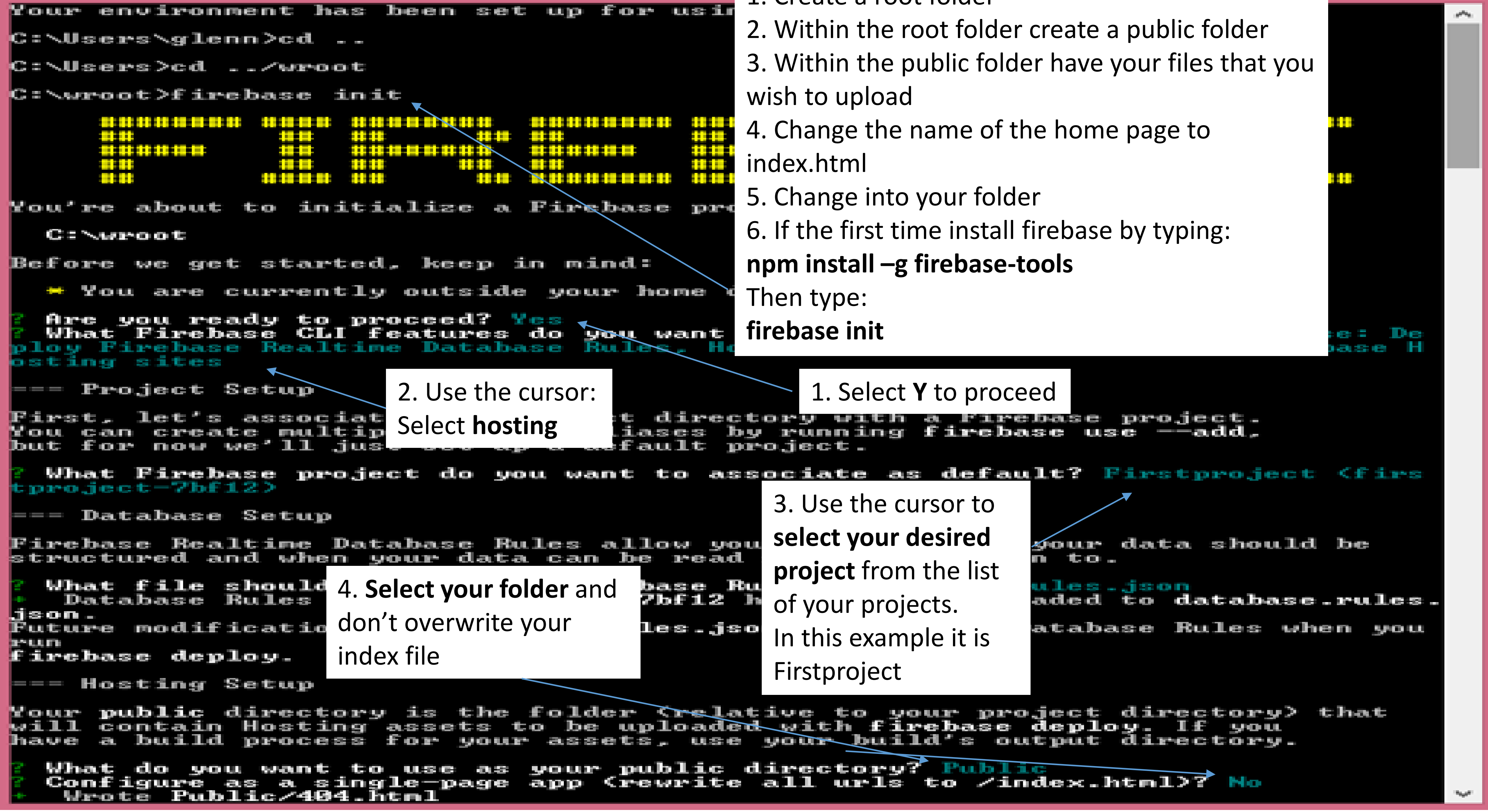
Firebase Hosting



```
Your environment has been set up for usi
C:\Users\glenn>cd ..
C:\Users>cd ../wroot
C:\wroot>firebase init

You're about to initialize a Firebase pro
  C:\wroot
Before we get started, keep in mind:
  * You are currently outside your home
? Are you ready to proceed? Yes
? What Firebase CLI features do you want                              e: De
ploy Firebase Realtime Database Rules, Ho                             base H
osting sites
=== Project Setup
First, let's associat            t directory with a Firebase project.
You can create multip            liases by running firebase use --add,
but for now we'll jus            default project.
? What Firebase project do you want to associate as default? Firstproject (firs
tproject-7bf12)
=== Database Setup
Firebase Realtime Database Rules allow you                 your data should be
structured and when your data can be read                  n to.
? What file should                base Ru                  ules.json
+ Database Rules                  7bf12 h                  aded to database.rules.
json.
Future modificatio                les.jso                  atabase Rules when you
run
firebase deploy.
=== Hosting Setup
Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.
? What do you want to use as your public directory? Public
? Configure as a single-page app (rewrite all urls to /index.html)? No
+ Wrote Public/404.html
```

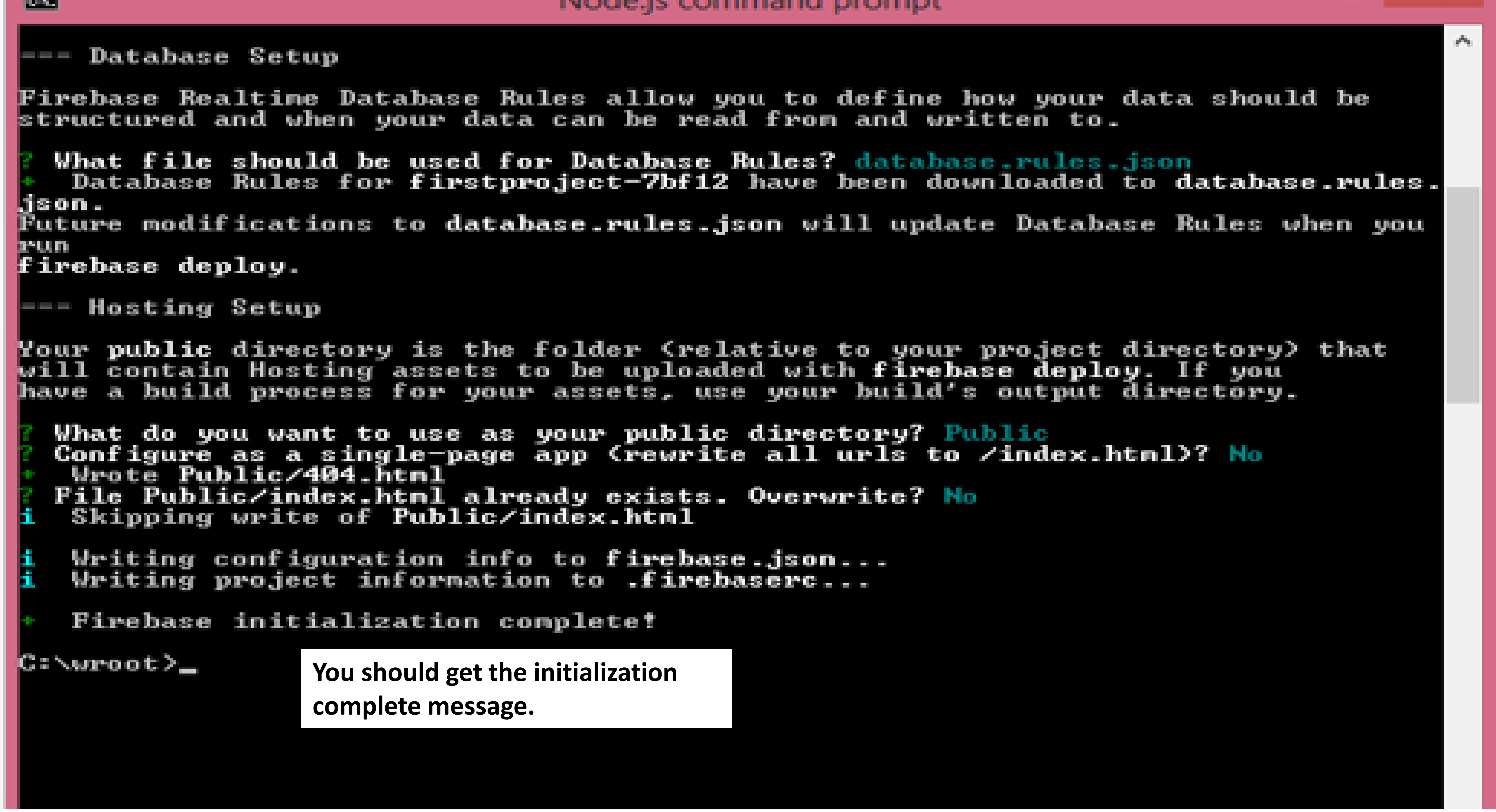
1. Create a root folder
2. Within the root folder create a public folder
3. Within the public folder have your files that you wish to upload
4. Change the name of the home page to index.html
5. Change into your folder
6. If the first time install firebase by typing:
npm install –g firebase-tools
Then type:
firebase init

1. Select **Y** to proceed

2. Use the cursor: Select **hosting**

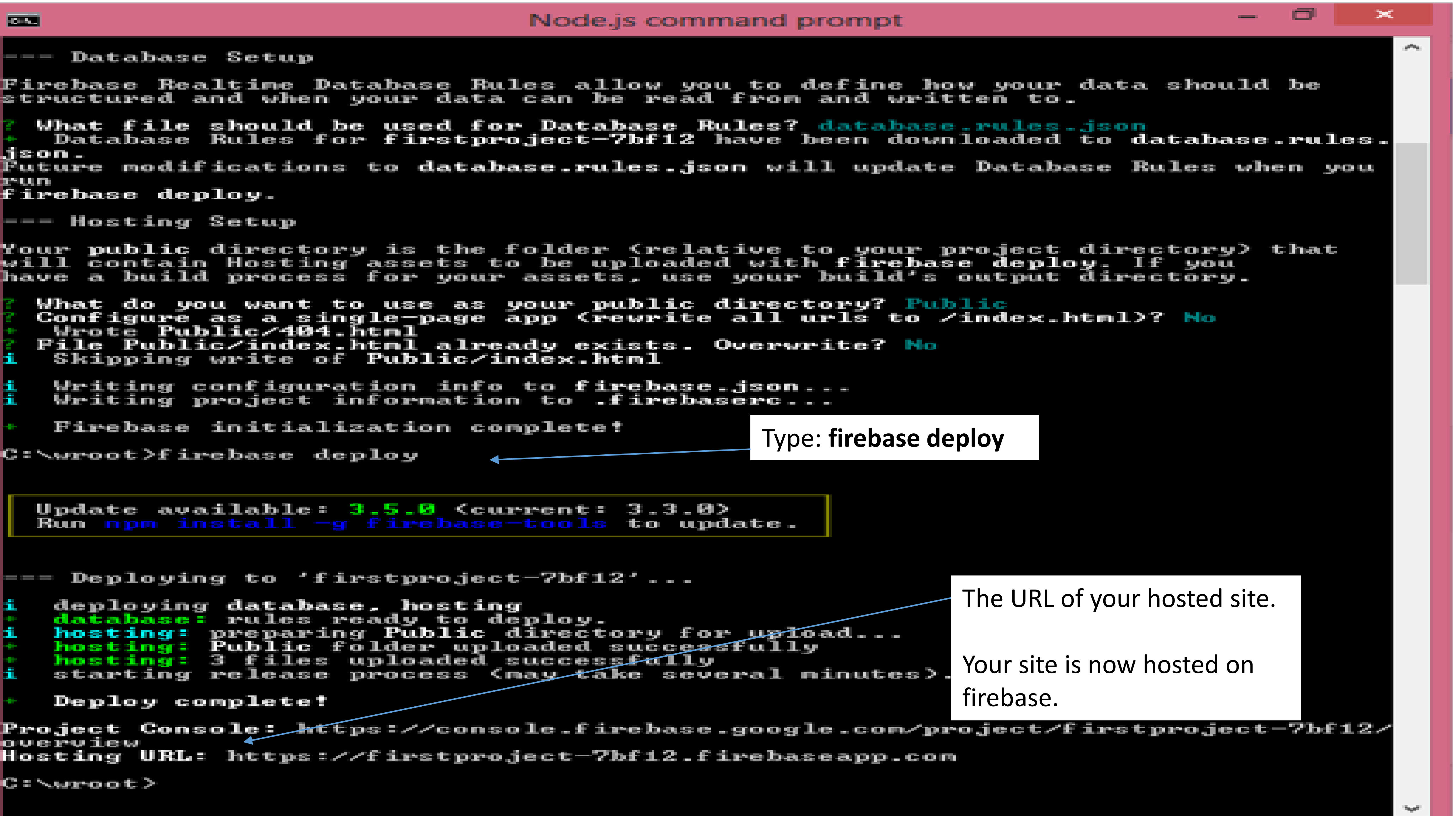
3. Use the cursor to **select your desired project** from the list of your projects. In this example it is Firstproject

4. **Select your folder** and don't overwrite your index file



```
=== Database Setup

Firebase Realtime Database Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Database Rules? database.rules.json
+  Database Rules for firstproject-7bf12 have been downloaded to database.rules.
json.
Future modifications to database.rules.json will update Database Rules when you
run
firebase deploy.

=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? Public
? Configure as a single-page app (rewrite all urls to /index.html)? No
+  Wrote Public/404.html
? File Public/index.html already exists. Overwrite? No
i  Skipping write of Public/index.html

i  Writing configuration info to firebase.json...
i  Writing project information to .firebaserc...

+  Firebase initialization complete!

C:\wroot>_
```

You should get the initialization complete message.



```
=== Database Setup

Firebase Realtime Database Rules allow you to define how your data should be
structured and when your data can be read from and written to.

? What file should be used for Database Rules? database.rules.json
+  Database Rules for firstproject-7bf12 have been downloaded to database.rules.
json.
Future modifications to database.rules.json will update Database Rules when you
run
firebase deploy.

=== Hosting Setup

Your public directory is the folder (relative to your project directory) that
will contain Hosting assets to be uploaded with firebase deploy. If you
have a build process for your assets, use your build's output directory.

? What do you want to use as your public directory? Public
? Configure as a single-page app (rewrite all urls to /index.html)? No
+  Wrote Public/404.html
? File Public/index.html already exists. Overwrite? No
i  Skipping write of Public/index.html

i  Writing configuration info to firebase.json...
i  Writing project information to .firebaserc...

+  Firebase initialization complete!

C:\wroot>firebase deploy


  Update available: 3.5.0 (current: 3.3.0)
  Run npm install -g firebase-tools to update.


=== Deploying to 'firstproject-7bf12'...

i  deploying database, hosting
+  database: rules ready to deploy.
i  hosting: preparing Public directory for upload...
+  hosting: Public folder uploaded successfully
+  hosting: 3 files uploaded successfully
i  starting release process (may take several minutes)...

+  Deploy complete!

Project Console: https://console.firebase.google.com/project/firstproject-7bf12/
overview
Hosting URL: https://firstproject-7bf12.firebaseapp.com

C:\wroot>
```

Type: **firebase deploy**

The URL of your hosted site.

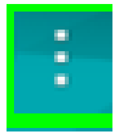
Your site is now hosted on firebase.

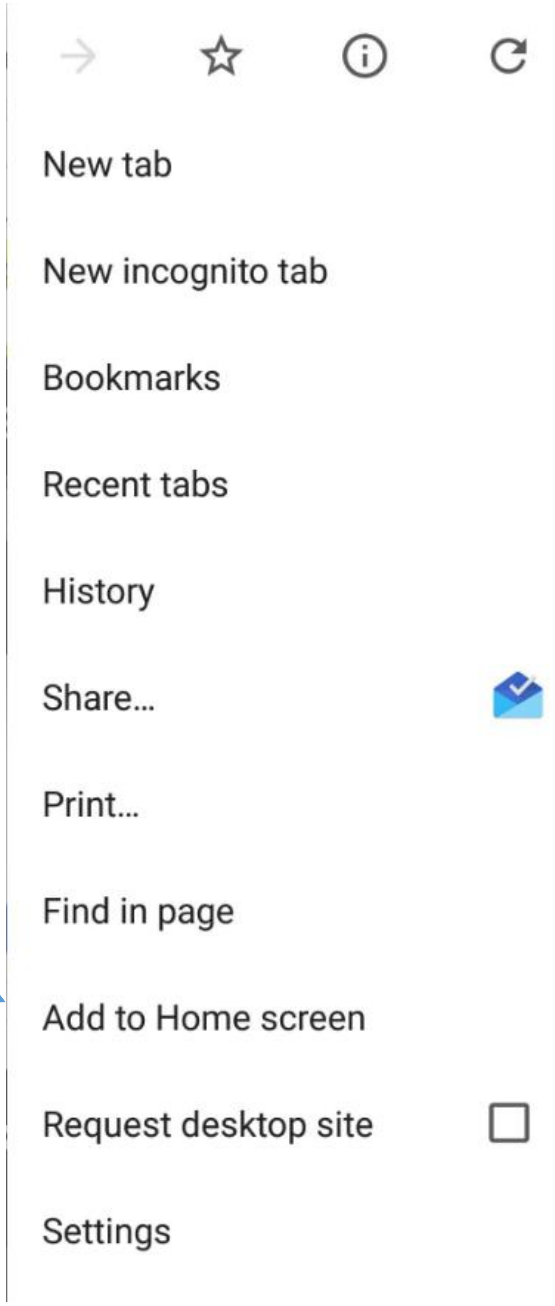
Web Apps may be Native Apps or Web Apps served from a Web site

Native App: A client-side application (for example developed using the Android SDK and installed on user devices in an APK file)

A Web App: a short cut to a web site that look and feels like an app to a user.

Installing as a Web App

- Easy install as a web app
  - Open site in chrome (Andriod)
  - Select menu - top right 
    - Select add to homescreen
- You should now see an icon for your app

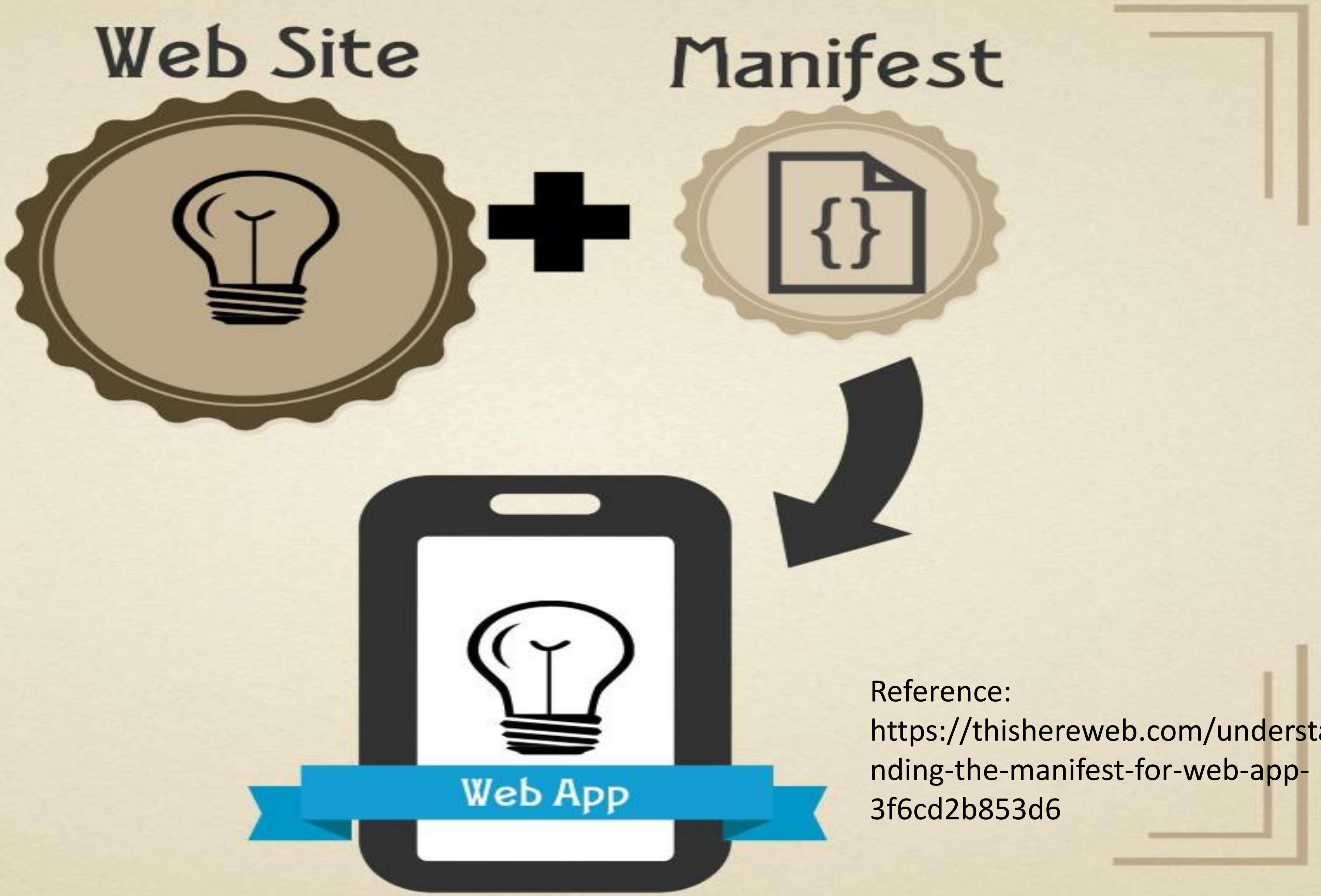


- Other Devices will be similar
  - Open site in browser
  - Go to menu options
    - Add to homescreen

The Manifest File

- However, for Web Apps to be successful, they need to work how the user would expect a native application to work.
- The manifest is a simple JSON file that gives the developer, the ability to control how the app appears and is launched by the user.

Reference: https://developers.google.com/web/updates/2014/11/Support-for-installable-web-apps-with-webapp-manifest-in-chrome-38-for-Android



Reference: https://thishereweb.com/understanding-the-manifest-for-web-app-3f6cd2b853d6

To integrate the manifest:

1. Create and deploy a manifest file (as part of your web site).

2. Add a link element from the pages in your app pointing to the manifest file.

Reference: https://developers.google.com/web/updates/2014/11/Support-for-installable-web-apps-with-webapp-manifest-in-chrome-38-for-Android

Example Manifest File: manifest.json

```
"short_name": “Glenns",

"name": “Glenns Amazing Application ",

"icons": [

  {
    "src": "launcher-icon-2x.png",
    "sizes": "96x96",
    "type": "image/png"
  },

  {
    "src": "launcher-icon-3x.png",
    "sizes": "144x144",
    "type": "image/png"
  },

//  you can continue to include other sizes for different devices

],

"start_url": "/index.html",

"display": "standalone",

"orientation": "landscape"}
```

- **name**: Name to show up in app listing or with icon
- **short Name**: when present used instead of name along with app listings
- **start_url**: this is the URL that is launched
- **display:** determine how the app should be presented to the user. Options are fullscreen, minimul-ui, standalone, or browser (opens in browser)
- **orientation:** default the app to a particular orientation e.g. “landscape”, “portrait”.

Reference:
https://thishereweb.com/understa
nding-the-manifest-for-web-app-
3f6cd2b853d6

- **Icons** is an array of objects that hold the characteristics for images used by the app presentation. Each object can have a few different values:
  - **src**: this points to the location of the asset (image file),
  - **type**: determines the type of the icon file.
  - **sizes**: this declares the image size.
    - each platform can and does require its own unique file sizes,

Reference:
https://thishereweb.com/understa
nding-the-manifest-for-web-app-
3f6cd2b853d6

Once you have the manifest created and it is hosted on your site, all you need to do is add a link tag from **All** the pages of your app:

```
<link rel="manifest" href="/manifest.json">
```

```
<!doctype>
<html>
<title>GlennsPage</title>
  <link rel="manifest" href=“/manifest.json">
```

You now have a web app that will look and feel like a native app to the user!

Web Apps vs Native Apps

Pros of Native Apps

- Since native apps work with the device’s built-in features, they perform faster on the device.

- Native apps get full support from the concerned app stores.
  - the user may be assured of more safety and security with the app.

Ref: https://www.lifewire.com/pros-and-cons-of-native-apps-and-mobile-web-apps-2373173

Cons of Native Apps

Native apps tend to be a more expensive proposition to the developer. e.g developing for multiple native platforms

The cost of updating and maintenance is also higher for native apps,

There is a process of getting the app approved at the app store and subsequent cost.

Ref: https://www.lifewire.com/pros-and-cons-of-native-apps-and-mobile-web-apps-2373173

- **Pros of Web Apps**

- Web apps are much easier to maintain, as they have a common code base across multiple mobile platforms.

- Web apps do not require developers to submit the app to an app store for approval.

- they can be released at any time and in any form, as per the developer's preferences.

- Users do not need an app store. (a web site and manifest)

Ref: https://www.lifewire.com/pros-and-cons-of-native-apps-and-mobile-web-apps-2373173

- **Cons of Web Apps**

- limited scope as far as accessing a mobile device’s features is concerned.

- Users may sometimes find it difficult to discover a Web app, as it is not systematically listed in any app store.

- Since there is no regularized quality control system for Web apps, users may not always be guaranteed safety and security.

Ref: https://www.lifewire.com/pros-and-cons-of-native-apps-and-mobile-web-apps-2373173

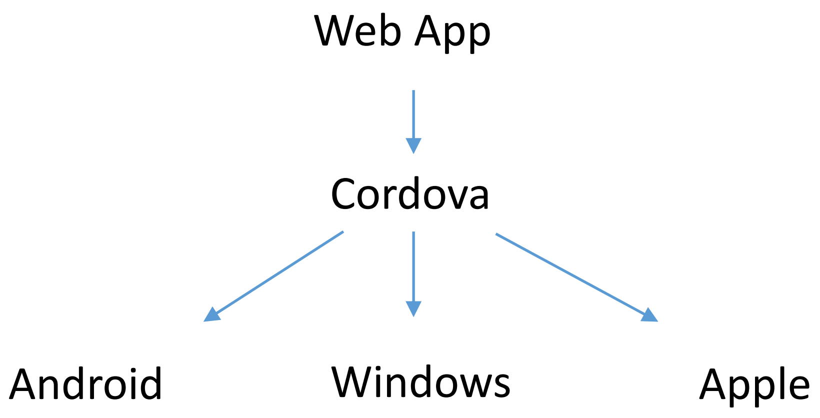
Apache Cordova

- Turning web apps into Native apps

Why? - using native device features that would otherwise be impossible.
Submitting to stores.

Apache Cordova has been a prevalent toolkit for developing cross-platform mobile client side applications.

It encapsulates the web app into a native app



The distinctive technical features of Cordova.

Cordova uses the web browsers available in almost all mobile platforms to host web-applications.

However, a Cordova app executes web-app code within a browser component that is hosted by a native app.

This makes it possible to add extra components to the app that can provide the web-app with access to native hardware and software features (e.g. the device’s camera)

Cordova’s popularity

Developers have adopted this approach because it allows them to use their web-development skills to produce native apps without having to learn to use the native platforms (and languages).

Cordova also allows them to build apps using familiar tools, while only needing to use native tools for the final compilation build.

Additional work the developer must to do to distribute a Cordova application.

The developer still needs to do the final build of the native app. That means that the native development platform must be available.

The native device features for a range of devices require separate Cordova APIs,

it also means that for each platform the app is built, there must be a separate version of the web app using the native API bridge functions.

The Cordova Build service

The Cordova build service assists by providing an online build service so that developers do not need to procure, install and maintain a range of native development platforms.

However, the different app versions must still be created.

Testing

All applications should be thoroughly tested prior to deployment and in-situ to ensure they perform adequately and according to specification.

During the development process, a mobile application should be tested within a controllable test environment.

Three main options (depending on app format):

- Test in an emulator,
- test on a Real Device attached to the development machine by network, Bluetooth or cable,
- test in a browser (web app only).

Emulator:

- tailored to target device (but no need to have access to target device), uses same code-base (in emulation) as target device, can simulate target device hardware (e.g. camera, GPS),

- BUT,
- slow execution speed – particularly during debugging,
- no possibility of simulating touch-based input
- May have difficulty simulating some features

e.g. Andriod emulator in cordova

Real Device:

- platform is exactly the same as the target platform, so best possible experience,
- faster
- full set of device features available,

- BUT,
- need device (may be costly if targeting multiple devices)

BROWSER:

only useful for web-apps,

fast testing environment,

good debug tools available,

BUT,

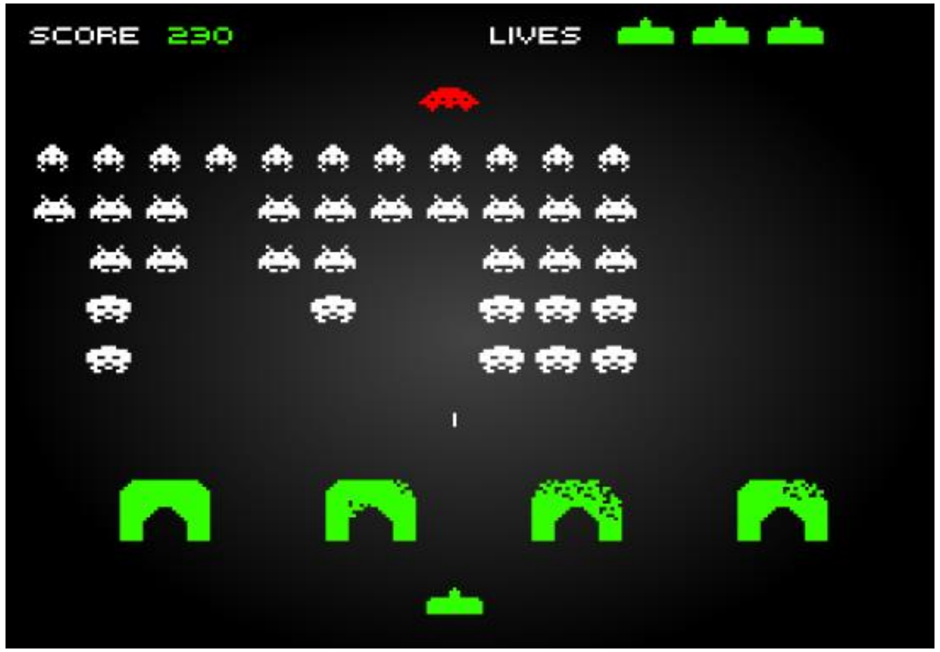
no actual, or simulation, of device hardware

In short it depends on the app and the functionality features you need to test which would be the best option choice for developmental testing.

Your Project and Assessment: Design a Basic Test Plan for your App

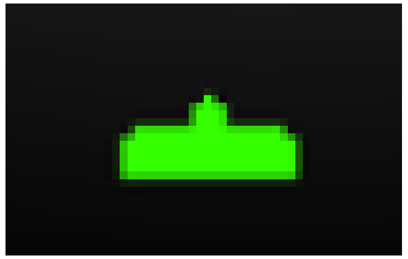
- Functionality list – What was included in your list – test and document whether it all works!
- Your Use case descriptions – check and document whether each use case does as expected!
- Build a Test Report (to be included in your final report).

Functionality list



Example From Space invaders

Example requirement: The **Player/Defender**



- The **Player** can move left to right at the bottom of the screen, but cannot move off-screen.
- The **player** can fire **bullets**.

Player firing
Id: Use Case 5
Actors: Player
Preconditions: Game is executing and no other firing is in progress.
Flow of events: 1 User presses the firing key. 2 The system creates a new upward moving missile at the ship’s location. 3 Use case ends.
Post Conditions: A player firing is in progress.

Test the preconditions, flow of events and the post conditions

Test case x

- On pressing the firing key the system creates a new upward moving missile at the ship’s location

- Test Result:

…

Move Ship
Id: Use Case 4
Actors: Player
Preconditions: Game is executing.
Flow of events: 1 User press the right or left arrow key. 2 The system moves the ship and displays the ship in its new position. 3 Use case ends.
Post Conditions: The ships position has changed.

Test and Report on each use case and requirement.

Test Case x

- User press the right arrow key.

Test Result:

The system moves the ship and displays the ship in its new position

Test Case x

- User press the left arrow key.

Test Result:

The system moves the ship and displays the ship in its new position but does not render smoothly

Token Interaction Diagram

	Bullets	Bombs	Aliens	Defender	Bases	Spacecraft	Game Area
Bullets fired	X						
Bombs dropped by aliens	Collision	X					
Aliens (Move)	Collision	X	X				
Defender (move)	X	Collision	Collision	X			
Bases	Collision	Collision	Collision	X	X		
Spacecraft	Collision	X	X	X	X	X	
Game area	Collision	Collision	Collision	Collision	X	Collision	X
Score	X	X	X	X	X	X	X
Lives	X	X	X	X	X	X	X

In a game like space invaders you would want to check all the collisions worked as expected.

Best to use a diagram for both design and testing

Test object interactions e.g. with a diagram

Test Report

- Ensure you test all user interactions and all expected behaviours are tested.

- Create a test report of the outcomes of your tests.

- **Not everything has to be perfect but it should all be tested!**