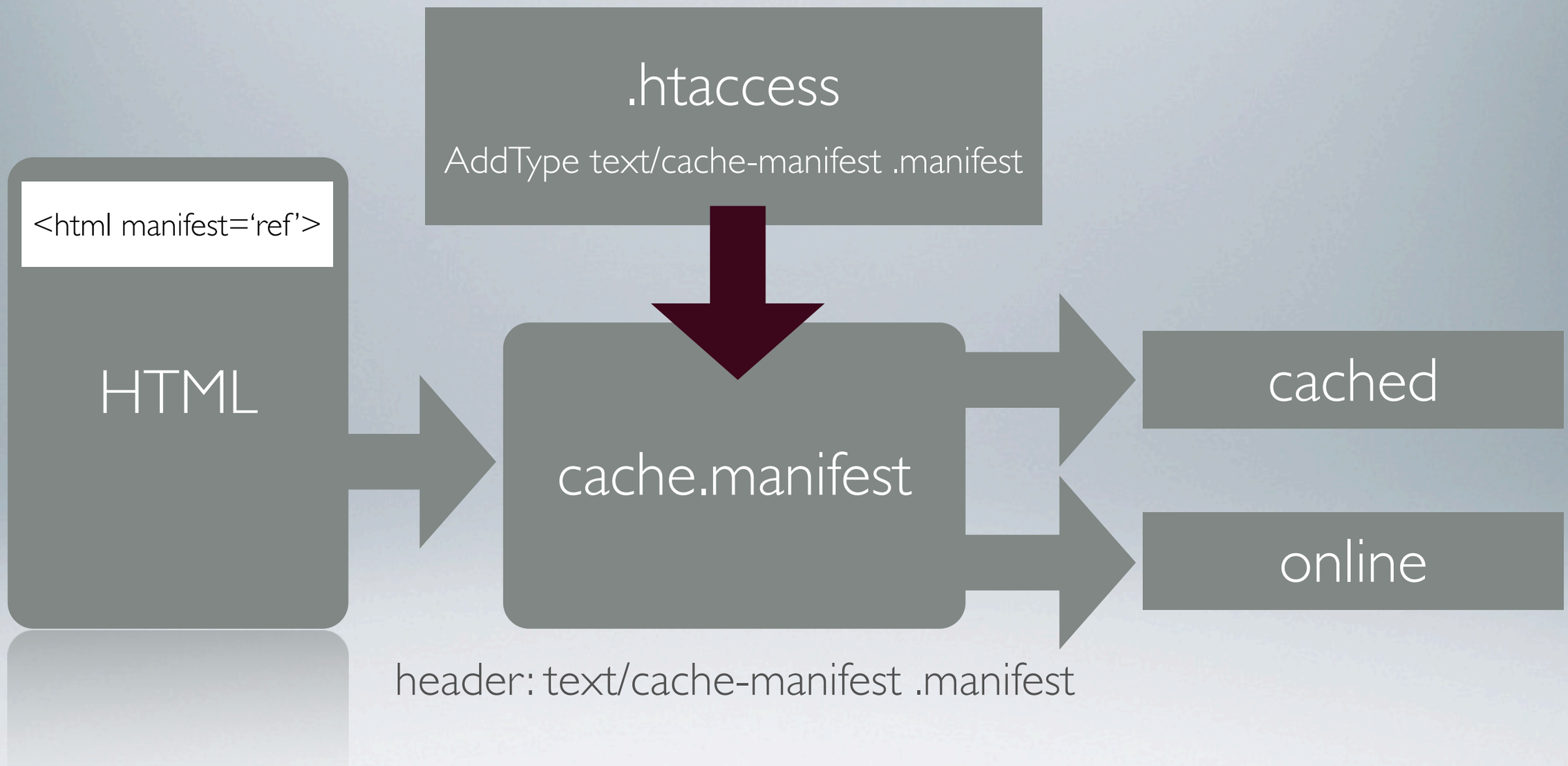
HTML 5

Cache Manifest



Browser Cache:

index.html, styles.css, script.js, image.png etc.

```
<html manifest="example.appcache">
  ...
</html>
```

```
CACHE MANIFEST
index.html
stylesheet.css
images/logo.png
scripts/main.js
```

```
CACHE MANIFEST
# 2010-06-18:v2

# Explicitly cached 'master entries'.
CACHE:
/favicon.ico
index.html
stylesheet.css
images/logo.png
scripts/main.js

# Resources that require the user to be online.
NETWORK:
login.php
/myapi
http://api.twitter.com

# static.html will be served if main.py is inaccessible
# offline.jpg will be served in place of all images in images/large/
# offline.html will be served in place of all other .html files
FALLBACK:
/main.py /static.html
images/large/ images/offline.jpg
*.html /offline.html
```

```
CACHE MANIFEST
# 2010-06-18:v3

# Explicitly cached entries
index.html
css/style.css

# offline.html will be displayed if the user is offline
FALLBACK:
/ /offline.html

# All other resources (e.g. sites) require the user to be online.
NETWORK:
*

# Additional resources to cache
CACHE:
images/logo1.png
images/logo2.png
images/logo3.png
```

IDL

```
interface ApplicationCache : EventTarget {

  // update status
  const unsigned short UNCACHED = 0;
  const unsigned short IDLE = 1;
  const unsigned short CHECKING = 2;
  const unsigned short DOWNLOADING = 3;
  const unsigned short UPDATEREADY = 4;
  const unsigned short OBSOLETE = 5;
  readonly attribute unsigned short status;

  // updates
  void update();
  void abort();
  void swapCache();

  // events
           attribute EventHandler onchecking;
           attribute EventHandler onerror;
           attribute EventHandler onnoupdate;
           attribute EventHandler ondownloading;
           attribute EventHandler onprogress;
           attribute EventHandler onupdateready;
           attribute EventHandler oncached;
           attribute EventHandler onobsolete;
};
```

```
var appCache = window.applicationCache;

switch (appCache.status) {
  case appCache.UNCACHED: // UNCACHED == 0
    return 'UNCACHED';
    break;
  case appCache.IDLE: // IDLE == 1
    return 'IDLE';
    break;
  case appCache.CHECKING: // CHECKING == 2
    return 'CHECKING';
    break;
  case appCache.DOWNLOADING: // DOWNLOADING == 3
    return 'DOWNLOADING';
    break;
  case appCache.UPDATEREADY:  // UPDATEREADY == 4
    return 'UPDATEREADY';
    break;
  case appCache.OBSOLETE: // OBSOLETE == 5
    return 'OBSOLETE';
    break;
  default:
    return 'UKNOWN CACHE STATUS';
    break;
};
```

```
var appCache = window.applicationCache;

appCache.update(); // Attempt to update the user's cache.

...

if (appCache.status == window.applicationCache.UPDATEREADY) {
  appCache.swapCache();  // The fetch was successful, swap in the new
cache.
}
```

```
// Check if a new cache is available on page load.
window.addEventListener('load', function(e) {

  window.applicationCache.addEventListener('updateready', function(e) {
    if (window.applicationCache.status ==
window.applicationCache.UPDATEREADY) {
      // Browser downloaded a new app cache.
      // Swap it in and reload the page to get the new hotness.
      window.applicationCache.swapCache();
      if (confirm('A new version of this site is available. Load it?')) {
        window.location.reload();
      }
    } else {
      // Manifest didn't changed. Nothing new to server.
    }
  }, false);

}, false);
```

```
function handleCacheEvent(e) {
  //...
}

function handleCacheError(e) {
  alert('Error: Cache failed to update!');
};

// Fired after the first cache of the manifest.
appCache.addEventListener('cached', handleCacheEvent, false);

// Checking for an update. Always the first event fired in the sequence.
appCache.addEventListener('checking', handleCacheEvent, false);

// An update was found. The browser is fetching resources.
appCache.addEventListener('downloading', handleCacheEvent, false);

// The manifest returns 404 or 410, the download failed,
// or the manifest changed while the download was in progress.
appCache.addEventListener('error', handleCacheError, false);

// Fired after the first download of the manifest.
appCache.addEventListener('noupdate', handleCacheEvent, false);

// Fired if the manifest file returns a 404 or 410.
// This results in the application cache being deleted.
appCache.addEventListener('obsolete', handleCacheEvent, false);

// Fired for each resource listed in the manifest as it is being fetched.
appCache.addEventListener('progress', handleCacheEvent, false);

// Fired when the manifest resources have been newly redownloaded.
appCache.addEventListener('updateready', handleCacheEvent, false);
```

```
function logEvent(event) {
    console.log(event.type);
    if(event.type == "updateready") {
      // update downloaded and restarting app to use
      alert("APP Update!\n Your app has been updated.\nIt will now restart...");
      location.reload();
    }
}

window.applicationCache.addEventListener('checking',logEvent,false);
window.applicationCache.addEventListener('noupdate',logEvent,false);
window.applicationCache.addEventListener('downloading',logEvent,false);
window.applicationCache.addEventListener('cached',logEvent,false);
window.applicationCache.addEventListener('updateready',logEvent,false);
window.applicationCache.addEventListener('obsolete',logEvent,false);
window.applicationCache.addEventListener('error',logEvent,false);
```

JQuery AJAX

```
$.ajaxSetup({
    cache: true
});
```