



# **WEB TECHNOLOGIES 1**

OFFLINE WEB APPLICATION

# OFFLINE WEB APPLICATION

- A typical web app consists of a variety of resource files like HTML for markup, CSS for styling, Javascript for application logic, and others like images, fonts, etc.
- On the other hand, the application is worth nothing without data that it either displays or stores (or both)



# WHY “OFFLINE APPS”

## Why “Offline Apps”



No Internet connection



Poor Internet connection



Improve performance

## Offline Apps Includes



Offline Storage

data on client-side



Application cache

html, js, css, images



# APPCACHE

- HTML5 allows us to create a “list” of files accessible after the browser is disconnected.
- The browser will always have to download the main HTML file and all the files in this “list” at least once.
- Offline web apps are composed of two parts:
  - Offline cache
  - Client side storage.
- You use offline cache (also known as AppCache) to make the resources available in offline mode .
- In client side storage users can add, update and remove data without connectivity.




# APPCACHE

- AppCache is communicated to the browser through a resource known as AppCache Manifest. You tell the browser about the manifest through manifest attribute of the html tag.
- When the browser sees the manifest, it will download and cache the manifest as well as the page with the manifest attribute.
- It then goes through the entries in the manifest and caches them all too.



## TO CACHE FILES, DO THE FOLLOWING:

- Create a manifest file that lists the files and other web resources you want to cache.
  - Reference the manifest file in the header of every page you want cached.
  - Every page with the manifest attribute specified will be cached when the user visits it.
  - If the manifest attribute is not specified, the page will not be cached (unless the page is specified directly in the manifest file).
- 

# THE MANIFEST

- The “list” is called the *manifest file*. This manifest is declared with an attribute of the `<html>` Tag.
- By convention you should use ‘appcache’ extension for the cache manifest “file”.
- The filename does not matter, but as a convention you may use names like : *manifest.appcache*, *myapplicaiton.manifest*.
- The cache manifest resource mime type should be set to ‘text/cache-manifest’.
- The content encoding of the file must be set to **UTF-8**.



# MANIFEST SECTIONS

- A manifest can have three distinct sections:
  - CACHE
  - NETWORK
  - FALLBACK.





# MANIFEST SECTIONS

- **CACHE: (required)** This is the default section for entries. Files listed under this header (or immediately after the CACHE MANIFEST) will be explicitly cached after they're downloaded for the first time.
- **CACHE MANIFEST.**

For example: CACHE MANIFEST

# This is a comment

index.html

css/style.css

js/script.js

- This file will tell the browser to keep three files in cache: index.html, style.css and script.js.
- So now, if you are disconnected and you refresh your browser, you will always be able to use your web page.

# MANIFEST SECTIONS

- NETWORK: Files listed in this section are explicitly NOT cached.  
“All requests to such resources bypass the cache, even if the user is offline”
- FALLBACK: This section will specifies fallback pages the browser will use when a resource is not accessible. Entries in this section lines of mapping urls: URL\_REQUESTED URL\_FALLBACK.
- Note :
  - All the files listed outside any sections are considered listed in the explicit CACHE section.
  - Lines starting with a hash symbol (#) are comment lines.



# MANIFEST FILE

## CACHE MANIFEST

# This is a comment

## CACHE:

index.html

css/style.css

js/script.js

## NETWORK:

# All requests to the Api need a  
network connection

/api

## FALLBACK:

add.html offline.htm

- This manifest file will caches 3 files, tell the browser to always try to download resources inside */api*, and if the *add.html* file is requested, the display the *offline.html* file.
- One side effect using application cache is that the browser won't try to download updates cached files until the manifest file itself is updated.



# MANIFEST FILE

- To solve this problem we can add a *version number* in a comment line at the top of the manifest file to force the browser to update its cache.
- The browser updates the application cache only when the manifest changes.
- We need to ensure that the manifest file changes whenever we build a new application version, so that browsers know that they need to update the application cache.

```
CACHE MANIFEST
# v1.0 : 10-08-2014

CACHE:
# pages
index.html

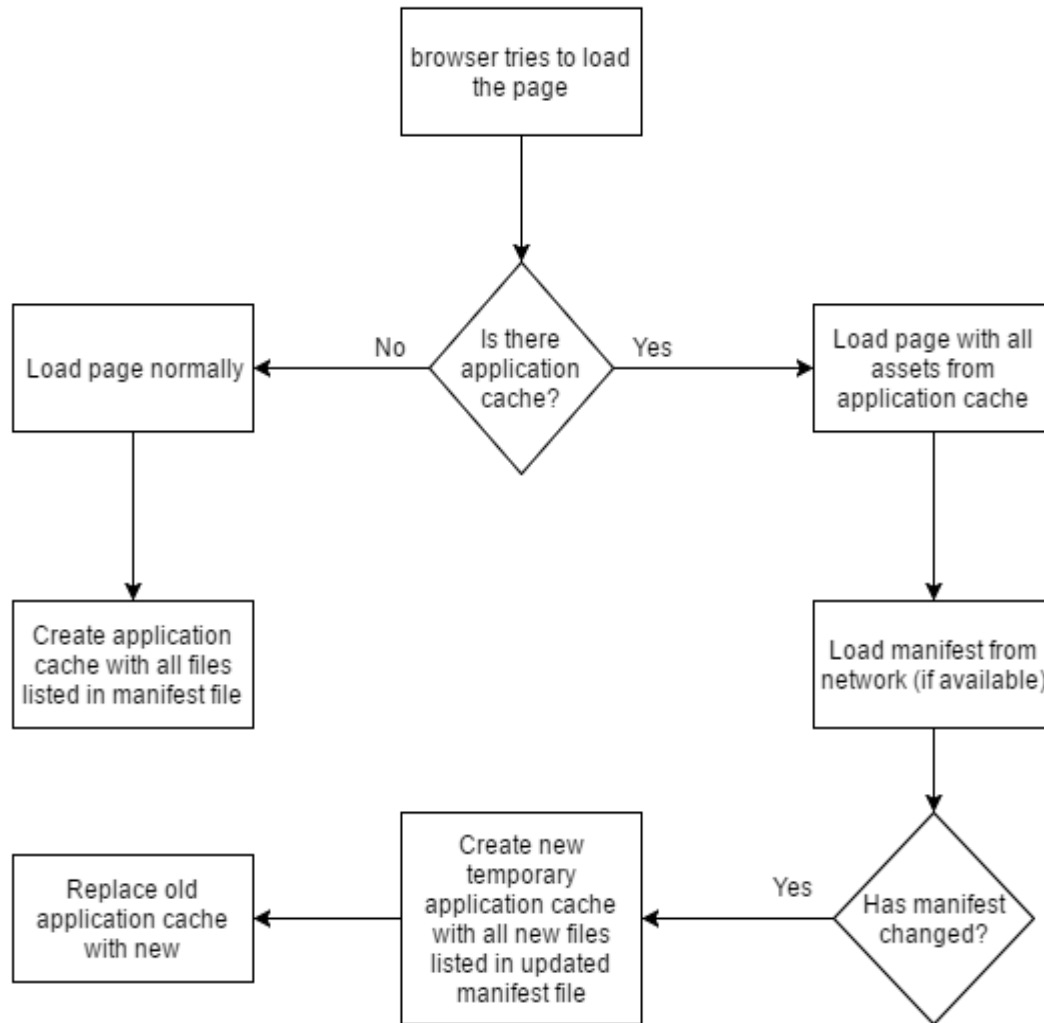
# styles & scripts
css/theme.css
js/jquery.min.js
js/default.js

# images
/favicon.ico
images/logo.png

NETWORK:
login.php

FALLBACK:
/ /offline.html|
```

# HOW DOES THE BROWSER USE THE MANIFEST FILE?




- If there is no application cache, the browser will create it when visiting the page.
- When there is an application cache, the browser will use cached files to load the page.
- It will also try to load the manifest from the network and check if it has changed.
- If there is a change, it will create a new temporary cache, with new files.
- When it's done, the browser will replace the old application cache with the new one.

# HOW TO DETECT WHEN A BROWSER GOES OFFLINE

- JavaScript code can detect when a browser is online or offline using *navigator.onLine*.
- This property is supported by most browsers and returns true when online or false when offline.
- Firefox also allows you to attach handlers to the online and offline events of the window object, e.g.

//standard event listeners

- `window.addEventListener('online',function(){ console.log('online');},false)`
  - `window.addEventListener('offline',function(){ console.log('offline');}, false)`
- 

# BENEFITS

An application cache provides the following benefits:

- **Offline browsing:** users can navigate a site or use a web application although they are offline
- **Speed:** cached resources are local, and therefore load faster
- **Reduce HTTP request and server load** — The browser will only have to download the updated/changed resources from the remote server that minimize the HTTP requests and saves precious bandwidth as well as reduce the load on the web server..

