Objects

JavaScript variables serve as containers for data values.

JavaScript objects can contain multiple values and serve as a way to store and organize related data.

Objects can be declared using the const keyword, making it a common practice.

Object Properties

Object properties are pairs of property and value. For instance:

firstName: Kevin

lastName: Durant

age: 35

eyeColor: black

You can access object properties using two methods: dot notation (e.g., objectName.propertyName) or bracket notation (e.g., objectName["property-name"]).

Manipulating Object Properties

You can add object properties to an object, even if they don't exist.

You can also delete object properties using the delete keyword.

Object Methods

Object methods are functions associated with and defined within objects, enabling specific operations or manipulations on the object's properties (data).

Methods are written as key-value pairs within objects, where the key is the method's name, and the value is a function definition.

Example:

```
propertyName: function() {
   return ("action you want to do with properties")
}
```

Accessing object methods is done by using the object name followed by the method name and parentheses:

```
objectName.methodName()
```

It's important to include the parentheses when calling a method. Omitting them will return the function definition.

JavaScript HTML DOM

When a web page loads, the browser constructs a Document Object Model (DOM) of the page.

The DOM is another built-in object in JavaScript, similar to console, Math, JSON, and localStorage. You access it using the document object.

The Power of the DOM

With the DOM, JavaScript gains the capability to create dynamic HTML, transforming the page's elements and attributes.

JavaScript can modify HTML elements, attributes, and CSS styles.

JavaScript can remove existing HTML elements and attributes.

JavaScript can add new HTML elements and attributes.

JavaScript can respond to all existing HTML events on the page.

JavaScript can even create entirely new HTML events within the page.

JavaScript - HTML DOM Methods

The HTML DOM can be accessed using JavaScript and other programming languages. In the DOM, all HTML elements are treated as objects.

The programming interface of the DOM consists of properties and methods for each object.

Example:

```
<html>
  <body>
```

```
    <p id="demo"></p>

    <script>

      document.getElementById("demo").innerHTML = "Hello World!";

    </script>

  </body>

</html>
```

Methods in the DOM allow you to perform actions, such as accessing or finding an HTML element.

Properties in the DOM represent values that you can get or set, like changing the content of an HTML element.

Arrays & Loops

Array Methods

Array methods are important for manipulating arrays. One such method is toString().

The toString() method converts an array to a string of comma-separated array values. For example:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];

fruits.toString();
```

The output will be "Banana,Orange,Apple,Mango."

Note that you can specify a custom separator by using the join() method. Everything is the same as toString(), but you can define your separator. For instance:

```
fruits.join(" * ");
```

Syntax of a For Loop

A for loop is a fundamental construct in JavaScript, comprising three essential components:

Counter initialization: This section is executed once before the loop starts. It's responsible for setting the loop variable (e.g., let i = 0).

Condition: The condition defines when the loop should continue running (e.g., i < 5).

Increment expression: This section increases the value of the loop variable (e.g., i++) after each iteration of the loop.

Here's an example of a for loop:

```
for (let i = 0; i < 5; i++) {

    text += "The number is " + i + "<br>";
```

}

You can initialize multiple values separated by commas.

The Do While Loop

The do-while loop is a variation of the while loop. It executes the code block at least once, even before checking if the condition is true, and continues as long as the condition remains true.

Here's the syntax:

```
do {
   // code block to be executed
} while (condition);
```

For example:

```
do {
   text += "The number is " + i;
   i++;
} while (i < 10);
```

REFERENCES:

JavaScript Full Course (2023) - Beginner to Pro - Part 1 - YouTube

javascript_tutorial.pdf (tutorialspoint.com)

JavaScript Tutorial (w3schools.com)

Learn JavaScript | Codecademy

Events and Event handlers - YouTube

That's the wrap for our JavaScript Fundamentals report.

Happy Coding 😊