

Template
HTML



```
graph LR; A[Template HTML] --> B[Template Data]; B --> C[Angular Interpreter]; C --> D[DOM];
```

The diagram illustrates the Angular rendering process as a linear sequence of four steps. It begins with a teal box labeled 'Template HTML', followed by a grey arrow pointing to a purple box labeled 'Template Data'. This is followed by another grey arrow pointing to a teal box labeled 'Angular Interpreter', and finally a third grey arrow pointing to a teal box labeled 'DOM'. The teal boxes are wider than the purple box, and all boxes are of equal height.

Template
Data

Angular
Interpreter

DOM

Template
HTML

```
graph LR; A[Template HTML] --> B[Template instructions]; B --> C[DOM];
```

The diagram illustrates a three-step process. It begins with a teal box labeled 'Template HTML'. A grey arrow points from this box to a purple box labeled 'Template instructions'. Another grey arrow points from the purple box to a final teal box labeled 'DOM'.

Template
instructions

DOM

```
import {someFn, unusedFn} from './third-party';

function main(x) {
    someFn();
    if (x) unusedFn();
    return x;
}

main(false);
```

Schematics

Schematics allow you to “do things” to your project.

- update = single command to update dependencies
 - *ng update = npm, code, RxJS, Material*
- add = instant code and library scaffolding
 - Material, Elements, PWA, bootstrap, clarity, nativescript
 - *ng add @angular/pwa = does all the setup.*
- generate = Per-library 'generate' schematics
 - *ng generate @angular/material:material-nav --name=main-nav*

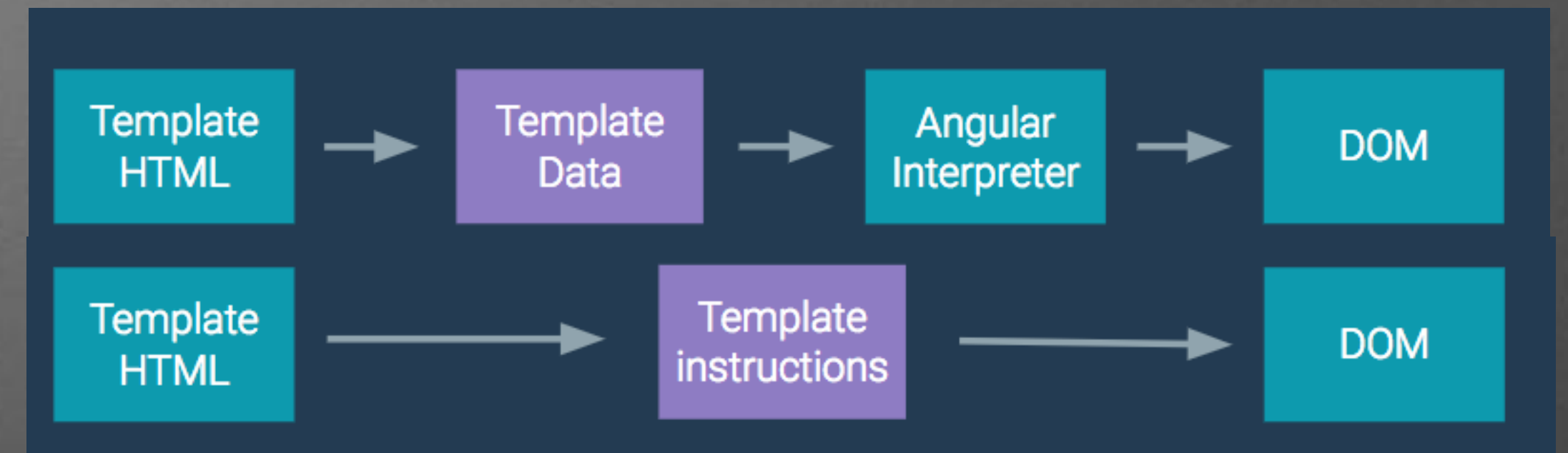
Vendors and Us can publish our own schematics. Goal is to make universal libraries easier to build: project scaffolding, test, build system. N

Using 'ng add'

```
> ng add @angular/pwa
> ng add @angular/elements
> ng add @angular/material
> ng add @ng-bootstrap/schematics
> ng add @clr/angular
> ng add @nativescript/schematics (real soon!)
```


Ivy

- Biggest benefit of Ivy, the rendering pipeline removes actually reads the code vs. static analysis. This removes imports that are never used, even if they are referenced in the code.
- Backwards Compatible - 6 months out ([status](#))
- Locality - Allows AOT dependencies - & no meta data. Local meaning only information which is directly tied to the annotation of the component
- Stack trace is much more accurate, and identifies the issue.
- Build size is insanely small!
 - Today Compressed → 36K
 - Minified → 7.3Kb
 - Ivy Compressed → 2.7Kb



```
import {someFn, unusedFn} from './third-party';

function main(x) {
  someFn();
  if (x) unusedFn();
  return x;
}

main(false);
```