Should I Use “X”

Should I use the CLI?

- Pros:
    - Scaffolding
    - Adding capabilities
    - Building / Serving 4. Keeping up to date
- Cons:
    - If you need full control of application
    - Can’t customize: Webpack, SystemJS, Rollup, PostCSS,
    - Can’t use: Bazel (for now)

Should I use Universal? … in my notes

Unit Tests? - Yes

E2E Tests? - Yes - not as many as unit tests

Reactive forms? - Yes

Should I build a PWA?

- If a more reliable and engaging web experience would be helpful, might be worth it.
- Ways to deliver on Mobile: Cordova > NativeScript > Native > PWA
    - PWA - not all items are in the browser yet. Fingerprint, Contacts, etc…
    - PWA - Great if you also have to deliver a web experience
    - PWA - Advanced dev teams pushing the boundaries
        - Building shareable controls for other environments
        - If your primary use base is IE, performance isn’t great.

RxJs6

- Start Doing Today:
    - pipe-able operators need to be used
- New:
    - Simplified imports - only two Ng Devs need to worry about: rxjs && rxjs/operators
    - new operator - throwIfEmpty - if completes without throwing values, it will error out.
- Breaking:
    - New unhandled error behavior - now no error handler will throw it as an async
        - do not depend on sync errors (try/catch)
        - this is going to break unit tests that have observables throw errors
    - Only can import “of” directly, it is no loner available on Observable.of
- Deprecating
    - result selectors for mergeMap - this is replaced by using a .pipe(map(…))
    - operator versions of concat, merge, zip - use them as static functions
- Migrate and Update
    1. Update to 5.5.10 > Update to 6.0.0
    2. ng update rxjs
    3. Install rxjs-compat
    4. Update to remove rxjs-compat
    5. yarn add rxjs-tslint
    6. run tslint -fix will fix everything to be 6.0 compliant! Wooo!
    7. should be able to remove rxjs-compat at this point

- Google is running RxJs6 in prod now!

Should I Use “X”

Should I use the CLI?

- Pros:
  - Scaffolding
  - Adding capabilities
  - Building / Serving 4. Keeping up to date
- Cons:
  - If you need full control of application
  - Can’t customize: Webpack, SystemJS, Rollup, PostCSS,
  - Can’t use: Bazel (for now)

Should I use Universal? … in my notes

Unit Tests? - Yes

E2E Tests? - Yes - not as many as unit tests

Reactive forms? - Yes

Should I build a PWA?

- If a more reliable and engaging web experience would be helpful, might be worth it.
- Ways to deliver on Mobile: Cordova > NativeScript > Native > PWA
  - PWA - not all items are in the browser yet. Fingerprint, Contacts, etc…
  - PWA - Great if you also have to deliver a web experience
  - PWA - Advanced dev teams pushing the boundaries
    - Building shareable controls for other environments
    - If your primary use base is IE, performance isn’t great.