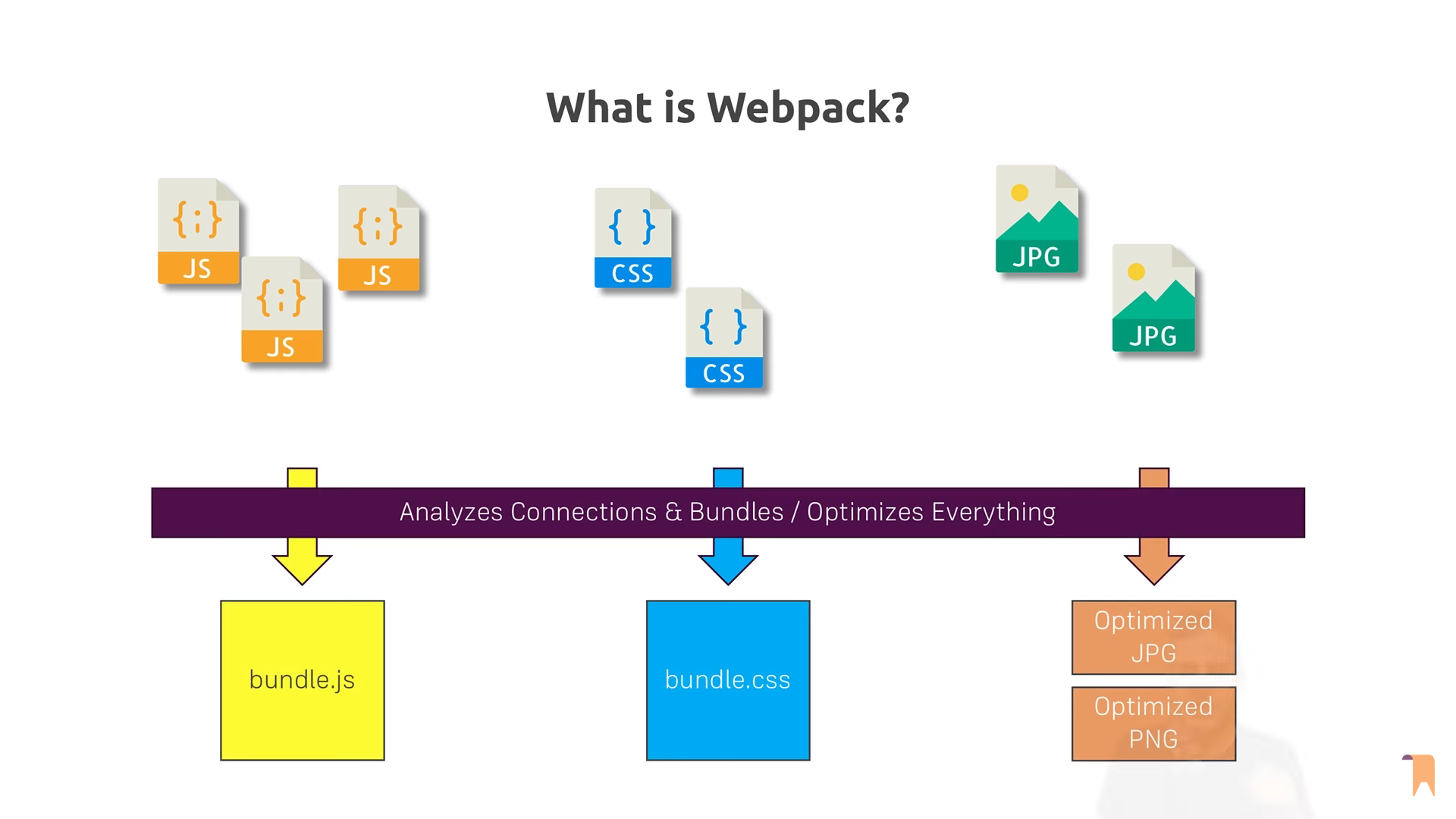
**Webpack**

**What is Webpack?**

**Webpack** is a famous module builder that we extensively use during our Web Development Journey. It is used for those applications that use Javascript.

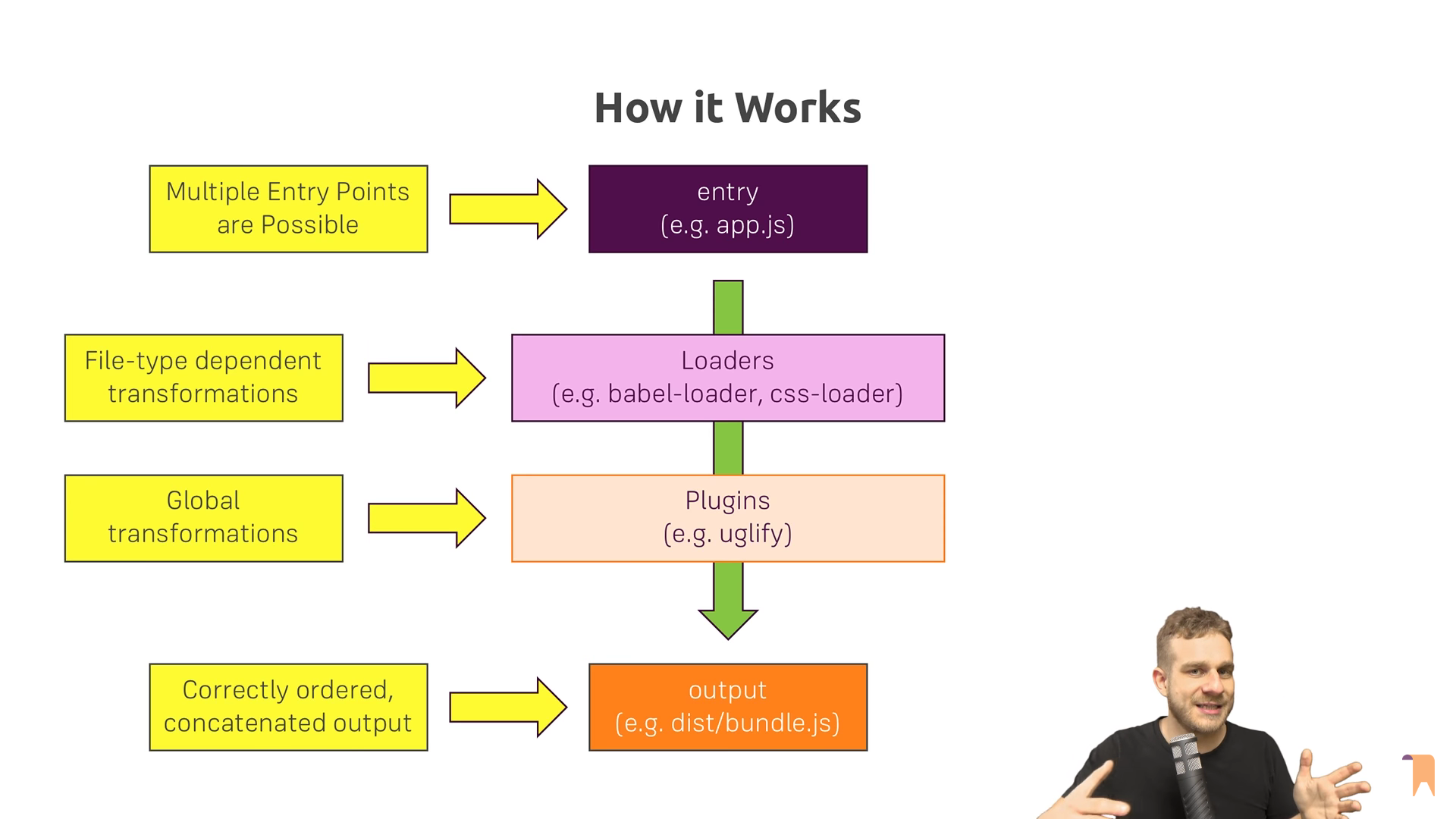
Before moving forward, we should note that Webpack only understands JavaScript and JSON. So it converts other frontend files like HTML and CSS into modules with the help of a loader and thus provides a complete frontend solution to us. It internally makes a dependency graph while processing any application.

****

**Working of Webpack:**

There are many features of webpack:

* **Entry:**We all know webpack makes a dependency graph and the starting of this graph is known as the entry or entry point. From the starting point of the dependency graph, it will follow all the dependencies to know what it has to bundle.
* **Output:**Output tells webpack where to put the bundles that it had made and what will be its format.
* **Loaders:**Loaders convert different types of files like images and CSS into a module before adding them to the dependency graph.
* **Plugins:**Plugins provide functionality. It can provide much functionality like printing something on running the webpack, minifying, optimization of bundles, etc.



**Steps to Install Webpack:**

If you want to use web pack then first you need to install that. So for installing the webpack using the node package manager use this command in the terminal.

npm i --save-dev webpack

**Application of Webpack:**

* It helps to use different JavaScript files without having tension that will load first.
* It makes code shorter.
* It helps in converting many files other than JavaSctript into modules.
* It compiles different JavaScript module.

**Module Federation:**

In Webpack, module federation is a feature that allows you to share code between separately built JavaScript bundles. It enables you to dynamically load code from one bundle into another at runtime. This is particularly useful for microfrontend architectures where multiple teams are working on separate codebases that need to be integrated into a single application.

With module federation, you can expose certain modules from one webpack bundle and consume them in another webpack bundle. This allows you to create a decentralized system where each application or microfrontend can manage its own dependencies and still share code with other parts of the application.

**Here's a basic example of how you might set up module federation in a webpack configuration:**

// webpack.config.js for the host application

const ModuleFederationPlugin = require("webpack/lib/container/ModuleFederationPlugin");

module.exports = {

// other webpack config options...

plugins: [

new ModuleFederationPlugin({

name: "app",

remotes: {

otherApp: "otherApp@http://localhost:3001/remoteEntry.js",

},

shared: ["react", "react-dom"],

}),

],

};

**name:** The name In this example under which this module is exposed.

**remotes:** Other applications or modules that this application will consume.

**shared:** An array of modules that will be shared between the applications.

And in the remote application (otherApp):

// webpack.config.js for the remote application

const ModuleFederationPlugin = require("webpack/lib/container/ModuleFederationPlugin");

module.exports = {

// other webpack config options...

plugins: [

new ModuleFederationPlugin({

name: "otherApp",

library: { type: "var", name: "otherApp" },

filename: "remoteEntry.js",

exposes: {

"./Button": "./src/Button",

},

shared: ["react", "react-dom"],

}),

],

};

In this example:

**name:** The name under which this module is exposed.

**exposes:** The modules that will be exposed to other applications.

**shared:** An array of modules that will be shared between the applications.

This setup allows the app to load the Button component from the otherApp bundle at runtime.