Overview

Ivy is a new backwards-compatible Angular renderer focused on further speed improvements, size reduction, and increased flexibility.

Ivy is currently not feature complete, but can be tested via `enableIvy: true` `angularCompilerOptions` flag.

We currently expect Ivy to remain behind the flag until it's feature complete and battle tested at Google. In the meantime you can check out this Hello World demo.

Implementation Status

The work can be divided into three categories:

- `@angular/compiler-cli` : TypeScript transformer pipeline which includes two command line tools:
  - `ngtsc` : (Angular TypeScript Compiler) Angular compiler which strips out `@Component` (and friends) and replaces it with `ɵɵdefineComponent` (and friends).
  - `ngcc` : (Angular Compatibility Compiler) NPM upgrade compiler which reads the `STORING_METADATA_IN_D.TS.json` files and `.js` files and adds `ɵɵdefineComponent` (and friends) into the `node_module` . This in effect converts a pre-ivy module into ivy module.
- `@angular/compiler` : Ivy Compiler which converts decorator into ivy
- `@angular/core` : Decorators which can be patched with `@angular/compiler` .

`@angular/compiler-cli` changes

`ngtsc` TSC compiler transformer

TSC transformer which removes and converts `@Pipe` , `@Component` , `@Directive` and `@NgModule` to the corresponding `ɵɵdefinePipe` , `ɵɵdefineComponent` , `ɵɵdefineDirective` and `ɵɵdefineInjector` .

- ✅ Basic setup of the transformer into `tsc`
- ✅ Can read STORING_METADATA_IN_D.TS from `.d.ts` (see: STORING_METADATA_IN_D.TS.md)
- ✅ Detect decorators and convert them to the `defineXXX` method using the `__Compiler` in `@angular/compiler` .
- ✅ Encode selectors into `.d.ts` file.
- ✅ support `extends` for `@Pipe` , `@Component` , `@Directive` and `@NgModule` .
- ❌ Documentation

`ngcc` Angular `node_module` compatibility compiler

A tool which "upgrades" `node_module` compiled with non-ivy `ngc` into ivy compliant format.

- ✅ Basic setup of stand alone executable
- ✅ Rewrite existing code by interpreting the associated STORING_METADATA_IN_D.TS
- ❌ Integration with WebPack (cli)
- ❌ Documentation

`@angular/compiler` changes

- ✅ Component compilation: Translates `@Component` `=>` `ɵɵdefineComponent`
  - ✅ `TemplateCompiler` (current known as `ViewCompiler`)
  - ✅ `StyleCompiler`
- ✅ `PipeCompiler` : Translates `@Pipe` `=>` `ɵɵdefinePipe`
- ✅ `DirectiveCompiler` : Translates `@Directive` `=>` `ɵɵdefineDirective`
- ✅ `InjectableCompiler` : Translates `@Injectable` `=>` `ɵɵdefineInjectable`
- ✅ `NgModuleCompiler` : Translates `@NgModule` `=>` `ɵɵdefineInjector` (and `ɵɵdefineNgModule` only in jit)
- ❌ Documentation

`@angular/core` changes

The goal is for the `@Component` (and friends) to be the compiler of template. Since decorators are functions which execute during parsing of the `.js` file, the decorator can compile the template into Ivy. The AOT compiler's job is to remove the `@Component` and replace it with call to `ɵɵdefineComponent`.

- ✅ `@angular/compiler` can patch itself onto:
  - ✅ `@Injectable`
  - ✅ `@NgModule`
  - ✅ `@Pipe`
  - ✅ `@Directive`
  - ✅ `@Component`
- ✅ `ResourceLoader.resolved: Promise<>` Returns true if all `templateUrl`s and `styleUrl` have been resolved and application is ready to be bootstrapped.

Testing / Debugging

- ✅ in debug mode publish components into DOM nodes for easier debugging.

Crosscutting

Decorators

Annotation	`defineXXX()`	Run time	Spec	Compiler
`@Component`	✅ `ɵɵdefineComponent()`	✅	✅	✅
`@Directive`	✅ `ɵɵdefineDirective()`	✅	✅	✅
`@Directive`	✅ `ɵɵdefineBase()`	✅	✅	✅
`@Pipe`	✅ `ɵɵdefinePipe()`	✅	✅	✅
`@Injectable`	✅ `ɵɵdefineInjectable()`	✅	✅	✅
`@NgModule`	✅ `ɵɵdefineInjector()`	✅	✅	✅
`@ConfigureInjector`	✅ `ɵɵdefineInjector()`	❌	❌	❌

Component Composition

Feature	Runtime	Spec	Compiler
creation reordering based on injection	✅	✅	✅
`class CompA extends CompB {}`	✅	✅	✅
`class CompA extends CompB { @Input }`	✅	✅	✅
`class CompA extends CompB { @Output }`	✅	✅	✅

Change Detection

Feature	Runtime
`markDirty()`	✅
`detectChanges()`	✅
`tick()`	✅
`attach()`	✅
`detach()`	✅
`ON_PUSH`	✅
`ALWAYS`	✅
`DIRTY`	✅
`ATTACHED`	✅

Bootstrap API

Feature	Runtime
`renderComponent()`	✅
`getHostElement()`	✅
`createInjector()`	✅

Template Compiler

Template Syntax

Feature	Runtime	Spec	Compiler
`<div>`	✅	✅	✅
`<div>{{exp}}</div>`	✅	✅	✅
`<div attr=value>`	✅	✅	✅

`<div (click)="stmt">`	✅	✅	✅
`<div #foo>`	✅	✅	✅
`<div #foo="bar">`	✅	✅	✅
`<div [value]="exp">`	✅	✅	✅
`<div title="Hello {{name}}!">`	✅	✅	✅
`<div [attr.value]="exp">`	✅	✅	❌
`<div class="literal">`	✅	✅	✅
`<div [class]="exp">`	✅	✅	✅
`<div [class.foo]="exp">`	✅	✅	✅
`<div style="literal">`	✅	✅	✅
`<div [style]="exp">`	✅	✅	✅
`<div [style.foo]="exp">`	✅	✅	✅
`<div xmlns:foo="url" foo:bar="baz">`	✅	✅	✅
`{{ ['literal', exp ] }}`	✅	✅	✅
`{{ { a: 'literal', b: exp } }}`	✅	✅	✅
`{{ exp | pipe: arg }}`	✅	✅	✅
`<svg:g svg:p>`	✅	✅	✅
`<img src=[userData]>` sanitization	✅	✅	✅
`<div (directiveOut)>`	✅	✅	✅
`<ng-template (directiveOut)>`	✅	✅	✅
`<ng-container>`	✅	✅	✅

Life Cycle Hooks

Feature	Runtime	Spec	Compiler
`onChanges()`	✅	✅	✅
`onDestroy()`	✅	✅	✅
`onInit()`	✅	✅	✅
`onChanges()`	✅	✅	✅
`doCheck()`	✅	✅	✅
`afterViewChecked()`	✅	✅	✅
`afterViewInit()`	✅	✅	✅

`afterContentChecked()`	✅	✅	✅
`afterContentInit()`	✅	✅	✅
listener teardown	✅	✅	✅

`@Query`

Feature	Runtime	Spec	Compiler
`@Query(descendants)`	✅	✅	n/a
`@Query(one)`	✅	✅	n/a
`@Query(read)`	✅	✅	n/a
`@Query(selector)`	✅	✅	n/a
`@Query(Type)`	✅	✅	n/a
`@ContentChildren`	✅	✅	✅
`@ContentChild`	✅	✅	✅
`@ViewChildren`	✅	✅	✅
`@ViewChild`	✅	✅	✅

Content Projection

Feature	Runtime	Spec	Compiler
`<ng-content>`	✅	✅	✅
`<ng-content selector="...">`	✅	✅	✅
container `ngProjectAs`	✅	✅	✅

Injection Features

Feature	Runtime	Spec	Compiler
`inject(Type)`	✅	✅	✅
`directiveInject(Type)`	✅	✅	✅
`inject(Type, SkipSelf)`	❌	❌	❌
`attribute('name')`	✅	✅	❌
`injectChangeDetectionRef()`	✅	✅	✅
`injectElementRef()`	✅	✅	✅
`injectViewContainerRef()`	✅	✅	✅
`injectTemplateRef()`	✅	✅	✅

`injectRenderer2()`	✅	✅	✅
default `inject()` with no injector	✅	✅	✅
sanitization with no injector	✅	✅	✅

I18N

Feature	Runtime	Spec	Compiler
i18nStart	✅	✅	✅
i18nEnd	✅	✅	✅
i18nAttributes	✅	✅	✅
i18nExp	✅	✅	✅
i18nApply	✅	✅	✅
ICU expressions	✅	✅	✅
closure support for g3	✅	✅	✅
`<ng-container>` support	✅	✅	✅
runtime service for external world	❌	❌	❌
migration tool	❌	❌	❌

View Encapsulation

Feature	Runtime	Spec	Compiler
Renderer3.None	✅	✅	✅
Renderer2.None	✅	✅	✅
Renderer2.Emulated	✅	✅	✅
Renderer2.Native	✅	✅	✅

`_______Ref`s

Method	View Container Ref	Template Ref	Embeded View Ref	View Ref	Element Ref	Change Detection Ref
`clear()`	✅	n/a	n/a	n/a	n/a	n/a
`get()`	✅	n/a	n/a	n/a	n/a	n/a
`createEmbededView()`	✅	✅	n/a	n/a	n/a	n/a
`createComponent()`	✅	n/a	n/a	n/a	n/a	n/a
`insert()`	✅	n/a	n/a	n/a	n/a	n/a

`move()`	✅	n/a	n/a	n/a	n/a	n/a
`indexOf()`	✅	n/a	n/a	n/a	n/a	n/a
`length()`	✅	n/a	n/a	n/a	n/a	n/a
`remove()`	✅	n/a	n/a	n/a	n/a	n/a
`destroy()`	n/a	n/a	✅	✅	n/a	n/a
`destroyed`	n/a	n/a	✅	✅	n/a	n/a
`onDestroy()`	n/a	n/a	✅	✅	n/a	n/a
`markForCheck()`	n/a	n/a	✅	✅	n/a	✅
`detach()`	✅	n/a	✅	✅	n/a	✅
`detachChanges()`	n/a	n/a	✅	✅	n/a	✅
`checkNoChanges()`	n/a	n/a	✅	✅	n/a	✅
`reattach()`	n/a	n/a	✅	✅	n/a	✅
`nativeElement()`	n/a	n/a	n/a	n/a	✅	n/a
`elementRef`	n/a	✅	n/a	n/a	n/a	n/a

Renderer2

Method	Runtime
`data()`	n/a
`destroy()`	✅
`createElement()`	✅
`createComment()`	✅
`createText()`	✅
`destroyNode()`	✅
`appendChild()`	✅
`insertBefore()`	✅
`removeChild()`	✅
`selectRootElement()`	✅
`parentNode()`	n/a
`nextSibling()`	n/a
`setAttribute()`	✅
`removeAttribute()`	✅

`addClass()`	✅
`removeClass()`	✅
`setStyle()`	✅
`removeStyle()`	✅
`setProperty()`	✅
`setValue()`	✅
`listen()`	✅