Js For Beginners

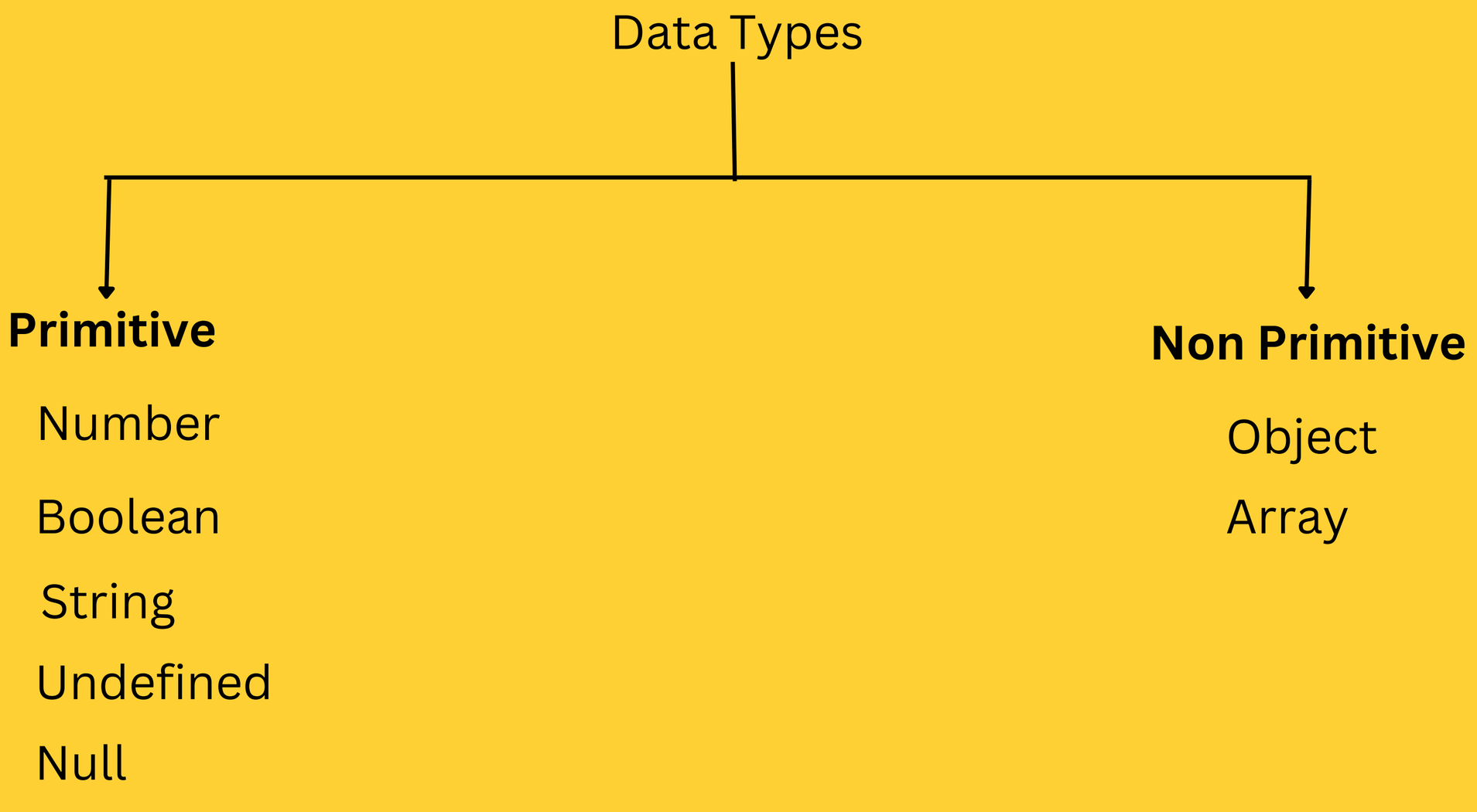
By Deepa Chaurasia

Agenda

Topics Covered

- JavaScript syntax and data types
- Variables and control structures
- Functions and scope
- Objects, arrays, and JSON
- DOM manipulation and event handling

Data Types


Data Types
Primitive
Number
Boolean
String
Undefined
Null
Non Primitive
Object
Array

JavaScript and Camel Case

- Hyphens are not allowed in JavaScript. They are reserved for subtractions.
- JavaScript programmers tend to use camel case that starts with a lowercase letter:
- Upper Camel Case -FirstName, LastName, MasterCard, InterCity.
- Lower Camel Case-firstName, lastName, masterCard, interCity.
- A JavaScript name must begin with:

  A letter (A-Z or a-z)

  A dollar sign ($)

  Or an underscore (_)

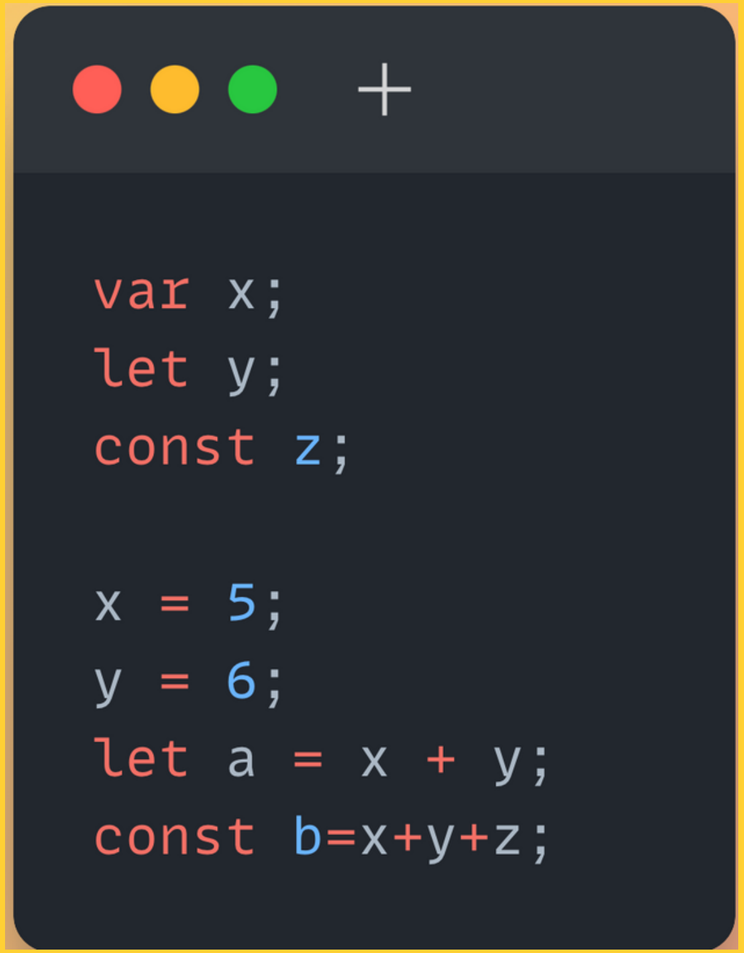
JavaScript is Case Sensitive

All JavaScript identifiers are case sensitive.
The variables lastName and lastname, are two different variables:

```
let lastname, lastName;
lastName = "Deepa";
lastname = "Chaurasia";
```

JavaScript does not interpret LET or Let as
the keyword let.

How to create a variable?



```
var x;
let y;
const z;

x = 5;
y = 6;
let a = x + y;
const b=x+y+z;
```

The var statement declares a **function-scoped** or **globally-scoped** variable, optionally initializing it to a value.

The let declaration declares a **block-scoped** local variable, optionally initializing it to a value.

The const declaration creates **block-scoped constants**, The value of a constant **can't be changed through reassignment** and it can't be redeclared . However, **if a constant is an object or array its properties or items can be updated or removed.**

JS Control Structure

1. If Statement
2. If else statement
3. else if statement
4. Switch statement
5. Do While

If Statement

```
let hour = new Date().getHours();
if (hour < 20) {
  document.getElementById("demo").innerHTML = "Good day";
}
```

Use if to specify a block of code to be executed, if a specified condition is true

If else Statement

```
let hour = new Date().getHours();
if (hour < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

The if/else statement executes a block of code if a specified condition is true. If the condition is false, another block of code can be executed.

else if Statement

```
var time = new Date().getHours();
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

Use else if to specify a new condition to test, if the first condition is false

Switch Statement

```
var time = new Date().getHours();
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

The switch statement executes a block of code depending on different cases. The switch statement is a part of JavaScript's "Conditional" Statements, which are used to perform different actions based on different conditions. Use switch to select one of many blocks of code to be executed. This is the perfect solution for long, nested if/else statements.

Do while

```
let text = "";
let i = 0;
do {
  text += i + "<br>";
  i++;
}
while (i < 5);
```

- The do...while statements combo defines a code block to be executed once, and repeated as long as a condition is true.
- The do...while is used when you want to run a code block at least one time.

Function in Js

Functions will be considered as First-Class Citizen in JavaScript if the functions:

- **store functions in a variable.**
- **pass a function as an argument to another function.**
- **return a function from another function.**

How to declare a function?

```
function add(a, b) {

    // a and b are the parameters of this
    // function code to do the operation
    return a + b; // return statement
    }

    // Invoking the function and 2, 3
    // are arguments here
    console.log(add(2, 3));
```

Function Expressions

When a function is stored inside a variable, it is called a function expression. This can be named or anonymous. If a function doesn’t have any name and is stored in a variable, then it would be known as an anonymous function expression.

```
// Anonymous function expression
const add = function (a, b){
    return a + b;
    }

    // Named function expression
    const subtractResult = function subtract(a, b){
    return a - b;
    }

    console.log(add(3, 2)); // 5
    console.log(subtractResult(3, 2)); // 1
```

Callback

A function that is passed as an argument into another function is known as a callback function.

```
1   function showLength(name, callback) {
2       callback(name);
3   }
4
5   // function expression `nameLength`
6   const nameLength = function (name) {
7     console.log(`Given Name ${name} which
8     is ${name.length} chars long`);
9   };
10
11  // Passing `nameLength` as a callback function
12  showLength("Codehub", nameLength);
13
```

Arrow Function

It is an expression or syntax which is simplified as well as a more compact version of a regular or normal function expression or syntax.

```
// arrow function
const nameLength =(name)=> {
  console.log(`Given Name ${name} which
  is ${name.length} chars long`);
};
```

Scope

It is a region of the program where a variable can be accessed.

- **Global scope**: **Variables declared outside of all functions** are known as global variables and in the global scope. Global variables are accessible anywhere in the program.
- **Function scope**: **Variables that are declared inside a function** are called local variables and in the function scope. Local variables are accessible anywhere inside the function.
- **Block scope**: **Variable that is declared inside a specific block & can’t be accessed outside of that block.** In order to access the variables of that specific block, we need to create an object for it.

Scope Chain

Whenever our code tries to access a variable during the function call, **it starts the search from local variables. And if the variable is not found, it’ll continue searching in its outer scope or parent functions’ scope until it reaches the global scop**e and completes searching for the variable there

```
let name = "Abhijit";
var sector = "Government";

{
  let name = "Souvik";

  // as `var` is NOT block scoped(globally s
  // coped here), it'll update the value
  var sector = "Private";
  console.log(name); //Souvik
  console.log(sector); //Private
}

console.log(name); //Abhijit
console.log(sector); //Private
```

Closure

It is an ability of a function to remember the variables and functions that are declared in its outer scope.

“The combination of a function bundled together with references to its surrounding state or the lexical environment“

function’s local environment along with its parent function’s environment forms a lexical environment.

```
function closureDemo(){
    const  a = 3;

    return function (){
        console.log(a);
    }
}

// Returns the definition of inner function
const innerFunction = closureDemo();
innerFunction(); // 3
```

In the above example, when the `closureDemo()` function is called, it'll return the inner function along with its lexical scope.
Then when we attempt to execute the returned function, it'll try to log the value of `a` and get the value from its lexical scope's reference.
This is called closure. Even after the outer function's execution, the returned function still holds the reference of the lexical scope.

Object

In JavaScript, almost "everything" is an object.

- Booleans can be objects (if defined with the new keyword)
- Numbers can be objects (if defined with the new keyword)
- Strings can be objects (if defined with the new keyword)
- Dates are always objects
- Maths are always objects
- Regular expressions are always objects
- Arrays are always objects
- Functions are always objects
- Objects are always objects

Accessing Object value

practice.js > ...

```
const person = {
    firstName: "John",
    lastName: "Doe",
    id: 5566,
    fullName: function() {
      return this.firstName + " " + this.lastName;
    }
  };
```

name = person.fullName;

name = person.fullName();

What is this??

In JavaScript, the this keyword refers to an object.

In an object method, this refers to the object.

Alone, this refers to the global object.

In a function, this refers to the global object.

In a function, in strict mode, this is undefined.

In an event, this refers to the element that received the event.

Array

An array is a special variable, which can hold more than one value

practice.js > ...

```
    //Syntax to access an array

    const names = ['deepa','devesh','jyoti'];

    const name=new Array['deepa','devesh','jyoti']
```

Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.

Json-Javascript Object notation

- JSON is a lightweight data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

Eg-{"firstName":"John", "lastName":"Doe"}

Converting JSON text to Js Object

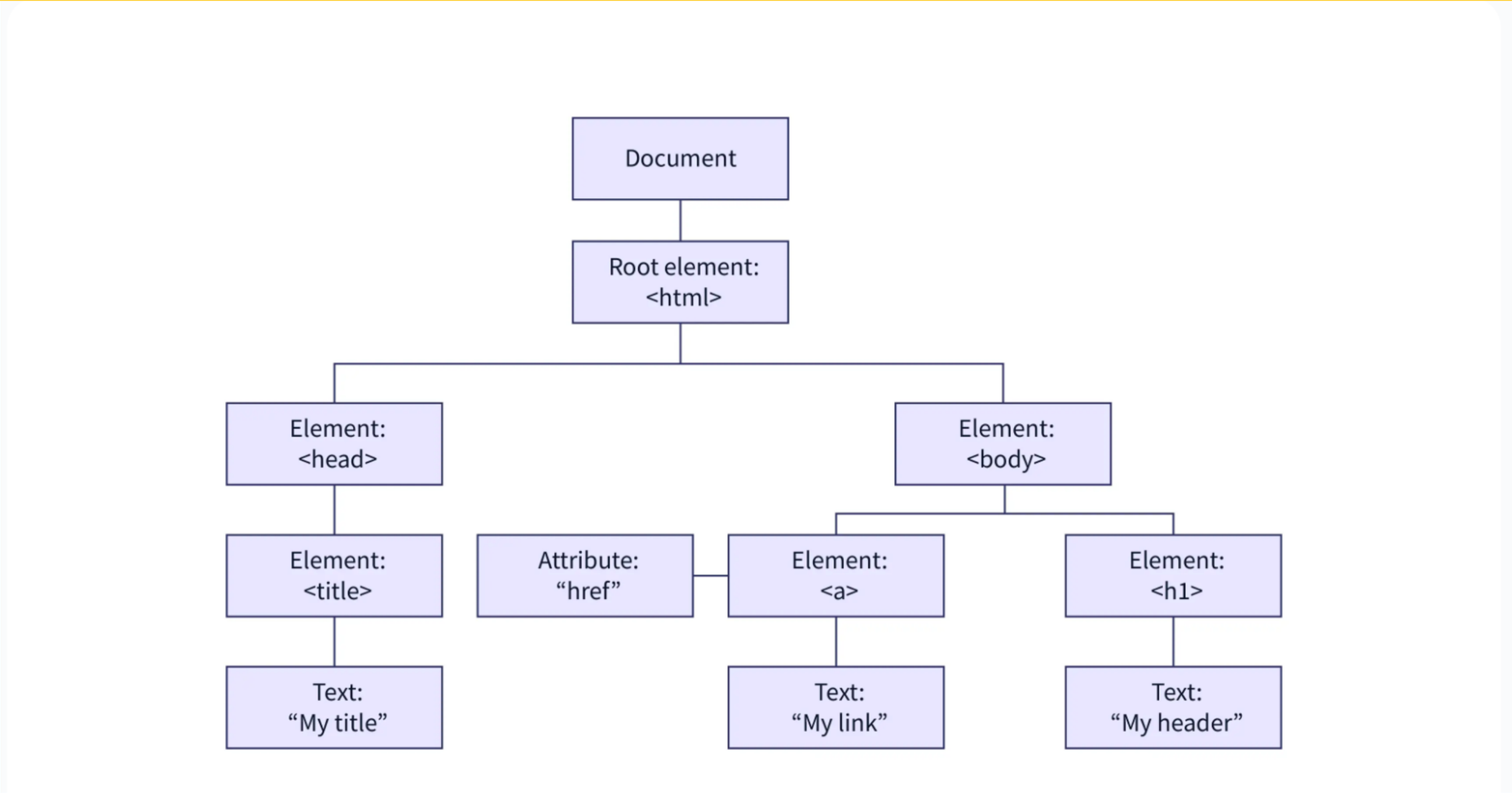
```
let text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';

const obj = JSON.parse(text);
document.getElementById("demo").innerHTML =
obj.employees[1].firstName + " " + obj.employees[1].lastName;

//Anna Smith
```

DOM Manipulation

It is the process of interacting with the DOM API to change or modify an HTML document that will be displayed in a web browser.

The DOM in dom manipulation in javascript stands for **Document Object Model.** The Document Object Model (DOM) is a tree-like structure illustrating the **hierarchical relationship between various HTML elements.**

DOM Tree


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
“href”
Element:
<a>
Element:
<h1>
Text:
“My title”
Text:
“My link”
Text:
“My header”

How to Select Elements in the DOM

- **getElementById**: returns an element whose id matches a passed string.
- **getElementsByTagName**: returns a collection of all the elements present in the document that have the specified tag name,
- **getElementsByClassName**: returns an HTMLCollection of elements that match the passed class name.
- **getElementsByName**: returns a NodeList Collection of the elements that match the value of the name attribute with the passed string.
- **querySelector**: returns the very first element within the document that matches the given selector.
- **querySelectorAll**: returns a static NodeList of elements that matches with one or a group of selectors.

Working with Events

- **Handling events**: includes HTML event handler attribute, element's event handler property, and addEventListener().
- **Page Load Events**: includes DOMContentLoaded, load, beforeunload, and unload.
- **load event**: includes dependent resources like JavaScript files, CSS files, and images.
- **Unload event**: The unload event is fired after before unload event and pagehide event.
- **Mouse events**: includes mousedown, mouseup, and click.
- **Keyboard events**: includes keydown, keypress, and keyup.
- **Scroll events**: includes scrollX and scrollY properties that returns the number of pixels that the document is currently scrolled horizontally and vertically.

Thankyou