# GCS 3205 – Mobile Web Application

Department of Computer Science
Gulu University

Daniel Ogenrwot

d.ogenrwot@gu.ac.ug
https://www.gu.ac.ug

October 20, 2020

# Lecture-5 Application Cache

- **Web application caching** is the process of storing dynamically generated data for reuse and leaving data closer to the end user.
- **HTML5 introduces application cache**, which means that a web application is cached, and accessible without an internet connection.

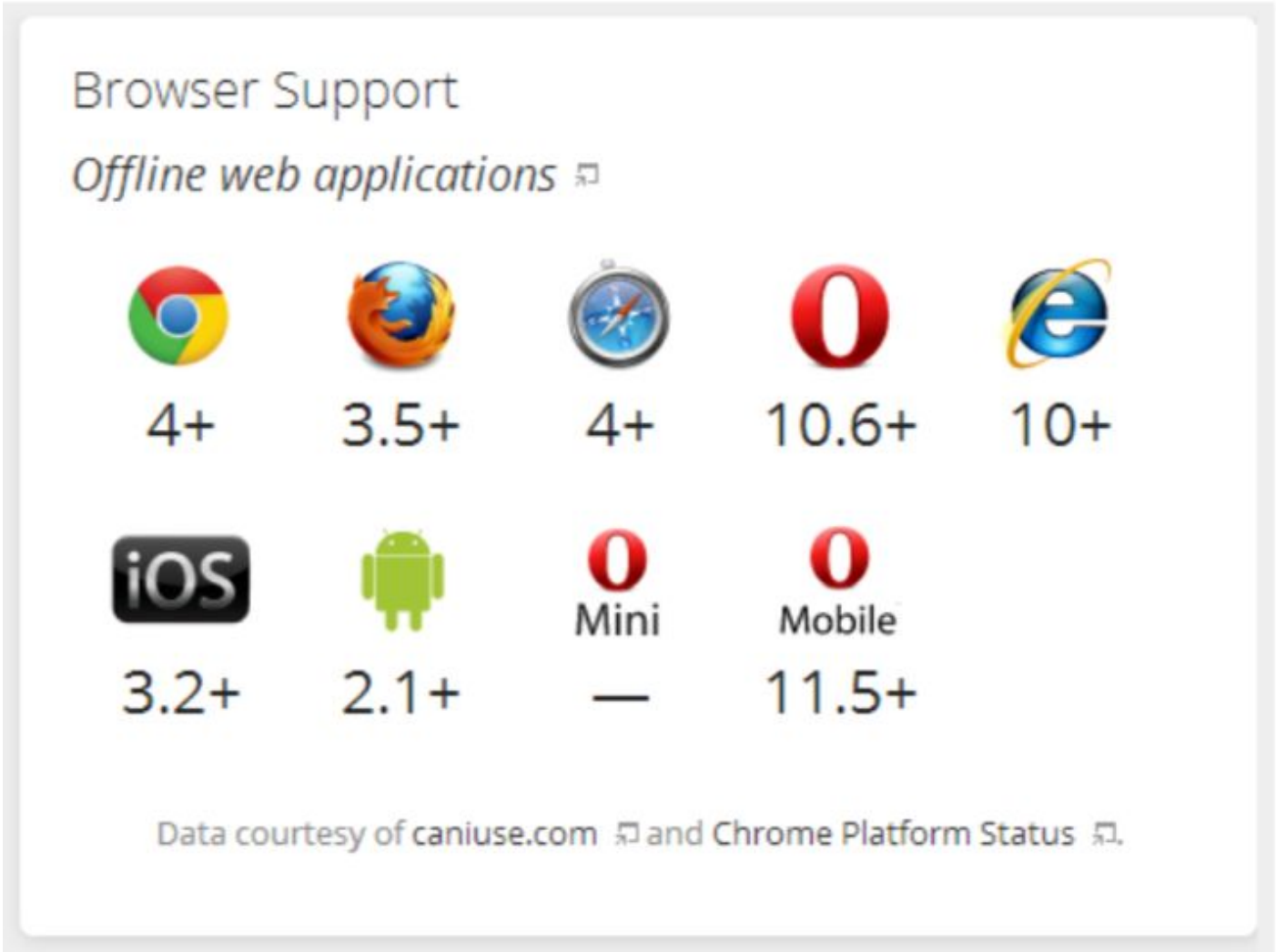
# Advantages

- Offline browsing - users can use the application when they're offline
- Speed - cached resources load faster
- Reduced server load - the browser will only download updated/changed resources from the server

# Disadvantages

- Stale Data
- Overhead
- Complexity

# Browser Support



# Type of Caching

- **Client Caching**
  - **- browser caches URLs for future uses**
  - **- Mozilla Firefox, Google Chrome**
- Proxy Caching
  - proxy server caches most requested URLs
  - Varnish ( reverse proxy ) - CDN ( forward proxy )
- Server-side Caching
  - server side cache reduces load on server
  - File System Cache
  - In-Memory Cache (MemCached, Redis)

# Cache Manifest Basics

To enable application cache, include the manifest attribute in the document's <html>

Tag:

```
<!DOCTYPE HTML>
<html manifest="demo.appcache">
...
</html>
```

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 recommended file extension for manifest files is: ".appcache"

*A manifest file needs to be served with the correct media type, which is "text/cache-manifest". Must be configured on the web server.*

# The Manifest file

The manifest file is a simple text file, which tells the browser what to cache (and what to never cache).

The manifest file has three sections:

**CACHE MANIFEST** - Files listed under this header will be cached after they are downloaded for the first time

**NETWORK** - Files listed under this header require a connection to the server, and will never be cached

**FALLBACK** - Files listed under this header specifies fallback pages if a page is inaccessible

# CACHE MANIFEST

The first line, CACHE MANIFEST, is required:

```
CACHE MANIFEST
/theme.css
/logo.gif
/main.js
```

The manifest file above lists three resources: a CSS file, a GIF image, and a JavaScript file. When the manifest file is loaded, the browser will download the three files from the root directory of the web site. Then, whenever the user is not connected to the internet, the resources will still be available.

# NETWORK

The NETWORK section below specifies that the file "login.asp" should never be cached, and will not be available offline:

```
NETWORK:
login.asp
```

An asterisk can be used to indicate that all other resources/files require an internet connection:

```
NETWORK:
*
```

# FALLBACK

The **FALLBACK** section below specifies that "offline.html" will be served in place of all files in the /html/ catalog, in case an internet connection cannot be established:

```
FALLBACK:
/html/ /offline.html
```

Note: The first URI is the resource, the second is the fallback.

# UPDATING THE CACHE

Once an application is cached, it remains cached until one of the following happens:

- The user clears the browser's cache
- The manifest file is modified (see tip below)
- The application cache is programmatically updated

Be careful with what you cache.

Once a file is cached, the browser will continue to show the cached version, even if you change the file on the server. To ensure the browser updates the cache, you need to change the manifest file.

*Note: Browsers may have different size limits for cached data (some browsers have a 5MB limit per site).*

# Full example of .appcache file

```
CACHE MANIFEST
# This is a comment
CACHE:
/css/screen.css
/css/offline.css
/js/screen.js
/img/logo.png
http://example.com/css/styles.css
FALLBACK:
/ /offline.html
NETWORK:
*
```

# THANK YOU