How to bundle an Angular app for production

Asked 3 years, 5 months ago Active 21 days ago Viewed 177k times

335

205

What is the best method to bundle Angular (version 2, 4, 6, ...) for production on a live web server.

Please include the Angular version within answers so we can track better when it moves to later releases.

angular webpack systemjs angular-cli

edited Aug 1 at 15:13
Community ♦
1 1

asked Jun 4 '16 at 13:59

Pat M
4,613 5 15 27

For now (rc1). Here are a few solutions stackoverflow.com/questions/37324511/how-to-bundle-angular2-rc1-with-systemjs – Pat M Jun 6 '16 at 17:35

And this one stackoverflow.com/questions/37098942/… – Pat M Jun 7 '16 at 21:33

rc3 now offers a bundled file versions that lowers the number of requests from 300+ to about 40. – Pat M Jun 29 '16 at 19:16

2 Hey. I also hate WebPack's and build steps in general. Sort of overkill for just trying to throw together a simple website. Thus I made this: github.com/schungx/angular2-bundle – Stephen Chung Oct 11 '16 at 5:55

Thank you Stephen. This would be a simple solution for the vendors part. Hoping this could be officially offered and updated. I suppose you use something like Gulp for the project`s files? – Pat M Oct 11 '16 at 14:00

11 Answers

339

OneTime Setup

- `npm install -g @angular/cli`
- `ng new projectFolder` creates a new application

Bundling Step

- `ng build --prod` (run in command line when directory is `projectFolder`)

  flag `prod` bundle for production (see the Angular documentation for the list of option included with the production flag).
- Compress using Brotli compression the resources using the following command

```
for i in dist/*; do brotli $i; done
```

*bundles are generated by default to **projectFolder/dist(/$projectFolder for 6)***

Output

Sizes with Angular `8.2.11` with CLI `8.3.13` and option CSS without Angular routing

- `dist/main-[es-version].[hash].js` Your application bundled [ES5 size: 188 KB for new Angular CLI application empty, **44 KB** compressed].
- `dist/polyfill-[es-version].[hash].bundle.js` the polyfill dependencies (@angular, RxJS...) bundled [ES5 size: 122 KB for new Angular CLI application empty, **36 KB** compressed].
- `dist/index.html` entry point of your application.
- `dist/runtime-[es-version].[hash].bundle.js` webpack loader
- `dist/style.[hash].bundle.css` the style definitions
- `dist/assets` resources copied from the Angular CLI assets configuration

Deployment

You can get a preview of your application using the `ng serve --prod` command that starts a local HTTP server such that the application with production files is accessible using http://localhost:4200.

edited Oct 23 at 18:24

answered Aug 4 '16 at 11:18

Nicolas Henneaux
7,570 5 38 58

I got the error when running npm install -g angular-cli@webpack: npm ERR! Please include the following file with any support request: ....\npm-debug.log. Do you know what's going on? – Chong Wang Aug 29 '16 at 21:09

2 @chrismarx it produces only one bundle including all the components with their html and styles. – Nicolas Henneaux Sep 27 '16 at 17:49

4 I had an application and i wanted to use this method, so i launch ng init from the project folder. I have done the rest of the steps but when i deploy my applications it seems to be empty. The only thing that appears is a "app works!" message, is ther some place where i have to set where to take my app files? – mautrok Oct 25 '16 at 10:52

2 ng-init has been removed from angular cli. github.com/angular/angular-cli/issues/5176 – Pat M May 12 '17 at 18:08

2 I finally marked this as the accepted answer. Although other solutions may work as well and even provide some extra flexibility (I posted one about using Webpack without CLI). Using Angular CLI is definitively the one that gives the less headaches. I ended up using Angular CLI and adapting my project so I can use AoT more easily. – Pat M May 16 '17 at 20:55

54

`2.0.1 Final` using Gulp (TypeScript - Target: ES5)

OneTime Setup

- `npm install` (run in cmd when direcory is projectFolder)

Bundling Steps

- `npm run bundle` (run in cmd when direcory is projectFolder)

  *bundles are generated to **projectFolder / bundles /***

Output

- `bundles/dependencies.bundle.js` [**size: ~ 1 MB** (as small as possible)]

- contains your project

File Structure

- **projectFolder / app /** (all components, directives, templates, etc)
- **projectFolder / gulpfile.js**

```
var gulp = require('gulp'),
  tsc = require('gulp-typescript'),
  Builder = require('systemjs-builder'),
  inlineNg2Template = require('gulp-inline-ng2-template');

gulp.task('bundle', ['bundle-app', 'bundle-dependencies'], function(){});

gulp.task('inline-templates', function () {
  return gulp.src('app/**/*.ts')
    .pipe(inlineNg2Template({ useRelativePaths: true, indent: 0, removeLineBreaks:
true}))
    .pipe(tsc({
      "target": "ES5",
      "module": "system",
      "moduleResolution": "node",
      "sourceMap": true,
      "emitDecoratorMetadata": true,
      "experimentalDecorators": true,
      "removeComments": true,
      "noImplicitAny": false
    }))
    .pipe(gulp.dest('dist/app'));
});

gulp.task('bundle-app', ['inline-templates'], function() {
  // optional constructor options
  // sets the baseURL and loads the configuration file
  var builder = new Builder('', 'dist-systemjs.config.js');

  return builder
    .bundle('dist/app/**/* - [@angular/**/*.js] - [rxjs/**/*.js]',
'bundles/app.bundle.js', { minify: true})
    .then(function() {
      console.log('Build complete');
    })
    .catch(function(err) {
```

```
});

gulp.task('bundle-dependencies', ['inline-templates'], function() {
  // optional constructor options
  // sets the baseURL and loads the configuration file
  var builder = new Builder('', 'dist-systemjs.config.js');

  return builder
    .bundle('dist/app/**/*.js - [dist/app/**/*.js]', 'bundles/dependencies.bundle.js', {
minify: true})
    .then(function() {
      console.log('Build complete');
    })
    .catch(function(err) {
      console.log('Build error');
      console.log(err);
    });
});
```

- **projectFolder / package.json** (same as Quickstart guide, just shown devDependencies and npm-scripts required to bundle)

```
{
  "name": "angular2-quickstart",
  "version": "1.0.0",
  "scripts": {
    ***
     "gulp": "gulp",
     "rimraf": "rimraf",
     "bundle": "gulp bundle",
     "postbundle": "rimraf dist"
  },
  "license": "ISC",
  "dependencies": {
    ***
  },
  "devDependencies": {
    "rimraf": "^2.5.2",
    "gulp": "^3.9.1",
    "gulp-typescript": "2.13.6",
    "gulp-inline-ng2-template": "2.0.1",
    "systemjs-builder": "^0.15.16"
  }
}
```

```
(function(global) {

  // map tells the System loader where to look for things
  var map = {
    'app':                        'app',
    'rxjs':                       'node_modules/rxjs',
    'angular2-in-memory-web-api': 'node_modules/angular2-in-memory-web-api',
    '@angular':                   'node_modules/@angular'
  };

  // packages tells the System loader how to load when no filename and/or no extension
  var packages = {
    'app':                        { main: 'app/boot.js',  defaultExtension: 'js' },
    'rxjs':                       { defaultExtension: 'js' },
    'angular2-in-memory-web-api': { defaultExtension: 'js' }
  };

  var packageNames = [
    '@angular/common',
    '@angular/compiler',
    '@angular/core',
    '@angular/forms',
    '@angular/http',
    '@angular/platform-browser',
    '@angular/platform-browser-dynamic',
    '@angular/router',
    '@angular/router-deprecated',
    '@angular/testing',
    '@angular/upgrade',
  ];

  // add package entries for angular packages in the form '@angular/common': { main:
'index.js', defaultExtension: 'js' }
  packageNames.forEach(function(pkgName) {
    packages[pkgName] = { main: 'index.js', defaultExtension: 'js' };
  });

  var config = {
    map: map,
    packages: packages
  };

  // filterSystemConfig - index.asp's chance to modify config before we register it.
  if (global.filterSystemConfig) { global.filterSystemConfig(config); }
```

- **projetcFolder / dist-systemjs.config.js** (just shown the difference with systemjs.config.json)

```
var map = {
    'app':                        'dist/app',
  };
```

- **projectFolder / index.html** (production) - *The order of the script tags is critical. Placing the `dist-systemjs.config.js` tag after the bundle tags would still allow the program to run but the dependency bundle would be ignored and dependencies would be loaded from the `node_modules` folder.*

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>
  <base href="/"/>
  <title>Angular</title>
  <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>

<my-app>
  loading...
</my-app>

<!-- Polyfill(s) for older browsers -->
<script src="node_modules/core-js/client/shim.min.js"></script>

<script src="node_modules/zone.js/dist/zone.min.js"></script>
<script src="node_modules/reflect-metadata/Reflect.js"></script>
<script src="node_modules/systemjs/dist/system.js"></script>

<script src="dist-systemjs.config.js"></script>
<!-- Project Bundles. Note that these have to be loaded AFTER the systemjs.config script
-->
<script src="bundles/dependencies.bundle.js"></script>
<script src="bundles/app.bundle.js"></script>

<script>
    System.import('app/boot').catch(function (err) {
      console.error(err);
    });
```

- **projectFolder / app / boot.ts** is where the bootstrap is.

The best I could do yet :)

edited Jul 12 '17 at 12:30
Igor
47.7k 5 60 122

answered Jun 17 '16 at 6:32

Ankit Singh
19.1k 4 51 82

2 Hi, the gulp script is creating the bundles, but I'm unsure what should be in the boot.ts file? Aren't all the files now in the bundle? Do we execute the bundle? – chrismarx Sep 27 '16 at 14:30

2 Huh, I guess I need to try again. I tried switching to builder.buildStatic and got errors from rxjs about not being loaded as a commonjs or amd module. I'll give your suggestion another try – chrismarx Sep 29 '16 at 14:00

1 I'm also unclear how the bundles actually get used in this setup? I seem to be running into the exact same issues as @chrismarx here. I can create the bundles, but then it seems everything is still being loaded from my transpiled and copied app folder (located at dist/app). If I look in my network panel I can see that my app related files are actually being loaded from there (components, etc), instead of everything app related coming from app.bundle.js. A_Singh, can you share your boot.ts? It seems I'm missing something here and would love some clarification. – jbgarr Nov 9 '16 at 0:39

1 A_Singh, I don't see how that helps. When `inline-templates` is run it inlines the templates then creates a copy of all app folders and files at `dist/app` . Then in `dist-systemjs.config.js` you map `app` to `dist/app` which is a folder that won't exist if you use the `dist` folder as root. Wouldn't you want to run your app from the `dist` folder? And if that's the case, you wouldn't have a `dist` folder nested in the root `dist` folder. I must be missing something else here. Don't you need to tell systemjs to use your bundled files and not the usual files found in the `dist/app` folder? – jbgarr Nov 9 '16 at 17:02

1 I'm encountering an issue with your solution, boot is something that does not exist here, and when I replace it by "app" I'va an error "module is not defined". – Sakuto Nov 21 '16 at 13:09

22

Angular 2 with Webpack (without CLI setup)

1- The tutorial by the Angular2 team

The Angular2 team published a tutorial for using Webpack

I created and placed the files from the tutorial in a small GitHub seed project. So you can quickly try the workflow.

- **npm install**
- **npm start**. For development. This will create a virtual "dist" folder that will be livereloaded at your localhost address.
- **npm run build**. For production. "This will create a physical "dist" folder version than can be sent to a webserver. The dist folder is 7.8MB but only 234KB is actually required to load the page in a web browser.

2 - A Webkit starter kit

This Webpack Starter Kit offers some more testing features than the above tutorial and seem quite popular.

edited Feb 3 '17 at 15:49

answered Oct 8 '16 at 20:41

Pat M

4,613 5 15 27

hi, is it possible to update the seed project with angular 2.1.0? The tutorial is using angular 2.1.0 now. I followed it and could not get it work. The error is http 404 - can't find app.component.html. – heq99 Oct 17 '16 at 22:30

I updated to angular 2.1.0 without problem. app.component.html is called from app.component.ts (templateUrl: './app.component.html'). you have both files in the same app folder? – Pat M Oct 22 '16 at 15:31

1 Tree-shaking, Minification & Gzipping can greatly reduce the size when you go for production. here is an excellent read with example, blog.mgechev.com/2016/06/26/… – Hamzeen Hameem Jan 12 '17 at 8:40

Angular 2 production workflow with SystemJs builder and gulp

15

Angular.io have quick start tutorial. I copied this tutorial and extended with some simple gulp tasks for bundling everything to dist folder which can be copied to server and work just like that. I tried to optimize everything to work well on Jenkis CI, so node_modules can be cached and don't need to be copied.

Source code with sample app on Github: https://github.com/Anjmao/angular2-production-workflow

Steps to production

1. Clean typescripts compiled js files and dist folder
2. Compile typescript files inside app folder

5. Copy everything inside assets folder to dist folder

Node: While you always can create your own build process, but I highly recommend to use angular-cli, because it have all needed workflows and it works perfectly now. We are already using it in production and don't have any issues with angular-cli at all.

edited Nov 4 '16 at 9:44

answered Sep 18 '16 at 18:59

Andzej Maciusovic

2,876 1 17 30

This is what I'm looking for. Sample app on github is very useful. Thanks – Shahriar Hasan Sayeed Nov 3 '16 at 15:45

14

Angular CLI 1.x.x (Works with Angular 4.x.x, 5.x.x)

This supports:

- Angular 2.x and 4.x
- Latest Webpack 2.x
- Angular AoT compiler
- Routing (normal and lazy)
- SCSS
- Custom file bundling (assets)
- Additional development tools (linter, unit & end-to-end test setups)

Initial Setup

```
ng new project-name --routing
```

You can add `--style=scss` for SASS .scss support.

You can add `--ng4` for using Angular 4 instead of Angular 2.

Bundle Steps

Inside the project folder:

```
ng build -prod
```

~~At the current version you need to to specify `--aot` manually, because it can be used in development mode (although that's not practical due to slowness).~~

This also performs AoT compilation for even smaller bundles (no Angular compiler, instead, generated compiler output). The bundles are much smaller with AoT if you use Angular 4 as the generated code is smaller.
You can test your app with AoT in development mode (sourcemaps, no minification) and AoT by running `ng build --aot`.

Output

The default output dir is `./dist`, although it can be changed in `./angular-cli.json`.

Deployable Files

The result of build step is the following:

(Note: `<content-hash>` refers to hash / fingerprint of the contents of the file that's meant to be a cache busting way, this is possible since Webpack writes the `script` tags by itself)

- `./dist/assets`
  Files copied as-is from `./src/assets/**`
- `./dist/index.html`
  From `./src/index.html`, after adding webpack scripts to it
  Source template file is configurable in `./angular-cli.json`
- `./dist/inline.js`
  Small webpack loader / polyfill
- `./dist/main.<content-hash>.bundle.js`
  The main .js file containing all the .js scripts generated / imported
- `./dist/styles.<content-hash>.bundle.js`
  When you use Webpack loaders for CSS, which is the CLI way, they are loaded via JS here

In older versions it also created gzipped versions for checking their size, and `.map` sourcemaps files, but this is no longer happening as people kept asking to remove these.

Other Files

In certain other occasions, you might find other unwanted files/folders:

- `./out-tsc/`
  From `./src/tsconfig.json` 's `outDir`
- `./out-tsc-e2e/`
  From `./e2e/tsconfig.json` 's `outDir`
- `./dist/ngfactory/`
  From AoT compiler (not configurable without forking the CLI as of beta 16)

edited Oct 19 '17 at 23:31

answered Oct 6 '16 at 10:39

Meligy
29.5k 9 70 94

Is it possible to separate the angular lib and their dependencies from my app? – Dominick Piganell Dec 12 '16 at 21:29

Not using the CLI, which is on purpose for tree-shaking to work. That is removing all Angular EcmaScript modules that are not used in your application. There is a plan to disable this in dev mode for speed (they call the libraries loaded as is "DLL"s), but no plan to separate in end result. It should be achievable if you are rolling your own Webpack stuff though without the CLI. – Meligy Dec 12 '16 at 23:32

How to check my app using dist folder. How can I host in my web server ? – raj m Jan 27 '17 at 5:25

You just copy it to the server. It's plain static website that can be served anyway. If you use routing, you might want to redirect all calls to the HTML file though, for that check Angular deployment docss on server configuration section angular.io/docs/ts/latest/guide/… – Meligy Mar 18 '17 at 23:56

@Meligy what if I remove `<content-hash>` from the bundles in prod. it may cause problems in getting latest bundle ? – k11k2 Feb 27 '18 at 9:08

5

As of today I still find the Ahead-of-Time Compilation cookbook as the best recipe for production bundling. You can find it here: https://angular.io/docs/ts/latest/cookbook/aot-compiler.html

My experience with Angular 2 so far is that AoT creates the smallest builds with almost no loading time. And most important as the question here is about - you only need to ship a few files to production.

This seems to be because the Angular compiler will not be shipped with the production builds as the templates are compiled "Ahead of Time". It's also very cool to see your HTML template markup transformed to javascript instructions that would be very hard to reverse engineer into the original HTML.

I've made a simple video where I demonstrate download size, number of files etc. for an Angular 2 app in dev vs AoT build - which you can see here:

https://youtu.be/ZoZDCgQwnmQ

You'll find the source code used in the video here:

https://github.com/fintechneo/angular2-templates

answered Jan 13 '17 at 19:55

Peter Salomonsen
3,397 1 13 26

2

```
        **Production build with

         - Angular Rc5
         - Gulp
         - typescripts
         - systemjs**

        1)con-cat all js files  and css files include on index.html using  "gulp-
concat".
          - styles.css (all css concat in this files)
          - shims.js(all js concat in this files)

        2)copy all images and fonts as well as html files  with gulp task to "/dist".

        3)Bundling -minify angular libraries and app components mentioned in
systemjs.config.js file.
         Using gulp  'systemjs-builder'

            SystemBuilder = require('systemjs-builder'),
            gulp.task('system-build', ['tsc'], function () {
                var builder = new SystemBuilder();
                return builder.loadConfig('systemjs.config.js')
                    .then(function () {
```

```
                    .then(function () {
                        del('temp')
                    })
            });


    4)Minify bundles  using 'gulp-uglify'

jsMinify = require('gulp-uglify'),

    gulp.task('minify', function () {
        var options = {
            mangle: false
        };
        var js = gulp.src('dist/app/shims.js')
            .pipe(jsMinify())
            .pipe(gulp.dest('dist/app/'));
        var js1 = gulp.src('dist/app/app_libs_bundle.js')
            .pipe(jsMinify(options))
            .pipe(gulp.dest('dist/app/'));
        var css = gulp.src('dist/css/styles.min.css');
        return merge(js,js1, css);
    });

5) In index.html for production

    <html>
    <head>
        <title>Hello</title>

        <meta name="viewport" content="width=device-width, initial-scale=1">
        <meta charset="utf-8" />

       <link rel="stylesheet" href="app/css/styles.min.css" />
       <script type="text/javascript" src="app/shims.js"></script>
       <base href="/">
    </head>
     <body>
    <my-app>Loading...</my-app>
     <script type="text/javascript" src="app/app_libs_bundle.js"></script>
    </body>

    </html>

 6) Now just copy your dist folder to '/www' in wamp server node need to copy
node modules in www.
```


Tushar Tibude
31 4

2

You can deploy your angular application on `github` using angular-cli-ghpages

check out the link to find how to deploy using this cli.

the deployed website will be stored in some branch in `github` typically

gh-pages

use can clone the git branch and use it like static website in your server

answered Sep 1 '17 at 7:26


Sunil Kumar
419 4 11

1

"Best" depends on the scenario. There are times when you only care about the smallest possible single bundle, but in large apps you may have to consider lazy loading. At some point it becomes impractical to serve the entire app as a single bundle.

In the latter case Webpack is generally the best way since it supports code splitting.

For a single bundle I would consider Rollup, or the Closure compiler if you are feeling brave :-)

I have created samples of all Angular bundlers I've ever used here: http://www.syntaxsuccess.com/viewarticle/angular-production-builds

The code can be found here: https://github.com/thelgevold/angular-2-samples

Angular version: 4.1.x

answered May 23 '17 at 1:53


TGH
36.1k 8 84 121

0

just follow the below github doc

https://github.com/roshan3133/angular2-webpack-starter

edited Nov 17 '17 at 5:58

Nicolas Henneaux
7,570 5 38 58

answered Sep 3 '17 at 12:04

AniketGole
549 6 16

0

Please try below CLI command in current project directory. It will create dist folder bundle. so you can upload all files within dist folder for deployments.

ng build --prod --aot --base-href.

answered May 15 at 7:42

Nagnath Mungade
1 4