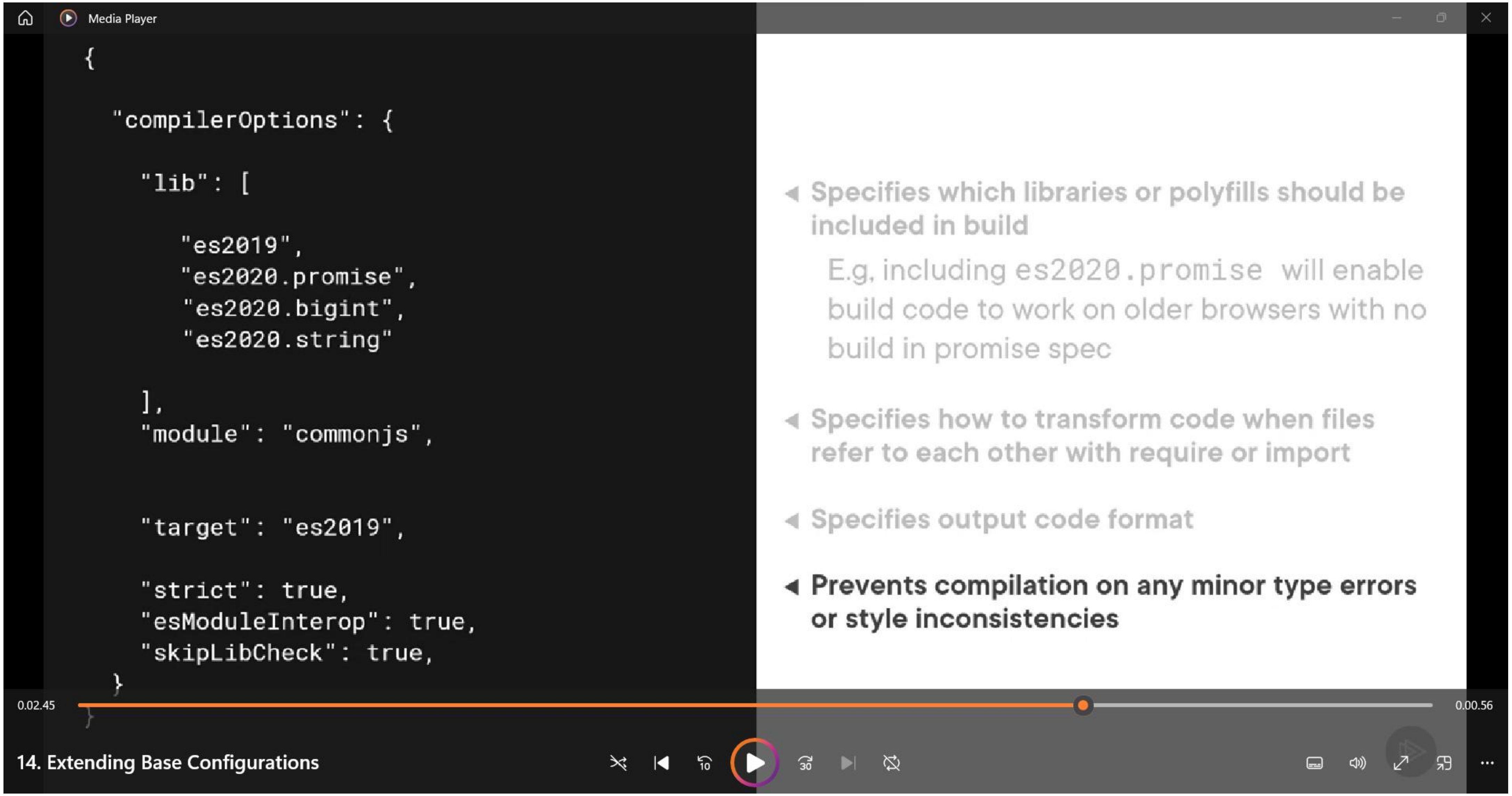

Media Player

```
{
  "compilerOptions": {
    "lib": [
      "es2019",
      "es2020.promise",
      "es2020.bigint",
      "es2020.string"
    ],
    "module": "commonjs",

    "target": "es2019",

    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
  }
}
```

- Specifies which libraries or polyfills should be included in build

  E.g, including `es2020.promise` will enable build code to work on older browsers with no build in promise spec

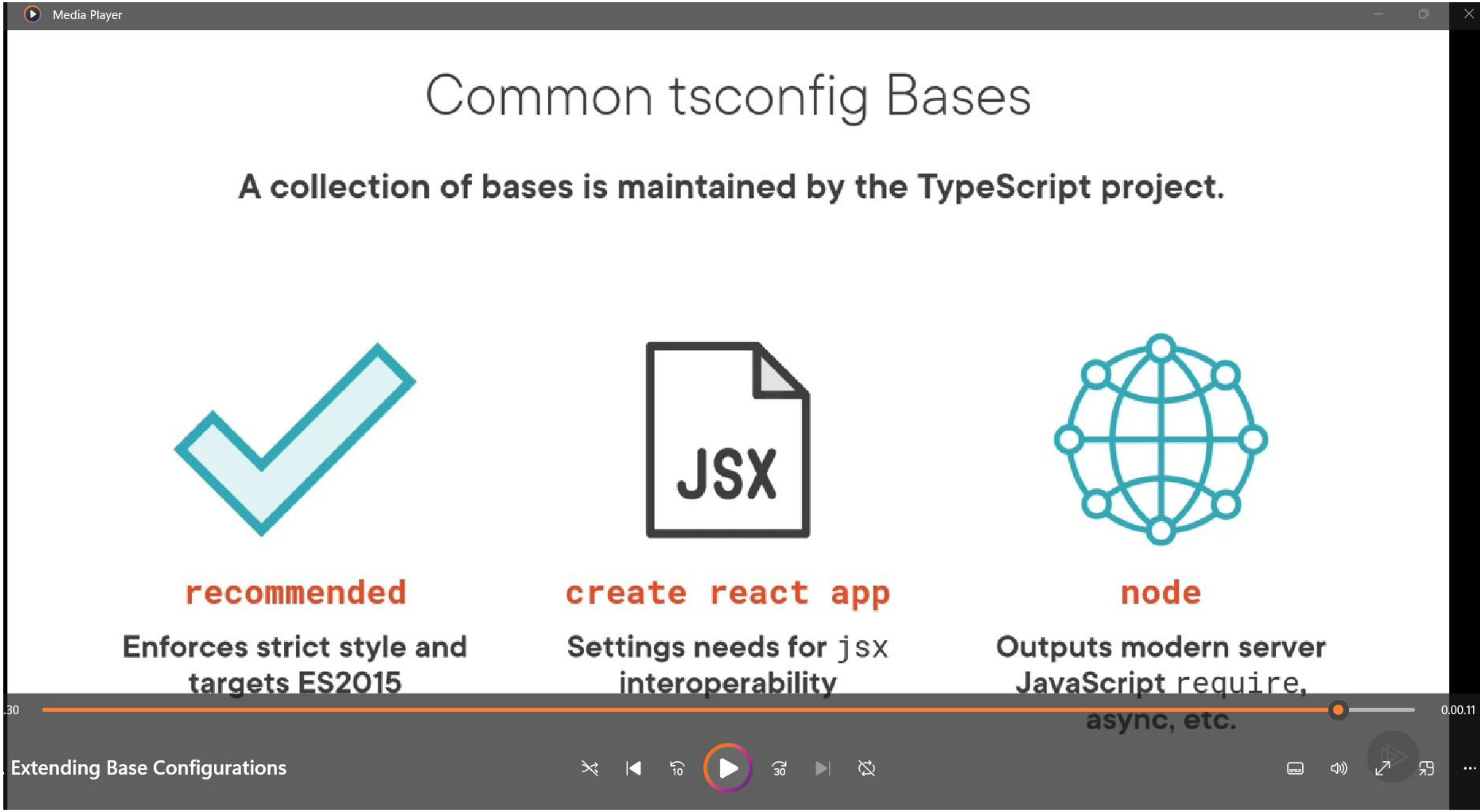
- Specifies how to transform code when files refer to each other with require or import

- Specifies output code format

- **Prevents compilation on any minor type errors or style inconsistencies**

0.02.45 0.00.56

14. Extending Base Configurations



Common tsconfig Bases

A collection of bases is maintained by the TypeScript project.

recommended

Enforces strict style and targets ES2015

create react app

Settings needs for `jsx` interoperability

node

Outputs modern server JavaScript `require`, async, etc.

Multi- and Single-file Compilation

Multi-file Compilation	Single-file Compilation
Creates one JavaScript file for every target TypeScript file	Combines all TypeScript files into one single JavaScript file
Each file must be loaded for the application to work in a browser	Only a single file must be loaded for the application to work in a browser
Files must be concatenated or use require to work in Node.js	Single file will work when invoked as a Node script
Possible to update just one generated file in production	Updated production code must be pushed in its entirety
Standard compilation option for TypeScript	Additional tooling (Webpack, Babel) needed

Single-file Compilation for Majority of Tasks

Greater support for isomorphic applications

Fewer HTTP requests, simpler deployment to web applications

Greater consistency across browser / Node versions

Compiling a TypeScript application to a single file generally makes it easier to deploy as both a web and a server-side application.