JavaScript for Web Development

**by Board Infinity**



About this Course

Three module/week intermediate Web Development Course.

Dive into the captivating world of JavaScript, a cornerstone language that has revolutionized web development by enabling dynamic and engaging web applications. This course, divided into three holistic modules, ensures learners gain a deep understanding of JavaScript from basic concepts to advanced implementations.

In the first module, you'll embark on a journey through the foundational elements of JavaScript. This foundational foray will familiarize you with the primary components such as data types, variables, operators, and fundamental functions, setting the stage for the subsequent modules.

The second module transitions to the vibrant world of web interactivity. Here, you'll delve into the intricacies of the Document Object Model (DOM) and understand how JavaScript seamlessly integrates with it. Mastering this module is essential for those aiming to craft web pages that dynamically adjust and respond to user interactions.

Finally the third module, we elevate our exploration to advanced techniques and the influential jQuery library. With jQuery, you'll discover streamlined methods for sophisticated DOM manipulations and enhanced event handling, paving the way for the creation of intricate web application functionalities.

By the end of the course, you will have a solid foundation in JavaScript programming and a clear understanding of how to use it for web development. You will also have the skills to design and develop dynamic, interactive web applications that resonate with the contemporary digital landscape. This course takes 3 weeks to complete.

**Table of Contents**

**Module 1: Introduction to JavaScript Programming**

This module is designed to provide learners with a comprehensive introduction to JavaScript programming. Throughout the module, learners will gain hands-on experience working with JavaScript and will develop a deep understanding of its core features and capabilities.

The module will cover the key concepts of JavaScript, including variables, data types, operators, control structures, and functions. Learners will understand how to use these concepts to create basic programs that are modular, maintainable, and easy to test. By the end of the module, learners will be proficient in creating simple and effective JavaScript programs. They will be able to apply best practices and coding standards to write high-quality code that meets the needs of their clients and users.

**16 lesson, 3 readings**

**Syllabus**

1. [**Course Introduction**](https://www.coursera.org/lecture/javascript-for-web-development/course-introduction-video-3k9Hn) **– JS for Web Development (0:53)**
2. **About this lesson (0:38)**
3. **Introduction to JavaScript (5:33)**
4. **Setting up your development environment (2:46)**
5. **Writing your first JavaScript program (4:39)**
6. **About this lesson (0:52)**
7. **Variables in Javascript (7:12)**
8. **Data Types in Javascript (8:58)**
9. **Using arithmetic and comparison operators (8:38)**
10. **Arrays in Javascript (5:41)**
11. **Working with Strings (6:17)**
12. **String Manipulation (6:34)**

**1.12.1 Variables, Operators, and Strings**

1. **About this lesson (0:48)**
2. **Conditional statements (if/else, switch) (7:17)**
3. **Looping structures (For, While, Do-While) (6:43)**
4. **Functions and Scope (7:50)**

**1.16.1 Control Statements**

**Module 2: DOM Manipulation and Event Handling**

This module is designed to provide learners with a comprehensive understanding of how to manipulate the Document Object Model (DOM) using JavaScript and how to handle events in JavaScript. Throughout the module, learners will gain hands-on experience working with the DOM and will develop a deep understanding of how to make dynamic and interactive web pages.

The module will cover the key concepts of the DOM, including how to access and manipulate DOM elements using JavaScript, and how to use event listeners to respond to user input, such as clicks and key presses. Learners will understand how to use these concepts to create interactive web pages that are responsive to user input. By the end of the module, learners will be proficient in manipulating the DOM using JavaScript and will be able to handle events to make their web pages more interactive. They will be able to apply best practices and coding standards to write high-quality code that meets the needs of their clients and users.

**13 lessons, 3 readings**

1. [**About the DOM**](https://www.coursera.org/lecture/javascript-for-web-development/about-this-lesson-IT7ub) **(0:46)**
2. **Understanding the Document Object Model (DOM) (3:51)**
3. **Accessing DOM elements with JavaScript (7:21)**
4. **Manipulating DOM elements with JavaScript (7:10)**
5. **Modifying styles and attributes with JavaScript (6:17)**
6. **DOM Manipulation**
7. **What you will learn in this Lesson (0:47)**
8. **Understanding events and event listeners (3:49)**
9. **Responding to user input (clicks, key presses) (6:16)**
10. **Creating interactive user interfaces with events (4:59)**
11. **Event Handling**
12. **What you will learn in this Lesson (0:59)**
13. **Creating and removing DOM elements with JavaScript (6:36)**
14. **Using AJAX to load content dynamically (7:03)**
15. **Handling errors and exceptions (5:53)**
16. **Advanced DOM Manipulation**

**Module 3: Advanced JavaScript Techniques and jQuery**

This module is designed to provide learners with a comprehensive understanding of advanced JavaScript techniques and the jQuery library. Throughout the module, learners will gain hands-on experience working with JavaScript and jQuery and will develop a deep understanding of how to make dynamic and interactive web pages.

Learners will understand how to use these techniques to write efficient and effective JavaScript code. In addition, learners will be introduced to the jQuery library, which simplifies DOM manipulation and event handling in JavaScript. They will learn how to use jQuery to create animations, handle events, and manipulate the DOM more easily. By the end of the module, lerners will build a hands on project that will bring thier learning to life.

12 lessons, 2 readings

1. [**What you will learn in this Lesson**](https://www.coursera.org/lecture/javascript-for-web-development/what-you-will-learn-in-this-lesson-ODHZ5) **(0:50)**
2. **Working with arrays and objects in JavaScript (8:18)**
3. **Understanding closures and the scope chain (11:42)**
4. **Debugging JavaScript code (7:51)**
   1. **Advanced JS Techniques**
5. **What you will learn in this Lesson (1:18)**
6. **What is jQuery? (5:01)**
7. **jQuery Selectors (7:39)**
8. **jQuery Effects (6:31)**
9. **jQuery Traversing (5:33)**
10. **jQuery**
11. **Project Building - Part 1 (5:35)**
12. **Project Building - Part 2 (10:46)**
13. **Project Building - Part 3 (6:03)**

**1.1 Syllabus**

This course is designed for beginners who want to learn how to use JavaScript to add dynamic and interactive features to their websites. In this course, you will learn the fundamentals of JavaScript programming, and how to apply them to web development.

Throughout the course, you will be guided through the process of using JavaScript to manipulate the Document Object Model (DOM) of a web page, handle events, and create dynamic and interactive user interfaces. You will also learn how to use JavaScript to make asynchronous requests to web servers and consume data from web APIs.

By the end of the course, you will have a solid foundation in JavaScript programming and a clear understanding of how to use it for web development. You will also have the skills to create dynamic and interactive websites that can respond to user input.

**Module 1: Introduction to JavaScript Programming**

This module is designed to provide you with a comprehensive introduction to JavaScript programming. Throughout the module, you will gain hands-on experience working with JavaScript and develop a deep understanding of its core features and capabilities.

The module will cover the key concepts of JavaScript, including variables, data types, operators, control structures, and functions. You will understand how to use these concepts to create basic programs that are modular, maintainable, and easy to test.

By the end of the module, you will be proficient in creating simple and effective JavaScript programs and will be able to apply best practices and coding standards to write high-quality code that meets the needs of your clients and users.

**Module 2: DOM Manipulation and Event Handling**

This module is designed to provide you with a comprehensive understanding of how to manipulate the Document Object Model (DOM) using JavaScript and how to handle events in JavaScript. Throughout the module, you will gain hands-on experience working with the DOM and will develop a deep understanding of how to make dynamic and interactive web pages.

The module will cover the key concepts of the DOM, including how to access and manipulate DOM elements using JavaScript, and how to use event listeners to respond to user input, such as clicks and key presses. You will understand how to use these concepts to create interactive web pages that are responsive to user input.

By the end of the module, you will be proficient in manipulating the DOM using JavaScript and will be able to handle events to make your web pages more interactive. You will be able to apply best practices and coding standards to write high-quality code that meets the needs of your clients and users.

**Module 3: Advanced JavaScript Techniques and jQuery**

This module is designed to provide you with a comprehensive understanding of advanced JavaScript techniques and the jQuery library. Throughout the module, you will gain hands-on experience working with JavaScript and jQuery and develop a deep understanding of how to make dynamic and interactive web pages.

You will understand how to use these techniques to write efficient and effective JavaScript code. In addition, you will be introduced to the jQuery library, which simplifies DOM manipulation and event handling in JavaScript. You will learn how to use jQuery to create animations, handle events, and manipulate the DOM more easily.

By the end of the module, you will build a hands-on project that will bring learning to life.

End syllabus. Begin Course:

1.1 Course Introduction: JS for Web Dev (0:53)

Welcome to the JavaScript for web development course. (BI-JSforWebDev)

In this course, you will learn how to use JavaScript, to create dynamic and interactive web application. The course is divided into three modules starting with an introduction to JavaScript programming.

Here you will learn the basics of JavaScript including data types, variables, operators and functions.

The second module covers DOM manipulation and Event handling. You will learn how to manipulate DOM that is Document Object Model using javascript and how to handle events such as mouse click and key presses.

Finally, in the third module, you will learn Advanced Java script techniques and jQuery. You will learn how to use JQuery to simplify DOM manipulation and how to use java script to create animations and effects.

By the end of this course, you will have a strong foundation in JavaScript programming and you will be able to create dynamic and interactive web applications. We are excited to start this journey with you, so, are you ready to learn JavaScript? See you in the next video.

1.2 About this Lesson (0:38)

Hi there. In this lesson, you will be introduced to the world of JavaScript,  a high level programming language that is widely used for web development.

You will learn about the basics of JavaScript programming, including the syntax, data, types and variables.  You will also set up your development environment to get started with writing  your own JavaScript code. By the end of this lesson,  you will have a solid understanding of the basics of JavaScript programming. And you will be able to create simple JavaScript programs. You will also be ready to move on to more advanced topics like JavaScript development. Whether you are a complete beginner or have some programming experience, this lesson will provide you with a solid foundation in JavaScript programming. See you in the next lesson.

1.3 Introduction to JavaScript (5:33)

Hi there. In this lesson we will learn some history and introduction related to JavaScript. So let's get started.

The very first question that comes to our mind is, what is JavaScript?

JavaScript is a programming language that is used to add interactive features to websites.

It is used often alongside HTML and CSS to make websites more dynamic and engaging. In simple terms, JavaScript allows you to add functionality to your web page. For example, you can use JavaScript to create interactive forms, add animations, and create dynamic effects when users interact with your web page.

One of the great things about JavaScript is that, it runs in the web browser, which means that it can be used on almost any device that has a web browser. This makes it a powerful tool for creating cross platform applications that can be used on desktop computers, tablets, and smartphones.

So why is it called JavaScript? JavaScript was originally created by Brendan Eich at Netscape Communications. The language was initially called Mocha, but it was later renamed to Live Script. The decision to rename the language to JavaScript was largely a marketing decision made by Netscape. At that time, the company was in heated browser war with Microsoft and they wanted to capitalize on the popularity of Java by making their language sound similar. This helped to create a perception that JavaScript was a complementary technology to Java, even though the two languages are quite different.

JavaScript has become one of the most widely used programming languages in the world, and it is an essential tool for web developers today. In fact, according to some estimates, JavaScript is used over 95% of all websites on the Internet.

Where else beyond the browser can JavaScript be executed? JavaScript is a versatile programming language that can be used to create both front end and backend applications. It is primarily known for its use in web development, where it is used to add interactivity and dynamic effects to web page.

However, JavaScript can also be used to build serverside applications that run on web server, as well as desktop and mobile applications, and even on IoT devices. This versatility is one of the greatest trends of JavaScript as it allows  developers to use a single programming language to build applications across a wide range of platforms and devices.

This not only makes development more efficient, but also enables developers to create interactive and responsive user interfaces that can respond to user input in real time. Web browsers use different JavaScript engines to interpret and execute JavaScript code. For example, V8 is used by Chrome, SpiderMonkey by Firefox, and Chakra by Internet Explorer.

Each engine has its own unique features and performance characteristics.

What are the tasks that in browser JavaScript can perform? Let's look at some. It can manipulate HTML and CSS. JavaScript can change the style and content of elements on a web page.

For example, by hiding or showing an element, or changing its color, or size, or maybe animating it. Handling user events, JavaScript can respond to user interactions with the page, such as clicking a button, submitting a form, or scrolling, and trigger corresponding actions such as, displaying a message or updating the page content. Validate forms, JavaScript can check the user input in a web form.

For example, to ensure that a required field is not left blank, that a phone number is valid, or that a password means a certain criteria. Add interactivity, JavaScript can add interactive elements to a web page such as drop down menus, image sliders, or pop up windows to enhance the user experience and engage users with the content.

Make HTTP requests, JavaScript can fetch data from a server or send data to a server using HTTP request. For example, to update the page content dynamically or submit a form without reloading the page. Last but not the least, it can store and manipulate data.

JavaScript can store data in variables, arrays, and objects, for example, to remember user preferences or to keep track of game scores and manipulate the data using built-in functions such as sorting or filtering.

Let's summarize this, JavaScript is a high level dynamic programming language  that is used for creating interactive web pages and web applications.  I would call it a 5th generation language feeding off its’ prior 4 levels. Originally called Mocha and later renamed to LiveScript, it was eventually rebranded as JavaScript for marketing purposes. While it is primarily associated with web development, JavaScript can also be used on servers and other devices that have a JavaScript engine. With its versatility and popularity, JavaScript has become a fundamental language for modern web development. This is all for this lesson. In the next lesson, we will see how to set up your development environment.

1.4 Setting Up Your Development Environment (2:46)

Hi there, in the previous video, we saw introduction to JavaScript.

Now in this video, we will set up our development environment for JavaScript.  So let's get started. To set up your development environment, you need a code editor, a web browser, and a JavaScript engine. A code editor enables us to write and organize our code with features such as syntax, highlight, autocompletion, and code formatting.

It also provides us with debugging tools to identify and fix errors in our code. A web browser is necessary for testing our JavaScript code in a real world environment. It allows to see how our code interacts with HTML and CSS on a web page and to test features such as user interactions and Http request.

A JavaScript engine enables us to run JavaScript code outside of a web browser, which is useful for building serverside, applications, command line tools, and other types of applications that do not rely on web browser.

You can follow these basic steps for your development environment. So the first step is you can install a code editor. You can choose from variety of free and paid code editors such as Visual Studio Code, Sublime Text, or Atom. These editors provide features such as syntax highlighting, autocompletion, and debugging tools that make writing and debugging JavaScript code easier. You can just go to Google and let's say if you want to install Visual Studio code, you can just type in the Google and you can just install it for your OS.

Second is to install a web browser.  You can choose from popular web browsers such as Google Chrome, Mozilla Firefox, or Microsoft Edge. These browsers have built in developer tools that enable you to inspect and debug your JavaScript code in real time.

Last but not the least, you have to install a JavaScript engine. So most modern web browsers come with their own JavaScript engine. But you can also install standalone engines such as Node, JS or Rhino. These engines allow you to execute JavaScript code outside of a web browser, for example, to build server side applications or command line tools.

So once you have these tools set up, you can start writing and running JavaScript code in your code editor and test it on your web browser or JavaScript engine. So this is all for this video. In the next video, we will write our very first JavaScript program. See you in the next video. Thank you.

1.5 Writing Your First JavaScript Program

Hi there, in the previous video, you have set up your development environment. Now in this lesson, you will write your first JavaScript program.

Let's get started. Let's look at some basic steps to write the first JavaScript program.

The first step is to open up a code editor. You can choose a code editor of your choice, such as Visual Studio Code, Sublime Text, or Atom and you have to create a new file with the extension.js.

The second step is to write the code. In your new file, you can write your first JavaScript program, for example, you can write a simple Hello World program.

The third step is to save the file. You have to save your file with a meaningful name and the .js extension, for example, hello.js.

The last step is to run the program. You can open up a terminal or command prompt and navigate to the directory where your program file is saved. There you can type node hello.js to run your program. You should see the output printed in the console. Let's actually follow these steps and let's try our first JavaScript program. I will go to my desktop and I will create a new folder and let's rename it as JavaScript.

**1.6 Variable, Data Types & Operators (7:12)**

Hi there. In this lesson we will learn variables in JavaScript, so let's get started.

Let's say that you have a box of crayons and you want to organize them by color. You could use different compartments in the box to store each color separately. Each compartment would be like a variable and the crayons would be the data stored in those variables. For example, you could have a compartment for red crayons, a compartment for blue crayons, and so on. The number of crayons in each compartment could vary depending on how many crayons you have of each color.

In JavaScript, variables work in a similar way. You can think of variable as a container that holds a piece of data like a crayon. The variable has a name that you can choose, just like the compartment in the crayon box has a label indicating the color of crayons it holds. You can then use these variables to perform calculations or manipulate the data stored in them.

Just like you can move crayons from one compartment to another, you can change the data stored in a variable as needed.

Most of the time, a JavaScript application needs to work with some information. Let's see some examples. The first is a social media app. In this case, the information might include the user's name, age, and number of followers.

The second is a music app. In this case, the information might include the user's favorite songs, playlist, and artist. Variables are used to store this information. To create a variable in JavaScript,  you can use let, var, and const keyword. Let's understand when to prefer what through some examples. Let's move to the VSCode.

Here I have index.html ready with us.

We can just create a new file and let's call it as variables.js. You can use var, let, and const, but when to use what lets see that. Before that, let's use let keyword and let's say let name to be John. This is how you create a variable.

Here you have two process. First is that you are saying let name. That means you are declaring a variable. Then you are assigning the value, that is John in this case, like this.

This is variable declaration. This is variable initialization. You can in fact do the initialization and declaration in the same line like this, you will use in this fashion. Let is the preferred way to declare a variable that can be reassigned later in the code. It is blocked scope, which means that it can only exist within the block of code where it was defined. For example, let's create a variable by the name of count and we can give it a value of 0.

Then you can say count to be 1 and then we can say console.log(count).

Now we can run this. Let's clear up the terminal and here we can say, let's save and let's say node variables.js and you can see 1 is printed here.

Next we can look at const. Const is the preferred way to declare a variable that cannot be reassigned. It is also blocked scope. For example, let's say if you want to use some constants, like let's say Pi.

We can say const PI and normally the value of Pi is approximately equal to 3.14, and then we can say console.log(PI).

If I click on "Save", if I run this code, you will see that we get the value. But now if someone tries to do this, let's say four or something, in that case, it will give an error, as you can see here, that you cannot assign it to a constant variable.

The third one that is left is var keyword. Var is older keyword for declaring variables in JavaScript. While it can still be used in modern JavaScript, it is generally considered outdated because it is not blocked scope. This means that variables declared with var are available throughout the entire function or global scope, which can make it harder to reason about your code and avoid bugs.

Now all these terms, functions and global scope you will be looking at in further videos in detail.

For now, let's also see how we can create a variable using var.

We can say var count = 0 and we can use our if condition here.

Again, we will see if conditions in detail in the conditional spot.

I can say if true, let's say var count to be 1 and then we can say console.log(count).

In the down, if you will say console.log(count) as well.

Let's clear up the terminal and let's see the output.

Here you can see that in both the cases, it is 1, that means it is not blocked scope.

It has shadowed the previous count at this point.

Earlier it was 0, even though it isn't a block, that is these curly brackets.

It has overridden the previous count, and now you can see that in both the console's, it is printing 1 and 1 each time.

Let's summarize this.

Overall variables are a way to store and manage data in your code.

They allow you to refer to a data by name, which makes your code more organized and easier to read.

To create a variable in JavaScript, you can use let, var, and const keyword.

Let and const are the preferred keywords for declaring variables in modern JavaScript because they are blocked scope, and offer better control over variable assignment.

While var can still be used, it is generally considered outdated and should be avoided in new code wherever possible.

In the next video, we will see data types in JavaScript.

* 1. Data Types in JavaScript (8:58)

Hi there. In the previous video we learned variables in JavaScript. Now in this video we will learn data types in JavaScript. So let's get started.

In JavaScript, a data type represents the type of data that can be stored and manipulated in a program. Let's look at some common data types in JavaScript. The first one is numbers, a data type that represents both integer and floating point numbers. In JavaScript, numbers are represented with the number keyword.

Let's see all these data types side by side with an example. Let's move to VS Code. And here I have a file, datatypes.js, ready with me. If you want to create a data with a number type, you can say let, and we can just say x. X is a variable name, and we can say 5, this is a number. This creates a variable x of type number with value of 5. Next one is BigInt.

For most purposes, we have a range that is +(2^53 - 1) and -(2^53 - 1) and this range is quite enough. But sometimes we need the entire range of really big integers. For example for cryptography of microsecond precision timestamps. Now BigInt was recently added to the language to represent integers of arbitrary length. A BigInt value is created by appending end to the end of an integer. The n means that it's a big end. Let's see how we can create one. Here I can say const and let's say bigInt, and here we can give it a very lengthy number. Let's just give any number and let's just say n. Now, if you want to check the type of any variable in JavaScript, you can say console.log, and we can say typeof bigInt. If I click on Save, let's run this program that is node datatype.js, and you can see it gives the type as bigInt in the console. BigInt rhombus are rarely needed.

But still if you want, you can use them like this. Next data type is String.

A data type that represents a sequence of characters is a String. In JavaScript, Strings are represented using a String keyword. Let's see an example. If you wanted to create a String, you can say let and we can say name, and the name we want to give is a string that is a sequence of characters. Let's give it John. This creates a variable name of type string with a value of John. Again, if you want to check the type, you can use this type of operator here, but this is how you represent a string.

The next one is Boolean, a data type that represents a logical value. It could be either true or false. In JavaScript, Booleans are represented using a Boolean keyword. Let's see an example.

To create a variable that holds Boolean value, you can say let and we can give any variable name like is True or something. And we can just give it a value of true. So this creates a variable isTrue of type Boolean with a value of true. Next is Null. It is a data type that represents the intestinal absence of any object value. It is represented using a null keyword. It is a separate data type, and if you want to create or give a value to a variable, you can just say let and you can create any variable, let's say x

* 1. **Using Arithmetic and Comparison Operators (8:38)**

Hi there. In the previous video we learned data types in JavaScript.

Now in this video we will learn about arithmetic and comparison operators in JavaScript.

Let's get started. Let's talk about arithmetic operators first. Arithmetic operators in JavaScript are used to perform mathematical calculations on numerical values. As you can see, one of the arithmetic operator is addition. Then we have subtraction, multiplication, division, modulus and exponentiation. These are some symbols that you can use for these operators.

Let's go to the VS code and understand them by examples.

I'm in my VS code and let's create two variables first.

We can say let x and let's give it a number 10. And we can say, let y and let's give it a value of 5. We want to perform arithmetic operations, so let's talk about addition. We can say let sum and addition would be x+ y. This is the addition operator. Then I can just say console.log(sum). The result that is stored for x +y is in the variable sum.

And as soon as I click on Save and run this program, let's say node operators.js, you can see the output is 15.

Similarly, let's take the other examples as well. If you want to calculate the difference, you can say let difference, and this would be equal to x- y.

This is a subtraction operator. For multiplication, you can say let product, and it would be x into y.

We use this star symbol. This will give you the multiplication of x and y. Then we have division, so we can say, let's create a variable called quotient, and let's make it equal to, x divided by y.

We use this symbol here. Then we have modulus, so we normally calculate remainders using a modulus, that is a percentage symbol. Let's create a variable, let's say remainder, and we can make it equal to x%y.

Then last we have exponentiation, so for that we have double multiplication symbols. What do I mean is, let's create a variable, let result. And if you want to calculate the exponential value, you can say x two times this and then y. Just like we were doing console.log(sum), let's get all the values here. We have console.log(sum) then let's copy and paste and put difference here, then we can put product here.

And then let's put quotient here, and then we can say remainder, and in the end, we can just say result. Let's click on Save. And let me clear up the terminal and I will run this program again. Let's say node and we have operators.js. You can see we get the output. Sum is 15, difference is 5 and product is 50.

Then quotient is 2, remainder is 0, and the result is 100,000. This is how you can actually use arithmetic operators in JavaScript.

Let's talk about comparison operators. Comparison operators in JavaScript are used to compare values, and in turn they return a Boolean value that is either true or false based on the comparison result.

The following are the comparison operators in JavaScript we have equal to, we have not equal to, then we have strict equal to, then we have strict not equal to, then greater than, greater than equal to, and similarly for less than and less than equal to.

Let's also understand these using examples.

Remember, the output will always be a Boolean value.

Let's go here and let's comment everything out. Let's clear up the terminal as well. And let's start with comparison operators. Let's take two values. Let's say let a to be 10 again and let's take b to be 5.

Now what we have to do is we have to compare. Let's use an equal to operator first. We can say console.log(a)== b. And what should be the output here? Say if I run this program, you will see we get the output as false, because 10 is not equal to 5.

Then we can say not equal to. We can just copy this, the only thing we have to change is the symbol that is not equal to.

And if I run this program now, you will see that we get the value false and the second one we get the value as true.

And of course, 10 is not equal to 5.

Similarly, we can check for strict equal. So strict also checks the data type for it, and then we have strict not equal as well. The only thing in both the cases is that you have to add one equal to sign extra.

This is strict equal to and this is strict not equal to. Again, the output would be same. If I run this program, you will see false true and false true. Then let's look at the other remaining operators as well. We have greater than, so we can say console.log, this would be a greater than b, and then we have greater than or equal to.

It will be like this.

Let's see the output, let me clear up the terminal.

And if I run this program you will see we get the output as true and

true in both the cases. a is yes, greater than b and it also satisfies this condition as well. Similarly, we have less than operators. Rather than doing this, we can use the opposite bracket.

And now we have less than and less than equal to.

If I try to run this program, you will see the last two values are false and false.

Let's summarize this, arithmetic operators in JavaScript are used for mathematical calculations such as addition, subtraction, multiplication, division, etc.

Comparison operators on the other hand, are used to compare values and return a Boolean value based on the comparison result, such as equal to, not, equal to, greater than, less than, and so on.

These operators are fundamental in JavaScript programming and understand standing how to use them is very essential for the beginners.

In the next lesson, we will understand arrays in JavaScript.

* 1. **Arrays in JavaScript (5:41)**

Hi there. In the previous video, we learned arithmetic and comparison operators in JavaScript.

Now in this lesson we will learn arrays in JavaScript so let's get started.

In programming an array is a collection of values that are stored under a single variable name.

Arrays can hold any type of value, including numbers, strings, and even other areas.

Each value in an array is accessed through its index, which starts at zero for the first value in the array.

Let's understand by an example. Let's say that we have array of numbers, and it normally starts with square brackets.

Let's say we have numbers as 10,20,30,40 and 50. In this example, we have an array and you can call it as my array or whatever the variable name you want to give. It contains five values, 10,20,30,40 and 50. Each value is stored in a separate slot in the array, which is identified by its index number and how we can identify it?

The first value in the array 10 is stored in slot zero, and that is called list index.

The second value 20 is stored in slot one, then two, then three, then four and then so on. Let's also see the array example in VS Code. You can think of array in JavaScript like a drawer with multiple compartments. Each compartment has a label and you can put things in each compartment, for example let's say that you have a drawer labeled fruits.

What we want here is inside the drawer, you have compartments labeled as apples, oranges, and bananas. You can put one or more fruits in each compartment. Similarly, in JavaScript, you can create an array called fruits and store multiple values in it.

The array can contain different types of values, like strings, numbers, or even other areas. You can access the values in the array by using an index number, just like opening a specific compartment in the draw.

Let's create an array of fruits.

We can say let's put some fruits here. This is an array and we can put apples, oranges and let's also put bananas. This is an array of strings, all these values are strings. How to access the first element in the array?

We can just say console.log. How to access this, you can see fruits and you can use this square bracket notation and you could pass the index.

Fruits at index zero will give you the output as apples. Let's test it out.

If I click on Save and if I run this program, that is node arrays, you will see that we get the values as outputs. If you say fruits at index one, you will get oranges. There are also some built-in methods in JavaScript for arrays, for example let's say if you want to add a new fruit to this array.

How we can do it, we can say fruits.push. Let's say we want to add grapes.

Now if I just print out the original array, so let's say console.log, and let's consume fruits. Let's run the program and you will see here that the grapes has been pushed at the end of the array. Similarly, we have a pop method as well.

What we can do is we can say fruits.pop. If I do console.log fruits again and let's run this program, you will see we get back the original array.

Why? Because the last value that is grapes is popped out from the end of the array. There are more built-in methods, but these are some, or you can say the most used methods in array. Let's summarize this.

Array is a collection of values that are stored under a single variable name. Arrays can hold any type of value including numbers, strings, and even other arrays. You can also see the syntax here.

Here we have array of numbers, array of strings and the last one is mixed array, that is array of mixed data types. You can create an array in JavaScript using square brackets and separating the values with comma.

You can access the values in an array using the index number of the value we want to retrieve. For example fruits at index one. You can modify the values in an array by assigning a new value to a specific index.

Arrays also have built-in methods that you can use to manipulate the data in the array. Just push to add a value to the end of the array and pop to remove the last value from the array. Understanding arrays is a very important concept in programming as they allow you to store and access multiple values under a single variable name.

In the next lesson, we will understand about strings and string manipulation in JavaScript.

* 1. **Working with Strings (6:17)**

Hi there. In the previous lesson, we learned about arrays. Now in this video we will learn strings in JavaScript. Let's get started.

In JavaScript, the textual data is stored as strings. There is no separate type for a single character. A string is a sequence of characters enclosed in quotes, either single quotes, double quotes, or backticks.

Let's look at a few examples of strings in JavaScript. Let's go to the VS Code, and here I have a file letter, strings.js. Let's create strings with all the quotes. Let's say let str1, this is a variable and we can use double quotes so we can say, hello, world.

Now, this is a string. If you want, you can check using type of operator that we will do. Second, we can say let str2, and we can just type anything.

This time, you will note that I'm using a single quote. Let's say this is our string. Lastly, we can use backticks as well. We can say let str3, this is the third variable, and let's say 123, and this is now a string.

Of course, if you can consolidate, you will get the outputs on the console, but lets just do the type of them. Let's say console.log, typeof str1 then comma, and say typeof str2, and then we can say typeof str3.

Let's open up the terminal and let's run this program. I'll say node and string.js. You will see that we get the output as a string, string, and string.

All these are type of steps. Single and double quotes are essentially the same. Tactics, however, allow us to embed any expression into the string.

That means they have some extended functionality. Let's see that as well. Here I would say console.log, and here we can just use backticks.

We can say 1 plus 2 to be equal 2, and you can embed an expression using a dollar and curly bracket sign. We can say 1 plus 2 and it will evaluate the expression and then convert it into the string.

Now if I tried to run this program again, you will see that we get 1 plus 2 and we get the evaluated expression that is 3 in this case. You can find the length of a string using the dot length property. Let's see how length property works. Here we have a string and let's say we are taking a str2.

I recommend the rest of the code. Then I can say console.log, and it gives us str2.length. Many people get confused and they try to access this like this. This is not a function, this is not a method, it is a property. Property does not have parentheses. Now, if I try to run this program, you will see that we have the output as 16. The length of this is 16.

You can access individual characters in a string using the square bracket notation. Just like an edit to get a character, add some position, let's say pause. You can use square brackets or use some built-in methods as well. Let's say if we want to get or access the first letter of the string or the first character of the string.

You can say console.log.

We can see here str and let's say you want to get at index zero.

If I click on Save and if I try to run this program, you will see that we get the value as P. You can access more characters using their indexes.

One more important thing to note is that strings are immutable. That means strings can be changed in JavaScript, it is impossible to change a character. What I will do is we have absolute position the character as P. Let's do what? Let's say str2 at index zero, and let's give it H, and now let's run this program. You will see that it is T but if I console this again in the end, let's see what do we get as the output.

If I run it again, you will see that we get, this is a string only and you can see the character at index 0 has not changed.

It is still T. That means strings are immutable. Let's summarize this. In JavaScript we have strings which are used to store textual data. Strings are a sequence of characters and closing quotes, including single quotes, double quotes, or backticks. Backticks are all expressions to be embedded into a string using dollar and curly brackets. The link property of a string returns its length.

To access the characters at specific position, square brackets or built-in methods can be used. Strings are immutable in JavaScript and cannot be changed. It means it is possible not to change a character. The length of a string can be found using.length property and individual characters accessed using the square bracket notation.

In the next lesson, we will see string manipulation in JavaScript. Thank you.

* 1. **String Manipulation (6:34)**

Hi there, in the previous lesson we learned strings. Now in this lesson we will learn string manipulation in JavaScript. Let's get started.

String manipulation refers to the process of modifying or manipulating a string of characters in some way to achieve a desired output. A string in a sequence of characters enclosed in quotes such as, Hello World. String manipulation in JavaScript involves manipulating and transforming strings which are sequence of characters.

Let's look at some most common techniques for manipulating strings in JavaScript.

The first one is, concatenation. Concatenation is the process of joining two or more strings together to create a new string. In JavaScript you can concatenate strings using the plus operator or the concat() method. Let's look at the examples.

Let's go to the VS code, and here I have a file that is stringManipulation.js.

Let's create two strings, let str1, and this would be Hello. And let's create another string that would be let str2, and this would be World. And let's use these two operators and methods. The plus operator can be used to concatenate two or more strings. We can say let result and it would be equal to str1.

Then we can say +. And then we have to add a space, so let's add some empty space. Then again, this + operator and then str2. If I do console.log(result), we should get the two added strings.  Meanwhile, let's also perform the same thing using a concat() method.

The concat() method can be used to concatenate two or more strings together. So here we have this result, and let's say, Let result 2.

And in this case we will use a concat() method.  We'll say str1.concat.

And we want to add a space, and then we want to add str2. And let's also do console.log(result2). Now if I click on Save, let's open up the terminal and let's run this program. Would say node stringManipulation.js and you will see in both the cases we get the same output. Basically, you can use either a plus operator or a concat operator for adding the strings.

The second one is substring. A substring is a portion of a string that you can extract. In JavaScript, you can use either the slice method or substring method to get a substring from a string. Let's look at both the methods.

Here we are in the VS code again. And let's look at how to get a substring. First let's use a slice method.

Let's comment these as well. We can create a single string, let's say str, and let's give it Hello World.

And here what we want to do is, we can say let result and we want to get a portion of it. We can say str.slice. And here we can pass let's say 7 and 12 as the indexes.

The slice method extracts a section of a string and it returns a new string without modifying the original string. You can pass one or two arguments.

Here we are passing two arguments.

If you pass one argument, the slice method will return all the characters from that index to the end of the string.

If you pass two arguments, in this case, the slice method will return all the characters from the first index up to, but not including the second index.

Here if I do console.log(result), We should get the output as World. Let's run this program, and you will see we get World as well. Let's also look at a substring method. Same what we can do is, we can say let result2 and here we can say str2.substring. And here we can pass again, 7 and 12.

This substring method extracts a section of a string, and again, it returns a new string without modifying the original string. Here also you can pass one or two arguments. Here we are passing a two argument. This method will return all the characters from the first index up to but not including the second index.

Here also, if I do console.log(result2), we should get the same output.

Let's run this program, and you will see that we get world and world in both the cases.

Let's summarize this, in JavaScript string manipulation is the process of performing operations on strings such as, concatenating, extracting, substrings.

And there are many more like replacing characters and changing the case of the letters. We have seen the most used one, you can of course explore the other ones as well. Concatenation can be achieved using a plus operator or the concat() method that we have seen.

Substrings() method can be extracted using the slice() method or the substring() method. You can also explore replace() method to uppercase or lowercase methods for casing of the letters. Then you can also explore trim() method for removing the white space.

Note that, each of these methods returns a new string and does not modify the original one. This is all for this lesson, see you in the next lesson.

**1.10 Learn more about Variables, Operators, and Strings**

**Understanding variables and data types**

JavaScript is a dynamically typed (also called loosely typed) scripting language. That is, in JavaScript variables can receive different data types over time. Datatypes are basically typed data that can be used and manipulated in a program.

Variables in JavaScript are containers that hold reusable data. It is the basic unit of storage in a program.

[Read more](https://www.geeksforgeeks.org/variables-datatypes-javascript/)

**Using arithmetic and comparison operators**

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators. The JavaScript comparison operator compares the two operands.

[Read more](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Math)

**Working with strings and string manipulation**

Strings are useful for holding data that can be represented in text form. Some of the most-used operations on strings are to check their length, build and concatenate them using the + and += string operators, check for the existence or location of substrings with the indexOf() method, or extract substrings with the substring() method.

1. Control Structures & Functions (0:48)

Hi there. In this lesson, you will learn about control structures and functions in JavaScript.

Control structures allows you to make decisions and repeat code based on certain conditions.

You will learn how to use conditional statements like if else and switch, as well as looping structures like for and while to create more dynamic and flexible programs.

Additionally, you will learn about functions in JavaScript, which are a way to group a set of related statements and execute them whenever you want.

You will learn about the syntax of creating functions, the different types of functions, and how to call them.

We will also explore scope, which refers to the visibility of variables and functions in different parts of your code.

By the end of this lesson, you will have a strong foundation in control structures and functions, which are fundamental concepts in JavaScript, and essential to build more complex programs.

1. Conditional Statements (if/else, switch) (7:17)

Hi there. In this video, we will learn about conditional statements like if/else, and search in JavaScript. Let's get started.

Let's talk about if/else first.

The if/else statement allows you to execute different blocks of code based on certain conditions.

Let's look at the general syntax.

We have if then we have a condition, and then we have a block, and then we have else, and then another block that is curly brackets.

If the if condition is true, you can see that in the comment it has written code to be executed if the condition is true, otherwise, it will go to the else condition, if the condition is false.

You can think of a situation where you are deciding whether to go outside or stay indoors based on the weather.

If the weather is sunny, you will go outside, otherwise, you will stay indoors.

Let's try to create an example using if/else statements around this.

Here I'm in my VS code and here I have a basic index.html structure ready along with index.js file.

Let's say const weather this would be a variable that we want to hold the value as sunny, it is a string, and then we can say if weather = = to sunny.

In that case, we can say console.log.

It would be double quotes here.

We can say, let's go outside, and then we can say else.

Let's print out console.log.

Let's stay indoors.

Now if I click on "Save", you can see that the weather is sunny here.

If I can go to the inspect and let's go to console, let's see what we get as the output.

We're getting an error here, the unexpected end of input. Add this.

Here we have to close the else block as well, like this, and now if I click on "Save" here, you can see that we get, let's go outside because in this case, the weather is sunny.

If you change the weather to, let's say rainy or something, it will print, let's stay indoors.

In this example, if the weather variable is equal to sunny, the code inside the first block will execute and it will output let's go outside to the console.

Otherwise, the code inside the else block will execute and output would be, let's stay indoors.

Let's look at a new type of conditionals, that is switch statements.

The switch statement is useful when you have multiple conditions to check against a single variable. It provides a concise way to select one of many code blocks to execute.

Let's look at the syntax of switch case statements. You have a switch and then you have an expression. Based on different values or you can say cases, there are different codes that gets executed. You can imagine that you are at a restaurant.

I'm choosing a dish from a menu based on the number associated with it.

Let's create an example around this. Again, I'm in my VS code and let's comment this out. We can say const and we can give our dishNumber variable to be, let's say two. Now we can say switch on the basis of dishNumber. See dishNumber could be anything yet in our case we are taking it as two.

It could be any number or even any data type that you want. We can say if the case is one, that means if the dishNumber is one, we can say console or we can just print out, you ordered a burger something like that.

Then we have to break out from the switch case statement.

We will discuss about this in a minute. Next word we can do is we can just copy paste this, and we can create a second case. In this particular case, what we can do is we can just say, if the case is 2, we can say you ordered pizza. Similarly, we can create as many cases as we want to.

Let's have case 3, and in the third case, it would be a different order.

We can say case 3 here, and let's say you ordered pasta. Then we can have our case for where we can just print out or let's say the order would be you ordered something else.

If you don't want this case 4, in the last, you can have a default case if, neither of these cases are true.

That means if dishNumber is not even 1, 2, or 3 so default, it would be, let's say console.log, and we can put here a message like sorry, that dish is not available. As easy as that.

If I click on "Save", if we go here to the console, you can see that you ordered pizza is printed because that is the case. If I do case 1 here, you will see that it will print you ordered a burger. This is all switch case statement works.

In this example, when this number is two, the code inside the switch case second block will get executed and it will output you ordered pizza to the console.

This break statement is important, insert statements. It ensures that the code inside the matching case block executes and prevents it from falling through the subsequent cases.

For example, if I remove the break statement from here and from here as well, you will see that since case 1 and case 2, 3 do not have any break statements, it will run all the cases here after two.

That means you order pizza, you ordered pasta and the dish is not available.

This is not what we want. Always include break statements to break out of the switch statement. Let's summarize this.

Conditional statements in JavaScript, such as if/else and switch, allows you to execute different blocks of code based on specific conditions.

If/else statements helps you to make decisions while switch statements provide a concise way to select code blocks based on different cases.

By using these conditional statements, you can control the flow of your program and execute specific code paths depending on various conditions.

In the next lesson, we will learn about looping structures for and while.

1. Looping Structures (For, While, Do..While) (6:43)

Hi there.

In the previous lesson we learned conditional statements. Now in this video we will learn about looping structures and specifically about for, while, and do-while loops.

So let's get started. Loops are programming constructs that allows you to repeat a block of code multiple times. They provide a way to automate repetitive tasks and iterate over a collection of data.

Loops are essential in programming because they help in reducing code duplication, improving efficiency, and making programs more dynamic.

Let's look at some **loops**. For loop, the for loop is used when you want to repeat a block of code for a specific number of times.

It normally consists of three parts, initialization, where you initialize a variable that will control the loop. A condition, that is, the condition that determines whether the loop should continue executing or stop.

And the third is, increment and decrement. And that means after each iteration of the loop, the variable is modified according to the increment or decrement specified there.

Let's understand this for loop through an example. Let's go here. And into the VS code and we can say for. This is how we start write a for loop.

And then we can say let i, so this is the first part that is initializing.

The second part is a condition. Condition here we can put is i should be less than or equal to 5. And then we can put the increment decrement, that is the third part. So we can say i++. And then we can say console.log, just print out i. Let's see the output first.

As expected, you can see in the console the output is 12345. In this example, the loop starts with i set to 1. The condition i less than equal to 5 checks if i is less than or equal to 5. And if true, the code inside the loop is executed, which simply locks the value of i to the console. After each iteration, i is incremented by 1 because we are putting here increment, that is, i++. The loop continues until i become 6, at which point this condition will become false and the loop will exit.

Let's look at the second loop, that is, while loop.

The while loop is used when you want to repeat a block of code while a condition is true.

It only has a condition and as long as the condition remains true, the loop will continue executing.

Let's do the same example using while loop.

We can say while the condition is true.

What is the condition here? Condition i, i is less than equal to 5. But then where is i? We have to declare it. We can declare it, outside, we can say let i to be 1. And then inside it, we can say console.log (i). But here, there is a problem i has not been incremented. We have to do i++ after printing the value. And that's it, if I click on save, you can see that we get the same output.

In this example, we start again with i set to 1. The condition is again set to i less than equal to 5, and it is checked before each iteration. As long as i is less than or equal to 5, the code inside the loop is executed, which locks the value of i to the console. And after each iteration, i is incremented by 1, that is i++.

The loop again continues until i become 6 at which point the condition will become false and the loop exists. Let's look at the third type of loop, that is, do-while loop. The do-while loop is similar to the while loop, but it checks the condition after the executing the loop block. This means that the code inside the loop will always execute at least once.

First, let's do the same code using do-while loop, and then let's look at the difference as well. We can say let i to be 1. And then we can say do, then a block of code, and then we can print out the value of i like this.

And after that, we can say i++, that means incrementing the value of i. And then we can check the condition, let's say while, this condition is true. While i is less than equal to 5, that's it.

Now if I click on Save, you will see that we get the same output. But then what is the difference between while and do-while loop? While they have been created in a different way. To understand this, I will make the condition false. I will say do run this code while i is greater than 5.

And you can see that the value of i is 1.

And when we say i is greater than 5, if I click on Save, you will see that still it prints 1 and then it checks the code. And then it checks the condition that i is not greater than 5, it is false, then it does not go into the next iteration, but it will execute once even if the condition is false.

And with the case of while loop, it will definitely not print anything.

If we say I is greater than 5, you will see that it will not print anything.

That is the difference between while and do-while loop. Let's summarize this. Looping structures in JavaScript such as for, while, and do-while loops allow you to repeat a block of code multiple times based on a specific conditions.

The for loop is used when you know the number of iterations in advance.

While the while loop is suitable for indefinite iterations, the do-while loop ensures that the code block executes at least once.

These loops are essential for automating repetitive tasks for controlling the program flow. In the next video, we will understand about functions and scope.

1. Functions and Scope (7:50)

Hi there. In the previous video we learned about looping structures such as for while and do-while loop in JavaScript. Now in this video, we will learn about functions and scope. Let's get started. Let's talk about functions in JavaScript first. Functions are a block of code that perform a specific task. They allow you to group reusable code and execute it whenever needed. Functions can accept input values called parameters, and it can return output values.

This is the syntax of a function. Let's break down the different parts of the syntax. The first is this function keyword. This function keyword is used to declare a function in JavaScript. It tells the interpreter that you are defining a new function.

Then we have function name. This is the name you give to your function. Always choose a meaningful and descriptive name that reflects the purpose of the function. The name should follow JavaScript naming convention, for example, starting with the letter, or white spaces, or using special characters.

Then we have something called parameters. Parameters are placeholders for values that the function expects to receive when it is called. They are optional and you can have zero or more parameters. Parameters are separated by commas and are enclosed in parentheses. These parameters act as local variables within the function.

Then we have this block of code, that is the function body. The function body is enclosed within curly braces. It contains the code that defines the actions or computations the function will perform. This is where you write the instructions that makes up the functionality of a function.

Then we have an optional return statement. The return statement is optional and is used to specify the value that the function should return back to the caller. If the return statement is present, it immediately ends the function's execution and return the specified value. If no return statement is used, the function will return undefined by default.

Let's look at an example of function that adds two numbers. I'm in my VS code here. We can say function. Let's give the function name as addNumbers. We can pass parameters. They're optional. You can pass zero or more parameters, and then we can say return. It is also optional. We just want to return a+b or where this a+b will come from. To call or invoke the function, you can use its name followed by parentheses, and then passing the arguments. That means you can just call them like this inside the parentheses and we can pass the numbers as 10 and 20, 10 would be a and 20 would be b.

If I click on Save, we are calling the function. Let's go to Inspect and to the console, we do not get any output. This is because this function is only returning. To print the return value, you have to put it in console statement. If I click on Save, now you can see that you get the output as 30.

Here, the function addNumbers is called with 10 and 20 as the arguments. The result is stored in the variable and then you can log into the console. If you want to store this addNumbers, you can store it in a variable sum as well. We can say const sum to be like this, and then we can say console.log sum, it gives you the same output as you can see, 30.

Let's talk about scope in JavaScript. Scope refers to the visibility or accessibility of variables within a specific part of the code. It determines where variables can be accessed and where they are not accessible.

There are two types of scope in JavaScript. We have global and local scope. Let's talk about global scope first. Variables declared outside of any function are considered global and can be accessed from anywhere in the code.

Let's try to understand them with an example. Let's comment this piece of code and then we can create a variable, let's call it as global variable. This is not inside any function. That is why it is a global variable. We can just say, store a string here, I am a global variable, something like that. Now we can say function logGlobal, something like that.

Here, we can just say console.log, global variable.

If I click on Save, we need to call this function as well. That is logGlobal.

Now it does not accept any arguments, so we are just calling it like this.

If I click on Save, you can see that it prints

the output that I am a global variable. In this example, this global variable is defined outside of the function and can be accessed inside the logGlobal function and almost anywhere in the code.

Next we have is local scope. Variables declared inside a function are considered local and can only be accessed within that function. For example, let's create a new function here, and let's name it as logLocal.

Inside this, we can create variable, let's say const localVariable.

We can just hold a string here. Let's say I am a local variable because it is created inside our function. Then you can say console.log local variable.

Now if I call this function that is logLocal, let's see what do we get as the output. You can see that the variable is printed here. But if you try to access it outside of the function, it will give an error that local variable is not defined.

In this example, what we're doing is local variable is defined inside the logLocal function and can only be accessed within this function.

If you attempted to access it outside of the function, it will result in an error that we have seen already. Let's call it like this.

It's important to note that variables declared within a block using let or const also have block scope, meaning that they are accessible only within that block. Let's summarize this.

Functions in JavaScript allows you to group reusable code and perform specific tasks. They can exert parameters and return values.

Scope, on the other hand, determines the visibility and accessibility of variables within a specific part of the code.

Understanding functions and scope is crucial for writing modular, reusable, and well organized JavaScript code.

By using functions and understanding scope, you can create more structured programs and avoid naming conflicts between variables.

1. Further Reading on Control Statements

**Conditional statements (if/else, switch)**

In JavaScript we have the following conditional statements:

* Use if to specify a block of code to be executed, if a specified condition is true
* Use else to specify a block of code to be executed, if the same condition is false
* Use else if to specify a new condition to test, if the first condition is false
* Use switch to specify many alternative blocks of code to be executed
* The if Statement - Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

**Looping structures (for, while)**

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

* for loop
* while loop
* do-while loop
* for-in loop

**Functions and scope**

Another essential concept in coding is functions, which allow you to store a piece of code that does a single task inside a defined block, and then call that code whenever you need it using a single short command — rather than having to type out the same code multiple times.

1. Module 2: Manipulating the DOM
2. About the DOM (0:46)

Hi there. In this lesson, we will be exploring the Document Object Model and how to manipulate it with JavaScript. The DOM is a programming interface for web documents that represents the page so that programs can change the document structure, style and content. We will learn how to access and manipulate DOM elements using JavaScript, which can be a powerful tool in creating dynamic and interactive web pages. We will also cover modifying styles and attributes with JavaScript that will allow us to update the appearance and behavior of web pages on the fly.

By the end of this lesson, you will have a solid understanding of the DOM and how to use JavaScript to interact with it. Which will provide you with the necessary skills to create more advanced and interactive web pages.

1. Understanding the Document Object Model (DOM) (3:51)

Hi, there. In this video we will learn DOM and HTML. Let's get started. DOM stands for Document Object Model. The Document Object Model is a programming interface for web document. It represents the HTML or XML document as a tree-like structure where each node in the tree corresponds to a different part of the document.

The dominance for web developers to interact with the contents of a webpage using JavaScript. It allows them to add, remove, or modify elements dynamically.

You can imagine a webpage as a tree where each branch represents an element in the webpage, such as a paragraph or an image.

Each leaf represents a property or attribute of that element, such as a text inside the paragraph or the source URL of the image.

This is essentially what DOM is. JavaScript can manipulate the DOM by selecting nodes and changing their properties or adding and removing nodes altogether.

For example, you could use JavaScript to change the texts inside a paragraph tag or add a new paragraph or image to a page, remove an element from the page entirely, or maybe change the attributes of an elements such as its size, color, or position.

This is how tree structure would look like. This tree diagram represents the hierarchical structure of the DOM for a given webpage. Each node in the tree corresponds to an HTML element or a piece of text within that element.

The nodes are arranged in a parent-child relationship with parent nodes higher up in the tree and child nodes branching off from their parent nodes below. For example, document is the parent tree of this HTML.

Html is a child of document. In this particular diagram, the root node is the document object, which represents the entire webpage.

The first level of the child nodes corresponds to the top-level HTML elements, such as HTML then we have head, then we have body as well.

Each of these nodes have one or more child nodes depending on the structure of the webpage. Head has further child nodes, body has further child nodes. For example, the body node might have child nodes corresponding to a header section, a main content area, and a footer section.

Each of these child nodes might have further child nodes, such as paragraphs, images, or other HTML's. For example here we have H1 in the body, and this one has a further child node of x that is welcome to my page.

This tree diagram is useful for visualizing the structure of a webpage and understanding how different elements are related to one another.

By traversing the tree and selecting different nodes, developers can use JavaScript to modify the contents of the webpage and create dynamic and interactive effects. Let's summarize this.

The Document Object Model is a programming interface for web documents that represents that HTML or XML document in a tree-like structure of nodes.

Each node in the tree corresponds to a different part of the document, and JavaScript can manipulate the DOM by selecting nodes and changing their properties or adding and removing nodes altogether.

The DOM is a powerful tool for web development and it allows developers to create dynamic interactive webpages.

In the next video, we will learn how to access and actually manipulate the DOMs with JavaScript.

1. Accessing DOM elements with JavaScript (7:21)

Hi there, in the previous lesson we learned DOM in JavaScript.

Now in this video we will see how to access DOM using JavaScript so let's get started. Accessing DOM elements with JavaScript is a key skill for web development.

As we know that the Document Object Model is a tree-like structure that represents the HTML elements of a web page. You can use JavaScript to access these elements in various ways. To get started you will need to know how to select an element.

Let's look at some commonly used DOM selectors in JavaScript. First is document.getElementById. It returns the element with the specific ID let's see how we can use that. Let's go to the VS code and here I have index.html file.

I can delete this one and let's create a very basic boilerplate using this shortcut and you can see we have a title document let's make it DOM.

And in the body we can say h1 of DOM Selectors.

If I click on save you can see that we have the output on the web page. We are talking about document.getElementById. Here we can have a div and let's say hello world.

And what we can do is we can create one more div and we can just say hi there. Now the first div we will just give it an ID of let's say hello.

We want to access this div out of all these elements we want to access the first div of ID hello.

Here I will have a script tag and I will say document or we can say console.log(document.getElementById) and we can pass the ID name here. The ID is hello we have to pass it in quotes. And now if I click on save let's go to inspect, let's go to console and you can see that we have got this div. It's cool, right?

We have got this on the basis of ID. The next one is document.getElementsByClassName. The little difference you will see that it is get elements earlier one was get element.

We can only get one element out of it but here we can get multiple elements by class name. It returns a collection of all the elements with the specified class name. For example, if I create an unordered list and inside that I have a li of class, let's say green.

And we can say item one and we can create one more li or let's say four lis out of which only two has the class of green. We can remove these options here. If I click on save and if I rather than doing this, what I can do is I can say let items, I can also put them in a variable.

I can say document.getElementsByClassName and we can pass the class name as green and then we can say console.log items.

If I click on save if I go here, if I refresh you will see that we get an HTML collection.

It is an array like structure and we get two Li items that has the class of green color. Let's go to the next one. Next one is document.getElementsByTagName.

Here also we can get multiple collections or you can say it returns a collection of all the elements with the specified tag name. We have multiple items li items here. What we can do is let's just copy paste this instead of document.getElementsByClassName we can say get elements by tag name.

Here the tag name is li. We have to pass it like this. If I click on save if I go here let's refresh, you can see that we are getting all the li items.

So this is now a collection of four elements.

Next one is document.querySelector. It returns the first element that matches the specified CSS selector. For example, we have here li and two class names are green. What I will do is I will say let item, let's comment this out. And we will use query selector here and I will say I want to get the item that is a class of green.

We have to add a CSS selector as well that is dot for class. Now you may notice that two of the li classes have class green. And if I do console.log item it returns only the first li element that has a text of item one.

It always returns the first element that matches the specified CSS selector.

But then if you want to get all the classes or you can say all the li items that has a class of green.

One way was to use get elements by class name. The other method is that is the last one that is document.querySelectorAll. Here what we can do is we can say query selector all and we are passing the same green class here. And now if I do items, if I click on save you will see that it returns a collection of all the items that match the specified CSS selector.

So let's summarize this.

We know that the Document Object Model is a tree-like structure that represents the HTML elements of a web page.

JavaScript provides various methods to access these elements such as getElementById, getElementsByClassName, getElementsByTagName, querySelector and querySelectorAll.

Once you have a reference to an element you can modify its attributes, content and style or create new elements using other methods that we would be looking at in a separate video.

With these tools you can create dynamic and interactive web pages that responds to user input and update in real time.

In the next lesson we will see how to manipulate DOM using JavaScript.

1. Modifying styles and attributes with JS (6:17)

Hi there. In the previous lesson, we learned how to manipulate DOM elements with JavaScript.

Now in this video we will learn how to modify styles and attributes using its own methods.

Let's get started. Modifying styles and attributes with JavaScript allows you to change the appearance and behavior of elements on a webpage dynamically.

If you want to modify the styles, you can use style property of an element to access and modify its CSS styles, and to modify the attributes you can use set attribute method to add or modify attributes on an element.

Let's understand all these methods and properties through some examples. I'm here in my VS Code, and here we have an index or HTML template ready with a script tag. We can start writing our code.

Here, in the body I want to create a div, inside this div, I will have a span tag.

Let's give this ID of one and let's give a name attribute of, let's say post. Then we can just write, it is a first spend, this is some text. Similarly, I want to create one more span and we can give it an ID of two. Let's give it a second and second span.

Now we need to write some JavaScript code. First we can select all the elements using DOM selectors. I'll say const live and we can see here a document.querySelector and you can pass the div. Then we can say const spanOne and you can make it equal to document.getElementById and we can pass the ID name as one.

Similarly, we can do it for two as well. We can say spanTwo and this would be two. Now let's start manipulating these tiles and attributes.

Let's talk about attributes first. If you want to get the attribute, you can say console.log and let's say if you want to get the attribute of spanOne so we can say spanOne.get attribute.

It is a method, it will take the attributes, we want to get the value of attribute name.

If I click on Save, if I go here, refresh, we get the output as first because the name has the value of forced at this point.

Let's say we want to change the name.

That means setting the attribute.

We can say spanOne.setAttribute.

This is the method, we can put the attribute name, let's say name and the value and the second argument.

Let's say a, b, c, d, e, if I click on Save, you will see nothing here but if you want to see the actual thing that happened, you can go to the elements section to the body and here inside that div, we will see that now the name has changed from first to a, b, c, d, e. If you want to remove the attribute, you can just see spanOne.removeAttribute and we can just remove the attribute of name.

If I click on "Save," you will see in this span, we only have the ID and name has been removed.

A very short feature is also there that you can just say spanOne.id and rather than using set attribute, you can directly give it an id name of, let's say any text.

If I click on "Save," if I go here, you will see that in the body now seems to this unique word, or letter, you can name it anything.

You can also change classes using class methods.

Let's see if I target spanTwo and I can just say spanTwo.classList.add and I can add the class, let's say a new class.

If I click on "Save," you will see that in the body under the span div ID two v gate a class that is new class.

If you want to remove, you can just say remove, like this and you can pause the class name that you want to remove.

You can actually manipulate the styles using a style property.

Let's do that as well.

We can say spanOne.style.color and let's give it red.

If I click on "Save", you will see that the text color has changed to red.

Similarly, if you want to change the background, you can say spanTwo.style.backgroundColor, you can put any property here and let's give it green.

If I click on Save, you will see that the background of the second span has changed to green.

Let's summarize this.

The style property can be used to access and modify CSS styles of an element and the classes property can add or remove classes to apply styles.

To modify attributes, the second attribute method can add or change attributes while the get attribute method can retrieve the attribute values.

Lastly, we have also seen remove attribute method through which you can remove an attribute from an element.

These capabilities allow developers to create interactive webpages that can respond to user input and change in real-time.

In the next video, we will learn about events and event listeners in JavaScript.

1. Read more about DOM Manipulation

**Understanding the Document Object Model (DOM)**

The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document.

The W3C DOM standard is separated into 3 different parts:

* Core DOM - standard model for all document types
* XML DOM - standard model for XML documents
* HTML DOM - standard model for HTML documents

[Read more](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction)

**Accessing and manipulating DOM elements with JavaScript**

When writing web pages and apps, one of the most common things you'll want to do is manipulate the document structure in some way. This is usually done by using the Document Object Model (DOM), a set of APIs for controlling HTML and styling information that makes heavy use of the Document object

[Read more](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents)

**Modifying styles and attributes with JavaScript**

The HTML DOM allows JavaScript to change the style of HTML elements.

To change the style of an HTML element, use this syntax:

**document.getElementById(id).style.property = new style**

[Read more](https://www.w3schools.com/js/js_htmldom_css.asp)

1. **Event Handling – What you will learn (0:47)**

Hi there.

In this lesson, you will learn about event handling in JavaScript.

Event refers to actions or occurrences that happen in the browser, such as user clicking a button or typing in the form.

Understanding how to handle events is crucial for creating dynamic and interactive web pages.

You will learn about event listeners and how to use them to respond to user input.

You will also learn about the common events such as click, mouse hover, and keypress, and how to use them to create interactive user interfaces.

Additionally, you will learn how to handle errors and exceptions that may arise in your code.

By the end of this lesson, you will have a solid understanding of how to handle events in JavaScript, and you will be able to create dynamic and engaging web pages that responds to user input.

1. Understanding Events & Event Listeners (3:49)

Hi there.

In the previous lesson, we learned how to modify styles and attributes with JavaScript. Now, in this lesson, we will understand events and event listeners. Let's get started. Events in JavaScript refers to actions or occurrences that happen on a web page, such as a mouse click, a keyboard press, or a page load.

JavaScript provides a way to listen for these events and this point to them through event listeners. Events are important in JavaScript because they allow us to create dynamic and interactive web pages.

That means, without events, web pages would be static and unresponsive.

Events provide a way for users to interact with a web page and for web pages to react to user input.

For example, you can consider a web page with a button, in that, when a user clicks the button, an event is triggered.

JavaScript can listen for this event and respond by changing the text on the page or performing some other action.

Another example is, when a user types something in an input field, an event is triggered.

JavaScript can listen for this event and respond by validating the input or performing some other action.

To listen for events in JavaScript, we use event listeners.

Event listeners are functions that are called when a particular event is triggered.

We can attach event listeners to the DOM elements using Add Event Listener method.

When the event occurs, the event listener function is called, and we can perform some action in response to the event.

Let's look at some common examples of events and how we might use them in JavaScript.

First is, click Events.

So these are triggered when the user clicks an element on the page, such as a button or a link.

We can use click events to perform actions like showing or hiding content, submitting a form, or navigating to a different page.

Input events, these are triggered when the user types into an input field or makes a selection in a drop down menu.

We can use input events to validate user input or update the page in real time.

Then we have load events, these are triggered when the page finishes loading.

We can use load events to perform actions that depend on the page's content being fully loaded, such as fetching data from an API, or initializing a plugin.

Then we have, mouseover events, these are triggered when the user hovers over an element on the page, such as an image or a link.

We can use mouseover events to add visual effects like toolstips or hover states.

These are just a few examples of the many types of events we can use in JavaScript.

By using event listeners, we can make our web pages more interactive and responsive to user actions.

So let's summarize this, events in JavaScript refer to actions or occurrences that happen on a web page, such as mouse click, a keyboard press, or a page load.

Events are important in JavaScript because they allows us to create dynamic and interactive web pages.

To listen for events in JavaScript, we use event listeners.

Event listeners are functions that are called when a particular event is triggered.

We can attach event listeners to DOM elements using the Add Event Listener method.

Overall, events are fundamental part of creating interactive web pages with JavaScript that allows us to create dynamic and responsive web experience for the user.

This is all for this video, in the next video, we will understand how to respond to a user input such as click or keypresses events.

1. Responding to User inputs (clicks, key presses) (6:16)

Hi there, in the previous lesson we learned events and event listeners in JavaScript.

Now in this lesson we will learn how to respond to a user input.

Let's get started. When building interactive applications or websites, it's important to be able to respond to user input such as clicks or key presses to provide a better user experience.

This involves using event listeners to detect when a user interacts with the application and executing specific code in response to those events.

Let's look at some examples.

First we have Click events. You can use JavaScript to add event listeners to HTML elements and detect when they are clicked. For example, you could have a button that when clicked, triggers a popup window to appear.

Let's also see one example in VS code.

Here I am in my VS code and I have an index or HTML page ready with minimum HTML template with a script tag as well. First let's create a button, and we can give here an ID of, let's say, my button, and then we can give it, let's say, a text of click me.

Next we can go inside the script tag and we can write some JavaScript. If I click on save first, we will have a button here.

What I want is currently you can see nothing is happening. I want to attach a click event listener on this button.

What I will do is I will first target the button in the script tag. I'll say const button, you can use any selector you want. I will use query selector, so document.queryselector, and we can pass here the ID of my button like this. Next we can attach event listener, so we can say button.addeventListener. It takes two arguments.

First, the type of the event that is click in our case, and a callback function.

This callback function runs as soon as it triggers this particular event. At this point I just want to alert, let's say, hello world. If I click on save, if I go here and let's click on Click Me, and you can now see that a pop up comes.

Why? Because it triggers this click event and this callback function runs that just alerts and print, hello world. In this example we are using the Add event listener function to attach a click event, and then we are passing a callback function that gets executed when the event is triggered.

Second is key press events. You can also detect when a user presses a key on their keyboard and execute code in response.

For example, you could have a search box that automatically updates search results as the user types. Let's create that example as well. For that, first of all, we need to have an input field.

Let's delete all this and let's create an input field. To this input, I'll just give it an ID of, let's say, search box, and a type of text.

Now in the script tag, we can start writing our JavaScript code.

Before that you can see the input has already come here on the screen.

We can see here const, search box, and this variable will hold the element, so we can say document.queryselector. And we can pass here the ID as search box.

Next, what we can do is we can just attach an event listener. This time the event listener would be key up. The event is key up, and on that, that function will run. And into this function, you can just write, code to update search results that would go here. Or for now, we can say console log.event triggered.

If I click on save, let's go to inspect, then to console. And here you will see as soon as I press a key, a, event is triggered. As soon as I print s, I have not released the key as of now, as soon as I release the event triggers second time. Whenever user release the key, this callback function will run.

Here in the add event listener function is used to attach a key up event to the search box input element.

Whenever the user releases a key while typing in the search box, the anonymous function passed as the second argument is executed, which could include the code to update search results.

Let's summarize this. Responding to user inputs such as clicks and key presses is an essential part of building interactive applications or websites.

This involves using event listeners to detect when a user interacts with the application, and executing specific code in response to those events.

The examples of how to respond to user input including click events to trigger a pop up window or maybe using keypress events to update search results as the user types.

To implement these functionalities, HTML elements are used to trigger the events and JavaScript code is used to attach the event listeners and execute the desired functionality.

In the next lesson we will create interactive user interfaces with events.

1. Creating interactive user interfaces with events (4:59)

Hi there. In the previous lesson, we learned how to respond to user input, such as clicks and key pressed event. Now in this video, we will create an interactive user interface using events. Let's get started. Creating interactive user interfaces with events is a key part of building engaging with applications.

Events are actions that happen in the browser, such as a user clicking a button or scrolling the page. They can be used to trigger specific functionality in your application.

In this context, events are used to create interactive user interfaces that respond to user actions in real time. Let's create an interactive interface. I'm here in my VS Code and I have a basic HTML setup ready along with script tag. Let's say that you are building a to-do list application and you want to allow users to mark items as completed by clicking on the item.

Let's see how we can do that. First in the body. Let's have an H1 and we can just create our to-do list. At this point, there should be coming in the webpage. Next we can have some unordered list. Inside this, we can have list items. Let's say we want to buy groceries.

You can see we want to do some laundry work.

Or we can have one mode, LI item, let's say clean house. We have a to-do list, and we have these three items in our to-do list. Next, we will go to our script tag and we can start writing of a JavaScript tool. Here I want to target all of the list items.

I can use different methods like class name or query selector. I will say cost list items. Here I will use a document.querySelectorAll. You can pass the tax letter here. It will target all of the list items. Then we can run a loop. I can say ListItem.forEach. For each item it takes a callback function.

We want to do something. On each item what we want to do is we want to attach an event listener. I can say item.addEventListener. The type of event would be click. Then we want to run a callback function.

Here we will say item.classList.toggle. We will add a class that is completed. Toggle means just like us on and off button. If the class is there it will remove it. If the class is not there, it will add it. We are talking about a completed class. We have to create this class.

What we can do is below the title, we can have a style tag and we can say.completed. We can say text decoration to be line through.

We can give it a color of gray as well. If I click on "Save" Let's test it out.

If I go here, if I click on, let's say buy groceries, it cuts. It has a line through here as well, here as well. When we click on again, since it already has the class, it will remove it.

This is how you can create interactive applications using JavaScript. In this example, the HTML page contains an unordered list with three to-do items. When a user clicks on the to-do item an event listener is triggered that toggles a CSS class on the item, causing it to be displayed with a line through and a gray color.

The CSS class is defined here in the head of the HTML using the style tag.

Let's summarize this. Using events is a crucial part of creating interactive user interfaces in web applications, events can be used to detect user actions, such as clicks and key presses, and trigger specific functionality in response.

By using event listeners to handle user input, web developers can create engaging and dynamic applications that provide a seamless user experience.

We have seen some examples of how events can be used to create interactive features in real-world applications. Such as to-do list app.

1. Further reading on event handling

**Understanding events and event listeners**

Javascript has events to provide a dynamic interface to a webpage. These events are hooked to elements in the Document Object Model(DOM).

These events by default use bubbling propagation i.e, upwards in the DOM from children to parent. We can bind events either as inline or in an external script.

[Read more](https://www.w3schools.com/css/css3_text_effects.asp)

**Responding to user input (clicks, key presses)**

Event handlers can be used to handle and verify user input, user actions, and browser actions:

* Things that should be done every time a page loads
* Things that should be done when the page is closed
* Action that should be performed when a user clicks a button
* Content that should be verified when a user inputs data

[Read more](https://www.w3schools.com/js/js_events.asp)

**Responding to user input (clicks, key presses)**

CSS is the mechanism for adding style to various web documents. Text Effects allows us to apply different types of effects on the text used in an HTML document.

[Read more](https://www.w3schools.com/css/css3_text_effects.asp)

**Creating interactive user interfaces with events**

The UI Events API defines a system for handling user interactions such as mouse and keyboard input. This includes:

events that are fired on specific user actions such keypresses or mouse clicks. Most of these events fire on the Element interface, but the events relating to loading and unloading resources fire on the Window interface.

[Read more](https://developer.mozilla.org/en-US/docs/Web/API/UI_Events)

1. Advanced DOM Manipulation (what you will learn) (0:59)

Hi there.

In this lesson, we will dive deeper into manipulating the Document Object Model with JavaScript.

You will learn how to create and remove DOM element dynamically using JavaScript.

This will enable you to build more interactive and dynamic web applications that can respond to user input and change their content dynamically.

You will also learn about AJAX, a powerful technique that allows you to load content dynamically from a server without refreshing the entire page.

This is a crucial skill for building fast and responsive Web applications that can update their content without interrupting the user's experience.

Finally, you will learn about handling errors and exception in JavaScript.

This is an important skill for writing robust and error free code, as it allows you to detect and handle unexpected errors that might occur during runtime.

You will learn how to use try-catch blocks to catch and handle errors, as well as best practices for debugging and testing your code.

1. Creating and removing DOM elements with JS (6:36)

Hi there. In this lesson, we will learn how to create and remove DOM elements with JavaScript in retail. Let's get started.

Creating and removing DOM elements with JavaScript is a fundamental technique using web development to dynamically modify the content of web pages.

JavaScript provides a variety of methods for creating and manipulating DOM elements, including createElement, appendChild, append, remove, removeChild, and so on.

Let's understand this using an example.

Let's go to the VS Code and here I have a basic HTML template ready.

We would also need to have a script tag so that we can write our JavaScript code.

Let's create a basic template first.

Let's say we are creating an H1 and let's say we're listing some favorite foods.

My favorite foods.

Then we can have an unordered list and we can give you idea of food list.

Here we can have list items.

Let's see, we have pizza and we have, let's say sushi as well.

Noteworthy will do is we will create two buttons.

Let's give this an ID of addButton.

As the ID use suggests, we will add items to it so we can say add Tacos.

Similarly, we can create one more button and we can see here removeButton.

As the name suggests, it removes.

We can say at this point, remove sushi.

Now we can start writing our JavaScript code.

If I click on Save, you will see that all of the operators coming here on the webpage, and let's manipulate it, create and remove some items from it.

Let's select the elements using DOM selectors.

I'll say const addButton.

We can make it equal to document.querySelector and we can just put here ID, so ID is #addButton.

Similarly, what we can do is we can target our removeButton as well.

We can just copy-paste at this point and similarly, we can target our food list.

Let's say const foodList and it would be again equal to document.querySelector.

We can pass the ID here like this.

Now what we want to do is we want to attach event listener.

I can say addButton.addEventListener and let's have an event of type click and then we can run a callback function.

Inside this function, we will say const newListItem.

Here we can say document.createElement and we want to create a new li element like this.

Then let's say const newListItemText and you can create it using a method that is document.create, and that is TextNode.

We want to pass it let's say, Tacos.

Then what we want to do is after that, we can say newListItem.append, or you can use append child and we want to say newListItemText.

Then what you want to do is we want to see foodList.appendChild, this time let's use this and you want to append newListItem.

Now if I click on Save, if we try to run this, you will see that when we say Add Tacos, it adds it to the list based on the event.

Similarly, let's create it for removeButton as well.

Let's put here removeButton and that is again a type of click and you want to do some things here.

We can say const sushi to be document.querySelector and we can say target li, that has an nth child of two.

Then what we can do is we can either use removedChild or remove.

Let's use removedChild at this point, we can say removeChild and you want to remove the sushi.

Now if I click on Save,

let's test both of them.

I will refresh. It adds tacos and it removes the sushi.

In this example, there are two buttons.

One that adds a new list item containing the text tacos to the unordered list, and one that removes the second item containing the text sushi from the unordered list.

The event listeners for both buttons use querySelector method to access the appropriate DOM elements.

Then we use appendChild or append, or you can say remove a removeChild method to create or remove elements from the DOM respectively.

Let's summarize this.

In web development, creating and removing DOM elements with JavaScript is a fundamental technique used to dynamically modify the content of web pages.

By using the methods like createElement, appendChild, removeChild, web developers can create dynamic and responsive web pages that provide a rich and engaging user experience for the users.

In the next lesson, we will see how to use Ajax to load content dynamically.

1. Using AJAX to load content dynamically (7:03)

Hi there. In the previous video, we learned how to create a new move DOM elements using JavaScript.

Now in this video, we will learn how to use AJAX to load content dynamically. Let's get started.

AJAX or Asynchronous JavaScript and XML is a technique used in web development to load content dynamically without requiring a full page refresh.

This technique allows webpages to load new content or data from the server in the background without interrupting the user's current interaction with the page.

To use AJAX in a webpage, we typically use the XMLHttpRequest object in JavaScript to send an HTTP request to the server and then process the response using JavaScript.

We can use this technique to load content from the server and display it on the webpage without having to reload the entire page.

Let's look at an example of how to use

AJAX to load content dynamically from a server.

Let's go to the VS Code, and here I have a basic HTML file ready.

Let's start creating a basic template here.

Let's say I wanted H1 and in this H1, you can just give it a heading off, let's say users.

Then we can just have an unordered list.

Let's give it an ID of user list.

Let's leave it empty for now.

Next, what we will do is we need to write JavaScript now.

We'll go inside the script tag and we can capture this user list using DOM selector.

Lets say const userList equals to document.querySelector.

We can pass the ID, that is hash userList.

Then you can say const xhr.

We can create this new XML request object.

It is XMLHttpRequest and we can create an object like this.

Next what we want to do is we can say xhr and we need to have an event that is on ready state change.

On this, we need to run a function.

Let's use a function here.

We can say if xhr.readyState is equal to XML httprequest.done, in that case, we want to have one more if condition.

Here we can say if xhr.status is 200, that means we got the response successfully.

We can say const, userData.

At this point, we can parse it. We can parse what?

We can parse the response text that we're getting from xhr object.

We can say responseText here.

Next, what we want to do is we can say userData and we want to loop through it.

We can say for each.

Here we can get a single user and then you want to run this callback function for every single user.

Then you want to say const, newUser.

You want to create an LI element for it.

We can say document.create element.

The element that you wanted to create is LI.

After that, we can say const, newUser let's say text.

This would be equal to document.CreateTextNode.

The node we want to create is coming from user that is named.

Let's use the name as the text that we want to append.

Now we can see newUser.appendChild, and let's append newUserText.

Then we can say userList.appendChild, and let's open a newUser.

This is for the if condition.

What if you want to have an else condition as well?

After this, we can just say else.

We can say console.log.

We can say there was a problem with the request, something like that, an error message.

After that, what we want to do is we want to make the request so we can say xhr.open.

We want to make a request of Typekit and you want to pass the URL.

The URL that we would be using is from JSON typi code.

You can search for jsontypicode.com.

Here we want the endpoint as /users.

Here you can see we will get a list of users.

Let's copy it and let's paste it here.

Then what we want to do is we just need to say xhr.send that will send the request.

If I click on "Save", we should get a user list here in the webpage.

This means it is working perfectly fine.

In this example, we use AJAX to load a list of users from the JSON Placeholder API and we are displaying them on the webpage.

We start by using XMLHttpRequest object to create a new HttpRequest and then we define a callback function to handle the response when it is received from the server.

Now, when the response is received, we check the status of the response to make sure that it was successfull.

Then we parse it using JSON.parse the response data into a JavaScript object.

Finally, we loop through them and dynamically create LI elements for each user, which are then appended to the userList that is UL on the webpage.

Let's summarize this.

Overall, using AJAX to load content dynamically from APIs is a powerful technique that can greatly enhance the user experience of the pages by allowing them to load new content in the background without requiring a full page refresh.

In the next lesson we will see how to handle errors and exceptions in JavaScript.

1. Handling Errors and Exceptions (5:53)

Hi there. In the previous lesson, we learned how to use Ajax to load content dynamically.

Now in this video, we will learn how to handle errors and exceptions in JavaScript. Let's get started.

Handling errors and exception is an important part of writing robust and reliable JavaScript code.

In JavaScript, we have try and catch keywords that are used together to handle errors and exceptions in the code.

Here is the basic syntax of try and catch block.

The try block contains the code that may potentially throw an error or exception.

If an error occurs, JavaScript will jump out of the try block and into the catch block.

The catch block contains code that will handle the error or exception such as logging an error message or taking corrective action.

Within the catch block, you will also notice this error variable.

This error variable is used to represent the error object that was thrown.

This variable can be used to access information about the error, such as its message, stack trace, and any additional data that may be attached to it. Let's understand this through an example. Let's go to the VS Code.

Here I have a file that is a JavaScript file handlingErrors.js. Let's start writing a code at this point. Let's create a function divide and this function will take a numerator and you can see denominator as the parameter.

Then what you want to do is, we can add a check here. We can say if denominator is equal, equal to zero, in that case, division is not possible.

What we want to do is we want to throw new error. This will throw a new object. We can pass a message here. We can say division by zero.

You can pass any error related to it. At this point, it seems more appropriate. Then what we want to do is we can say return numerator divided by denominator. This is the function. Then you want to call a function. But before calling the function, we will use try and catch to check for any errors.

We can say try this piece of this block of code. We can say const result, and result would be a part of, let's say divide. Whatever divide this returning, you can show it in result. We will pass here 10, 0 because we want to get an error. Then we can say console.log result.

As you can see, it is throwing some error that catcher finally expected.

So it is always important to use catch method at this point.

We can say catch error.

We can see console.log error.

It is an object and we want to put a message at this point.

We can say error.message.

Let's try to run this program.

I've clicked on "Save" and let's open up the terminal.

Let's clear it and let's run node handlingErrors.js.

You can see we get the message here that is division by zero.

Also right here something like this to actually give more idea about that this is an error rather than a normal message.

If I try to run this again, you can see it says error division by zero.

In this example we define a function divide that performs a division operation on two numbers, that is numerator and denominator.

Before performing the division, we checked to make sure that the denominator is not zero.

If it is, we throw a new error object with a custom error message that is error division by zero in this case.

We then call the divide function with arguments 10 and 0 within a try block.

Since this will result in a divide-by-zero error, JavaScript will jump out of the try block into the catch block.

Within the catch block, we log the error message to the console using console.log error.message.

That's all we are getting the output here.

Let's summarize this.

In JavaScript, try and catch are used together to handle errors and exceptions in the code.

The try block contains the code that may potentially throw an error or exception, and the catch block contains code that will handle the error or exception.

Within the catch block, the error variable is used to represent the error object that was thrown.

This variable can be used to access information about the error such as its message or maybe stack trace and any additional data that may be attached to it.

Using try-catch blocks can help us write more reliable and maintainable JavaScript code by gracefully handling errors and exceptions.

1. Read more on Advanced DOM Manipulation

**Creating and removing DOM elements with JavaScript**

In an HTML document, the document.createElement() method creates the HTML element specified by tagName, or an HTMLUnknownElement if tagName isn't recognized

The Element.remove() method removes the element from the DOM.

[Read more for Creating Element](https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement)

[Read more for Removing Element](https://developer.mozilla.org/en-US/docs/Web/API/Element/remove)

**Using AJAX to load content dynamically**

Ajax is not a technology, but rather a programming concept. HTML and CSS can be used in combination to mark up and style information. The webpage can be modified by JavaScript to dynamically display—and allow the user to interact with the new information. The built-in XMLHttpRequest object is used to execute Ajax on webpages, allowing websites to load content onto the screen without refreshing the page. Ajax is not a new technology, nor is it a new language. Instead, it is existing technologies used in a new way.

[Read more](https://phppot.com/jquery/dynamic-content-load-using-jquery-ajax/)

**Handling errors and exceptions**

There are three types of errors in programming: (a) Syntax Errors, (b) Runtime Errors, and (c) Logical Errors.

[Read more](https://www.tutorialspoint.com/javascript/javascript_error_handling.htm)

Module 3: What you will learn (0:50)

Hi there.

In this lesson, we will delve into more advanced Java script techniques.

You will learn how to work with arrays and objects in Java script and understand how to manipulate their properties and methods.

Additionally, you will learn about closures and how they can be used to create private variables and functions.

Furthermore, we will explore the scope chain, which is the set of variables that are available to a function at a given point in the code.

This will help you understand how variables are scoped and how to avoid conflicts when writing complex applications.

Finally, we will discuss debugging techniques in Java script.

You will learn about how to use browser developing tools to diagnose and fix errors in your code.

This is an important skill to have when building complex Java script applications.

Working with Arrays and Objects in JavaScript (8:18)

Hi, there. In this lesson, we will learn about arrays and objects in JavaScript. Let's get started.

Let's first talk about arrays in JavaScript. An array is a data structure in JavaScript that allows you to store multiple values in a single variable. It can hold values of any data type such as: numbers, strings, Booleans, or even other arrays. Arrays are zero-indexed. That means the first element is at Index 0, the second would be at Index 1, and then so on.

Let's understand working with arrays with the help of an example. Here, I'm in my VS Code and I have a basic index.html page, linked with index.js file. Let's first start by creating an array.

We can say let fruits, and we can create an array using the square brackets and we can put multiple items here, let's say array of strings.

We can say apple.

Let's put banana, and then we can put orange as well.

This is how you can create an array.

If you do console.log fruits, you will get array into the console.

Let's go to Inspect and console.

You can see that we have array with three strings, that is apple, banana, and orange and you can see the index starts from zero.

Apple's index is 0, banana is 1, and orange is 2.

You can access the element based on the index.

If I say fruits at Index 0, guess the output, the output would be apple here.

If I do, let's say 2, it would be orange, as you can see here.

If you want to modify an array element, let's say we can do something like fruits add index 1 and I just want to put, let's say, grape here.

Let's console it again. I can say console.log fruits.

At this point, you will see that at Index 1, we have grape now, if you want to check a length of the array, you can do it by using a.length property like this, it will give you the length of the array that is three in this case.

You can do a lot of things using array methods and properties.

We're just seeing the important ones and the commonly used.

Let's also see how we can add elements to this array.

We can say fruits.

There is a method called push that push the elements into the end of the array.

Let's say we want to push mango and if I do console.log fruits again, you will see that the mango has been added at Index 3.

If you want to pop or remove the last element, you can use a method that is pop.

We can say,.pop mango or you don't have to mention here anything because it will directly remove the last element and if I console log groups again, you will see that first it had mango and then when we use pop, it popped off from the array.

Of course, there are a couple of more methods that you can use, but these are the common ones.

You can also iterate the array using a for loop.

We can say for let i to be zero, we want to iterate through fruits.length and then we can say i++.

In this case, if I do console.log fruits at index I, you will see that it has iterated over all of the strings and printed in the console.

Now let's talk about objects in JavaScript.

Objects in JavaScript are more complex data types that allows you to store key-value pairs.

They can not represent real-world entities or concepts and provide a way to organize related data and functionality together.

Objects have properties that is called as keys and corresponding values, which can be of any data type.

Let's now try to understand objects using an example.

Let me remove this code and let's create a person object.

Just like we create an array using a square bracket, we can create an object using curly brackets.

Then we can create a key.

Let's say the name of the person is John.

Then it can have any number of attributes, let's say age to be 25.

Then we can say is a student so that would be true in this case.

Again, it can hold any data type.

As you can see, we're holding string, number, and Boolean, although you can have areas and objects here as well. How to access the properties?

You can just say console.log and we can say person.name.

This is one notation that is dot notation and when we say person.name, you will get John printed here. There is one more way to access the properties using a square bracket notation. We can just say square bracket here and we can pass the key age.

If I write this syntax, you will see we get the age here as well in the console. Let's say we want to modify the properties here so we can see person.age to be 30. Then we can say console.log person.age. Earlier it was 25 now you can see the age has been changed to 30.

You can add more properties to it.

Let's say if I want to add a city of the person, let's say person.city to be, let's say, New York.

Now if I do something like console.log person, you will see that last console brings all the properties, including new property, that is city as well.

If you want to iterate over the properties of an object you can use a for in loop.

You can say for let key in the person object.

It will iterate to all of the keys and then we can say print or console.log p plus, let say some colon. Then you can say plus person and square bracket key.

If I click on "Save," you can see that key will give you name, age isStudent, and city and this gives you the values, and in-between we have colons here.

This is how you can iterate over the properties. In this example, an object person is created with properties like name, age, and isStudent.

Then we have access using dot notation and back-end notation as well and you can also modify existing properties by assigning new values and add new properties to an object dynamically. Let's summarize this.

Arrays and objects are powerful data structures in JavaScript that enable you to organize, manipulate, and represent complex data.

Understanding how to work with arrays and objects is crucial for handling data. In the next lesson, we will understand about closures and scope chain in JavaScript.

Understanding Closures and the Scope Chain (11:42)

Hi there. In the previous lesson, we learned about arrays and objects in JavaScript.

Now in this lesson we will understand closures and scope chain.

Let's get started. Before diving into closures, let's first understand what is scope. Scope in JavaScript refers to the visibility and accessibility of variables, functions, and objects in some particular part of your code during runtime.

JavaScript has function scope which means that variables defined within a function are only accessible within that function.

This concept helps with variable encapsulation and prevents naming conflicts. Let's understand the scope using an example. Let's go to the VS code I have index HTML and index JS file linked to each other, and let's create a function called great.

In this function, let's create a message using const and here we can put let's say Hello at this point. If I do something like console log message, and let's call great here, if I click on save, let's go to inspect and to console.

You can see that we get the output as Hello. But if after the greet I try to log this message, you will see that error.

That message is not defined. In this example the message variable is defined within the greet function. It is accessible and can be used within this function only. However, if you try to access this outside of the function like this, you will receive an error because the variable is not defined in that scope.

Now we can move to the part of closures. Closure is a combination of a function and the lexical environment within which that function was declared. It allows a function to access variables from its outer scope, even after the outer function has finished executing. This is a very important concept, so let's try to understand this using an example.

Here what we can do is, we can create a function and let's give the name as outer. Now into this function, we can create a variable and let's call it as outer variable. And I can just put up a value here or a string here that I am from outer, something like that. And then inside this, we can create a function inner.

And here we can say console.log outer variable. And then after this function we can return this inner function at this point. Now we can say const closureFunction.

And let's put outer and call the outer function here. Outer function is called here and it will return inner function into this closure function variable, and then what we can do is, we can call this closure function like this.

If I click on Save you will see that we get the output I am from outer. In this example, the outer function defines a variable called outer variable here. It also declares an inner function within it, the inner function has access to the outer variable even though it is declared inside the outer function.

When outer function is invoked and assigned to the closure function variable, it returns the inner function. Even though at this point the outer function has finished executing, closure function still retains access to the outer variable through something called closure.

When closure function is called, it locks the value of outer variable at this point. If you want to test this out, you can also go to sources to index.js and where we are consoling it here.

Let's put a debugger at this point and click on Refresh. Now you can see that this particular function forms a closure with the outer function, and that's how it is able to access the outer variable even though the outer function was finished executing. Closures are useful for preserving data privacy and creating private variables.

They allow you to create functions that can access and manipulate specific data, while keeping that data inaccessible from outside of the function. Now let's move to another important topic, that is scope chain.

The scope chain is the hierarchy of scopes in which variables and functions are accessible. It is created during the lexical analysis phase of code execution.

When a variable or function is referenced, JavaScript searches for it in the current scope. If it doesn't find it, it continues searching in the outer scope, and this process continues until the variable or function is found or until the global scope is reached.

Let's understand the scope chain using an example. We can create a function again with the name outer, and inside this, we can say const outervariable, and let's put the value as outer string here.

And inside this we can create a function inner, and then we can say const innervariable, and let's put the value as inner. And then we can say console.log innervariable and we can put some space here, and then we can say plus then outervariable. And after this we can call our inner function, and after this we can call our outer function.

Let's see the output first. If we go here, let's remove the debugger and let's refresh it again, you can see that we get the output as inner and outer. In this example, the inner function has access to both the inner variable and the outer variable. This is possible because the inner function is within the scope of the outer function. It can access variables in both its scope and outer scope as well.

Let's summarize this.

Understanding closures and the scope chain is crucial for writing JavaScript code that effectively uses variable encapsulation, maintains data privacy, and leverages the flexibility of different scopes. With these concepts you can create powerful and modular JavaScript applications.

In the next lesson we will understand about debugging in JavaScript.

Debugging JavaScript Code (7:51)

Hi, there.

In the previous lesson, we learned about closures and the scope chain in JavaScript. Now in this lesson, we will understand how to debug JavaScript code. Let's get started.

Debugging JavaScript code is an important skill for developers to identify and fix errors or issues in their code. Let's look at some debugging concepts and techniques in JavaScript. Identifying errors, so when you encounter an error in your JavaScript code, the browser's console is a useful tool for identifying and understanding the error.

The console provides information about the error type, the line number where the error occurred, and any accompanying error messages.

You can just console these statements. Let's go to the VS Code, and here I have index.js and index.html file linked together, and you can directly use a console.log statement, and you can log anything that would be printed in the console.

If I click on Save, I have already opened index.html with live server.

Let's go to Inspect and Console, and you can see the console statements here.

When running this code, the browser's console will display an error message, indicating a syntax error and the line number where the error occurred if there is any error.

For example, if I do something like this, you can see that it displays a type error at this point.

Next is to use console statements.

Console statements are helpful for debugging code by printing values or messages to the console.

They actually allow you to see the state of variables or the flow of execution at specific points in your code.

For example, let's create a function and let's call it as multiply. This would take parameters a,b, and then we can say onsole.log('Multiplying', a, "and", b);. Then, we can say return a \* b;.

After that, let's create a result variable, And let's invoke the function multiply(5,2,); and let's also say console.log('Result: ', result);.

In this example, the console.log statement is used to display messages and the values of the variables during the execution of the multiply function.

By inspecting the console output, you can verify if the function is receiving the correct values and producing the expected result.

If I click on Save, you will see that we are getting Multiplying 5 and 2, and the result is 10.

Next technique is to set breakpoints. Setting breakpoints in your code allows you to pause the execution at specific lines enabling you to examine the state of variables and step through the code line by line.

Most modern web browsers provide developer tools with built-in debugging features including setting breakpoints.

For this particular case, let's modify the example and we can say let result = a \* b; and then we can say return the result;.

Next, we are just calling this multiply function storing the result in the result variable and then logging the result again.

But just before returning the result, I will just add debugger key here, and this will pause the code.

In this example, this debugger statement is added to pause the execution at that line. When you open this browser tools now, you can see the code has been stopped and it allows us to inspect the variables in step to the code and analyze the program's flow. You can see that at this point, a is 5, b is 2, and the result is 10.

Last but not the least, we can read error messages. When encountering an error, JavaScript provides error messages that can offer clues about the issue. Understanding and interpreting these error messages can help you identify the problem and fix it.

Let's try to create an error, so what we can do here is we can say function divide(a, b). Then, we can say if(b === 0) {throw new Error('Cannot divide by zero' ); as this is not possible. And after this if this condition is false, we can just return A / b;.

Next, we can say try and into this try block, we can say let result = divide(10,0);.

And let's create an error, so we can say console.log('Result: ', result);.

And we can also have a catch, so it will catch for any errors, and then we can say console.log('Error: ',error.message).

In this particular example, the divide function throws an error if the second argument b is 0.

When running this code, an error is thrown, so let's check that out.

Let's refresh, and you can see in the console, we have error cannot divide by zero, and here the catch block is executed.

The error message can be accessed using this error.message property that allows you to display custom error messages or handle errors gracefully, so let's summarize this.

By understanding these debugging techniques and concepts, beginners can effectively identify and resolve issues in their JavaScript code.

Debugging plays a vital role in development process and helps improve the functionality and quality of your JavaScript applications.

Learn more about Advance JS Techniques

**Working with arrays and objects in JavaScript**

Arrays are generally described as "list-like objects"; they are basically single objects that contain multiple values stored in a list. Array objects can be stored in variables and dealt with in much the same way as any other type of value, the difference being that we can access each value inside the list individually, and do super useful and efficient things with the list, like loop through it and do the same thing to every value.

[Read more](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Arrays)

**Understanding closures and the scope chain**

Scope is the context environment (also known as lexical environment) created when a function is written. This context defines what other data it has access to.

Closures are the best form of privacy for functions and variables. This is evident in the use of many module patterns. A module pattern returns an object to expose a public API. It also keeps other methods and variables private. Closures are used in event handling and callbacks.

[Read more](https://www.freecodecamp.org/news/deep-dive-into-scope-chains-and-closures-21ee18b71dd9/)

**Debugging JavaScript code**

Refer to the link below for a step-by-step approach.

[Read more](https://developer.chrome.com/docs/devtools/javascript/)

Intro to jQuery – What you will learn (1:18)

Hi there. In this lesson, you will learn about basics of jQuery, a popular JavaScript library that makes it easier to work with HTML documents.

We will start with introduction to jQuery, where you will learn what jQuery is and why it is useful for web development.

You will also learn how to include jQuery in your web pages and how to use it to manipulate the DOM.

Next, we will cover jQuery selectors, which allows you to select and manipulate HTML elements in your web pages.

You will learn how to use CSS style selectors to find and select elements in the DOM and how to manipulate their attributes and properties using jQuery.

After that, we will dive into jQuery effects, which allows you to animate and add visual effects to your web pages.

You will learn how to use jQuery to create smooth animations, feed elements in and out, and change the visibility of the elements in the page.

Finally, we will explore jQuery traversal, which allows you to move up and down the DOM tree to find and select elements.

You will learn how to use jQuery to traverse the DOM and select elements based on their relationships to other elements on the page.

By the end of this lesson, you will have a good understanding of the basics of jQuery and how to use it to create interactive and dynamic web pages.

What is jQuery? (5:01)

Hi there. In this lesson we will learn about what is jQuery. Let's get started.

jQuery is a popular, fast and lightweight JavaScript library that is designed to simplify client side web development.

It provides a range of features and functionalities that makes it easier to manipulate HTML elements, handle events, perform animations, make AJAX request, and much more.

jQuery is built on top of JavaScript and offers a simplified syntax and cross-browser compatibility, making it widely used in web development.

Let's try to consider an analogy to understand jQuery.

Imagine that you have a toolbox filled with various tools that help you perform specific task.

Each tool serves a different purpose and make your work easier.

In this analogy, JavaScript is like a general purpose tool, while jQuery is a specialized tool that focuses on specific tasks related to web development.

Just as you would reach for a specific tool in your toolbox to perform a specific task efficiently, you can read for jQuery to simplify and streamline common web development task, such as selecting and manipulating HTML elements, handling events, and make AJAX requests.

Another thing you can imagine is that you are visiting a foreign country where you don't speak the native language.

However, you have a translator who can help you communicate effectively.

In this anthology, JavaScript is like the native language and jQuery acts as the translator that simplifies and bridges the gap between JavaScript and HTML.

Just as the translator helps you understand and communicate with the locals in the language, jQuery act as a translator between JavaScript and HTML, making it easier for developers to write code that interacts with and manipulates HTML elements.

Let's look at some key features of jQuery, DOM manipulation.

jQuery simplifies the process of selecting and manipulating HTML elements in the document object model, often called as DOM.

It provides a concise syntax for accessing and modifying elements such as changing their content, attributes, or CSS styles.

With jQuery, you can easily create dynamic and interactive web pages.

Even handling: jQuery makes it easier to handle browser events such as clicks, key presses, mouse movements, and form submissions.

It provides methods to attach event handlers to elements, and execute code when that particular event occurs.

This allows you to enhance user interactivity and respond to user actions effectively.

Animations and effects: jQuery offers a wide range of animations and effects that can be applied to HTML elements.

You can create smooth transitions, fields, slides, and other visually appealing effects using simple jQuery methods.

This enables you to add engaging in emissions to your web pages without complex JavaScript coding.

AJAX: jQuery simplifies the process of making asynchronous requests to the server and handling the responses without requiring a page reload.

It provides a set of AJAX methods makes it easier to fetch data from a server, send data, and update parts of a webpage dynamically.

AJAX allows for seamless and interactive user experiences.

Cross-browser compatibility: One of jQuery's significant advantages is its ability to handle cross-browser compatibility issues.

It provides a consistent API that works across different web browsers, reducing the need to write browser specific code.

This ensures that your web application functions reliably and consistently across various platforms.

Last but not the least, plugin integration.

jQuery has a vast ecosystem of plugins contributed by the community.

These plugins extend jQuery's functionality and allows you to incorporate additional features and behaviors into your web projects.

You can find plugins for things like sliders, carousels, form validation, date pickers, and much more. Let's summarize this.

Overall jQuery is a versatile library that simplifies many common tasks in web development.

Its application span from basic DOM manipulation and event handling to more advanced features like AJAX request animation effects.

By using jQuery, developers can write concise, cross-browser compatible code and create rich and interactive web experiences with less effort.

In the next lesson, we will understand about selectors.

Selectors (7:39)

Hi there. In the previous video we learned about what is jQuery.

Now in this video we will understand jQuery selectors, so let's get started.

jQuery selectors are a powerful feature that allows you to target and select HTML elements in the DOM.

They provide a concise and efficient way to identify elements based on various criteria, such as element types, classes, IDs, attributes, and more.

Let's look at some commonly used jQuery selectors.

We have element selector, so it selects elements based on their tag name.

You can use tag names like div, p, h1, etc, to select specific elements.

Then we have a class selector where you can select the elements based on any class.

Then we have an ID selector you can select the element with its unique ID and then we have attribute selectors, so you can select an element based on its attribute and you can also combine all these selectors and pick up a single element.

Let's try to understand all of them using examples.

Let's go to the VS Code, and here I have index.html file ready where I have a bunch of HTML tags and this is the output as of now.

We have h1, two p tags with a class of highlight, some div with class of boxes, two links, then again headings; h2s and h3s, then input with attributes, then we have buttons and a form.

Now we have to select them based on jQuery selectors.

Let's start by simple element selector.

If I want to select let's say a p tag, so we're using a p tag here and what we can do is we can say dollar and we can pass a p tag here.

We can say dot text and we can change the text to, let's say this text is changed using jQuery, something like that.

Now if I click on "Save", you will see that both of the p tags text is changed.

This is cool, this is what is called as an element selector where you pick up the elements based on their tag names.

Here we have picked up p. If you want to pick h1, you can say h1, h2, and h3. The problem here is that it will pick all the elements, all the p tags from the particular HTML page.

We want to create or we want to select a unique element based either on the class or an ID, or you can say attribute.

Let's look at those examples as well.

Let's say if I want to select elements with the class of highlight, so we can say dollar and we can pass the name of the class in quotes, let's say ".highlight".

Then we can say.css and you want to change the color to let's say red.

Now if I click on "Save" you will see that the color of this particular paragraph or the particular element with the class of highlight, the text color has been changed and you can see this p tag only has the class of highlight.

Let's select the element based on the ID.

What we can do is let's say dollar and the ID is my elements.

Just like you say.class, you say #ID.

#MyElement and then we can say.hide.

Let's see if we have an element by MyElement ID, so we don't have it.

What we can do is we can give here this p tag an ID of my element and let's click on "Save".

Let's go here. You can see that the second paragraph or that element that had the ID of my element has been hidden.

Let's look at attribute selector, so you can select elements based on their attributes and attribute values. Let's try that.

What we can do here is we can say dollar and we want to target the input element, and the attribute of that would come in the square brackets, so let's say required.

Now you can see that we have three inputs here out of which first and third has the required attribute.

We want to pick them and we want to add a new class and that class would be let's say required-field.

Now if I click on "Save" you will see that they have this required field added here.

This required-field we have already created here, so it has just the border of 2px solid and red.

You can see that only those required; that is one and third, have been given the borders.

This is what is called as an attribute selector.

Next we can look at combination of selectors, or you can say in easier words, combining selectors.

You can combine multiple selectors to target elements that meet specific criteria.

For example, you can combine class and elements selectors or use multiple selectors separated by comma.

We have here anchor tag.

Here, let's do what? Let's target them.

We can say dollar and let's put here a.nav-link.

What is this nav-link?

This nav-link is the class.

We are combining here the tag selector and the class selector, and then we can say.addClass and let's add an active class.

If I click on "Save" you will see in both the elements we now have the active class.

An active class is created here or it's not created, so we can say.active and let's give it a background of green.

If I click on "Save", you can see that we have navigation links and let's target that again.

We have.active here, it should be ve.

Now you can see that both have been given the background color.

These are just few examples of jQuery selectors, there are many more selectors options available such as parent-child relationships, sibling relationships, form related selectors, and much more.

jQuery selector syntax is designed to closely reassemble CSS selectors, making it easy to grasp and use for selecting elements.

In the next video, we will learn about jQuery effects.

jQuery Effects (6:31)

Hi there. In the previous video, we learned about jQuery selectors.

Now in this video we will learn jQuery effects. Let's get started.

jQuery effects allows you to add visual transitions and animations to HTML elements.

Effects can make your webpages more engaging and interactive by providing smooth transitions, fades, slides, and other animations.

Let's understand this through an example.

Let's go to the VS Code and I have index.html file we created here.

Let's have a basic template and let's say intitled to jQuery effects and let's have an h1 for now of let's say effect.

If I click on Save, we should get effect as the output here.

Let's change it to effects example.

You have what the structure would be is, let's have a button and to this button, let's give a text of a toggle box.

We will create a toggle example here and then you can have a div.

Let's give this an id of myBox and we can also give it a class of box like this and that's it.

Let's leave it empty. Now let's take this box and here we can have a style tag, so let's say dot box, and we can give it a width of, let's say, 200 pixels and a height of 200 pixels as well.

Let's also give it a background color of, let's say, light blue and let's also give it a display of none.

Before that, let's see the output first.

We should get a square here, lightbox.

Now what we want to do is initially we want it to be hidden.

We can say display of none.

Now if I click on Save, you will see that it is hidden for now.

After this style tag, let's also use a script tag and we will include the jQuery CDN, that is content delivery network.

You can search it on Google.

Let me just type it here.

It is https://code.jquery.com and then we can say slash

jQuery and then you can put the version.

I will use 3.6.0.

You can use any version here and the name would be.min.js. That's it.

Let's test this out if it is working or not.

We can have a script tag just below here.

Now in this script tag, let's start creating an effect.

First of all, we will say document.ready, and then we can have a function inside it.

Let's create a function. Here, we will add our toggle functionality.

You can say toggle the visibility of the element with a fade in and fade out effect on the button click. Which button? This button.

We can see dollar and I want to target my button like this using the selector and then we can add even listener that is click an onclick.

I just want to run a function again.

Here we can target our div of id myBox.

We have to pass here hash and name of the id that is my box and then we can use an effect that is fadeToggle and that's it.

Let's test this out.

If I click on Save, if I go here, we have toggle and boxes coming and then boxes going on the other clip.

This is a fadeToggle example that we have created.

Make sure that you include necessarily jQuery library along with the script tag.

Inside the script tag, this code is wrapped in document.ready function, ensuring that jQuery code executes when the document, in this case, HTML page, has finished loading.

Within this document.ready function, we haven't event handler that is attached with a button using this.click.

This means that when the button is clicked, the function provided as the argument will be executed.

Now inside the click event handler, the fadeToggle method is used.

This method toggles the visibility of the element with the ID myBox by feeding it in if it's hidden or feeding it out if it's visible.

jQuery effects allows you to add visual transitions and animations to HTML elements.

Effects can be used to show or hide elements, change dimensions, feed elements in or out, and much more.

jQuery provides a range of built-in methods for applying effects, such as show, hide, toggle, fade-in, fade-out, fadeToggle, slide up, slide down, slide toggle and animated.

Effects can enhance the user experience and make webpages more engaging and provide smooth transitions between different states of elements.

By utilizing jQuery effects, you can easily add interactivity and visual appeal to your web applications without writing complex animations from scratch.

In summary, jQuery effects are for a simple and a powerful way to add dynamic and interactive elements. We have webpages.

They provide an extensive set of methods that can be used to create eye-catching transitions and animations with ease.

This is all for this video.

In the next video, we will learn about jQuery diversing.

See you in the next video. Thank you.

jQuery Traversing (5:33)

Hi there. In the previous video, we learned about jQuery effects.

Now in this video we will understand about jQuery traversing. Let's get started.

JQuery provides powerful methods for traversing the DOM, that is document object model, allowing you to select and manipulate specific elements based on their relationship to other elements.

Traversing methods in jQuery help you navigate through the DOM tree and select elements that match certain criteria.

Let's understand this using an example.

Let's go to the VS Code.

Here I have a index.html file ready with a script tag that has a CDN of jQuery attached.

In this script tag, we can start writing our jQuery code.

Before that, let's create a basic UI.

Here, let's have our unordered list, and let's have multiple list items.

Let's say Item 1, and then we can say Item 2, and then let's have Item 3 as well.

All of them would be listed here.

Now let's start writing our jQuery code.

First I will say dollar and document, and then I would say.ready.

When the document is ready, I just want to run this function.

Inside this function, we want to filter the list items and select the ones containing two.

We can say, var filteredItems, let's create a variable here, and we would target LI items based on this selector.

Then you can say.filter.

Now, this is the traversing method, and here we can call a function.

This function, we'll just return.

Let's say this.text,.includes, and it should include this two here.

Then what we can do is then we can apply a CSS class to the filtered list item.

What we can do here is we can see filteredItems.addClass, and we can say, highlight, at this point.

Let's also create a highlight part here in the style tag, and we can say.highlight, let's just give it a background color of light blue or let's say blue.

If I click on "Save," if I go here, you can see that this particular element that includes two is being given the background color of blue.

This filter method is applied to this list items to filter and select only the list items that meet a specific criteria.

Inside this filter method, we are providing a filtering function, as you can see here, that returns true or false for each list item.

In this case, the function checks if the text of each list item includes the string 2, using the includes method.

If it does, the function returns true, indicating that the list item should be included in the filtered set.

The filter list items are then stored in filter items variable here.

Then we can perform any operations on the filtered elements.

In this example, we apply a CSS class named highlight to the filter list items using the add class method.

The highlight class adds a blue background color to the list item through the defined CSS.

Let's summarize this.

JQuery traversing methods provide a convenient way to select and navigate through elements in the document object model.

Traversing allows you to target specific elements based on their relationships to other elements or their properties, making it easier to manipulate or interact with them.

There are many traversing methods like parent, children, siblings, and filter that we have used, now that allows you to select elements based on their position in the DOM or specific criteria.

The parent method selects a direct parent element of the master element, while children selects the direct children elements.

These methods help you navigate up or down the DOM tree.

The siblings method selects all sibling elements of the matched element.

This allows you to target the elements that share the same parent.

The filter method lets you select a subset of elements from a larger set based on a specific criteria.

In our case, it was the text that has 2 in it.

You can use filtering functions to evaluate elements based on their properties, attributes or contents.

JQuery traversing methods provides flexibility and power when working with complex DOM structures, allowing you to precisely target and manipulate elements as needed.

By utilizing these traversing methods, you can easily navigate through the DOM and select the desired elements for further manipulation.

Traversing helps you write concise and efficient jQuery code, making it an essential skill for working with complex web applications.

Further Reading on jQuery

**What is Jquery?**

jQuery is a fast, small, cross-platform and feature-rich JavaScript library. It is designed to simplify the client-side scripting of HTML. It makes things like HTML document traversal and manipulation, animation, event handling, and AJAX very simple with an easy-to-use API that works on a lot of different type of browsers.

The main purpose of jQuery is to provide an easy way to use JavaScript on your website to make it more interactive and attractive. It is also used to add animation.

[Read more](https://www.javatpoint.com/what-is-jquery)

**Selectors**

jQuery Selectors are used to select and manipulate HTML elements. They are very important part of jQuery library.

[Read more](https://www.javatpoint.com/jquery-selectors)

**Effects**

The jQuery library provides several techniques for adding animation to a web page. These include simple, standard animations that are frequently used, and the ability to craft sophisticated custom effects.

[Read more](https://api.jquery.com/category/effects/)

**Traversing**

In jQuery, traversing means moving through or over the HTML elements to find, filter or select a particular or entire element.

[Read more](https://www.geeksforgeeks.org/jquery-traversing/)

Project Building, Part 1 (5:35)

Hi there.

In this video we will talk about how to create the project structure and how to get started in creating a Todo app.

Let's get started. Let's look at the problem statement first. You are tasked with developing a ToDo application. The app should allow users to add and delete tasks.

It should provide a user-friendly interface for adding tasks, displaying the them in a list and allowing users to delete the task.

The app should handle user input and create task items dynamically and update the list accordingly.

Key points here are that the main functionality that we have to add is add and delete task and the list should be added dynamically.

Let's look at the step-by-step approach of solving this problem statement.

First is HTML structure.

We need to create the HTML structure for the ToDo app including a form to add new task and a container to display the tasks.

Then we can add necessary HTML elements such as input fields, buttons, containers with appropriate classes or IDs for styling and manipulation.

Next we can start with JavaScript functionality.

You can write JavaScript functions to handle the core functionality of the ToDo app can create a function to capture user input from the form and add new task to the task list.

Then you can implement a function to delete task when the user interacts with the corresponding buttons. You need to ensure that the app updates the task list dynamically when a task is added or deleted. That is one of the core points in the problem statement.

Then we can add some event listeners. You can attach event listeners to the appropriate HTML elements, for example form submission or button clicks to trigger the corresponding JavaScript functions.

Need to ensure that the app responds to user interaction and updates the UI accordingly. With the problem statement and step by step approach, let's start building our ToDo application. Let's move to the Vs code and here let's create three files.

I have index HTML file, a script JS file and let's also create a style CSS dot file. Now first let's start with the structure. What we can do is, we can just create a basic HTML template here and let's give the title as todo app and then we can use a link tag to link our external style sheet. We can say relation of style sheet and let's say type of text CSS and an href of style.CSS.

Next in the body tag we can have an h1 for now let's say todo app and here we can have a script tag.

Let's give it a defer attribute so it will load asynchronously and let's also use source and pass the path that is script.JS. If I click on save now we should see todo app here. Now let's first create the structure. Here we want a div and let's give this a class of container.

Let's have a h1 or heading of todo app And then let's have a form. Let's give this an ID of task form.

And in this form we want an input where user will type the task and we can give it a type of text and ID of let's say task input.

We can also give it a placeholder, let's say add a new task and then we can have a button of type submit and then we can say here just a text of add task. If I click on save you will see this as the output. We have a basic to do heading input and a button and once the user clicks on the button we should be able to show a list. For that after the form we can have an ordered list and let's give it an ID of task list like this and it should be ul.

Through this we have completed the first approach that is creating our HTML structure. And this would be empty because the list would be generated dynamically, right, the task would be generated dynamically whenever user types in the input.

In the next video we will try to add functionality and before that we will add some styles to it.

Project Building, Part 2 (10:46)

Hi there, in the previous video we created the HTML structure of our Todo app. Now in this video we will add styles to it and also some functionality to the app. Let's move back to the VS code and this was our HTML structure. Let's open this side by side and style it.

First that I want to style is this class container. Let's give here a max width of, let's say 400 pixels and then we can say margin of 20 pixels and auto from left and right.

And then let's give it a background color of #fff. Then let's give a padding of 20 pixels from all sites. And let's also give it a box shadow of 0px, 2px and 5px. And the color that we want would be rgba and let's give it a black shade. We can say 0,0,0 and the opacity would be 0.1. If I click on save you can see that we have a light box shadow here.

Now let's target our h1, so we can say h1 and we just want to align it to center.

It aligns to the center now. And then we can target our form. And to this form we can just give it a margin bottom of let's say 10px.

And then we can target the input of type text like this.

And in this case, we just want to give it a width of, let's say 70%.

And let's give it a padding from all sides of 5px and let's give it a font size of 16px as well. If I click on Save we have this as the output. Now let's target this button. We can say button and let's give it a padding of let's say from top and bottom 5px, and from left and right 10px.

Let's give it a background color of a green shade. That would be 4caf 50 and then let's give it a color of fff. And then let's give it a border of none and let's also give it a cursor to be pointer. If I click on Save, this is our button and it looks cooler.

Next we can target our unordered list as well since there is nothing here, we can just say list, style, type of none and a padding of 0.

Since there is nothing here, we will not be able to see any output at this point, but it will be visible as soon as we start dynamically add using JavaScript.

The styling part is complete. Now let's go to the script tag and let's create the functionality. In the script tag first I want to target the st elements, or we can say get the html elements.

We only want the elements that we want to work with.

Let's say if we want to get the form, we can say const taskForm and we have a method for it. We can say document get elementbyid and we can say get the id of taskForm like this.

And then we can say const taskInput.

To this input we can say document.get element by id and the id is taskInput.

Let's also check if we have given the ids or not. To the input we have given id of taskInput and taskForm, so this is correct. Otherwise you can do console and check if you are getting the element in the variable and then in the UI you can say UL, we have given it an id of task list.

We can say const taskList to be document.getelementByid and we can say taskList here. We have targeted all the elements. Now we will create a task counter.

Let's say let task id to be 1 and now we can write a logic or a function to add a new task.

Let's create this function. We can say function addTask and we will get an event here because this will be passed as a callback to the event listener. And then we can say event prevent default.

This will prevent the page from refreshing. Now, side by side, let's also add the event listener. We can say here add event listener to the form. What we can do here is we can say taskForm.addEventListener and we want to add a type of event that is submit. And then we can say the function that we have to run is addTask.

This is the task. What happens is if I do not use this particular line, if I comment it out, you would see that if I click on Add Task the page is reloading, right? Although it is fast, you can see here it is reloading.

We don't want to remove the data because the data will be lost when you refresh the page. In this case we can say e prevent default or even prevent default. Now you can see the page has stopped refreshing.

Now we can write our logic here.

We can target the text from the input first. We can say const taskText and this would be coming from taskInput.value and we can trim for any spaces.

Next we can say if this is one of the edge cases, we can say if taskText is not equal to empty string, only then we want to move further, right? Because we do not want to add an empty value. In that case we can do what, we can create a taskItem and that is nothing but ally item, right?

We can say document.createElement and the element that we want to create is li.

Next we want to add a text and some button to it so we can say taskItem.innerhtml and we can use backticks to add a span. Let's add a span tag and inside this span tag we will add the text whatever the user has typed in the input and that is taskText. And then we can close our span tag and then we can also add a button for delete. We can say button of class. Let's give it a class of delete-button and then we can say delete here as a text and let's close this button tag.

After creating a new li element, what we can do is we can say taskList and this is our UI and we want to append taskItem to it.

And then we can say taskInput.value to be an empty string.

This is because once we add it to the list right, we do not want to show that value to the user on the input.

We are removing it and then we can do something like taskId ++, so it will increment the taskId.

And then what we can do is we can add an event to the delete button.

How? We can say const deleteButton or we can say btn to be taskItem querySelector and we can select the element with the class of delete button. We can say .delete-btn and then we can say delete button.addEventListener and we can add a type of click. And let's also run a new function that is deleteTask.

Now let's create a function called deleteTask and that's it. We have written the code for add task function and we have added the event as well.

This is all for this video, in the next video we will test this function out and you will complete our application and then we can also discuss about how you can enhance this application.

Project Building, Part 3 (6:03)

Hi there, in the previous videos we have created the HTML structure and added the functionality for add a task.

Now in this lesson we will complete the app and make it working.

Let's go to the VS code and here we already created the function add task. Let's test this out. If I click on save, let's go here and add a new task.

Let's say music. And as soon as I click on add, you can see that nothing is working, it is not getting added. And we are getting an error here. That means cannot read properties of null, and we have trim here. Okay, this is not area value, this is just value.

Now if I click on save, let's put music and add. And you can see it gets added here along with the delete button. And if you want to check it you can say the text gets removed inside the input.

If I add, let's say, study and click on add task, you can see it also gets deleted from the input.

Now we want to make this delete functionality working.

We have already created or attached an event listener here and created a delete task function.

We will again get an event from here, event object. And we can say alert, clicked, just to check if the event is working or not on the click of the delete button.

If I go here, let's add a and b. If I click on delete, you can see that it is clicked and it is clicked as well. It is working fine. Now what we can do is we can define or write a logic here to delete the task. We can say const taskItem, and we can say event.target.

First of all, let's console this as well. Let's see what do we get in the task item.

If I click on save, let's add a and b again. If I want to click on delete, you can see that we get button of delete button class here. Now we don't want to delete only the button, right? We want to delete the whole task.

In that case, we want to get its parent element. When I say parent element, e.target.parentElement, and if I click on save, let's say a, add it again, and let's click on delete. You can see that now the whole li element gets targeted here. And now we want to remove it from the unordered list. What we can do here is we can just say, taskList, that is the unordered list,dotremovechild, and we can remove this task item.

And if I click on save, you would see that if I do add a and b, and if I want to delete a, it gets deleted and only b is left. We have created the Working ToDo app, and we have added the functionalities like Add and Delete, based on the problem statement.

Let's also discuss how you can enhance your application. So you can add functionality like to do task completion. So you can allow users to mark tasks as completed by adding a checkbox or toggle button next to each task, right?

For example, you can go here and you can add a checkbox for each task.

And when a task is marked as completed, you can apply a different style to indicate its completion.

Next you can add is task editing. You can implement the ability for users to edit task names, and this can be done by allowing users to click on the task text and edit it directly.

You can add an event listener to the task text and enable editing mode when clicked. You can also provide a save button or press Enter to save the changes of the edited task.

You can also add a task count so you can display the total number of tasks as the top of the app. You can update the count dynamically as tasks are added or deleted.

You can also add a functionality like task filtering, you can add options to filter tasks based on their status, completed or pending.

And you can implement a drop down or radio buttons to allow users to switch between different filtering options. You can update the task list to show only the task that match the selected filter.

You can also add local storage persistence so you can implement local storage functionality to persist the task across page reloads. You can save the task array to local storage whenever a task is added, deleted, or modified.

Then you can retrieve the task from local storage when the app is loaded to populate the task list. You can also implement Clear All task functionality. You can add a button to clear all tasks at once.

When clicked, you can prompt the user for confirmation before deleting all tasks from the list. Last but not the least, you can also add responsive design. That means you can make the app responsive so that it adapts well to different screen sizes and devices.

You can ensure that the app layout and styling remain consistent and user friendly on both desktop and mobile devices.

Thank you.

The end…