CONTROL FLOW IN JAVASCRIPT

Fatima Arshad

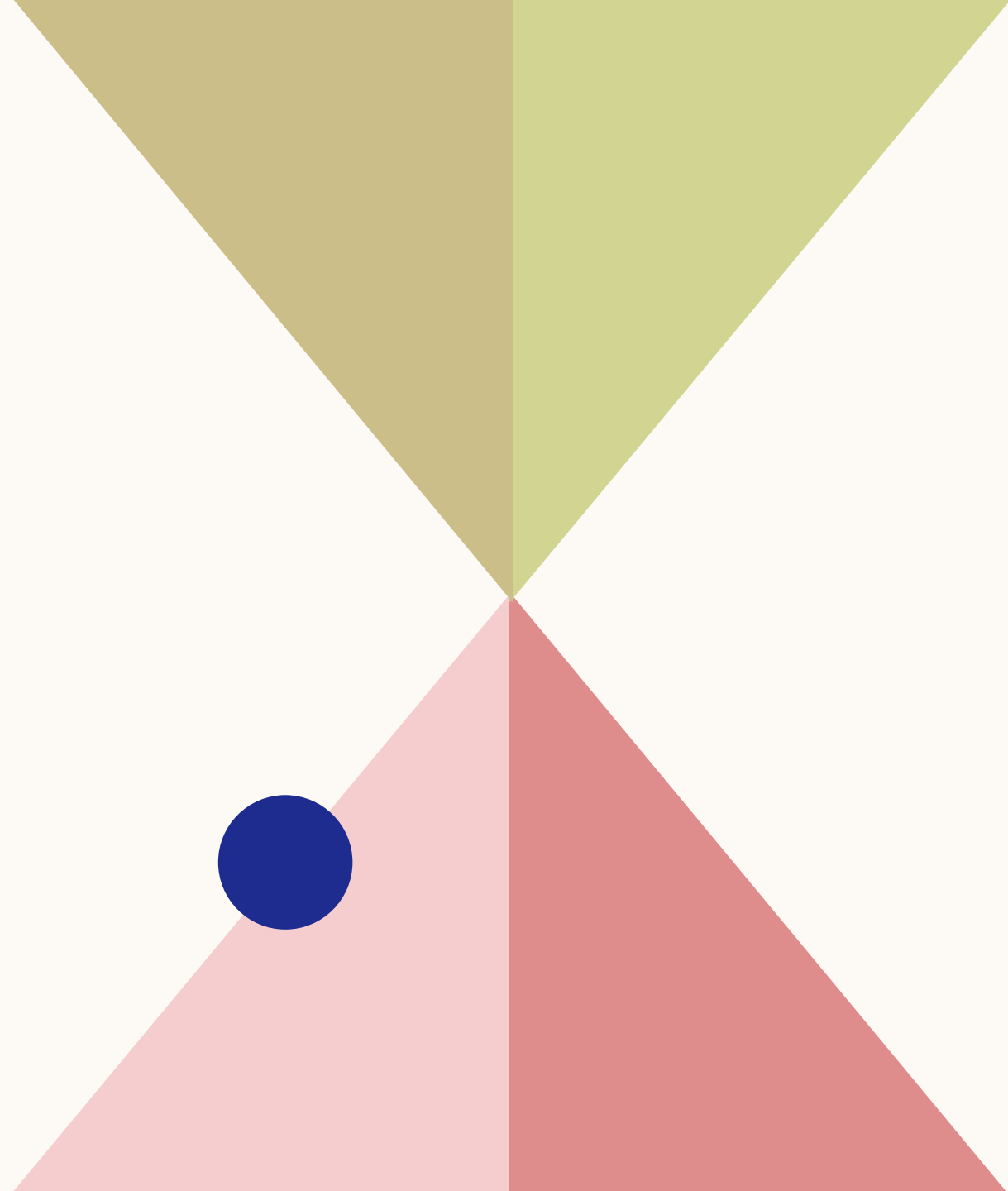
AGENDA

Variables

Operators

Conditions

Loops



UNDERSTANDING DATA TYPES IN JAVASCRIPT:

Definition :

1. Data types define the kind of data that can be stored and manipulated in JavaScript.
2. JavaScript has dynamic typing, meaning variables can hold values of different types.

JavaScript has data types:

1. Primitive
2. Complex

PRIMITIVE DATA TYPES

NUMBER

- Represents numeric values.
- Example: 5, 3.14, -10

STRING

- Represents textual data enclosed in single or double quotes.
- Example: "Hello, World!", 'JavaScript'

BOOLEAN

- Represents either true or false.
- Example: true, false

UNDEFINED

- Represents a variable that has been declared but has no value assigned to it.
- Example: let x;

NULL

- Represents the intentional absence of any object value.
- Example: let y = null;

COMPLEX DATA TYPES

OBJECT

- Represents a collection of key-value pairs.
- Example: { name: "John", age: 25 }

ARRAY

- Represents an ordered list of values.
- Example: [1, 2, 3, 4]

JAVASCRIPT VARIABLES

Definition :

1. Variables are used to store and manipulate data in JavaScript.
2. They provide a way to label and reference values within your code.

Variable Declaration:

Variables in JavaScript are declared using the

(var , let , const) => **keywords**

Example :

```
1   var x ;
2   let name ;
3   const PI = 3.14
```

Variable Initialization:

You can declare and assign a value to a variable in a single step, known as initialization.

Example :

```
5   var y = 5;
6   let age = 25;
7   const URL = "https://example.com"
```

Variable Naming Rules:

Variable names in JavaScript must follow certain rules:

- They can contain letters, digits, underscores, or dollar signs.
- They must start with a letter, underscore, or dollar sign (not a digit).
- They are case-sensitive.

Example of valid variable names :

‘myVariable’

‘_count’

Example of Invalid variable names :

‘2test’

‘my-variable’

‘@name’

Variable Scope

The scope of a variable refers to its visibility and accessibility within the code.

Global Scope:

1. Variables declared outside of any function or block have global scope.
2. Global variables can be accessed from anywhere in the code.

Example:

```
// global Scope
var global = 'global scope'
function myFunction (){
    console.log(global)
}
console.log(global)
```

Local Scope :

1. Variables declared inside a function or block have local scope.
2. Local variables are only accessible within the function or block they are declared in.

Example:

```
16  // local Scope
17  function myFunction (){
18      let local = 'local scope'
19      console.log(local)
20  }
```

Block Scope (let, const):

- Variable declared with (let , const) keyword have block Scope.
- Block scoped variables are limited to the block they are declared in , such as condition or loops.

Reserved Word :

Reserved words that have special meanings and cannot be used as variables, functions, methods, loop labels, or any object names.

These reserved words are part of the JavaScript language syntax and serve specific purposes.

Example:

abstract	else	Instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

OPERATORS IN JAVASCRIPT

Operators are symbols used to perform operations on values in JavaScript.

They enable

- Arithmetic calculations,
- Comparisons,
- Logical operations, and more.

ARITHMETIC OPERATORS

Operator	Description
Addition: +	Adds two operands Ex: A + B will give 30
Subtraction: -	Subtracts the second operand from the first Ex: A - B will give -10
Multiplication: *	Multiply both operands Ex: A * B will give 200
Division: /	Divide the numerator by the denominator Ex: B / A will give 2
Remainder (Modulus): %	Outputs the remainder of an integer division Ex: B % A will give 0
Increment: ++	Increases an integer value by one Ex: A++ will give 11
Decrement: --	Decreases an integer value by one Ex: A-- will give 9

COMPARISON OPERATORS:

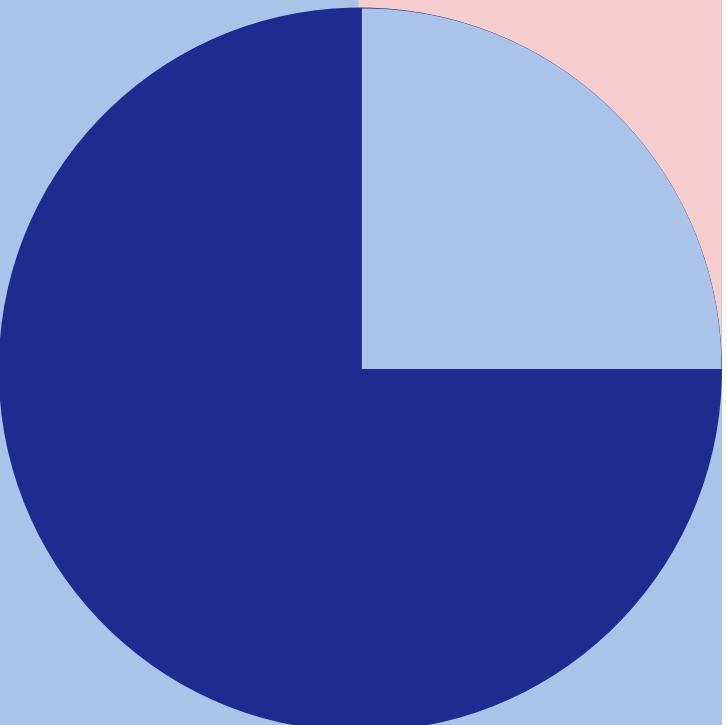
COMPARISON OPERATORS COMPARE VALUES AND RETURN A BOOLEAN RESULT.

Operator	Description
Equal to: ==	Checks if the value of two operands are equal or not, if yes, then the condition becomes true. Ex: (A == B) is not true.
Not equal to: !=	Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. Ex: (A != B) is true.
Greater than: >	Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. Ex: (A > B) is not true.
Less than: <	Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. Ex: (A < B) is true.
Greater than or equal to: >=	Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A >= B) is not true.
Less than or equal to: <=	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A <= B) is true.

LOGICAL OPERATORS:

PERFORM LOGICAL OPERATIONS AND RETURN BOOLEAN RESULTS.

Operator	Description
AND: &&	If both the operands are non-zero, then the condition becomes true. Ex: (A && B) is true.
OR: ||	If any of the two operands are non-zero, then the condition becomes true. Ex: (A || B) is true
NOT: !	Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. Ex: ! (A && B) is false



CONDITIONAL STATEMENTS: IF-ELSE

- Conditional statements allow the execution of different code blocks based on certain conditions.
- The "if-else" statement is one of the most commonly used conditional statements in JavaScript.

```
// Condition if else
let age = 18;

if (age >= 18) {
  console.log("You are an adult.");
} else {
  console.log("You are a minor.");
}
```

if...else if... Statement

- The 'if...else if..' statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions

LOOPS: WHILE AND FOR

Loops are used to repeatedly execute a block of code until a certain condition is met.
JavaScript provides two commonly used loop structures: "while" and "for" loops.

While loop :

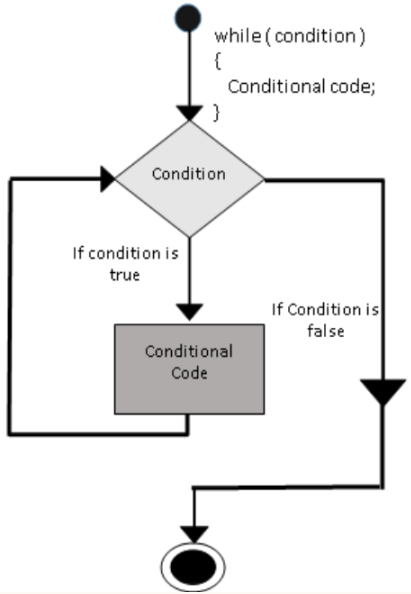
The purpose of a while loop is to execute a statement or code block repeatedly as long as an expression is true. Once the expression becomes false, the loop terminates.

```
// while loop
let count = 0;

while (count < 5) {
  console.log("Count: " + count);
  count++;
}
```

Flow Chart

The flow chart of **while loop** looks as follows:



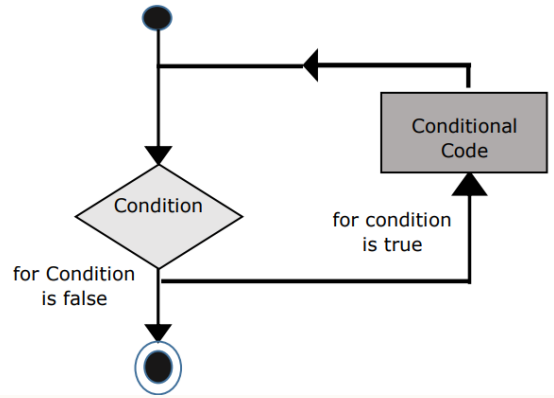
LOOPS: FOR

The 'for' loop is the most compact form of looping. It includes the following three important parts:

1. The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
2. The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
3. The **iteration statement** where you can increase or decrease your counter.

Flow Chart

The flow chart of a **for** loop in JavaScript would be as follows:



LOOPS: FOR


```
// for loop
for (let i = 0; i < 5; i++) {
    console.log("Value of i: " + i);
}
```

CODE PROBLEM QUESTIONS

1. Write a JavaScript program to declare a variable name and assign it a string value. Print the value of the name variable to the console.
2. Write a JavaScript program to declare two variables num1 and num2 and assign them numerical values. Calculate the sum of num1 and num2 and store the result in a variable called sum. Print the value of sum to the console.
3. Write a JavaScript program to declare a constant variable PI and assign it the value of 3.14159. Declare another variable radius and assign it a numerical value. Calculate the area of a circle using the formula area = PI * radius * radius and store the result in a variable called circle Area. Print the value of circle Area to the console.
4. Write a JavaScript program that takes a number as input and checks if it is positive, negative, or zero. Print an appropriate message to the console based on the input.
5. Write a JavaScript program using a while loop to print the numbers from 1 to 10 to the console.
6. Write a JavaScript program using a for loop to print the numbers from 1 to 10 to the console.

THANK YOU