3.2.8 various objects provided by java script

JavaScript provides a range of built-in objects that help in various tasks such as manipulating interacting with the web environment, and handling different types of values. Here's an overview of so the most commonly used JavaScript objects:

1. *Object*

The fundamental object from which all other objects inherit. It is the base object for all obje JavaScript. It provides methods to work with objects and can be used to create new objects.

Example:

```
const person = new Object();
person.name = "Alice";
person.age = 30;
```

2. ***Function***

Represents a function and provides methods to define and invoke functions.

Example:

```
function greet(name) {
 return `Hello, ${name}!`;
}
const greetFunc = new Function('name', 'return `Hello, ${name}!`;');
```

3. ***Array***

Used for storing ordered collections of values. Arrays can hold items of any type and provide methods for manipulation.

Example:

```
const numbers = [1, 2, 3, 4, 5];
numbers.push(6); // Adds 6 to the end
```

4. ***String***

Provides methods to work with text (string) values.

Example:

```
const message = "Hello, world!";
const length = message.length; // Length of the string
const upperMessage = message.toUpperCase(); // "HELLO, WORLD!"
```

5. ***Number***

Represents numeric values and provides methods for mathematical operations.

Example:

```
const pi = 3.14159;
const roundedPi = Math.round(pi); // 3
const piString = pi.toString(); // "3.14159"
```

6. ***Boolean***

Represents a logical entity with two values: true and false.

Example:

```
const isTrue = true;
const isFalse = false;
```

7. *Date*

Used to work with dates and times.

Example:

```
const now = new Date();
const year = now.getFullYear(); // Current year
const formattedDate = now.toDateString(); // "Wed Sep 10 2024"
```

8. *Math*

Provides mathematical constants and functions.

Example:

```
const sqrt = Math.sqrt(16); // 4
const randomNumber = Math.random(); // Random number between 0 and 1
```

9. *JSON*

Provides methods for working with JSON data, including parsing and stringifying.

Example:

```
const jsonString = '{"name": "Alice", "age": 30}';
const obj = JSON.parse(jsonString); // Converts JSON string to object
const jsonStringified = JSON.stringify(obj); // Converts object to JSON string
```

10. *Promise*

Represents the eventual completion (or failure) of an asynchronous operation and its resulting va

Example:

```
const promise = new Promise((resolve, reject) => {
setTimeout(() => resolve("Done!"), 1000);
});

promise.then(result => console.log(result)); // "Done!"
```

11. *Symbol*

A primitive data type that is used to create unique identifiers for object properties.

Example:

```
onst uniqueId = Symbol('id');
nst obj = { [uniqueId]: 123 };
nsole.log(obj[uniqueId]); // 123
```

12. Window

Represents the browser window and provides methods for interacting with the global environment.

Example:

```
window.alert('Hello, world!'); // Shows an alert dialog
```

13. Document

Represents the web page and provides methods to interact with the content of the page.

Example:

```
const title = document.title; // Gets the title of the document
document.getElementById('myElement').innerHTML = 'Hello, world!'; // Sets content of an element
```

14. Navigator

Provides information about the browser and the operating environment.

Example:

```
const browserName = navigator.appName; // Gets the name of the browser
```

15. TypeError

Indicates that a value is not of the expected type.

Example:

```
try {
null.f(); // This will throw a TypeError
} catch (error) {
console.log(error instanceofTypeError); // true
}
```

16. ReferenceError

Indicates that a variable is not defined.

Example:

```
try {
 console.log(nonExistentVariable); // This will throw a ReferenceError
} catch (error) {
console.log(error instanceofReferenceError); // true
}
```

3.2.8.1 Math Object

Math object is a built-in static object with properties and methods for mathematical co functions. It is used to perform complex math operations. Math is not a constructor.

Math Properties (Constants):

Syntax: Math.property

The math object properties are;

1. **Math.E:** It represents the Euler's number and the base of natural logarithms.
2. **Math.PI:** This property represents the ratio of the circumference of a circle to its diameter, app 3.1415
3. **Math.SQRT2:** Square root of 2; approximately 1.414.
4. **Math.SQRT1_2:** Square root of ½; approximately 0.707.
5. **Math.LN2:** Natural logarithm of 2, approximately 0.693.
6. **Math.LN10:** Natural logarithm of 10; approximately 2.303.
7. **Math.LOG2E:** Base-2 logarithm of E; approximately 1.443.

There are many math methods. They are;

1. **Math.abs(x)** It will return the absolute value (+ve) of the given value.
2. **Math.sin(x)** It will return the sine of a given angle in radians.
3. **Math.cos(x)** It will return the cosine of a given angle in radians.
4. **Math.tan(x)** Returns the tangent of a number.
5. **Math.min(a, b, ...)** Returns the smallest number passed in the method.
6. **Math.max(a, b, ...)** This method returns the largest value of the given numbers.
7. **Math.pow(x, y)** Returns the result of a value 1 (base) raise to the power of value 2 (exponent)
8. **Math.exp(x)** It will return Ex (Exponential value of the number.), where x is the argument, Euler's constant, the base of the natural logarithms.
9. **Math.sqrt(x)** It will return a square root of a given value.
10. **Math.log(x)** This method is used to return the natural logarithm (base e) of a number.
11. **Math.ceil(x)** This method returns the smallest integer, greater than or equal to the given numbe
12. **Math.floor(x)** It returns the greatest integer, smaller than or equal to the given number.
13. **Math.round(x)** This will round the number passed as a parameter to its nearest integer.
14. **Math.random()** This method will return a pseudo-random value n , where 0 <= n<1
15. **Math.trunc(x)** Returns the integer part of the given value and discards its fractional part.

In JavaScript, the **Math Object** provides different properties and methods to quickly perfo of mathematical operations. It's not a function object. Math object works with the Number type. properties and methods of Math are static and can be called by using Math as an object without creati

3.2.8.2 String Object

The String object is probably the most used of the built-in JavaScript objects. A new String object can be explicitly created using the newString constructor, passing the literal string as a parameter:

```
var sObject = new String("Sample string");
```

The String object has several methods, some associated with working with HTML, and several not. One of the non-HTML-specific methods, concat, takes two strings and returns a result with the second string concatenated onto the first. Below example demonstrates how to create a String object and use the concat method.

Example : Creating a String object and calling the concat method

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Exploring String</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<script type="text/javascript">
//<![CDATA[
var sObj = new String( );
var sTxt = sObj.concat("This is a ", "new string");
document.writeln(sTxt);
//]]>
</script>
</body>
</html>
```

The properties and methods available with the String object are listed below

String Object methods

Method	Description	Arguments
valueOf	Returns the string literal the `String` object is wrapping	None
length	Property, not method, with the length of the string literal	Use without parentheses

Method	Description	Arguments
anchor	Creates HTML anchor	String with anchor title
big, blink, bold, italics, small, strike, sub, sup	Formats and returns `String` object's literal value as HTML	None
charAt, charCodeAt	Returns either character (`charAt`) or character code (`charCodeAt`) at given position	Integer representing position starting at position zero (0)
indexOf	Returns starting position of first occurrence of substring	Search substring
lastIndexOf	Returns starting position of last occurrence of substring	Search substring
link	Returns HTML for link	URL for `href` attribute
concat	Concatenates strings together	Strings to concatenate onto `String`'s literal string
split	Splits string into tokens based on some separator	Separator and maximum number of splits
slice	Returns a slice from the string	Beginning and ending position of slice

The HTML formatting methods—anchor, link, big, blink, bold, italics, sub, sup, small, str

3.2.8.3 Date Object

Date objects are created with the new Date() constructor.

There are **9 ways** to create a new date object:

```
new Date()
new Date(date string)
new Date(year,month)
new Date(year,month,day)
new Date(year,month,day,hours)
new Date(year,month,day,hours,minutes)
new Date(year,month,day,hours,minutes,seconds)
new Date(year,month,day,hours,minutes,seconds,ms)
new Date(milliseconds)
```

2.8.4 Boolean and Number object

In JavaScript, Boolean and Number are two fundamental objects that represent primitive data types: Boolean values and numeric values, respectively. Understanding these objects helps in manipulating and managing boolean and numeric data efficiently. Here's a detailed overview of each:

1. Boolean Object:

Definition: The Boolean object is a wrapper for the primitive boolean values (true and false). While most of the time, you work with boolean values directly, Boolean can be used as an object.

Creating Boolean Objects:

```
const boolObjTrue = new Boolean(true);
const boolObjFalse = new Boolean(false);
```

Primitive Boolean Values:

```
const boolPrimitiveTrue = true;
const boolPrimitiveFalse = false;
```

2. Number Object:

Definition: The Number object is a wrapper for the primitive number data type. It provides methods for working with numbers and can be used to create numeric values.

Creating Number Objects:

```
const numObj = new Number(123);
```

Primitive Numbers:

```
numPrimitive = 123;
```