In 1886, the first internal combustion engine-powered car

was invented by Karl Benz.

Soon, there were many cars

that used the combustion engine,

airplanes, motorbikes, and boats followed shortly after.

People's ingenuity led to using

the combustion engine in

places that the inventor might not have imagined.

The same happened to JavaScript.

New JavaScript environments such as

Node and npm are now available.

JavaScript's home is in the browser.

That's the environment in which it

was used for several years.

Essentially, it was a front end only language.

Then in 2009, a developer named Ryan Dao

decided to use Google's JavaScript V8 engine

and make it work on the server.

It's not always easy being an innovator.

Many doubted whether it was even possible.

However, the idea caught

on and more people started getting on board.

This is how Node.js was born and how

JavaScript became a language

for both front-end and back-end.

Node.js is a separate standalone environment.

This means that Node.js can run in multiple settings.

For example, on the command line,

in a desktop application,

or on the back end of a web app.

Before the introduction of Node.js,

developers had to build backends in

other technologies and languages such as PHP,

Python, C-sharp, Ruby, and Java.

After Node.js became available,

it was possible to use JavaScript on

the backend or on the server-side.

This means that today you can

write full-stack JavaScript programs.

In other words, you can write

JavaScript on the client and on the server.

Node.js comes with a package manager called npm,

which stands for Node Package Manager.

The package manager allows you to use a large number of

libraries and frameworks as Node.js modules.

An npm module is a standalone piece of

code that has been published on the npm website.

Sometimes an npm module is

also referred to as an npm package.

Now that you've learned about Node.js,

you may be wondering how you can use it locally.

Node.js and npm are either pre-installed on your machine,

or you need to install them.

Once installed, you can interact with

Node.js and npm from the command line.

For example, you can run

the node command inside your computer's command line.

This is also called a shell, or a terminal.

In the same way, you can run the npm command.

You may be wondering why you need to run those commands.

Well, this is because you use

the node command to run a JavaScript file,

or to directly execute JavaScript code.

You use the npm command to install

any node module from the npm repository.

For example, you could install

the module named lowercase.

Once installed, you can import this module and use

its functions to transform

strings to lowercase in your code.

When you want to start a new project, first,

open a folder on

your machine where you want to place your project files,

then run the npm command.

These projects can be different shapes and sizes,

but they all have at least one thing in common,

the package.json file that gets

created after you run the npm command.

The package.json file holds all the instructions on

all the node modules that are pulled from

the npm repository of open source modules.

There are about 11 million modules in the npm repository.

It means that you can get thousands of hours worth of

other developers' coding by running

the npm install command and adding the package name.

Examples of libraries you can install include React,

Webpack, Bootstrap, and Angular Core.

The package.json file updates

when you install a new package.

It keeps track of everything you need

to have installed in your project.

This makes such projects easily portable.

For example, if you have built

a project with a specific number

of different node packages,

they're all listed inside the package.json file.

All you need to do is share this file with,

for example, your co-workers.

They can have the exact same setup on their machines

simply by running the command npm install.

This install command reads the contents of the

package.json and installs all the necessary packages,

also referred to as dependencies.

Sometimes dependencies also come

with their own dependencies.

It often happens that when you run

the npm install command,

several 100 megabytes worth of node packages get

installed into your project

under the node modules folder.

In this video, you learned about the origins of Node.js.

You should also be able to describe what Node.js and npm

are and how to use

the basic Node and npm commands on the command line.