**JavaScript Tutorial for Beginners [JS Crash Course 2024]**

**Intro and Course Overview**

0:00

congrats on deciding to learn JavaScript this crash course is actually a very small part of the software development

0:07

mini boot Cam that I teach where you learn every part of the software development life cycle to become a

0:14

software engineer or understand the entire concept and then go and pursue a

0:19

career in Cloud engineering or devops engineering Etc but in this video we'll

0:25

dive into JavaScript and really understand not only the syntax and the

0:30

basics but also the important and very interesting Concepts behind JavaScript

0:36

we will start by looking at the big picture first so we can understand where JavaScript fits in exactly when thinking

0:43

about website programming and then we'll jump into the demo and program with

0:48

JavaScript so let's get started right

**How websites are built (What is HTML, CSS and JavaScript)**

0:54

away imagine the following scenario you wake up in the morning get your coffee

1:00

open your laptop and open Facebook in your browser so let's pause there what happened when you typed in Facebook in

1:06

your browser and you saw a fresh new feed with all the new stuff that has happened while you were asleep what

1:13

happened there in the background technologically so Facebook is an application that some developers sitting

1:19

mostly in California are developing and these guys and girls are writing a bunch

1:24

of code and all that code that makes up Facebook lives in California so what

1:30

happens when you open Facebook on your laptop is that part of that code that lives in California travels all the way

1:38

to your laptop to your location and ends up temporarily living on your laptop

1:45

that's why sometimes when you refresh and see a blank page or loading and you become annoyed because you can't see

1:52

your feed within 3 seconds that's when all the data is coming or traveling from

1:58

somewhere to your laptop so it may need some time so how does that travel there

2:04

are internet cables everywhere in the world actual physical cables that let

2:09

electricity travel through from one place in the world to another that's how

2:15

everything is connected everywhere and that's why in some remote places in

2:20

Africa or Asia or just remote Villages anywhere in the world where there is no

2:25

internet and you're wondering like wow how is this possible no internet that's super crazy but that's pretty simple

2:32

actually because nobody built those internet cables near those areas and regions because of economic reasons and

2:40

so on now if you think about it it would be super inefficient if every time someone in Europe open Facebook and all

2:47

the data needed to travel to these places all the way from California so Facebook has built locations evenly

2:55

distributed all around the world that store part of the code and data that doesn't change much so it can get to you

3:02

much faster so back to that part of code that just traveled to you that's HTML

3:08

CSS and JavaScript so what are these when you see a website that just looks

3:13

ugly so we just text no fun colors no styling that's a naked HTML it's

3:20

necessary but no one wants to see some plain ugly text without any styling so

3:26

what makes the text pretty and it colors and some fun to it is CSS it's like

3:32

flash on a skeleton now you have a flash that looks good but it doesn't do anything it doesn't move it doesn't talk

3:38

doesn't respond you can interact with it and it's the same way with HTML plus CSS

3:44

page typical use cases are visit card websites where you can read texts

3:49

displayed in nice style but you can't leave a comment you can send a message you can do anything with the website so

3:57

all that action comes purely from JavaScript so you want to upload a photo on Facebook that's JavaScript you want

4:04

to send a message through a messenger that's JavaScript you want to like your friend's picture or leave a comment all

4:11

JavaScript so everything that you interact with in the UI which stands for

4:16

user interface is through JavaScript and the combination of these three HTML CSS

4:23

and JavaScript is called front end it's the Facebook code that everyone sees so

4:28

it's in the front and as I said all these three files leave on your laptop

4:34

when you open Facebook so you can see and even edit them so I could go and

4:40

change Facebook style like this and this won't affect anyone else in the world

4:45

since everyone has their own copy of these files on their laptops but there

4:50

is another part of the code that doesn't travel to you the beckon code that is in

4:56

the back and no one can see so you may be now wondering if I can do anything in

5:01

front end like upload images message someone leave a comment like what's the

5:07

backend for what does it do well when you upload a picture on Facebook you surely want your friends to see that too

5:14

or when you message someone you want to keep a history of these messages that means you want all this data to persist

5:21

or in other words the data to be saved over time now if JavaScript only leaves

5:26

on your laptop it wouldn't make sense to just have that photo only locally on your computer that's where the beon

5:33

comes in so what happens is that JavaScript takes that image that you uploaded puts it in an envelope and

5:39

hands it over to internet saying hey can you deliver this to backand at this

5:45

specific location so now we can follow this delivery all the way to the backend

5:50

and see what happens there or what what does backend do when it gets that image

5:56

now Beck in code can be written in Java python PHP or even JavaScript so there

6:03

are different technologies that you can use to write all that backend part of the code so let's say in our example

6:10

it's Java so what does Java do it checks and examines the photo it sees is it a

6:16

valid delivery there's no threat no danger I know the sender I trust it so

6:21

I'll open the envelope see everything is fine and hands it over to database to

6:28

save it for lat so database is like a warehouse it's a

6:33

big space where you can keep all the images messages likes comments that different users are producing it's all

6:40

in there so to have a more specific understanding let's actually see how the

6:46

location where beckon code and database leave quote unquote actually look like

6:52

so as I mentioned before Facebook has all these buildings distributed all around the world actual physical build

6:59

buildings with servers inside and I don't know what you imagine when you think about servers but whatever it is

7:06

it's probably not very close to reality so let me show an example of a server building of a typical big company like

7:13

Facebook so it's a location where machines uh that're interconnected with

7:18

each other are running 24/7 in order to make sure that Facebook or this

7:24

application is available for everybody in the world all the time

7:29

and that's where Beck and code and database actually leave so that's where

7:35

all the magic happens so when the envelope gets delivered to your backend that's where it ends up so the address

7:43

that it's sent to is one of those machines in among these hundreds or

7:48

thousands of machines that will take your image validate and then save it

7:54

into a database so it happens all inside there so now consider your friend logs

8:00

into Facebook on her computer same way front and code travels to her laptop

8:05

along with new data like images messages from friends Etc all coming from these

8:11

servers so how does Facebook know what data you should see on your news feed

8:16

now in the database Warehouse Facebook creates isolated spaces for each user

8:22

with their personal data images Etc and obviously on Facebook you want to have

8:27

some privacy meaning you want only your friends to see your posts so Facebook

8:33

creates Connections in the database between users that are friends and

8:39

that's how Facebook decides what information to send to a user when they log into the application along with new

8:46

data like images and messages from friends including the image that you

8:51

just uploaded all coming from these servers so to summarize a web

8:56

application is always made up of three components you have a front-end code backend and

9:02

database front end is what you see and interact with in your browser in the

9:07

user interface of the application backend is the code that waits for new messages from Front End about the

9:14

changes you make like new uploads new messages likes and handles them and

9:21

database is where all these changes are stored for future usage like when you come back the next day to use the

9:27

application or when your friend ens the application most of the applications are written by teams of developers and not

9:35

just a single developer and there are a lot of developers that are full stick meaning they know all three they can

9:42

work with front end backend and database however each of these three areas alone

9:49

is its own it field so some developers actually choose to specialize in just

9:55

one area and become for example a frontend developer so in big companies

10:00

like Facebook you would have teams with specialized frontend developers backend

10:05

developers and database developers and these things like front-end backend database and virtual servers you

10:12

actually learn all these in our software engineering mini boot camp so if you're a complete beginner in this field and

10:19

have no it knowledge and feel overwhelmed by all these Concepts and

10:25

components and don't know where to start then this mini boot camp is going to be exactly for you so make sure to check

10:32

out the link to the course in the video description so now that you know the basics of how the websites are built and

**Introduction to JavaScript (What is JS)**

10:40

generally how the web Works let's start by learning programming Basics with JavaScript so let's see what JavaScript

10:48

is and how it's used all modern web applications which are applications that

10:54

are used through internet browser are all written using JavaScript because it

10:59

has become the standard in developing web applications the way these web applications are built is that you have

11:07

a layer of HTML which gets styled and made more beautiful by CSS in order to

11:13

make this beautiful surface also called user interface smarter and more

11:19

intelligent we use JavaScript to write logic to create features examples of

11:25

such features are user registration booking a ticket online shopping sharing

11:30

photos writing messages Etc so when thinking about how websites are built

11:36

one can separate the design layer and logic layer and in big complex

11:42

applications used by millions of users each part becomes big enough on its own

11:48

so as a result you also have website designers also called UI designers and JavaScript developers as separate jobs

11:55

maybe you've heard of programming languages like Java python C Etc in one

12:01

sentence all these are backend programming languages and JavaScript is a frontend programming language in fact

12:08

JavaScript is currently the only programming language you can use to write front and code now to make all

12:15

these more tangible let's see how HTML CSS and JavaScript code actually looks

12:20

like HTML CSS and JavaScript are simple files that you write in a text editor

12:27

and these files have texts and keywords in them which have specific meanings to the browser in other words browser will

12:34

know exactly what to do with these keywords so the question is when you

12:39

create file with JavaScript keywords inside how does browser know that it's JavaScript and not some other text it

12:47

knows because you tell it explicitly using the file extension the typical

12:52

simple file will have a text extension txt for example but when you create file

13:00

called for example app.js browser knows it must interpret

13:06

all the contents of the file as JavaScript code and it works exactly the same with HTML and CSS to note here

13:14

browsers are also written by developers using some code so this functionality to interpret HTML CSS and JavaScript was

13:22

explicitly programmed and made part of all modern browsers it's important to

13:27

mention here that learning HTML and CSS is pretty easy since there are just a handful of Concepts you need to

13:35

understand and learn they don't change much over time and they're also less logical so it's more learning syntax by

13:42

heart to use them JavaScript on the other hand is very Dynamic and much more

13:47

powerful since you write almost all the logic with it so JavaScript changes a lot and improves continuously to make

13:54

writing all these logic easier and more structured so obviously when you look at at a simple web application Logic for

14:01

simple payment or uploading a picture or writing a message can be straightforward and easy but think of the complex

14:07

websites use daily that have hundreds of functionalities so that's a lot of

14:13

JavaScript code that developers have to manage and work with and that can be very challenging because of that over

14:21

time many developers or teams of developers that use JavaScript came up with different ways to improve it and

14:27

make it easier and better and they did it in parallel independent of each other

14:32

or also based on each other's ideas and as a result different JavaScript quote

14:38

unquote flavors or as we call them Frameworks emerged so framework is

14:43

basically an improved version of pure vanilla JavaScript which offers its own

14:49

additional Concepts and syntax on top of JavaScript Concepts and syntax that

14:54

makes writing complex JavaScript code easier and most of these Frameworks help you pretty much do the same but the

15:01

difference between them is in how they help you write the code structure it and

15:07

manage it and naturally when other developers tried all these different Frameworks a handful of the best of

15:13

these Frameworks became widely accepted and some of the most popular are reactjs

15:19

angularjs and vuejs so both pure JavaScript and JavaScript Frameworks are

15:24

continuously developing and improving in parallel now to mention here nodejs

15:30

which is also a JavaScript framework is different from all the others because it's meant not for frontend code but for

15:37

backand code so it's an alternative to Java python C Etc this means that if you

15:44

learn JavaScript you will be able to basically program the whole web application from front end to backend

15:50

using just JavaScript which could be a one incentive to learn JavaScript this

15:56

means that in order to learn any of these Frameworks that I just mentioned you first need to understand basic

16:02

JavaScript Concepts and syntax because all of these Frameworks are based on

16:07

JavaScript and once you've learned basic concepts of JavaScript then you can go

16:13

ahead and choose one of these Frameworks to learn and it will be much easier for you to learn Concepts and syntax of that

16:21

framework on top of JavaScript and in our software engineering course we learn

16:26

JavaScript in depth and then we learn vue.js framework specifically and how to

16:31

build applications with it plus in the course I point out exactly and compare

16:37

the advantages of using a framework compared to vanilla JavaScript to build

16:42

front-end applications so in the next sections we're going to cover the basic concepts and syntax of JavaScript so

16:50

that you have a foundation to learn any JavaScript framework so let's start by

**Datatypes and Variables**

16:56

learning the basic building blocks of JavaScript which are the data types and

17:01

variables and we will look at javascript's five main data types and then I'll show you what variables are

17:08

and why we need them know that these two building blocks are the same for

17:13

whatever programming language you learn so if I was doing a Java tutorial the concepts would be the same the

17:20

difference is in the syntax and this is a good thing because unlike learning natural languages when you learn one

17:26

programming language it will be much easier to learn another one so with that said let's Dive Right In now the

17:34

simplest way to start writing JavaScript is directly into the browser since all modern browsers are able to interpret

17:41

JavaScript and come packaged with so-called developer tools so open either

17:48

Firefox or Chrome browser on your laptop if you don't have these browsers

17:53

installed I suggest you install them because using browsers like Internet Explorer to run and execute JavaScript

17:59

is not a good idea so I would download either Firefox or Chrome and in your browser right click somewhere on the

18:07

surface and select inspect if your browser is in your native language then

18:12

it would be called something else but it should translate to inspect and when you click that you see something like

18:20

this and this space right here is called developer tools and here you should see

18:26

a console Tab and when you click on it you get this space where you can write

18:31

JavaScript directly in the next section I will show you all the different places where you can write and execute

18:38

JavaScript and which one you will usually use as a developer but in this part we're going to use the browser

18:44

console so that I can demonstrate some simple examples so the first data type

**Datatypes**

18:49

that we're going to talk about is numbers obviously when you're writing an application you need numbers some of the

18:56

examples are uh YouTube displays number of subscribers for each Channel or

19:01

number of likes for each video or how many videos one has when you search something on web application like Google

19:08

for example you usually see total heits uh Etc so these are all use cases where

19:14

you need numbers and these examples use whole numbers but for example when you

19:19

have an online shop where you need to display prices prices obviously won't be whole numbers always but rather decimal

19:26

numbers with send de ision also Amazon ratings right they can be 4.5 for

19:32

example and in JavaScript there are own data types for whole numbers and for decimal numbers the whole number data

19:39

type is called integer and these are numbers like 0 300 minus 45 so positive

19:46

negative numbers they're all integers and decimal numbers have a data type of

19:52

decimal and again they're positive or negative or zero so

19:59

0.0 um 90 777 minus

20:04

50.5 they're all decimals the second data type is string the way I like to

20:11

explain what string is is basically everything you see on your keyboard so

20:16

obviously all the letters no matter what language they're in but also all the special characters like M percent plus

20:24

minus and also the space character on your keyboard you also have numbers and numbers are characters or strings also

20:32

so how does JavaScript know whether you mean a number like integer or decimal

20:37

like five for example or a character five it knows it using quotes So you

20:43

tell JavaScript that you're using string by using quotes it could be single

20:48

quotes or double quotes So this single characters or any combination of them is

20:55

a string so let's say a couple of examples so three in quotes is a string

21:01

a o with umls letters from non-english alphabets combination of

21:09

letters and numbers special characters like in email or in password and also an

21:15

empty space character is also a string so just to mention here so difference

21:21

between character and string so character is basically just one letter or one number one special character and

21:29

string is a character or a combination of multiple characters so these are all

21:34

strings and this single keys are characters another data type in

21:42

programming languages is Boolean now to understand this let's say you enter

21:47

wrong password when logging into your application usually you get a message saying you provided a wrong password or

21:55

you accidentally enter wrong credit card number uh when buying tickets um or

22:01

invalid email address when registering so in the background developers write JavaScript code that validates whether

22:08

the information that you provided is wrong or right or correct and to express

22:15

that wrong or correct state booleans are used so word true expresses correct

22:22

State and false expresses wrong state so again just like with numbers you can

22:28

write a simple string false with quotes and JavaScript know that you mean a string without quotes it will be a

22:35

Boolean expression false that's how JavaScript can differentiate between those two in addition to that true false

22:43

can also be used for simple yes no situations like is the apartment on

22:48

Airbnb available to book on this date yes no that will be true or false in

22:53

Boolean terms or is user logged in depending depending on whether logged in

22:59

or not you will see different web page that's a Boolean expression is it a premium user true false also bulling

23:07

expression Etc another data type is array that expresses lists think of

23:13

applications that display lists of the same kind of data like list of friends

23:18

on Facebook or list of apartments on Airbnb or list of comments for your

23:24

Instagram picture lists have their own data types in all programming languages

23:30

in JavaScript its data type is an array for example list of friends names will

23:36

look like this it will be an array that's an array that includes

23:43

multiple names and now the syntax of the array that encloses square brackets so

23:51

that's how JavaScript knows or can interpret that it's dealing with an array or you could also have a list of

23:58

ratings for Amazon products for example so this will be an array with

24:05

numbers as you see array contains other data types so here we have uh list of

24:11

strings and here we have numbers and in JavaScript you can mix different data

24:16

types inside an array here we have integers and decimals in one array you

24:21

can also write an array like this where you have string in integer and Boolean

24:29

expression and the last data type in JavaScript is an object now in the

24:34

Amazon review for example you don't just see a list of ratings you also see the

24:39

author's name next to the rating and text they wrote In addition so you need all this information grouped in one item

24:47

in the array you express that using object data type so a single rating

24:53

object will look like this and not the syntax of curly braces

24:59

so as we saw array was expressed using square brackets and object is written

25:05

using curly braces so that syntax is important so that JavaScript understands

25:10

what data type you're writing another example is an apartment object on Airbnb

25:17

which will include all the attributes it has like location price ratings

25:23

description availability images Etc so an object for that information will look

25:29

something like this just with more information inside and here you see that the syntax highlighting of the browser

25:37

is also helping to see that strings booleans and numbers are colored uh in

25:42

different way meaning that JavaScript understands the difference between

25:48

them so as you see object has key value pairs so that you know which value

25:53

stands for what so all these author name user are 12 rating five these are key

26:00

value pairs where the key basically describes what the value stands for and the name of that key is totally up to

26:07

you you decide what that's going to be also you can use any data type as a

26:13

value inside of an object so here we have a string integer Boolean and array

26:19

as a value and you can also use another object as a value inside of an object

26:25

and finally uh if go back to the ratings you will have multiple rating objects so

26:31

a list of rating objects so the final list of these rating objects will look

26:37

like this so you will be an array with square brackets and it will have bunch

26:43

of objects that are comma separated now this is probably the most complex structure you will deal with most of the

26:51

time so it shouldn't get much more complex than

**Working with Numbers**

26:56

this now we saw all these data type example values but how do you use them to write

27:03

a so-called logic in JavaScript the simplest use case is some basic addition

27:09

subtraction etc for numbers you can use Simple arithmetic from elementary school

27:14

to calculate basic things and here I will use the chance to break this common misconception that you have to be super

27:21

good at math to learn programming which is completely 100% wrong I was a

27:27

software engineer for many years and worked in a lot of different projects and I have barely used anything more

27:34

advanced than simple arithmetics like plus minus multiply and divide when writing web applications so in

27:41

JavaScript you can do all of this you can do subtraction

27:48

multiply divide and you can also combine them just like you would with a calculator

27:55

and of course you can do the same with desk decimals some real life use cases for these basic arithmetics are for

28:03

example when you add multiple items to your shopping card on Amazon you see the sum and you also see the price breakdown

28:10

which is item prices plus shipment costs or when you have product ratings based

28:16

on all the individual ratings you display the average rating or on Uber for example you see the distance from

28:22

pickup locations to the destination in kilometers or miles and in minutes

28:28

all of these are simple calculations which are possible in JavaScript now obviously all these

28:34

operators are meant for numbers but there is a case where we can use the plus operator for string data types so

28:44

what will happen if I write string 12 +

28:49

string 12 so we get 1,212 so what happened here is that

28:54

JavaScript knows that these two are are strings and not numbers and instead of adding them it handles them as strings

29:04

by gluing them to each other and in programming that's called string concatenation and no you don't have to

29:11

remember that word just so you know that there are weird names for simple things in programming the same way you can glue

29:18

a string to any other data type so you can do blah to 12 and the number will

29:26

also turn into a string so to say or you can do the same with Boolean

29:32

values because JavaScript interprets that as you want to create another

29:38

string out of these two values and this examples probably don't

29:44

make much sense but in the next section of this video I will show you some more real life examples why this concept is

29:51

actually very important now let's understand one of the most important building blocks of

**What is a Variable, Why they are useful**

29:57

any programming language variables and let's see what variables are to

30:03

understand the concept of variables think about the following scenario you change your username on Facebook so

30:10

obviously your changed username is displayed on your profile but also all the comments that you wrote before the

30:16

name change should now appear with your new username and also in all your

30:22

friends lists your new username will appear all right so this means if you

30:28

had Facebook code where your username is written in all those different places so

30:34

you have your username in the profile section you have your username in comment one comment 2 Etc and in all

30:41

your friends lists right written directly as is or let's consider another

30:47

example think about online shop each product has a price and it's displayed

30:52

in a list in the products own detail page and maybe also in combination to

30:57

other products so if the store now offers a discount and the price changes

31:04

obviously the price should be updated in each location where it's displayed so

31:09

again in code it will look something like this obviously this is not a valid

31:15

JavaScript code but just to give you an idea so somewhere in one Javascript file

31:20

you have the list of products where the product price is directly written as is

31:26

and you'll have the same in detail page section of the product and

31:31

combination now when the username on product changes in those two scenarios all these

31:38

variables need to be overwritten which means that you will have to go and change the price here and here and here

31:45

and same with the username change it in all places where it's used now that

31:50

would be absolutely inconvenient considering applications are so Dynamic and things change a lot

31:58

so that's where variables come in so instead of writing the actual value in

32:03

10 different places you write the value once and then reference it 10 times from

32:10

10 different places and that reference to the actual value is called a variable

32:16

and you give variables a name that makes sense for that value so for example

32:22

product a price equals 50 so now you create a reference for the value 50

32:30

which is called Product a price or username equals app user a so you take

32:37

the actual value which is string and create a reference to it but in addition to that you should also let JavaScript

32:44

know explicitly that this random name that you just came up with is a variable

32:50

and you do that using VAR keyword like this and remember keywords

32:56

are words that have special meaning to JavaScript and because of that if you

33:01

noticed the coloring or the highlighting of the word changed once I added VAR in

33:09

front of it so now JavaScript knows that this is a variable keyword and this is the name of that reference or variable

33:16

or the same with product a price variable and in code it will now look like this so first you'll create that

33:23

reference somewhere like this and then in all the locations or all the places

33:30

in JavaScript code where you need that value that actual number value you use

33:36

the reference instead of the actual value so you have your text and the

33:41

reference to the number in all those different locations and here note the plus operator that I showed you earlier

33:49

this is an example of string cenation meaning string is glued to another value

33:56

and this this is where this concept is very useful so when you run or execute

34:02

this line what happens is that JavaScript in the background replaces this reference with the actual value so

34:09

when I execute this I see product a price 50 and this is a real example

34:15

where plus operator with strings is actually very important and you will use

34:20

this a lot because you will use variables a lot in order to make this variable concept stick more let's

34:27

consider two additional use cases first consider multilanguage applications on

34:33

most applications you can select a preferred language and see all your navigation buttons Etc in your language

34:42

now if application supports 10 languages obviously developers don't create 10 different websites instead the same text

34:50

is translated in 10 different languages and reference using variables and

34:55

depending on which language which the user selects the correct reference is

35:00

used and the Second Use case which is very important is user input so when you

35:06

sign up for a web application you need to enter your name last name email password these are all user inputs so

35:14

what you input in the application as a developer when you when you prepare that code for future usage you don't know

35:21

what these values are going to be because users can enter anything but what you know is the name of the

35:27

variables that will reference these values and this way you can use the actual values without even knowing what

35:34

they are now the question is when we create

**Where to write and execute JS**

35:39

JavaScript files and write lots of JavaScript logic inside how do we run

35:44

this logic in browser how do we give that JavaScript code to the browser to execute it there are three ways to

35:52

execute your JavaScript code the simplest way to start writing JavaScript is directly into the

35:59

browser this is a place where you can write JavaScript directly and the browser will interpret and execute your

36:06

JavaScript code right here so you don't have to set up anything and this is the easiest way to start especially if you

36:13

want to try out some simple commands so let's actually write some JavaScript code there so you can create variables

36:20

here or you can do some calculations and you can

36:27

at number one and number two Etc so you can do many things here however if I

36:33

refresh the page of course all the code will be gone and also you can execute

36:38

only one command at a time and usually when you're developing you want to save

36:44

that code and execute the whole thing at once and not have it disappear every time you refresh the browser so for that

36:51

you need to write JavaScript in a file and then give that file to browser in order to execute the code

36:58

inside and the file that browser accepts to execute JavaScript code is HTML I'm

37:05

going to create a basic HTML outline so that our browser can execute the JavaScript code so how can I write the

37:12

HTML file with JavaScript code in an editor so if you are on Windows you may

37:18

have a notepad a simple text editor or ionm have a text edit and I can create a

37:24

new document if I open that I can write just normal text here so I'm going to

37:30

write some simple html text and HTML as we saw at the beginning is a separate

37:36

scripting language to basically write the structure of a website and I teach

37:42

complete HTML and CSS in the software development miniot camp but here our

37:47

focus is on JavaScript so we're going to use the most basic HTML possible to be

37:53

able to execute JavaScript on the website so so maybe you're wondering if I'm writing HTML in the same editor

38:00

where I would write normal text how does browser know that it's an HTML file well

38:07

when you save the file instead of text document I'm going to choose HTML so

38:13

HTML extension and I'm going to call this file index which is a standard or common name for the main HTML file so

38:20

when I save this you also see the icon that has Chrome logo in it or maybe some

38:26

other browser logo that means it's a browser executable and since this is

38:32

just HTML in order to add JavaScript code I'm going to add tag called Script

38:37

that's where JavaScript goes and and this is a part where we'd write

38:43

all the JavaScript code so for example here I can create a variable called some

38:50

name and so on now I said you can write the whole HTML and JavaScript in here

38:58

but the problem with using the simple editor is that they don't help you in writing code so here you see they don't

39:06

highlight the keywords they don't tell you you made a syntax error or any other tips so for example in browser when we

39:13

wrote VAR name some name you get the highlighting of the keyword of the

39:19

variable name of the data type for example if you have a number it has a

39:25

different highlighting Etc so you don't have any of that in simple editor and it

39:31

also looks pretty ugly so instead there are special editors for different

39:36

programming languages that help you with highlighting and other features to write

39:41

the code and there are special ones for writing HTML and JavaScript code

39:47

specifically the one that we'll be using in this course is webstorm from

39:52

jetbrains which is actually an integrated development environment M or IDE and not a simple code editor what

40:00

does that mean exactly well when you install webstorm you basically have everything that you need for website

40:07

development in there so various features Beyond just writing code are included

40:12

out of the box or in other words you get an IDE with full batteries included and

**Download Webstorm IDE and create new Project**

40:19

we'll see some of its cool features throughout the course so if I type in webstorm download we can check out their

40:26

download page for your specific operating system and if I click on

40:31

developer tools you actually see they have a list of multiple idees and they

40:36

have dedicated idees for different programming languages and webstorm is

40:41

specifically for JavaScript so it gives us these awesome features specifically

40:47

for JavaScript development and the great thing is they have special offers for students universities startups for all

40:54

their idees including webstorm so let's go ahead and download the webstorm IDE

41:00

for free to develop our JavaScript application I'm using Mac OS so I'm going to download this one and once

41:07

downloaded you just double click on this installer file let's drag webstorm to

41:12

the applications and there you go we have webstorm available locally let's double click again and it's going to

41:18

open the webstorm IDE this is the welcome page and let's

41:24

go ahead and create a new product project let's call this JavaScript app

41:29

and create and in the webstor settings I'm actually going to zoom in so that you can see the code better so I'm going

41:37

to set it to 150 and there you go this is what we are starting

**Create index.html file**

41:45

with so now let's create a new file in our JavaScript application project if

41:51

you go here and right click with your mouse you see these options and we we can create a new generic file or if we

41:59

know that we're going to be writing a JavaScript or HTML file we can actually choose specifically that type of file

42:06

and this way webstorm actually knows that you want to write HTML or JavaScript code and it actually helps

42:14

you create the boilerplate code or the initial code for that file specifically

42:19

so let's see what happens if I choose HTML file from the options and let's

42:25

call this file index.html like we did before and know that we have HTML extension in the file

42:33

name and if I hit enter there you go webstorm actually knows that this is a

42:39

basic HTML structure all these tags are needed for a proper HTML page so it

42:46

autogenerated all these so you don't have to type it out from scratch this is super convenient and let's give our

42:54

index.html web page a tiple let's call it JS

42:59

app just like we did here and that's it now you can ignore actually all this

43:05

code here that was autogenerated this is kind of a structure a main structure for

43:10

a properly formatted HTML so everything that is needed is right here what we

43:16

want to focus on is writing our JavaScript code so if we check our index

43:22

HTML file that we wrote before we have this script text with some JavaScript code inside so I'm going to copy that

43:29

block we already have the body tags and the script tags usually go inside the

43:35

head Texs so between those opening and closing head Texs or between the body

43:41

Texs so in our case we're going to add them in the head and as soon as I paste

43:47

in the script block webstorm IDE already highlights some of the issues with our

43:55

syntax so because we use this simple text edit editor we have some wrong

44:01

characters that can't be interpreted by the browser so we need to fix them and know that we have these red underlines

44:07

here that are for error and then we have these orange underlines which are for warning so this need to be fixed because

44:15

otherwise our JavaScript is not going to work so we're simply going to replace those with a standard quotes basically

44:22

and that's it and as you see with the syntax highlighting the value assignment to the variable is working now and the

44:28

error is gone now you notice we have another underline here which is a

44:34

warning this means the code is going to work fine without any issues but we

**var vs let vs const**

44:40

could make some improvement to this code it says VAR is used instead of let or

44:45

const so what is it about back in the day when JavaScript was in the early

44:51

phases VAR keyword was used to declare the variables later as the JavaScript

44:58

programming language evolved and improved and Engineers who were developing JavaScript programming

45:04

language made those improvements these two Alternatives were added to defining

45:09

variables the let and const keywords and the reason for that Improvement or the

45:14

main benefit of that is to prevent basically accidentally overwriting the

45:21

variable values because JavaScript is very flexible which is great but it also means means that you can make lots of

45:27

accidental mistakes and one of them being defining the same variable multiple times and accidentally

45:33

overwriting the previous variable because you didn't know that you already had that variable in your code and

45:38

because of that we have these two Alternatives so the con is basically for

45:43

use cases where Define a variable so basically a value that you want to reference in multiple places but that

45:49

value should never change for example if we're developing an e-commerce application for Nike the name of the

45:57

shop or the brand name Nike should never change right imagine someone overwriting

46:03

the name of the brand or name of the shop from Nike to something else so this

46:09

value should not change dynamically within the program or within the application so we had thousands of lines

46:16

of code here there should not be a line that overrides the value of brand if

46:21

that happens accidentally Again by the mistake of developer then and the fact

46:26

that we're using const in front of it will tell JavaScript that program cannot

46:32

overwrite the value of this variable and for all the other use cases where we

46:37

want to allow program to change the variable value we can use let instead of

46:43

the VAR example again would be if we had some prices for different items and then

46:49

on some days we have a campaign so that price can dynamically change when discounts get applied for example or if

46:56

we have a customer name and they change their username or whatever that value could change as well so using let for

47:04

all the other variables instead of VAR is a good practice and as you see webstorm itself is helping us especially

47:10

as beginner developers to learn those syntax issues as well as good

47:16

programming practices so now we have a fully functioning HTML file with JavaScript code inside that is working

**Execute HTML file in Browser**

47:23

and properly written now the question is is how do we execute the index HTML file

47:29

in the browser so how do we take this file and contents and make it work in

47:34

the browser now there are a couple of simple ways to do that and I'm going to show you all the options so you can

47:41

compare them the first very easy option is to drag and drop this file into the

47:47

browser so just open a new tab on the browser and we need to locate this file

47:54

in the file explorer and we can do that by navigating directly to this file using this

48:00

location that you see here so this is my home directory called Nana and inside that I

48:07

have these webstorm projects which automatically gets created by the way by webstorm and inside that I have Js app

48:14

and index.html or you can also save yourself all that just right click on the file

48:21

itself open in finder and there you go

48:26

so now back to the browser one way to execute this index.html in the browser is just take this file and drop it

48:35

inside the browser and there you go now we don't see anything here it's just White Blank Page because we are not

48:43

displaying anything we're just creating a variable not even using it but if we open the inspect or the

48:51

developer tools of the browser we can see that this is exactly our our index.html with our script inside so

48:59

that's one way a second also very simple way is to Simply double click on

49:05

index.html and this will open a fresh new browser tab like this

49:12

again the same page with the same contents and note that we have this

49:17

title here this JS app displayed which we defined right here so that's the

49:25

title of the web page now there are two more ways of executing our index.html

49:32

file specifically using web stor and these are actually way more convenient

49:38

because you can open this file directly from here using one of the browser icons

49:44

so webstorm actually detects the browsers that you have available and it will show you the icons so you can open

49:51

this HTML file in one of your browsers so if I click inside side again this

49:57

opens a new browser tab automatically this is our title it even has this

50:02

webstorm logo here as you see the value inside the browser URL is different

50:08

because it's coming directly from web store and inspect again there you go

50:16

this is our code and final probably the most convenient way of opening your

50:24

index.html file a browser is using this web storms built-in browser preview so

50:31

let's click on this and see what happens and as you see it shows us the browser

50:36

integrated directly in the web storms IDE and that leads me back to the name

50:42

IDE which stands for integrated development environment so basically this is an environment where all the

50:49

tools and the things you need for application development are directly integrated in into this environment and

50:58

the convenient part about this is obviously that you don't need to switch between the IDE and the browser you have

51:04

everything in one place you have your folder and file structure here you have the code editor here and you have the

51:10

browser and a cool thing about it is that right now we don't have any text displayed on the website but let's

51:17

actually display something so we can see how it works this is our

51:23

awesome JavaScript application and as soon as I save and by

51:29

the way shortcut for saving is command s on Mech or control s on Windows as soon

51:35

as I save notice here the text is displayed right away right and again add

51:41

something here and all those changes are directly displayed here this saves a lot

51:47

of time when you're actually actively developing and making thousands of changes so you don't have to refresh and

51:53

reload all the time so this is available out of the box in our case we're

51:59

actually going to stick to the browser because we want to

52:05

use the browser developer tools because we're going to be outputting some stuff

52:10

here for our tutorial specifically however if I was developing myself and

52:16

writing the HTML code then I would actually go for this preview feature so let's close this and we already have the

52:24

application open in browser and you noticed that on the browser as well

52:30

without refreshing or reloading the page the changes are automatically applied so

52:35

let's go back and remove this and again let's save go back and the changes are

52:41

applied so we have the same auto sync or automatically reload on Save when we

52:47

open this index.html using one of these browser icons here which again as you know

52:55

notice is different from when we directly drag and drop the file in the browser here we would have to explicitly

53:01

refresh to apply the changes now this may look like a small thing when you're just getting started however when you're

53:09

a little bit more experienced and you're developing actively this may save you a lot of time and a lot of nerves awesome

**Console.log**

53:16

now if you want to also display stuff in the browser console we can add a line

53:23

here console do log which basically outputs whatever we pass in to the

53:30

browser so we can pass in the variable reference

53:35

or a stream directly like this again as soon as I

53:43

save this will automatically be reloaded in the browser and there you go we have

53:49

the variable value and we have whatever I wrote directly in the console.log so

53:56

console.log basically prints out stuff directly in this console tab right here

**JavaScript in a separate File (Project Structure)**

54:02

now imagine we actually wrote Our HTML web page structure right here within the

54:08

body Tex which can be actually a very large HTML file and in addition to that

54:15

we wrote a lot of JavaScript functionality within those script texts

54:20

that will be pretty inconvenient because now we have lots of lines of code within

54:25

one file so we have HTML here we have JavaScript here and maybe thousands of

54:31

lines of code which makes it difficult for us to have an overview of all the things this file contains and we're kind

54:38

of mixing two different things within one file so in programming as your code

54:44

grows it's always a good practice to keep your code clean have multiple files

54:50

where each file contains a limited number of code lines so that you can easily navigate through your application

54:58

code or if you have new team members new developers joining your project team

55:04

they can easily navigate through the application code so it's always a good practice to have your code clean and

55:10

especially to separate the functionalities from HTML CSS and JavaScript that means instead of having

55:16

our JavaScript code directly inside we want to actually have it in a separate Javascript file and Link it here and

55:23

it's super simple actually so first thing we want to do is to create own dedicated JavaScript file

55:29

instead of having JavaScript directly in the HTML file so going back here right click with your mouse again and this

55:36

time we're going to create a new Javascript file and we're going to call this

55:42

.js and again notice the JS extension in the file name this way browser will know

55:49

this is a Javascript file hit enter and you also see these different icons that

55:54

webstorm displays for different file types so even if we did not have the extensions for the files just by looking

56:02

at the icons you will know this is Javascript file this is HTML and

56:07

obviously other types of files will have their own icons as well this is so now

56:14

we're going to Simply take all this JavaScript code that is defined within the script text we're going to cut that

56:21

out or you can just use the shortcut and we're going to paste it directly in

56:28

here and you notice in JavaScript file we don't need any text we can just write JavaScript line by line like this and

56:36

you notice again webstorm detecting that it is a Javascript file actually

56:41

provides some additional functionality even to our code and one of them is showing the data type of the variable So

56:48

based on the value it's automatically detected that this is a string variable

56:54

and it's basically showing that as an additional hint to help us in programming however we're not done yet

57:01

because now we have those empty scripts but there's no linking or connection

57:07

between our HTML and JavaScript file we need to actually tell this HTML file to

57:13

get the script from somewhere so we need the source of the new JavaScript code

57:19

and the way we Define it is we leave this section empty so nothing actually

57:24

goes between the script tags and instead within the opening script tag we Define

57:31

a source SRC so this basically is a pointer to

57:36

JavaScript file that contains the script that we want to load as part of HTML

57:43

page and as you see again super conveniently webstorm is automatically

57:48

suggesting us all the files that we have in our current project and it tells us

57:54

you want to to reference epto JS and if we do if I just click on it it basically

58:00

automatically sets that as the source name right here and that gives us the

58:07

link that we want and again super convenient especially as a beginner that the IDE itself provides all these

58:14

additional explainers telling you what different tags are and so on and as I said nothing actually goes between those

58:21

script tags so we can just leave it empty and this will load the same script that we had defined directly here now

58:29

from a separate Javascript file so in order to test this I'm going to make a change in the JavaScript code and we're

58:37

going to see whether that is actually reloaded in the browser once we save it

58:42

so I'm going to say output in console from app.js

58:48

file and I'm going to save this make sure to save both changes in both files

58:54

because if if you don't save the changes in webstorm it's not going to be reloaded in the browser so now let's

59:01

check back with our browser and without any refresh you see output in console

59:07

from .js file and if I go back to the elements which is the HTML tags body and

59:15

so on we see that script is linking to app.js and this is another element that

59:23

is also loaded in the browser along with HTML file and that is also very interesting and important so these

59:30

are the HTML elements on the page however if I go to the sources tab right

59:37

here you see our index.html file this is the file that we wrote and

59:44

this is some additional stuff that webstorm actually adds to our index HTML

59:49

automatically to give us this automatic loading and then we have our .j s file

59:55

also loaded alongside and if I check the sources of

1:00:01

our previous index.html that had the script directly you see that we don't have any

1:00:07

Javascript file it's just HTML now let's learn about the next important concepts

**Conditionals and Operators**

1:00:13

of conditionals and comparison operators which are also common Concepts across

1:00:18

all programming languages and if those terms sound scary to you I'm going to explain those with very simple examples

1:00:25

and visualizations to make sure you understand them very easily so first let's see what

**Comparison Operator**

1:00:32

comparison operators are let's consider as an example

1:00:37

Facebook featured to display birthdays of its users so basically if today is

1:00:44

your birthday Facebook will show a birthday icon and will send

1:00:49

notifications to all your friends and for Facebook to check whether it's your birthday today basically it has to

1:00:56

compare today with your birth date right so that comparison or equals in

1:01:04

JavaScript is expressed with three comparison signs like this so one

1:01:10

comparison sign is reserved for assigning variables so double or triple

1:01:17

equal signs are used to check the equality of two values so let's see the

1:01:23

difference between the two so let's say I have a variable H 30 so if I compare value of variable H

1:01:33

to 30 then I get true right if I triple

1:01:39

compare that I also get true so same results now if I

1:01:46

compare the value with a string representation of 30 I also get

1:01:52

true however if if I triple check I get false and this is the

1:01:59

difference between these two the double comparison checks equality of the values

1:02:05

on two sides so checks the value here checks the value here and Compares them

1:02:10

by the value the triple equality Compares value and also the type the

1:02:16

data type of those values so if I have a number here and a number here which are

1:02:23

the same values then it's going to be true if I have a number value here which

1:02:29

is defined here and a string value here of the same value I get false because the data types are not the same so

1:02:36

that's the difference and obviously if I put here another value it's going to be false in any case right like this or

1:02:45

like this doesn't really matter and that little difference can actually have a

1:02:50

big impact that's why this concept is important in JavaScript general rule is that you should in most cases use the

1:02:58

triple equation to compare two values now let's consider another example where

1:03:04

you order something on Amazon and says that shipping is free above purchase of

1:03:09

€20 so when you add things to your card program checks whether your order total

1:03:15

is above 20 or below 20 and then calculates the shipment cost based on

1:03:20

that and let's say it charges you5 if your if your total order is under €20 so

1:03:28

now we don't have the equal check anymore but rather is it higher or is it

1:03:34

lower so the way to check that is let's clear this so let's define the variables

1:03:40

so you have the total price let's say you gathered items that are €19 and now

1:03:46

the program checks if total price is greater than €

1:03:52

20 you get a bullan expression of false because this condition is wrong in the

1:04:00

same way the program can also check the opposite direction so we can check whether your total price is below 20 but

1:04:09

notice that we don't check for Value 20 so what happens if the total price is

1:04:14

exactly 20 so basically here we need to check that if total price is below 20 or

1:04:21

€20 in both cases you get charged for the shipping so in order to do that you

1:04:27

can combine those two expressions and say like this so either it should be

1:04:32

less than 20 or it should be exactly

**Conditionals (if / else statement)**

1:04:39

20 and as you see each of these comparisons return to you a Boolean

1:04:44

expression of either false or true only one of these outcomes is possible so

1:04:52

what this means is that in the program you check if this condition comes out as

1:04:58

true then you want to offer free shipping otherwise you want to charge €5

1:05:05

for shipping and the way it's going to be written inside of the program using JavaScript syntax is if tour price is

1:05:13

greater than 20 here will be code that let's say calculates um or sets the

1:05:19

price of shipping so here we set it to zero there's no shipping cost otherwise

1:05:26

is expressed with else shipping cost equals to 5 so this is how the

1:05:35

conditions are used in JavaScript to check the condition and based on that do either this or if we switched that

1:05:44

condition we can also check if total price is less than or equal to 20 then

1:05:51

shipping cost will be set to five otherwise the shipping cost will be zero

1:05:59

so to go back to some technical terms in JavaScript this here is called if else

1:06:05

statements whatever is inside if is called a condition so this thing here

1:06:11

that evaluates to either true or false is a condition and this here that compares those two values is called a

1:06:18

comparison operator so greater smaller equals Etc all these are comparison

1:06:25

operators but let's say you have a case where you have three different shipping costs right so for example if the price

1:06:32

is below €1 you get charged 5 shipping cost if it's below 20 you get charged

1:06:40

only €3 and if it's above 20 you get charged nothing now you don't have if

1:06:46

else only but you have three conditions it's also super simple to do that with

1:06:52

if else so basically you check again if total price is less than or equal to 10

1:07:01

you get charged 5 and here you are going to say else if

1:07:07

which is another statement where you can write the conditional where we're going to check whether total price

1:07:15

is less than or equal to 20 in that case you get charged €3 and there comes our final else which

1:07:23

is zero so now I have three conditions here so the first condition here says it's

1:07:31

less than or equal to 10 the first one checks another condition and the third

1:07:36

one um important thing to note here that the last else or the the else here

1:07:43

doesn't have a condition explicitly but it implies that both of these above

1:07:50

conditions were false so both of them were wrong be and because it's implicit

1:07:56

we don't have to check it explicitly here so you could also go here and do

1:08:02

that else if price is greater than 20 and this

1:08:09

will work just fine however because it's implicit we don't need that so we saw

1:08:15

comparisons between two values using equals higher or lower the final

1:08:21

operator is not equals for example consider a feature where a web

1:08:27

application checks whether user is a premium member or not obviously if you

1:08:33

are not a premium user you won't see the cool features they have or you won't be

1:08:38

able to access some premium content so when you log in program will check is

1:08:45

user membership premium so with equals or it can check user membership not

1:08:52

premium and in order to express that negative comparison you use this syntax

1:08:59

so not equals in JavaScript is expressed with exclamation mark and double equals

1:09:05

and to show that in IFL statement let's say if user membership is not premium

1:09:11

show user non premium contents or maybe even recommend a premium upgrade as a

1:09:19

marketing measure and else show premium contents and finally you can combine all

**Logical Operator (AND, OR operator)**

1:09:27

these

1:09:32

conditions so for example if today is your birthday and you have displayed

1:09:38

that on your profile only then the program will show the notification so

1:09:43

both conditions must be true so if today's date is the same as your birth

1:09:49

date and birthday displayed is true so here you can use a comparison with uh

1:09:55

Boolean true or false so both these conditions must be true for notification

1:10:01

to be sent and in JavaScript the Syntax for end is this so This basically is

1:10:09

also a valid condition that you can also put inside of the if condition so if

1:10:14

this whole thing is true then show notification else do not or another

1:10:20

example back to the shipment cost let's say you don't pay for the shipment if

1:10:26

the total price is over €20 or you have Amazon Prime account so the way to

1:10:32

express that would be total price over €20 or Amazon Prime

1:10:40

equals true so here only one of them has to be true for you to get free shipment

1:10:47

or in JavaScript is expressed with double pipes and again you can put that

1:10:52

in if statement and say if either this or this

1:10:59

is true then give a free shipment otherwise don't and you can also use

1:11:06

negation in conditions for example you want to check whether this whole thing

1:11:12

is false and the way to do that is using the exclamation mark and the brackets so

1:11:21

basically this will check whether this whole condition is

**Functions in JavaScript**

1:11:29

false now let's learn about one of the most important Concepts in any

1:11:35

programming language which is functions and let's understand with simple examples what they are and why we

1:11:42

need them so let's go back to our JS app project and we're going to work in this

1:11:47

EP tojs file so I'm going to delete this and instead we're going to work with

1:11:53

this simple shipping cost calculation logic so this if else statement is our

1:12:01

JavaScript logic that basically calculates based on whatever total price is how much shipping cost a customer

1:12:08

will have to pay and let's actually output whatever the shipping cost is in

1:12:13

the console so we have this application up and running open in our browser so

1:12:19

whatever we output with console.log is going to appear right here and in

1:12:24

webstorm actually there is a very convenient way of writing console.log if you're lazy like me and don't want to

1:12:31

type out the whole thing you can simply do log enter and that basically

1:12:38

autocompletes console.log now we want to log whatever shipping cost value was

1:12:44

calculated by our logic so I have just set 19 as a total price you can set

1:12:50

whatever value you want and then we're going to output the result I'm going to

1:12:55

save it and if we go back to the browser you see the output here however I don't want to see just number I want to see

1:13:03

what is it exactly what does a number refers to so I want to have a full sentence like

1:13:09

shipping cost for you is whatever the value of shipping cost variable is so

1:13:16

basically we have a string here string value and with that string value we also

1:13:21

want to Output the value of a variable and there is one way to do that in JavaScript which is just combining these

1:13:28

two with the plus sign so it's not an arithmetic operation like with numbers

1:13:34

but in this case it basically takes whatever string value of this is and then whatever value this variable has

1:13:42

and outputs it as one sentence or one string so if I save this go back there

1:13:47

you go you see the full sentence however there is more elegant way to do that

1:13:52

which is is way nicer which is we remove these double quotes here

1:13:59

and instead we use what's called back ticks like this for the entire thing

1:14:05

including the variable and then to tell JavaScript these are all text and this

1:14:12

is actually name of the variable it's not the string itself we use this syntax

1:14:17

which is dollar sign and curly braces that's it and you see in webstorm itself

1:14:23

that the highlighting the syntax highlighting changed so this is treated

1:14:29

as a variable so it's value will be extracted and this is treated as the normal string again let's save go back

1:14:36

and we have the same result okay now this whole thing here is a fully

1:14:41

functioning logic that we wrote in JavaScript that takes whatever total

1:14:47

prices and calculates the shipping cost based on that and then outputs it in the console now imagine if we really had an

1:14:55

e-commerce application we would have thousands or tens of thousands of users

1:15:00

maybe and everyone will be shopping in their own shopping cart and adding items

1:15:08

so we would actually need to calculate this for every single purchase they make

1:15:15

on our e-commerce application so this logic this whole thing May repeat in our

1:15:22

application code multiple times and imagine we had 10 different places where

1:15:29

this calculation needs to be done and we would have to basically replicate this

1:15:34

code in 10 different places throughout our application code that seems like a

1:15:39

very inefficient way of coding right because we're repeating the same thing

1:15:45

10 times like I explained with variables where we use the same value multiple times except in this case it's even

1:15:51

worse because we're repeating the the entire code block multiple

1:15:56

times and that's where functions come in so you can think of a function as kind

1:16:04

of a variable reference so just like we saved a certain value once in a variable

1:16:11

and then we reference that variable every time we needed to grab that value

1:16:16

the same way we can save an entire code Block in a function and then whenever we

1:16:22

need that code block or that logic we can reference it by using the function name so I'm going to put this entire

1:16:29

code block including the assignment to total price variable inside a function

1:16:36

and the Syntax for that is actually super easy what we do is just like with

1:16:42

variables we have VAR or let or const keyword with functions we have one

1:16:48

keyword called function then then we have the name of

1:16:54

the function which is usually descriptive that describes whatever that code logic that the function references

1:17:01

actually does in our case let's call this calculate shipping cost because that's what the code block does right

1:17:11

calculate shipping cost and note at the end of the function name we have this

1:17:17

brackets opening and closing brackets and after that we have curly braces and

1:17:23

these curly braces will basically contain the block the code block that

1:17:28

function references so if I take this whole thing and paste it in here and

1:17:36

there you go we have just created or declared a function and know this

1:17:43

indentation right here so with this curly braces we're telling JavaScript that this is the start of the function

1:17:50

code block and this is the end so whatever is between those curly braces is the code block or logic that will get

1:17:58

executed whenever we reference this function name and just like here we have

1:18:04

this indentation here we have the entire function body indented and webstorm

1:18:09

actually handles the indentation so we don't have to fix anything now you

1:18:15

probably also notice this underlines here this orange underline again

1:18:20

webstorm is helping us understand some improvements that we could make to our

1:18:26

code so this is not an error this would actually work however there is a room

1:18:31

for improvement and let's say as a beginner you see these underlines and

1:18:36

you hover over it and you see this description like implicitly declared and you're thinking I don't know what that

1:18:43

means so I don't know how to fix that warning another really cool thing that webstorm has that I have used

1:18:50

extensively in my programming projects is kind of an Autos suggest for the fix

1:18:57

so what you can do is whenever you see this kind of underline you can use this shortcut that you see right

1:19:04

here which on MacBook is option enter with the mouse inside that code block

1:19:10

that needs to be fixed I'm going to do option enter you see that webstorm

1:19:16

actually gives you suggestions to how to fix that so I'm going to choose the first one and let's see what happens

1:19:23

there you go so you see the warning was about the variable shipping cost not

1:19:28

being declared and webstorm automatically edit the Declaration in the

1:19:34

code so let's put it here and there you go we have no warnings no errors and we

1:19:41

just added an improvement to our code now since we didn't change any values the output of this function should be

1:19:48

the same now let's save this code and let's see what happens when we switch to

1:19:53

the browser let's see what we see there so I'm going to switch to the browser and you see we have an empty output so

1:20:01

what happened here you see we have an empty output so this line of code

1:20:07

obviously didn't get executed because we don't see the sentence in the console output so what happened here we have

1:20:14

exactly the same code as before this time just in a function and that is a very important concept about functions

1:20:22

that you need to understand as a beginner in programming that just like with variables we declare the variable

1:20:29

which means we save the value into the variable reference and then later at

1:20:35

some point we use that variable the same way we first declare the function or we

1:20:42

create the function and we assign whatever code block logic to it and then

1:20:48

later we can use that function or code block inside that function to execute

1:20:54

the logic that means function declaration itself does not execute the code it just saves whatever code you put

1:21:01

inside the function block and you also see webstorm actually highlighting this

1:21:07

function name as gray and if I hover over it it says unused function so we

1:21:14

have declared the function but we're not using it and using a function is also called in programming calling a function

1:21:22

and the way to call a function is function name and the brackets and you

1:21:29

see the color of the function name changed and it doesn't say unused function anymore so now let's save this

1:21:36

and see the output in the browser if I switch back you see that function code

1:21:41

was executed and we see the result right here and the great thing is now whenever

1:21:48

we need this entire code block to execute we simply need to

1:21:54

call the function with the name and that's it so let's say there was some logic

1:22:01

here and somewhere down here we need that logic executed

1:22:06

again we simply call the function and now it's going to execute same code

1:22:11

block for that function call and by the way this is a Syntax for

1:22:18

comments so whenever you start a line in JavaScript with Slash two times you also

1:22:25

see the color is gray whatever you type in

1:22:31

here this tells JavaScript that this is not a code that needs to be executed this is simply a comment a self note for

1:22:39

yourself or for your programming colleagues which usually is used to

1:22:45

document the logic that you write in the function if it's not understandable just

1:22:51

from the code so let's clean this up and let's see another very interesting

**Function Parameters**

1:22:56

concept of

1:23:01

functions now we know that the total price is a dynamic value so every time

1:23:08

customers are shopping that value will be different so we can't hardcode it

1:23:13

here instead it will always be different for each purchase of each customer so we

1:23:20

need to use a different value each time we calculate the shipping cost that means we have to remove it from here and

1:23:28

when we do that you see automatically this warning from webstorm that tells us

1:23:33

this variable is not declared anywhere there is nowhere in the code where we're setting a value for this variable which

1:23:41

means we're referencing a variable that doesn't exist that we haven't created so

1:23:46

we need to do both first we need to create a variable and then we have to assign some value to that so so let's

1:23:52

say at the beginning we have this variable total

1:23:58

price which is a simple Declaration of the variable we're not assigning a value

1:24:04

to it because the value will be dynamically set when a user gets on our

1:24:10

platform and starts shopping right so let's say we have some

1:24:17

logic here that basically tracks the user behavior on a platform let's say a

1:24:22

shopping cart and user is putting the stuff in the shopping cart

1:24:28

and the total price of whatever user is purchasing is now let's say €1 and the

1:24:36

user continues shopping adding more

1:24:43

items and total price gets to €1 19 for example and then at some point the

1:24:50

customer is ready to order they have everything they need

1:24:56

and now they want to basically pay and place the order for all the items so

1:25:01

when that happens we want to calculate the shipping cost and again imagine this

1:25:07

is all code in our application we have tons of logic and at some point we are

1:25:13

calling the calculate shipment cost function to calculate the cost based on

1:25:19

the total price for the customer so we want to do is we want to give that total

1:25:24

price as input to our calculate shipping cost function which in programming is

1:25:31

also called passing a value to a function as a parameter as an input

1:25:37

parameter and that is also super easy we basically take the value or in this case

1:25:43

variable that references the value and pass it between those brackets as an

1:25:49

input parameter to the function and now you understand what the brackets are

1:25:55

there for for a function when we have no input parameters that we're passing to a

1:26:00

function then the brackets are empty or we can pass in one two or 10 parameters

1:26:06

within those brackets and then on the function definition we need to

1:26:11

explicitly say this function is expecting a parameter an

1:26:17

input so whenever someone either us the same engineer who declar the function or

1:26:23

your teammates other Engineers who also want to use the same function they know oh this function is expecting a

1:26:30

parameter which is total price so I need to pass that input because otherwise obviously it can't do its job because

1:26:37

the total price value will not be set and to do that we Define that in the

1:26:43

function definition like this just the name of

1:26:48

the variable and this is another very important thing these two names do not have to be

1:26:56

the same so we're going to call this let's call this total price param or parameter so this is a

1:27:04

variable reference to this value that we're passing here and obviously that's

1:27:11

what we need to use in our logic within our function and something that is very

1:27:17

helpful to understand this concept is that when we call this function with

1:27:22

value 19 passed in as total price variable what happens in the background

1:27:29

in the function execution is the following this variable is created

1:27:35

within the function with a value 19 that's exactly what happens in the

1:27:41

background when we pass a parameter to a function So within the function that is

1:27:47

created as a variable with that value assigned and now we can use that variable within the function block wherever we

1:27:54

need that value and we don't need to do that because it happens in the background and this is our complete

1:28:01

logic so now the total price is dynamic it's calculated somewhere else in our

1:28:07

application code whenever user is adding stuff to the shopping cart and when they

1:28:13

click on check out or ready to order or whatever then we are calling this

1:28:18

function and passing the total price as a parameter so this will be calculated

1:28:24

and we can even add another console log here with our shortcut and

1:28:31

say for total price

1:28:39

of whatever the total price is I'm going to save this go back to the browser and

1:28:47

there you go our logic is fully functional

**Const Use Cases**

1:28:56

now let's do one final Improvement to our shipping cost calculation logic and let's say once a year in our company in

1:29:04

our e-commerce company we change the shipping costs so last year it was five

1:29:11

as a standard shipping cost and for little bit discounted version we had three but this year we want to increase

1:29:17

those prices because of inflation or whatever so in this case the engineers

1:29:23

or us in this case need to change the code to update the shipping cost values

1:29:30

and we may be using these two values in other parts of the application as well

1:29:36

for different logic like here we're doing the shipping cost calculation maybe we have a logic for discounts and

1:29:43

these values are used there as well or in the checkout logic or in the payment

1:29:48

function and so on so if these values change because of our company policy we need to go to our application find every

1:29:56

single JavaScript file and code block where this is hardcoded and we need to

1:30:02

update them and again this is a perfect case for using variables for values that

1:30:08

are referenced in multiple different places and that may change at some point

1:30:13

now in this case this is not a variable that changes in the program like the total price for example it's not a

1:30:20

dynamic value it's a Val that changes less frequently and whenever that happens we manually have to update it in

1:30:28

the code and those are variables that we usually Define with const keyword so

1:30:35

that means right here we're going to create two variables for the standard shipping cost and the discounted one and

1:30:43

we're going to call that exactly that standard shipping

1:30:48

cost let's say we raised it to six 6 and we have

1:30:56

discounted shimy cost which is 4 and as I explained the difference

1:31:04

between these two is that the variables that are defined as const or constant those values cannot be changed

1:31:13

programmatically you can only go here and update the value but the program

1:31:18

itself cannot change it so JavaScript will not allow it so for for example somewhere in the code right here if I

1:31:25

was like let's set it to seven whatever you see this error that says attempting

1:31:31

to assign to constant or read only variable so this is not going to work which again prevents accidental

1:31:40

rewrite of the variables and now we can reference the

1:31:46

constants in our application logic and this is another very important Concept

1:31:53

in programming especially when we're writing functions that as you see we

1:31:59

started with a lot of hardcoded values and then step by step we remove them and

1:32:04

now almost everything in our logic is referenced as variables and we don't

1:32:10

have many hardcoded values here and this is a best practice in programming in

1:32:15

general so now let's test our application again and let's say here the

1:32:20

user started shopping they decided maybe we're ready to check out so we

1:32:26

calculated shipping cost they remembered they forgot one item so they continued shopping eded that in the shopping cart

1:32:33

now they ready to check out and we calculate the shipping cost again so in the first case we're at €10 which means

1:32:40

they're paying the standard shipping cost as we have defined in our logic here with 19 they see discounted

1:32:47

shipping cost so they will get charged the disc discounted shipping cost and when they

1:32:53

see that they think you know what I would like to save myself that shipping cost entirely so I'm going to add one

1:32:59

more item to my shopping cart to qualify for a free shipping with zero cost so

1:33:06

here they add one more

1:33:14

item that brings the total price to let's say

1:33:20

24 they will see the new shipping cost calculated which should be zero and here

1:33:30

they place the order so let's save our logic and see

1:33:36

the output and to differentiate between the outputs let's actually add like a

1:33:42

decoration line kind of thing which basically outputs something like this or

1:33:49

a divider just just a visual divider for us in the console log so let's save it

1:33:55

go back and there you go shipping cost is six for the total price of 10 then we

1:34:02

have four discounted one and here with 24 we are at free

**Naming Conventions**

1:34:10

shipping and one final note that I want to make is you probably ask yourself

1:34:16

about the variable and function names what you see here is called chemel case

1:34:22

so you basically start with a lowercase character and then every other word that

1:34:27

you add in the name is capitalized like this and this is a standard naming

1:34:33

convention for functions and variables in JavaScript specifically know that different programming languages have

1:34:38

their own different conventions that means you can name these actually whatever you want you can

1:34:47

do this or you can do underline

1:34:52

like this all of this will work because there are variable names you can name them as you wish however in JavaScript

1:34:59

this camel case naming is a standard and with that congratulations you have

1:35:05

completed the entire JavaScript crash course you have learned tons of very

1:35:11

important Concepts that are the main building blocks of any programming

1:35:16

language out there and you have a good basic understanding of JavaScript

**Next Steps in your Learning Journey**

1:35:24

however this is just the beginning there are way more exciting and interesting

1:35:29

things awaiting in the JavaScript world and generally in the software development world if you want to

1:35:35

continue and really dive into this world of programming and the best most fun way

1:35:42

to do this will be with our software engineering mini Boot Camp or what we

1:35:48

also call an IT beginners course and I will be more than happy to continue

1:35:53

teaching you there in it beginners course we're going to dive in deeper into the web development you're going to

1:35:59

learn JavaScript Frameworks for front-end development and for backend development you're going to learn how to

1:36:05

build proper websites with database connection and tools for managing dependencies of an application learn

1:36:13

about HTTP requests and protocols as well as how to take the web application

1:36:18

you just created and deploy it on a simple virtual server on a cloud

1:36:24

platform learn about firewall configurations operating system commands

1:36:29

and if any of these Concepts sound completely foreign to you and sound very difficult trust me you will learn all of

1:36:37

this with the simplest explanations and examples that I use throughout the entire course to make this Learning

1:36:44

Journey for you as simple and fun as possible so at the end of our software

1:36:49

engineering miniot camp once you have learned all of this you will have a

1:36:55

solid knowledge equivalent to a junior software engineer so absolutely make

1:37:00

sure to check out our software engineering mini boot camp in the video description or directly linked here in

1:37:06

the video and hope to see you there with that thank you for watching and see you

1:37:12

in the next video