JavaScript built-in objects are predefined objects that are part of the JavaScript language and can be used to perform various tasks in web development. One such built-in object is the Date object, which allows developers to work with dates and times.

Interacting with browsers is also a critical aspect of JavaScript programming, as it allows developers to create dynamic and interactive web pages that can respond to user actions.

Understanding these concepts is essential for any JavaScript programmer, as they provide powerful tools for creating robust web applications. In this lecture, we will cover the basics of JavaScript built-in objects, with a focus on the Date object, and explore how to interact with browsers to create dynamic web content.

some of the most commonly used Built-in Objects in JavaScript with applied examples with code

There are several commonly used built-in objects in JavaScript that provide useful functionalities to programmers.

Here are some of the most commonly used built-in objects in JavaScript:

1. **String: The String object represents a sequence of characters and provides various methods to manipulate strings. For example, the length property returns the length of a string, and the concat method joins two or more strings.**

Example code:

```
let str = "Hello, World!";
console.log(str.length); // Output: 13
console.log(str.concat(" How are you?")); // Output: Hello, World!
How are you?
```

2. Number: The Number object represents numerical values and provides various methods to manipulate numbers. For example, the toFixed method rounds a number to a specified number of decimal places.

Example code:

```
let num = 3.14159265359;

console.log(num.toFixed(2)); // Output: 3.14
```

3. Array: The Array object represents a collection of elements and provides various methods to manipulate arrays. For example, the push method adds an element to the end of an array, and the pop method removes the last element from an array.

Example code:

```
let fruits = ["apple", "banana", "cherry"];

fruits.push("orange");

console.log(fruits); // Output: ["apple", "banana", "cherry",
"orange"]
```

```
fruits.pop();

console.log(fruits); // Output: ["apple", "banana", "cherry"]
```

4. Object: The Object object represents a collection of key-value pairs and provides various methods to manipulate objects. For example, the keys method returns an array of the object's keys, and the values method returns an array of the object's values.

Example code:

```
let person = {name: "John", age: 30, city: "New York"};

console.log(Object.keys(person)); // Output: ["name", "age", "city"]

console.log(Object.values(person)); // Output: ["John", 30, "New
York"]
```

These are just a few examples of the many built-in objects available in JavaScript that programmers can use to make their code more efficient and effective.

the Date object in JavaScript.

The Date object is a built-in object in JavaScript that allows developers to work with dates and times. It provides a way to create and manipulate dates and times, and perform operations on them. The Date object is very useful in web development when you need to display, process or manipulate dates and times.

To create a Date object, you can use the `new` keyword followed by the `Date()` constructor function, like this:

```
const today = new Date();

console.log(today);
```

This will create a new Date object with the current date and time, and print it to the console.

The Date object provides a number of methods that allow you to perform operations on dates and times. Some of the most commonly used methods include:

`getFullYear()`**: Returns the year of a date as a four-digit number.**

```
const today = new Date();

console.log(today.getFullYear());
```

`getMonth()`**: Returns the month of a date as a number (0-11).**

```
const today = new Date();

console.log(today.getMonth());
```

`getDate()`**: Returns the day of a date as a number (1-31).**

```
const today = new Date();
console.log(today.getDate());
```

`getHours()`**: Returns the hour of a date as a number (0-23).**
const today = new Date();
console.log(today.getHours());

`getMinutes()`**: Returns the minutes of a date as a number (0-59).**
const today = new Date();
console.log(today.getMinutes());

`getSeconds()`**: Returns the seconds of a date as a number (0-59).**
const today = new Date();
console.log(today.getSeconds());

You can also set specific dates and times using the `setFullYear()`**,** `setMonth()`**,** `setDate()`**,** `setHours()`**,** `setMinutes()`**, and** `setSeconds()` **methods.**

const myDate = new Date();
myDate.setFullYear(2024);
console.log(myDate);

This will set the year of the `myDate` **object to 2022.**

The Date object can also be used to perform calculations on dates and times. For example, you can calculate the difference between two dates using the `getTime()` **method.**

```
const today = new Date();
const futureDate = new Date("June 30, 2023");
const difference = futureDate.getTime() - today.getTime();
console.log(difference);
```

This will calculate the number of milliseconds between today's date and June 30, 2023.