Extended Diagnostics

There are many coding patterns that are technically valid to the compiler or runtime, but which may have complex nuances or caveats. These patterns may not have the intended effect expected by a developer, which often leads to bugs. The Angular compiler includes "extended diagnostics" which identify many of these patterns to warn developers about the potential issues and enforce common best practices within a codebase.

Diagnostics

Currently, Angular supports the following extended diagnostics:

- NG8101 - `invalidBananaInBox`
- NG8102 - `nullishCoalescingNotNullable`

Configuration

Extended diagnostics are warnings by default and do not block compilation. Each diagnostic can be configured as either:

- `warning` (default) - The compiler emits the diagnostic as a warning but does not block compilation. The compiler will still exit with status code 0, even if warnings are emitted.
- `error` - The compiler emits the diagnostic as an error and fails the compilation. The compiler will exit with a non-zero status code if one or more errors are emitted.
- `suppress` - The compiler does *not* emit the diagnostic at all.

Check severity can be configured in the project's `tsconfig.json` file:

```
{
  "angularCompilerOptions": {
    "extendedDiagnostics": {
      // The categories to use for specific diagnostics.
      "checks": {
        // Maps check name to its category.
        "invalidBananaInBox": "suppress"
      },

      // The category to use for any diagnostics not listed in `checks` above.
      "defaultCategory": "error"
    }
  }
}
```

The `checks` field maps the name of individual diagnostics to their associated category. See Diagnostics for a complete list of extended diagnostics and the name to use for configuring them.

The `defaultCategory` field is used for any diagnostics that are not explicitly listed under `checks`. If not set, such diagnostics will be treated as `warning`.

Extended diagnostics will emit when `strictTemplates` is enabled. This is required to allow the compiler to better understand Angular template types and provide accurate and meaningful diagnostics.

Semantic Versioning

The Angular team intends to add or enable new extended diagnostics in **minor** versions of Angular (see semver). This means that upgrading Angular may show new warnings in your existing codebase. This enables the team to deliver features more quickly and to make extended diagnostics more accessible to developers.

However, setting `"defaultCategory": "error"` will promote such warnings to hard errors. This can cause a minor version upgrade to introduce compilation errors, which may be seen as a semver non-compliant breaking change. Any new diagnostics can be suppressed or demoted to warnings via the above configuration, so the impact of a new diagnostic should be minimal to projects that treat extended diagnostics as errors by default. Defaulting to error is a very powerful tool; just be aware of this semver caveat when deciding if `error` is the right default for your project.

New Diagnostics

The Angular team is always open to suggestions about new diagnostics that could be added. Extended diagnostics should generally:

- Detect a common, non-obvious developer mistake with Angular templates.
- Clearly articulate why this pattern can lead to bugs or unintended behavior.
- Suggest one or more clear solutions.
- Have a low (preferrably zero) false-positive rate.
- Apply to the vast majority of Angular applications (not specific to an unofficial library).
- Improve program correctness or performance (not style, that responsibility falls to a linter).

If you have an idea for a compiler check which fits these criteria, consider filing a feature request.