

1. **Setup Fiddler**
2. **Open all links in tabs**
3. **Branch from codebase**

****

1. **Capture w/ Network tab**
   1. Filters
   2. Totals – load is red line, DOMContentLoaded is blue line
   3. Size & Content – size is what was transferred on the wire, content is size of actual useable asset
   4. Time & Latency – time is total time, latency is waiting time where we weren’t getting bytes. Also the more transparent part of the timeline
   5. Select a resource, show headers and preview
   6. Show preserve log – make additional requests
   7. Copy all as HAR, display in HAR Viewer, show page timeline and hide statistics
2. **Audits & PageSpeed**
   1. Reload Page and Audit on Load
   2. Explain that I have an extension, others exist
   3. Red should be fixed first, and so on
   4. PageSpeed is another extension, that is similar to Audits, but has separate features
3. **Combine/Minify**
   1. Change comments in \_Layout and show BundleConfig
4. **Compression**
   1. Change two values from false to true in web.config
   2. Show changes to Size/Content
5. **HTTP Caching**
   1. Uncomment caching section in web.config
   2. Toggle Disable Cache option in Chrome DevTools
   3. Show 304’s appear in Network tab
   4. Demo cache headers at http://redbot.org/ with <http://www.codepalousa.com/>
6. **Re-Run Audits and PageSpeed**
   1. Commit in Git
7. **Sprite Demo**
   1. Open Texas page and show off the HTTP count/size
   2. Generate sprites for Texas
   3. Rename .png.css file to .css to avoid bug
   4. Update \_layout & League
   5. Demo
8. **Image Optimization**
   1. Show Sprite, and optimize with <http://tinypng.com/>
   2. Show savings in Network tab
9. **DataUri Demo**
   1. Show ball/H1 icon on homepage
   2. Drop ball into <http://dataurl.net/>
   3. Use Visual Studio to change the ball to a DataUri in screen.css
   4. Reload to show asset missing
10. **Revert changes & Disable Fiddler**



1. **Async Script Demo**
   1. Uncomment ThirdPartyScript – these we can’t always control the placement of
   2. Reload page and notice the 3 second delay due to blocked parsing
   3. Replace with snippet
   4. Refresh



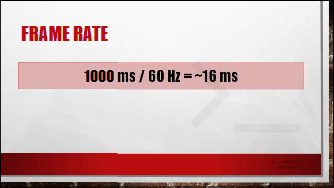
1. **Prerender Demo**
   1. Close Dev Tools!
   2. View Source on Home Page to show link tag
   3. Open chrome://net-internals/#prerender and Task Manager (Shift+Esc) to show hidden tab being used



1. **Profile Application**
   1. Stop IIS Express
   2. Cover types of Profiling Modes
   3. Use IIS Express, Chrome, and Yellow Option
   4. Allow CPU to flatten after start – mention this should be automated
   5. Click on All Leagues
   6. Stop Profiler
   7. Examine Outgoing HTTP Requests
   8. Select the All hump and describe the hit count and time w/ children.
   9. Toggle percent and milliseconds
   10. Right click on the Call Tree and “Expand the most expensive call stack”
   11. Find the 160 queries, switch to DB view
   12. See the problem in the decompiled code.
   13. Fix the problem with a .Include(m => m.Team.League) in the All() method
   14. Re-profile and see savings



1. **Profile Application**
   1. Explain lack of CPU intensive JS in MiLB Mascots and browse to http://octane-benchmark.googlecode.com/svn/latest/index.html
   2. Show similarities
   3. MS or % view
   4. Show off flame chart – time across stack vertically



1. **Timeline Demo**
   1. Record All Page
   2. Show Events
   3. Show Frames
   4. Zoom In on a Few Segments



1. **Paint Demo 1**
   1. Enable “Show Paint Rectangles” & Show composited layer boarders
   2. Scroll and show red boxes + blue boxes and explain the difference
   3. Add translate(0) to promote layer
   4. Record in chrome://tracing and show off layers there, stress how beta it is



1. **Paint Demo 2**
   1. Enable “Continuous page repainting” while “Show paint rectangles” is on
   2. See the whole page is red
   3. Turn off show rectangles
   4. Select a .mascot-card and start to toggle them on and off to see time drop
   5. Dig in and disable box-shadow and border-radius