The `appcache` package stores the static parts of a Meteor application (the client side Javascript, HTML, CSS, and images) in the browser's application cache. To enable caching simply add the `appcache` package to your project.

- Once a user has visited a Meteor application for the first time and the application has been cached, on subsequent visits the web page loads faster because the browser can load the application out of the cache without contacting the server first.
- Hot code pushes are loaded by the browser in the background while the app continues to run. Once the new code has been fully loaded the browser is able to switch over to the new code quickly.
- The application cache allows the application to be loaded even when the browser doesn't have an Internet connection, and so enables using the app offline.

(Note however that the `appcache` package by itself doesn't make *data* available offline: in an application loaded offline, a Meteor Collection will appear to be empty in the client until the Internet becomes available and the browser is able to establish a DDP connection).

To turn AppCache off for specific browsers use:

```
Meteor.AppCache.config({
  chrome: false,
  firefox: false
});
```

The supported browsers that can be enabled or disabled include, but are not limited to, `android`, `chrome`, `chromium`, `chromeMobileIOS`, `firefox`, `ie`, `mobileSafari` and `safari`.

Browsers limit the amount of data they will put in the application cache, which can vary due to factors such as how much disk space is free. Unfortunately if your application goes over the limit rather than disabling the application cache altogether and running the application online, the browser will instead fail that particular *update* of the cache, leaving your users running old code.

Thus it's best to keep the size of the cache below 5MB. The `appcache` package will print a warning on the Meteor server console if the total size of the resources being cached is over 5MB.

Starting from `appcache@1.2.5`, if you need more advanced logic to enable/disable the cache, you can use the `enableCallback` option that is evaluated on a per-request basis. For example:

```
// Enable offline mode using a value from database and certificate validation
Meteor.AppCache.config({
  // This option is available starting from appcache@1.2.4
  enableCallback: () => {
    if (!getSettingsFromDb("public.appcache_enabled")) {
      return false;
    }

    const validation = validateClientCert({
      clientCertPayload: req.headers["x-client-cert"],
      url: req.url.href,
    });

    return validation.passed;
```

```
  },
});
```

If you have files too large to fit in the cache you can disable caching by URL prefix. For example,

```
Meteor.AppCache.config({ onlineOnly: ['/online/'] });
```

causes files in your `public/online` directory to not be cached, and so they will only be available online. You can then move your large files into that directory and refer to them at the new URL:

```
<img src="/online/bigimage.jpg">
```

If you'd prefer not to move your files, you can use the file names themselves as the URL prefix:

```
Meteor.AppCache.config({
  onlineOnly: [
    '/bigimage.jpg',
    '/largedata.json'
  ]
});
```

though keep in mind that since the exclusion is by prefix (this is a limitation of the application cache manifest), excluding `/largedata.json` will also exclude such URLs as `/largedata.json.orig` and `/largedata.json/file1`.

For more information about how Meteor interacts with the application cache, see the AppCache page in the Meteor wiki.