Type checking in JavaScript files with `// @ts-check` and `--checkJs`

TypeScript has long had an option for gradually migrating your files from JavaScript to TypeScript using the `--allowJs` flag; however, one of the common pain-points we heard from JavaScript users was that migrating JavaScript codebases and getting early benefits from TypeScript was difficult. That's why in TypeScript 2.3, we're experimenting with a new "soft" form of checking in `.js` files, which brings many of the advantages of writing TypeScript without actually writing `.ts` files.

This new checking mode uses comments to specify types on regular JavaScript declarations. Just like in TypeScript, these annotations are *completely optional*, and inference will usually pick up the slack from there. But in this mode, your code is still runnable and doesn't need to go through any new transformations.

You can try this all out without even needing to touch your current build tools. If you've already got TypeScript installed (`npm install -g typescript`), getting started is easy! First create a `tsconfig.json` in your project's root directory:

```
{
    "compilerOptions": {
        "noEmit": true,
        "allowJs": true
    },
    "include": [
        "./src/"
    ]
}
```

> Note: We're assuming our files are in `src`. Your folder names might be different.

Now all you need to do to type-check a file is to add a comment with `// @ts-check` to the top. Now run `tsc` from the same folder as your `tsconfig.json` **and that's it**.

```
// @ts-check

/**
 * @param {string} input
 */
function foo(input) {
    input.tolowercase()
    //    ~~~~~~~~~~~ Error! Should be toLowerCase
}
```

We just assumed you didn't want to bring TypeScript into your build pipeline at all, but TypeScript is very flexible about how you want to set up your project. Maybe you wanted to have *all* JavaScript files in your project checked with the `checkJs` flag instead of using `// @ts-check` comments. Maybe you wanted TypeScript to also compile down your ES2015+ code while checking it. Here's a `tsconfig.json` that does just that:

```
{
    "compilerOptions": {
        "target": "es5",
        "module": "commonjs",
        "allowJs": true,
        "checkJs": true,
        "outDir": "./lib"
    },
    "include": [
        "./src/**/*"
    ]
}
```

> Note: Since TypeScript is creating new files, we had to set `outDir` to another folder like `lib`. That might not be necessary if you use tools like Webpack, Gulp, etc.

Next, you can start using TypeScript declaration files (`.d.ts` files) for any of your favorite libraries and benefit from any new code you start writing. We think you'll be especially happy getting code completion and error checking based on these definitions, and chances are, you may've already tried it.

This JavaScript checking mode also allows for two other comments in `.js` files:

1. `// @ts-nocheck` to disable a file from being checked when `--checkJs` is on
2. `// @ts-ignore` to ignore errors on the following line.

You might already be thinking of this experience as something similar to linting; however, that doesn't mean we're trying to replace your linter! We see this as something complementary that can run side-by-side with existing tools like ESLint on your JavaScript. Each tool can play to its strengths.

If you're already using TypeScript, we're sure you have a JavaScript codebase lying around you can turn this on to quickly catch some real bugs in. But if you're new to TypeScript, we think that this mode will really help show you what TypeScript has to offer without needing to jump straight in or commit.