JavaScript (JS) is a programming language used to develop and implement interactive features on web pages. It's also used in other environments like Node.js, Adobe Acrobat, and Apache CouchDB.

Here are some things to know about JavaScript:

Syntax

JavaScript's syntax is based on Java and C, and many structures from those languages apply to JavaScript.

Dynamic

JavaScript is a dynamic language that allows you to create interactive web pages.

Weakly typed

JavaScript is a weakly typed language, which means that a variable can store any data type at runtime.

Object-oriented

JavaScript supports object-oriented programming with object prototypes and classes.

Cookies

JavaScript allows you to create, read, and delete cookies to improve users' web browsing experience.

Hiding HTML elements

JavaScript allows you to hide HTML elements by changing the display style.

what are the datatypes supported by js ?

JavaScript includes primitive and non-primitive data types. The primitive data types in JavaScript include string, number, boolean, undefined, null, and symbol. The non-primitive data type includes the object. A variable of primitive data type can contain only a single value.

== and ===

== checks only value

=== checks type

closure

closure is the combination of function bundled together with reference to its surrounding state (the lexical env) , closure gives the function access to its outer scope . Closures are created every time a function is created at function creation time

function init(){

var a ="Meghana";

function inner(){

console.log(a);

}

inner();

}

init();

Hoisting is JavaScript's default behavior of moving all declarations to the top of the current scope (to the top of the current script or the current function).

elem = document.getElementById("demo"); // Find an element

elem.innerHTML = x;

x=5;

console.log(x);

var x;

temporal dead zone

In JavaScript, a temporal dead zone (TDZ) is a block of code where a variable is inaccessible until it's initialized with a value

this happens when the variable is declared with let , const , class . before being initialized . this means that variable is existing , but cant be accessed at this time

this keyword

In JavaScript, the this keyword refers to an object.

The this keyword refers to different objects depending on how it is used:

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.

Methods like call(), apply(), and bind() can refer this to any object.

const person ={

firstName : "John",

lastName : "Clover"

phone : 9194523456

id : 50

fullname : function (){

return this.firstName +" "+this.lastName

}

}

when used in an object method, this refers to the object.

In the example on top of this page, this refers to the person object.

Because the fullName method is a method of the person object.

fullName : function() {

return this.firstName + " " + this.lastName;

}

this alone

this Alone

When used alone, this refers to the global object.

Because this is running in the global scope.

In a browser window the global object is [object Window]:

Example

let x = this;

<!DOCTYPE html>

<html>

<body>

<h1>The JavaScript <i>this</i> Keyword</h1>

<p id="demo">In this example, <b>this</b> refers to the <b>person</b> object.</p>

<p>Because <b>fullName</b> is a method of the person object.</p>

<script>

// Create an object:

const person = {

firstName: "John",

lastName: "Doe",

id: 5566,

fullName : function() {

return this.firstName + " " + this.lastName;

}

};

// Display data from the object:

document.getElementById("demo").innerHTML = person.fullName();

</script>

</body>

</html>

javascript prototype

A prototype is an existing inbuilt functionality in JavaScript. Whenever we create a JavaScript function, JavaScript adds a prototype property to that function. A prototype is an object, where it can add new variables and methods to the existing object. i.e., Prototype is a base class for all the objects, and it helps us to achieve the inheritance

function Employee() {

this.name = 'Arun';

this.role = 'QA';

//this.salary='300'

}

Employee.prototype.salary=300;

var empObj1 = new Employee();

//empObj1.salary = 30000;

console.log(empObj1.salary); // 15

var empObj2 = new Employee();

console.log(empObj2.salary); // undefined

strict mode

The "use strict" directive was new in ECMAScript version 5.

It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.

The purpose of "use strict" is to indicate that the code should be executed in "strict mode".

With strict mode, you can not, for example, use undeclared variables.

All modern browsers support "use strict" except Internet Explorer 9 and lower:

You can use strict mode in all your programs. It helps you to write cleaner code, like preventing you from using undeclared variables.

"use strict" is just a string, so IE 9 will not throw an error even if it does not understand it.

undefined

When a variable is declared but not initialized, or when a function does not return a value, the variable or the function’s result is undefined.

Accessing an object property or array element that does not exist also results in undefined.

It is a primitive value.

null

It is a deliberate assignment that represents the absence of any object value.

It is often used to explicitly indicate that a variable or object property should have no value or no reference to any object.

It is also a primitive value.

difference between <script>, <script async> and <script defer>

\* { box-sizing: border-box; }

difference between: function Person(){}, var person = Person(), and var person = new Person()?

Kalyan Krishna

12:16 PM

What is the outcome of the two alerts below?

var foo = "Hello";

(function() {

var bar = " World";

alert(foo + bar);

})();

alert(foo + bar);

Kalyan Krishna

12:17 PM

What does the following code print?

console.log('one');

setTimeout(function() {

console.log('two');

}, 0);

Promise.resolve().then(function() {

console.log('three');

})

console.log('four');

Kalyan Krishna

12:19 PM

(function(){

var a = b = 3;

})();

What kind of function is this

What is the difference between var and let keyword?

Kalyan Krishna

12:20 PM

What is the key difference between “==” and “===” in Javascript?

differences on the usage of foo between function foo() {} and var foo = function() {}

Kalyan Krishna

12:22 PM

What is the difference between an "attribute" and a "property"?

Can you explain what Function.call and Function.apply do?

Kalyan Krishna

12:23 PM

"this"