person = {
firstName: "John",
lastName: "Doe",
age: 50,
eyeColor: "blue"
};

By creating instance of object directly

var objectname = new object();

It is used to create new object

<script>
var emp = new object();
emp.id = 101;
emp.name = "Ravi Malik";
emp.salary = 50000;
document.write(emp.id + " " + emp.name + " " + emp.salary);
</script>

Object properties:-

→ The name:values pairs in Javascript objects are called properties.

property	propertyValue
firstname	John
lastname	Doe

Accessing object properties:-

objectName.propertyName
or
objectName["propertyName"]

<script>
const person = {
firstNam: "John",
last name: "Doe",
id: 5566
};

→ Javascript object by object literal

object = {property1: value1, property2: value2}

<script>
emp = {id: 102, name: "shyam kumar", salary: 40000};
document.write(emp.id + " " + emp.name + " " + emp.salary);
</script>

o/p:- 102 shyam kumar 40000

Object Methods:-

→ Objects can also have methods.
→ methods are actions that can be performed on objects.
→ methods are stored in properties as function definitions.
→ A function stored as a property.

```
firstname: "John"
lastname: "Doe"
id: 5566
fullname: function() {
  return this.firstName + " " + this.lastname;
}
};
```

→ this.firstName means the firstName property of person.

→ this.lastName means the lastName property of person.

This:-

→ In Javascript, this keyword refers to an object.

→ Which object depends on how this is called.

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

→ In an object method, this keyword refers to an object.

→ Alone, this refers to the 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.

→ ~~The~~ In a function definition, this refers to the owner of the function.

→ In above example, this is the person object that "owns" the fullName function.

→ In other words, this.firstName means the firstname property of this object.

→ Accessing object Methods:

objectName.methodName()

parameters, it will return the function definition.

Do Not Declare Strings, Numbers and Booleans as objects

When a Javascript variable is declared with the keyword "new" the variable is created as object.

```
x=new String();
y=new Number();
z=new Boolean();
```

Javascript Events:-

→ Events are things that happen to HTML elements.
→ When Javascript is used in HTML pages, Javascript can "react" on these events.
→ An HTML event can be something the browser does, or something a user does.
→ Here are some examples of HTML events:-
An HTML web page has finished loading
An HTML input field was changed
An HTML button was clicked.
→ Often, when events happen, you may want to do something.
→ Javascript lets you execute code when events are detected.

```
<element event='some Javascript'>
```

with double quotes:-

```
<element event="some Javascript">
```

In the following example, an onclick attribute (with code) is added to <button>

```
<button onclick="document.getElementById
("demo").innerHTML=Date()">The time is?
```

```
<script>
function emp(id, name, salary) {
  this.id = id;
  this.name = name;
  this.salary = salary;
}
e=new emp(103, "Vimal Jaiswal", 30000);
document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

Method in javascript object:-

```
<script>
function emp(id, name, salary) {
  this.id = id;
  this.name = name;
  this.salary = salary;
  this.changesalary = changesalary;
  function changesalary(otherSalary) {
    this.salary = otherSalary;
  }
}
e=new emp(103, "sonoo Jaiswal", 3000);
document.write(e.id+" "+e.name+" "+e.salary);
e.changesalary(4500);
document.write("<br>"+e.id+" "+e.name+" "+
e.salary);
</script>
```

103 sonoo Jaiswal 3000 0

103 sonoo Jaiswal 4500 a

ensure that your web page on the web. In html, utf-8 characters, including special characters, symbols from different languages.

→ In html we can specify the character encoding using the <meta> tag. within the head section.
→ This tells the browser how to interpret the characters in the document.
→ utf-8 uses 1-byte for ASCII characters (0-127), while utf-16 uses a minimum of 2-bytes for every character even for ascii characters.
→ utf-8 result smaller transmission lines and faster transmission over the internet.

String Interpolation:-

Syntax:-

'string text'

'string text ${expression} string text'

Template literals in Javascript:-

→

Instance of:-

→ operators class has much more to offer as it only provides operator as an operator but also operator instance.

→ the instanceof operator is used to check if an object belongs to a particular class or not.

```
let arr = [1,2,3];
let obj = {name: "John", age: 30};
console.log(arr.instanceOf Array); → true
Console.log(arr.instanceOf Object); → true
```

IsArray in Array:-

→ the function Array.isArray() is used to check whether the value passed is an array or not.

```
let arr = [1,2,3];
let obj = {name: "John", age: 30};
Console.log(Array.isArray(arr)); → true
Console.log(Array.isArray(obj)); → false
```

```
let num=5;
let str="Hello";
let bool=true;
let func=function(){};
let arr=[1,2,3];
let obj={name:"John", age:30};
console.log(object.prototype.tostring.call(num));  → object number
console.log(object.prototype.tostring.call(str));  → object string
console.log(object.prototype.tostring.call(bool))
console.log(object.prototype.tostring.call(func));
Console.log(object.prototype.tostring.call(arr));
Console.log(object.prototype.tostring.call(obj));
```

O/P:→ [object number] [object function]
[object string] [object Array]
[object Boolean] [object object]

→ It is remember that typeof behaves strangely sometimes when working with certain types, such as functions and null.

→ There is a known contradiction in Javascript where typeof nullvalue returns "object" rather than null.

→
```
let result1=10/0;
let result2="Hello"/5;
console.log(isNaN(result1)); → false
console.log(isNaN(result2)); → true
```

Javascript may return NaN (not a number) when working with numeric numbers, especially when performing operations like division by zero or improper mathematical operations.

→ The isNaN() method can be used to check for NaN.

→ Since result1 is infinity, which is regarded as a numeric number, isNaN(result1) returns false.

→ Since dividing a string by a number is an improper operation, isNaN(result2) returns true since result2 is NaN.

→
```
let x;
let y;
```
→

declared & initialized.

```
let person = {name: "Alice", age: 30};
console.log("name" in person); → o/p:→ true
Console.log("gender" in person); o/p:→ false
Conside.log(person.hasOwnProperty("name")); o/p:→ true
console.log(person.hasOwnProperty("gender"))
                o/p:→ false.
function greet() {
  console.log("Hello!"); }
let person = {name: "Bob", age: 25};
console.log(typeof greet === "function"); → True
Console.log(typeof person === "function"); → false
```

→ In Javascript also we can use ; sign.
→ variables are case-sensitive.

Javascript strings:

→ Javascript strings are storing and manipulating text. A Javascript string is zero or more characters written inside quotes.

→ string length:

```
let text = "ABCDEF.... XYZ";
let length = text.length;
```

Escape character:

\b - Backspace
\f - form feed
\n - Newline
\r - carriage return
\t - Horizontal Tabular
\v - vertical Tabulator

string Interpolation:

→ string interpolation is a process in Javascript that is used to embed an expression, variable or function into a string of text. With the help of this we can embed an expression we use the template literals or backticks(`) instead of normal quotes.

Javascript string objects:

Javascript strings are primitive values, created from literals.

2-ways to create:
→ By string literal
→ By string object (new)

```
let x = "John";
```

But strings can also be defined as objects with the keyword new:-

```
var name = "value";
let y = new string("John");
```

→ Do not create string objects.
→ the new keyword complicates the code and slows down execution speed.
→ string objects can produce unexpected results when using the == operator, x and y are equal.

```
let x = "John";          // string
    y = new string("John")   // object
document.getElementById("demo").
innerHTML = (x==y);
</script>
```

string, object not comparable.

let y = new String ("John"); object.

document.getElementById ("demo").innerHTML = (x == y);

</script>

=> False →

(3) let x = new String ("John");
let y = new String ("John");
document.getElementById ("demo").innerHTML
= (x == y);
</script> → objects cannot be compared.
=> False

→ comparing two

(4) let x = new String ("John");
let y = new String ("John");
document.getElementById ("demo").innerHTML
= (x === y); → false, objects not compared.
</script>
=> False

→ comparing two javascript objects always returns false.

Javascript String Methods:-

String length()
String slice()
String substring()
String substr()
String replace()
String replaceAll()
String toUpperCase()
String toLowerCase()
String Concat()
String trim()
String trimStart()
String trimEnd
String padStart()
String padEnd()
String charAt()
String charCodeAt()
String split()