GN Check

GN has several different ways to check dependencies. Many of them are checked by the `gn check` command. Running checks involve opening and scanning all source files so this isn't run every time a build is updated. To run check on an existing build:

```
gn check out/mybuild
```

To run the check as part of the "gen" command to update the build (this is what the bots do):

```
gn gen out/mybuild --check
```

Contents

Concepts

Visibility

Targets can control which other targets may depend on them by specifying `visibility`. Visibility is always checked when running any GN command (not just `gn check`.

By default, targets are "public" meaning any target can depend on them. If you supply a list, visibility will be listed to those targets (possibly including wildcards):

```
visibility = [
  ":*",  # All targets in this file.
  "//content/*",  # All targets in content and any subdirectory thereof.
  "//tools:doom_melon",  # This specific target.
]
```

See `gn help visibility` for more details and examples.

Public header files

Targets can control which headers may be included by dependent targets so as to define a public API. If your target specifies only `sources`, then all headers listed there are public and can be included by all dependents.

If your target defines a `public` variable, only the files listed in that list will be public. Files in `sources` but not `public` (they can be in both or only one) may not be included by dependent targets.

```
source_set("foo") {
  public = [
    "foo.h",
    "foo_config.h",
  ]
  sources = [
    "foo.cc",
    "foo.h",
    "bar.cc",
    "bar.h",
  ]
}
```

Public dependencies

In order to include files from your target, that target must be listed in your target's dependencies. By default, transitively depending on a target doesn't give your files this privilege.

If a target exposes a dependency as part of its public API, then it can list that dependency as a `public_deps`:

```
source_set("foo") {
  sources = [ ... ]
  public_deps = [
    "//base",
  ]
  deps = [
    "//tools/doom_melon",
  ]
}
```

Targets that depend on `foo` can include files from `base` but not from `doom_melon`. To include public headers from `doom_melon, a target would need to depend directly on it.

Public dependencies work transitively, so listing a target as a public dependency also exposes that target's public dependencies. Along with the ability to include headers, public dependencies forward the `public_configs` which allow settings like defines and include directories to apply to dependents.

Putting it all together

In order to include a header from target Y in a file that is part of target X:

- X must be in Y's `visibility` list (or B must have no `visibility` defined).
- The header must be in Y's `public` headers (or Y must have no `public` variable defined).
- X must depend directly on Y, or there must be a path from X to Y following only public dependencies.

What gets checked

Chrome currently doesn't come close to passing a `gn check` pass. You can check specific targets or subtrees for issues:

```
gn check out/mybuild //base

gn check out/mybuild "//mojo/*"
```