Optimizing for Speed: Advanced JavaScript/CSS Management in DNN

Ian Robinson
Enliven

Why are we here?

- Performance is a feature *(an important one)*
  - Fast sites lead to satisfied users
- DNN was largely optimized on the server side, was not so much on the client side
- DNN 6.1+ introduced a new framework for managing CSS and JS

Session Outcomes

- You understand
  - the client resource “problem domain”
  - the solution foundation built into DNN
  - why and how to use it going forward
    - As a developer
    - As a site administrator

80% *of the end-user response time is* ***spent on the front-end****.*

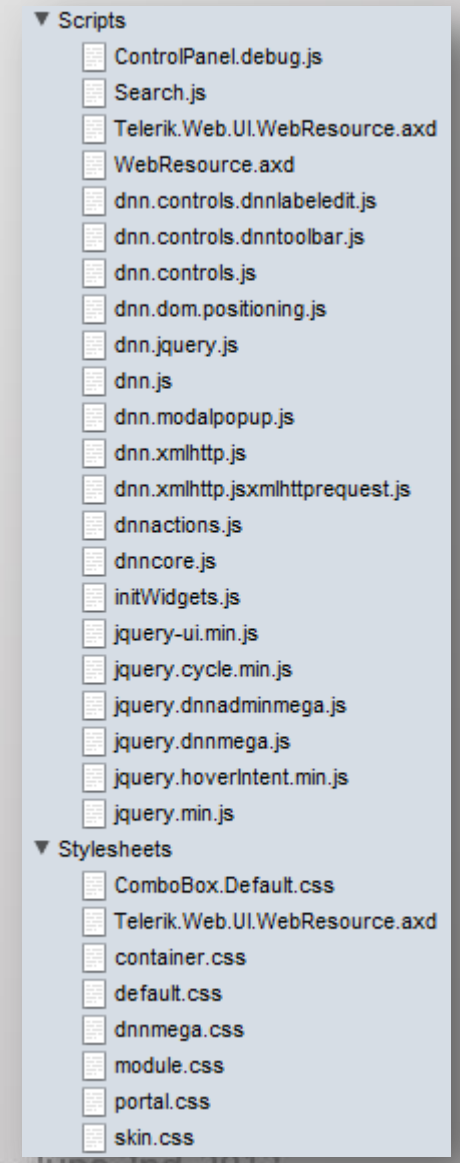
Most of this time is tied up in ***downloading all the components*** *in the page: images,* ***stylesheets, scripts****, Flash, etc.*

Reducing the number of components *in turn reduces the number of HTTP requests required to render the page.* ***This is the key to faster pages****.*

-Yahoo! Exceptional Performance Team

DNN 6.o Resources Overview

- Clean install, home page
  - unauthenticated
    - 6 CSS Files
    - 13 JavaScript Files
  - Logged in as host
    - 8 CSS Files
    - 22 JavaScript Files



Strategy (DNN 6.1+)

- Reduce the file size of each resource
  - Action: minify content
- Only deliver a resource that is needed
  - Action: Only load files that are requested, only request what is actually going to be used
- Reduce the number of resources into as few as possible
  - Action: file combination

Client Resource Management

- DNN 6.1.0 – 6.1.5
  - Settings managed directly in web.config
- DNN 6.2+
  - Settings managed through DNN UI
  - Host defines defaults
  - Each site (i.e. portal) can override defaults
- All versions
  - Debug=true in web.config **disables combination**

Request what is needed

- Client Resource Manament
  - DNN core library used for requesting that a file be loaded
  - Request a file in code through public API
  - Or indirectly through a user control

File Registration: API

```
ClientResourceManager.RegisterScript(page, scriptPath);
```

- Parameters
  - Page
    - The current page
  - FilePath
    - string - Local file path or absolute URL
  - Priority
    - int – relative to other priorities sent in
  - Provider (location)
    - string – name of provider to use

File Registration: Controls

```
<%@ Register TagPrefix="dnn"
             Namespace="DotNetNuke.Web.Client.ClientResourceManagement"
             Assembly="DotNetNuke.Web.Client" %>

<dnn:DnnJsInclude runat="server"
                  FilePath="jquery.cycle.min.js"
                  PathNameAlias="SkinPath" />

<dnn:DnnJsInclude runat="server"
                  FilePath="/DNNStandard/StandardMenu.js"
                  PathNameAlias="SkinPath" />
```

Minify and combine

- Sorts through all requests and weeds out duplicates
- Handles versioning and caching/persistence as well
- Oh yeah, and file ordering too

Render on the page

- File registration is scoped to page lifecycle
- We end up with <script> & <style> actually on the page for requested files.
- File placement can be dictated when registering, defaults to the top of the page for compatibility reasons

Details: Versioning

- When combining:
  - Snapshot in time of your CSS/JS files
  - Version is incremented in the host/admin UI
  - It is an integer value
  - Automatically increments when you upgrade an extension or edit portal.css in the browser

Details: Caching & Persistence

- When combined
  - ASP.NET Output Caching
  - Cache to disk
    - App_Data\ClientDependency is default

Details: Tips

- Priority
- Provider
  - ForceProvider="DnnFormBottomProvider"
- PathNameAlias
  - SkinPath
  - SharedScripts
    - 6.1 – json2.js
    - 6.2 – knockout.js, jquery history plugin

6.1 versus 6.2

- Web.config - don’t touch in 6.2
- Cache busting – not done as frequently in 6.2

Resources

- Google “Client Resource Management” and click on the second link. That’s the DNN Wiki.

Thanks to all our Generous Sponsors!

DotNetNuke™

POWER DNN
DOTNETNUKE® DONE RIGHT!