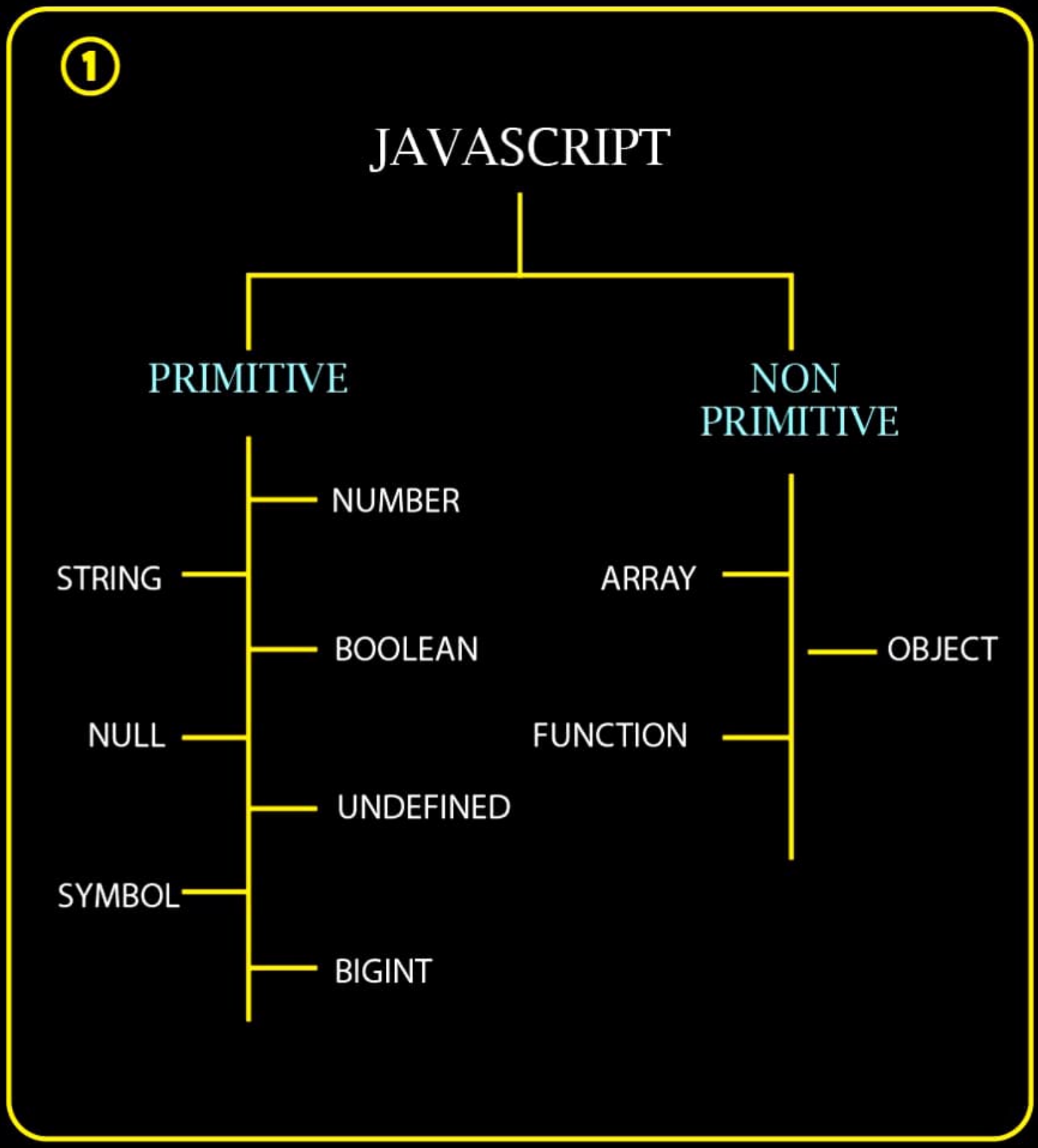
TOPIC
1

JAVASCRIPT

DATATYPES


1
JAVASCRIPT
PRIMITIVE
NON PRIMITIVE
NUMBER
STRING
BOOLEAN
NULL
UNDEFINED
SYMBOL
BIGINT
ARRAY
OBJECT
FUNCTION

2

PRIMITIVE DATA TYPE

- In javascript, primitive data types are the basic building blocks of data.
- These are immutable, meaning their values cannot be changed once set.

NUMBER

- Represents numeric data, including integers and floating-point numbers.
- Return type : number
- ```
  let age = 25      // integer

  let price = 99.99   // floating-point number
  ```
- ```
  console.log (typeof age)   // output : number
  ```

3

STRING

- Represents textual data enclosed within single or double quotes.
- Return type : string
- let firstName = 'john'
  let greet = "hello"
- console.log (typeof firstName) // o/p : string

BOOLEAN

- Represents a logical entity with two values: true or false.
- Return type : boolean
- let isTrue = true
  let isFalse = false
- console.log (typeof isTrue) // o/p : boolean

4

UNDEFINED

- Represents a variable that has been declared but not assigned a value.
- Return type : undefined
- let user // only declared not assigned
- console.log (typeof user) // o/p : undefined

NULL

- Represents the intentional absence of an object value.
- Return type : object
- let data = null
- console.log (typeof data) // o/p : object

null is considered an object type due to legacy reasons.

5

SYMBOL

- Represents a unique identifier.
- Symbols are often used to add properties to objects without the risk of name collisions.
- Return type : symbol
- let newId = Symbol('id')
- console.log (typeof newId) // o/p : symbol

BIGINT

- Represents whole numbers larger than the Number type can represent.
- Return type : bigint
- let num = 1234567890123456789012345n
- console.log (typeof num) // o/p : bigint

6

NON-PRIMITIVE DATA TYPE

- These are mutable, meaning they can be changed after creation.

OBJECT

- Represents a collection of key-value pairs.
- Objects can be created using {} or new Object().
- .
- Return type : object
- ```
  let user = {
    name: 'Alice',
    age: 30
  }
  ```
- ```
  console.log (user.name)   // o/p : Alice
  ```
- ```
  console.log (user.age)    // o/p : 30
  ```
  ```

7

ARRAY

- Represents a list-like collection of elements, where each element can be accessed via an index.
- Return type : object
- let number = [1,2,3,4,5]
- console.log (number[2]) // o/p : 3

FUNCTION

- Represents a reusable block of code that can be invoked by name.
- Return type : object function
- ```
  let  myFunction = function() {
     console.log ("hello world")
  }
  ```
- console.log (myFunction) // o/p : hello world
  ```