Storing Metadata in `.d.ts` files

Previous version of Angular used `metadata.json` files to store information about directives/component/pipes/ng-modules. `ngc` compiler would than do a global analysis to generate the `.ngfactory.ts` files from the `metadata.json`. Ivy strives for locality, which means that `ngtsc` should not need any global information in order to compile the system. The above is mostly true. Unfortunately, in order for `ngtsc` to generate code which is tree shakable `ngtsc` does need to have global knowledge.

Here is an abbreviated example of breakage of tree-shake-ability.

```
@Directive({
  selector: '[tooltip]'
})
export class TooltipDirective {
  // ngtsc generates this:
  static ɵdir = ɵɵdefineDirective(...);
}

@Component({
  selector: 'app-root',
  template: 'Hello World!'
})
class MyAppComponent {
  // ngtsc generates this:
  static ɵdir = ɵɵdefineComponent({
    ...
    directives: [
      // BREAKS TREE-SHAKING!!!
      // TooltipDirective included here because it was declared in the NgModule
      // ngtsc does not know it can be omitted.
      // Only way for ngtsc to know that it can omit TooltipDirective is if it knows
      // its selector and see if the selector matches the current component's
template.
      TooltipDirective
    ]
  });
}

@NgModule({
  declarations: [MyAppComponent, TooltipDirective],
  bootstrap: [MyAppComponent],
})
class MyAppModule {
    // ngtsc generates this:
  static ɵmod = ɵɵdefineNgModule(...);
}
```

Notice that `ngtsc` can't remove `TooltipDirective` because it would need to know its selector and see if the directive matches in the component's template. Knowing the selector breaks locality and so we make an exception for some locality information such as selector, inputs and outputs. Since we are breaking the locality rule, we need to

store the information someplace since `ngtsc` can't have access to the `TooltipDirective` source. We store the information in the `.d.ts` file like so.

```
class TooltipDirective {
  static ɵdir: DirectiveDeclaration<TooltipDirective, '[tooltip]', '', {}, {}, []>
}
```