CSC 581: Mobile App Development

Spring 2019

Progressive Web Apps

- saving Web pages to iOS home screen
- Progressive Web Apps
  - icon
  - manifest file
  - service worker
- PWA vs. native tradeoffs

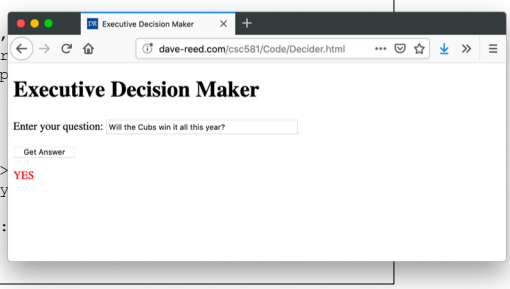
Web vs. App development

developing a Web page is generally much simpler than a native app

- does not require development tools – can create using a simple text editor
- HTML/CSS/JavaScript are easy to pick up (and lots of online support)
- PWAs can run on all platforms (so no need for separate iOS & Android apps)
- the user does not have to download an app, no app store required

```
<!doctype html>
<html>
  <head>
    <title>Executive Decision Maker</title>
    <script>
      function showAnswer() {
        var responses = ["YES", "NO",
        var index = Math.floor(Math.r
        document.getElementById("outp
      }
    </script>
  </head>
  <body>
    <h1>Executive Decision Maker</h1>
    <p>Enter your question: <input ty
    <p><input type=button value="Get
    <div id="outputDiv" style="color:
  </body>
</html>
```

simple Web page for an executive decision maker



Pseudo-apps

however, a Web page in a browser is not as convenient as an app

- e.g., an app has an icon on the home screen, so easier to find and access

iOS does allow you to save a Web page on the home screen

- essentially, creates a pseudo-app that looks like other apps (but is actually the Web page in a browser window)

1. load the page in Safari (or Chrome)
2. click on the Share icon  and select "Add to Home Screen"
3. this puts a pseudo-app on the screen
   - its icon is a screen shot of the Web page
   - when you click on the icon, it opens the page in a browser window

since pseudo-apps are just Web pages viewed in the browser

- they can't access any of the native phone features (sensors, GPS, photos, …)
- they can't work when offline or save their state if closed or interrupted

3

Progressive Web App

a PWA is a Web page with supporting files that allows it to

- specify its icon image
- provide information as to how the page is to be displayed (manifest file)
- provide code so that it works offline and can save app data (service worker)
- can also access device sensors, native features, push notices, …

recall: PWAs were the 3rd-party dev. model initially envisioned by Apple

- 3rd-party native apps & the App Store were added with Phone OS 2

as of 2018, all major mobile Web browsers support PWAs

- PWAs are not *enthusiastically* supported by Apple
- Apple wants to control app quality/security through the App Store approval process
- PWAs do not need to be downloaded through the App Store – can link directly to a Web site and download the PWA without Apple intervention
- many PWA developers will still post them to the App Store for visibility

4

Executive Decision Maker PWA

we can transform our simple Web page to a PWA by adding resources (here, organized in a PWAsupport folder)


```
<!doctype html>
<html>
  <head>
    <title>Executive Decision Maker</title>

    <link rel="apple-touch-icon" sizes="1024x1024" href="PWAsupport/icon-1024.png">
    <link rel="manifest" href="PWAsupport/manifest.json">
    <script src="PWAsupport/app.js"></script>

    <script>
      function showAnswer() {
        var responses = ["YES", "NO", "MAYBE"];
        var index = Math.floor(Math.random()*responses.length);
        document.getElementById("outputDiv").innerHTML = responses[index];
      }
    </script>
  </head>
  <body>
    <h1>Executive Decision Maker</h1>
    <p>Enter your question: <input type="text" size=40></p>
    <p><input type=button value="Get Answer" onclick="showAnswer()"></p>
    <div id="outputDiv" style="color:red"></div>
  </body>
</html>
```

5

Manifest file

the manifest file (usually a .json file), contains info about the app

- may include PWA name, title, author, default style settings, icons, …
- used by the browser to make the PWA look like an actual app

- for this example, manifest.json:

```
{
  "name": "Executive Decision Maker",
  "short_name": "Decider",
  "start_url": "../DeciderPWA.html",
  "display": "standalone",
  "orientation": "portrait",
  "lang": "en-US"
}
```

6

Service worker

a service worker acts a proxy for the Web server if it is unavailable

- e.g., if the phone is offline, the service worker can restore the PWA to its last saved state and run from that
- can also allow access to push notifications and some native APIs

```
// serviceWorker.js
////////////////////////////////////////////////////////////////////////////////////////

var cacheName ='version: 1.0.0';

var cacheFiles = [ '../DeciderPWA.html', './app.js',
                   './serviceWorker.js', './manifest.json', './icon-1024.png' ]

self.addEventListener('install', function(e) {
  console.log('[Service Worker] Installed');
  e.waitUntil(
    caches.open(cacheName).then(function(cache) {
      console.log('[Service Worker] Caching cacheFiles');
      return cache.addAll(cacheFiles);
    }) ); });

self.addEventListener('activate', function(e) {
  console.log('[Service Worker] Activated');
  e.waitUntil(caches.keys().then(function(cacheNames){
    return Promise.all(cacheNames.map(function(thisCacheName){
      if(thisCacheName != cacheName){
        console.log('[Service Worker] Removing Cached Files from Cache- ', thisCacheName);
        return caches.delete(thisCacheName);
      } })) }) ); })

self.addEventListener('fetch', function(e) {
  console.log('[ServiceWorker] Fetch', e.request.url);
  e.respondWith(

    . . .
```


Registering a service worker

a service worker must be registered with the browser

- this is done the first time the PWA is loaded, again every 24 hours to keep current
- for our PWA, the app.js script is run by the page to register serviceWorker.js

```
// app.js
////////////////////////////////////////////////////////////

if ('serviceWorker' in navigator) {

  navigator.serviceWorker
    .register('serviceWorker.js')
    .then(function(registration) {
      console.log("[Service Worker] Registered");
    })
    .catch(function(err) {
      console.log("[Service Worker] Failed to Register", err);
    })

}
```

Downloading a PWA

the PWA page and supporting resources must be stored together

- when you load the page into your mobile browser and Add to Home Screen, it downloads the supporting resources along with the page

- will use the specified image for the PWA icon
- will display the page using the style settings in manifest.json
- will register the service worker via the app.js script
- will provide the proxy and API services of serviceWorker.js

since PWAs are running through the browser, they are platform independent

- the same PWA can run on an Android device or an iOS device

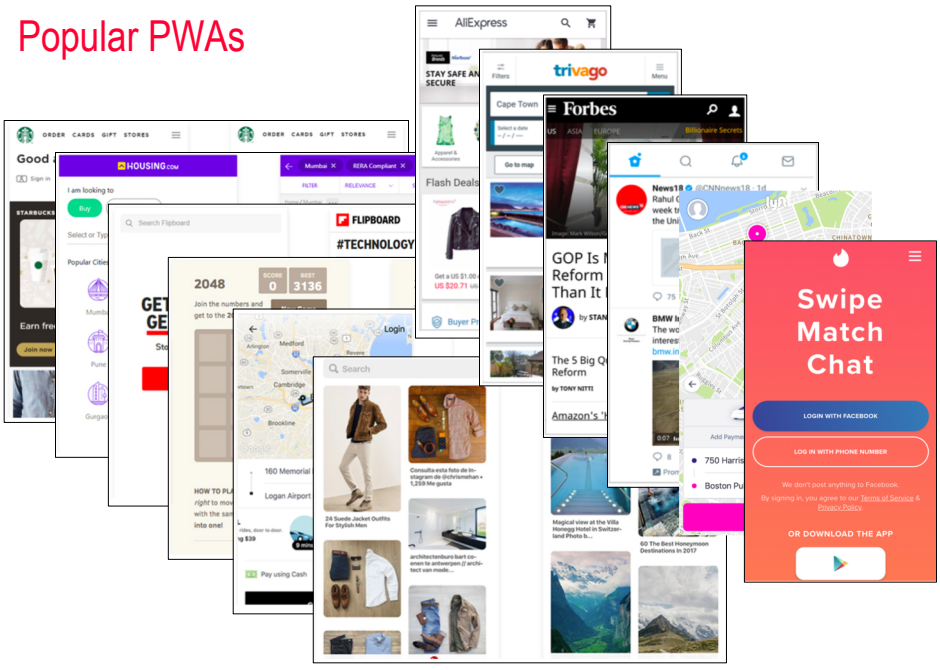
HUGE ADVANTAGE OVER NATIVE DEVELOPMENT!

tradeoff: PWAs do not have the full range of native features

- may not have same look & feel, or be as responsive as native apps

9

Popular PWAs



10

Questions

1. Describe in what ways a Progressive Web App is similar to a Web page. Describe how it is different.

2. Describe in what ways a Progressive Web App is similar to a mobile app. Describe how it is different.

3. What role does the manifest file play in a Progressive Web App? What role does the service worker play?

4. After reading about PWAs and their advantages and disadvantages, do you believe that PWAs will grow in popularity, replacing native apps for many tasks? Explain your answer.

11