javascript is a programming language

that people love to hate a long time ago

developers used to pride themselves on

being back in developers because it was

more challenging and complex javascript

was just a toy for annoying people on

websites but it's 2019 and today we have

tools like the cloud docker and many API

is that abstract way the complexity of

back-end development if you're starting

up a company today there's a good chance

you'll do the majority of your

development on the front end with

JavaScript so today I want to show you

how to write good JavaScript with modern

features and avoid bad JavaScript if

you're just finding me like and

subscribe and if you have your own

JavaScript pro tips let me know in the

comments one lucky comment will win this

one-of-a-kind t-shirt via livestream

next week

so the first thing you need to know is

how to debug your JavaScript like a pro

and the way we do that of course is by

console logging stuff and I'm saying

that with a straight face there are good

ways to console log and bad ways let's

imagine we have three different objects

each one assigned to its own variable

the obvious way to log these is just one

after the other but the main problem is

we don't know the name of the variable

when this gets logged but there's a

trick we can use here called computed

property names where we add the

variables to an object not only does

this reduce the code footprint but it

also tells us exactly which variable

define this data one line of code one

console log and all the information we

need but maybe this data is extra

important so we want to make it stand

out with some custom CSS styling you can

substitute data and also CSS styles by

using a percent sign so we'll add

percent C and then have the second

argument be our actual CSS style and now

we get this bold orange color in the

console one thing you might have noticed

is that the objects all share common

properties so maybe we could display

those as a table this is really useful

when you have an array of objects just

do console table with the array I could

probably do a whole video on console

logging so I'll just show you a couple

more things here if you're benchmarking

performance you can actually keep track

of time in the console will first define

our timer and we'll give it a name of

looper then we'll set up a while loop

and we'll go through a million

iterations in that loop when that's done

we'll run console time and as you can

see here it takes an average of about 4

to 5 milliseconds to loop a million

times

keeping track of time is great but what

if you need to know where a console.log

originated from let's imagine we have a

really important function that deletes

items from our database and we want to

make sure that we don't accidentally

call this method twice you can add a

console trace to your function and it

will give you a stack trace for where it

was called and what defined it if we run

this code we'll get a console log that

tells us the function was defined on

line 35 and then called on lines 37 and

38 so now that you're a debugging expert

I'm going to show you a few different

ways you can make your code as concise

and efficient as possible let's imagine

we have an object with some animal data

and we need a function that will tell us

how to feed the animal this is just a

function that returns a string so inside

that string will interpolate a few

values from that object this doesn't

look too bad but you'll notice that

we're repeating the word animal over and

over again there's a technique called

object D structuring that we can use to

eliminate most of the repetition here if

we have a function that takes an object

but we only need to use a handful of its

properties we can D structure those in

the argument itself we just wrap it in

brackets and then pass in the names of

the object properties that we want to

use so now we can format the same string

but we never actually have to use the

word animal directly this might seem

like a modest gain on this simple

function but when you have a big object

with a lot of properties this can make a

huge difference and I completely

understand that some people don't like

that bracket syntax in the object

argument so there's actually another way

we can do this which is just as good

this time we pass in the animal object

like we did originally but then we set

up a variable that has the names of the

properties in that object set equal to

the object and now we can use those

properties like variables throughout the

function and this tends to be the better

way to go if you have multiple objects

2d structure and a single function the

next thing we'll look at is template

literals which we've already been using

in the code but there's more to talk

about here when I first started

programming about 10 years ago

jQuery was the cool thing in JavaScript

kind of like react is today but

unfortunately it didn't have all of the

awesome things that we have in

JavaScript now for example you would see

a lot of string concatenation that looks

like this where you have a variable plus

a string and you have to manage the

spaces in between plus an expression

plus a whole bunch of other stuff this

type of code is

incredibly annoying to deal with but

template literals in modern JavaScript

solve this problem completely

instead of concatenating values together

we can actually interpolate them

directly into the string you can do this

by defining your string with backticks

and then use dollar sign brackets and

then whatever variable or expression you

want inside up there so we'll go ahead

and grab the properties we need with

object D structuring that we learned

just a minute ago

and then we'll interpolate those into

the string itself which is a lot more

readable and a lot easier to maintain

but you can actually take things a step

further and build strings in a purely

functional way so we'll write a function

here called horse age that takes in

array of strings as the first argument

and then it can take whatever other

arguments that wants after that we can

look at the arguments to this function

and use them to compose a string so here

we're going to look at the age of the

animal and if it's older than five we'll

say it's old otherwise we'll say it's

young and the last thing we'll do is

return the actual value of the string

that's a pretty standard looking

function but the interesting thing here

is that instead of passing a regular

argument to this function we can

actually just attach it to a template

literal and it will parse the arguments

in it this might seem kind of weird at

first but instead of doing parentheses

with the arguments we'll just attach our

template literal and it will parse all

of the string segments as an array of

strings as the first argument to the

function that we defined then it will

handle all the other arguments in the

order in which they appear inside of the

dollar sign brackets in other words you

can take a single argument and use it to

compose multiple values in the return

string this can be a very powerful

concept for templating and it's actually

used in the polymer project now via a

library called lit HTML so now that we

know how to work with strings we're

going to move on to the spread syntax to

work with objects and arrays let's

imagine we have one object for a Pokemon

and the other one for the stats that

define its various attributes let's say

we want to assign the properties from

the stats object to the Pikachu object

one way to do that is to just redefine

them one by one on the original Pikachu

object for one this is just really ugly

and verbose but we're also mutating the

original object when we most likely want

to create a new immutable object because

let's say that our Pokemon

levels up over time we want to represent

each level up as its own object we could

use object to sign here and take the

original object and merge it in with the

stats and this will merge them together

from left to right or if we just wanted

to update a single property we could add

an object with that property in it this

isn't too bad but there's a more concise

way to do this with the spread syntax by

creating a new object and placing our

existing objects in it with three dots

in front of them it will compose a new

object from left to right so the

property is farthest to the right will

have the priority again this is mostly

just syntactic sugar and it just makes

your code more readable and easier to

maintain and it's also possible to use

the spread syntax on arrays so let's

imagine we have an array of strings and

we need to push additional items to this

array the old-school way to do this

would be to just push new items to the

array one by one but in today's world we

can reduce these three lines of code to

just one by defining an array with the

new items and in the spread syntax on

the original array if we add the three

dots to the beginning then it's the

equivalent of doing an array push

because it will append the new items to

the end of the array but the nice thing

is we could just add this to the end of

the array and then we have the

equivalent of a race shift and we might

even take the original values and just

splice them in the middle of the array

giving us even more flexibility and this

code brings back some memories you see

this little trailing comma here this

used to be the kind of thing that would

break an entire JavaScript program and

be really difficult to figure out but

luckily in modern JavaScript it just

works and it's actually kind of

considered a good practice because you

can reduce the number of lines that

change when you do get commits now it's

time to move on to loops let's imagine

we have an array of numbers here that

represent the order totals that we've

had throughout the day in our app now

let's say we need to compute some values

based on this array such as the order

total maybe we need to add some tax to

each one and filter out the high value

orders to be reviewed by a manager one

option is to just use a classic for loop

like you'll find in pretty much every

programming language we have an integer

that starts at zero while that integer

is less than the orders length we will

increment it by one

personally I hate loops that look like

this and I almost never use them in

JavaScript but while we're in the loop

we can start computing

values for the total we will just do

plus equals with the order total then to

create a new array with the tax added to

it we'll go ahead and take the order

amount and multiply it by 1.1 to add 10%

tax then if we have order values that

are greater than 100 we'll go ahead and

add those to the high value order array

this code is a very ugly and B it's

mutating values that might make our code

a little more unpredictable luckily we

can reduce this down to just three lines

of code by using modern JavaScript array

methods if we want to take an array and

then have it accumulate to a value that

equals say a total amount we can use

array reduce it takes a callback

function as the argument where the first

argument is the accumulated value and

the second argument is the current value

in the loop so if we want to sum up all

of the items in the array we can just do

the accumulated values plus the current

value and when the loop finishes that

will give us the total of all elements

in that array mapping and filtering

values is even easier if we want to add

tax to all the items in the array we can

just take the values in the array then

map them to their value times 1.1 and

lastly we can use filter to create an

array that only has the values greater

than 100 in it whenever the callback

function equals true it's going to allow

a value through so in other words if the

value is greater than 100 it will allow

that value through to the new array I

wanted to save the best for last and

that of course is async/await let's

create a method called random that

returns a promise that resolves to a

random number asynchronously now let's

imagine that we want to retrieve three

different asynchronous numbers one after

the other and then add them all together

at the end that might seem like a silly

example but that's actually how things

work a lot of times in the real world

when you have to retrieve one item from

the database get some data retrieving

another item from an API and so on with

promises you wait for an asynchronous

value to resolve and then you handle it

with a callback function inside of then

once you have your data you can return

another promise and then chain another

then call back to it and continue this

pattern on for as long as you need to

you end up with this ridiculous looking

code where you keep saying and then

and then and then but fortunately

there's a really nice solution to this

which is async/await

basically it allows us to express a

singer s code in a synchronous format we

can come down here and rewrite our

promise chain the only difference is

adding async in front of the function

which will force it to return a promise

but the real benefit here is that we can

use a weight in front of our promises

and have them resolve to an actual

variable value so instead of using those

then callbacks we can just say Const

first equals await random and do the

same thing for the second and third

number and now it's much easier to read

and understand this code because we can

just go line by line and see that we're

waiting one number awaiting another

number and so on

a sync wait is one of the most awesome

things to ever happen to JavaScript and

it really deserves its own video right

now we're out of time but in the future

I'll show you some of the cool things

you can do with it like use it in

conditional statements or use it in for

loops and things like that I'm gonna go

ahead and wrap things up there if this

video helped you please like and

subscribe and make sure to check out the

livestream next week to see if you want

some free swag and if you're serious

about building apps consider becoming a

pro member at angular firebase comm

you'll get all kinds of exclusive

content designed to help you build and

ship your app faster thanks for watching

and I'll talk to you soon