Creating Fast Websites:

The Many Facets of Web Performance

Christopher Haupt, CTO
Michael Slater, CEO
sales@webvanta.com
888.670.6793

www.webvanta.com

Agenda

- Aspects of website performance
- Structuring your front-end code for peak performance
- Writing WebvantaScript for peak performance
- Compressing and combining images, JavaScript files, and CSS files
- Understanding server-side and client-side caching behavior

Aspects of Site Performance

- Speed of internet connection
- Number and size of files being downloaded
- Response time of server to provide those files
  - App and database time, if non-cached dynamic page
- Caching in the server and the browser
- Time for the browser to render the page
  - Degree of parallelism the code allows
  - Time to render

Optimizing Front-End Code

- Relatively simple things you can do to improve page load and display
  - Well structured markup, properly positioned related content like CSS and JavaScript, right-sized images
- Some techniques are done for you by Webvanta infrastructure
  - Cache “hints”, compression over the wire
- See Steve Souders’s books in reference section
  - We’ll preview a few of his rules

Rule 1: Minimize Number of Requests

- HTTP requests are expensive, so reduce the number of items that must be loaded
  - Combine stylesheets into one
  - Combine JavaScript files into one or a small set of combinations
  - Consider using CSS image sprites to combine lots of little pictures

http://stevesouders.com/hpws/combo-none.php

Using CSS Sprites

- Loading one composite image is much faster than lots of smaller images
- Use sprite as a background image and use background-position to choose a part

```
#icon1 {
     background: url(sprite.jpg) 0 -200px no-repeat;
 }
 #icon2 {
     background: url(sprite.jpg) -96px -200px no-repeat;
 }
```

Rule 5: Put Stylesheets at the Top

- Progressive rendering may be blocked until CSS is loaded
- Stylesheets in the <head>
  - Use <link> vs @import
  - Using <link> outside of <head> violates spec (and has worse behavior)
  - Avoids Blank Screen on older IE
  - Avoids Flash of Unstyled Content (FOUC)

http://stevesouders.com/hpws/css-fouc.php

Rule 6: Put Scripts at the Bottom*

- JavaScript is guaranteed to run in the order it is specified
- JavaScript *may block* parallel downloads (in older browsers) until it is done
- JavaScript **will** block progressive rendering even if content is available
  - Best Practice: Put scripts below visible content (above closing </body> tag) when you can
  - *Sometimes, you need to put stuff at the top

http://stevesouders.com/hpws/js-middle.php

**usually*

Rule 6a: What About In-Line Scripts?

- Positioning of in-line scripts impacts performance too
  - Block execution (rendering & sometimes downloads)
  - Solution: Move to bottom (but still blocks rendering)
  - Solution: Execute asynchronously (setTimeout for short code bits, onload handler for most other things)
    - jQuery makes this relatively easy via load or ready handlers:

```
jQuery(document).ready(function(){ //your code });
```

Rule 8: Make JS and CSS External

- Technically, inline should be faster, as it requires fewer downloads
- A typical visit includes at least a few pages
  - Leverage the browser cache by making CSS and JS external
- Use a shared source, such as Google's CDN, for common libraries such as jQuery

http://stevesouders.com/hpws/rule-inline.php

WebvantaScript

- WebvantaScript is executed by the server, to create the page contents
  - Increases time to deliver HTML page, if not cached on the server
- How you structure your pages can have large impact on time to deliver an uncached page
- Common to pull in too much data
- Generally, more complex structure == slower

Loop Through As Few Items As Possible

- More data == slower rendering times

```
// NO!
<w:kb:item:each type="posts">
  <w:if condition="published_at > now-5.days">
    <p>Found <w:name /></p>
  </w:if>
</w:kb:item:each>

// YES!
<ul>
<w:kb:item:each type="posts" condition="published_at > now-5.days">
  <li>Found: <w:name /></li>
</w:kb:item:each>
</ul>

<w:kb:item:if_iterator_preflight_size type="posts"
condition="published_at > now-5.days" >
 // Your Loop plus wrapper here
</w:kb:item:if_iterator_preflight_size>
```

Flat is Faster

- Single-layer Custom Item Types are faster than those with lots of related items
- Regions are faster than Snippets
- Avoid Snippets that include Snippets
- Often you must make tradeoffs between
  - The structure that is easiest to maintain
  - The structure that performs best

Careful With Embedded JS and CSS

- If you put JavaScript or CSS in Snippets it may:
  - End up inline
  - End up ***repeating*** in an iterator
  - End up making an invalid DOM (e.g., reuse same DOM ID)

Use Ajax to Load Secondary Content

- Use when most of a page is static but part must be dynamic
  - Create static page that is cacheable (and is indexed by search engines)
  - Use document ready handler to fire off Ajax request to update dynamic parts of the page
    - Personalize for a user
    - Show information that changes frequently
    - Show content that is slow to deliver

CSS: Combine, Don't Compress

- Combine all CSS files into one (or the smallest practical number)
- Should you compress CSS?
  - Typically just removes whitespace, unneeded punctuation, adjusts color codes, etc.
  - Tricky to automate compression of rules
  - Tradeoff between maintainability and size rarely worth it... make it readable and neat
- Tools: BBEdit, Dreamweaver, Online Tools

CSS Compressed vs. Source

```
.table{clear:both;margin:
10px 0;-moz-box-shadow:0 1px
3px rgba(0,0,0,.3);-webkit-
box-shadow:0 1px 3px
rgba(0,0,0,.3);box-shadow:0
1px 3px rgba(0,0,0,.3);-moz-
border-radius:6px;-webkit-
border-radius:6px;border-
radius:6px;}
```

OR

```
.table {
        clear: both;
        margin: 10px 0;
        -moz-box-shadow: 0 1px 3px rgba(0,0,0,.3);
        -webkit-box-shadow: 0 1px 3px rgba(0,0,0,.3);
        box-shadow: 0 1px 3px rgba(0,0,0,.3);
        -moz-border-radius: 6px;
        -webkit-border-radius: 6px;
        border-radius: 6px;
}
```

Minimizing JavaScript Load Time

- Best Practice: Combine all of your common .js files into one (e.g. jQuery, jQuery.ui, plugins, ...)
- Compress (aka minimize) *after* combining
  - unlike CSS, minimized code makes a BIG difference
- Keep both original and compressed files around
  - Use original files during development or to debug
  - Use compressed file for live pages
  - Never edit compressed JavaScript
- Tools: UglifyJS

Compressed JavaScript Not Editable

```
(function(a,b){function cA(a){return f.isWindow(a)?
a:a.nodeType===9?a.defaultView||a.parentWindow:!1}
function cx(a){if(!cm[a]){var b=c.body,d=f("<"+a
+">").appendTo(b),e=d.css("display");
```

OR

```
(function( window, undefined ) {

// Use the correct document accordingly with window argument (sandbox)
var document = window.document,
        navigator = window.navigator,
        location = window.location;
var jQuery = (function() {

// Define a local copy of jQuery
var jQuery = function( selector, context ) {
                // The jQuery object is actually just the init constructor 'enhanced'
                return new jQuery.fn.init( selector, context, rootjQuery );
        },

        // Map over jQuery in case of overwrite
        _jQuery = window.jQuery,

....
```

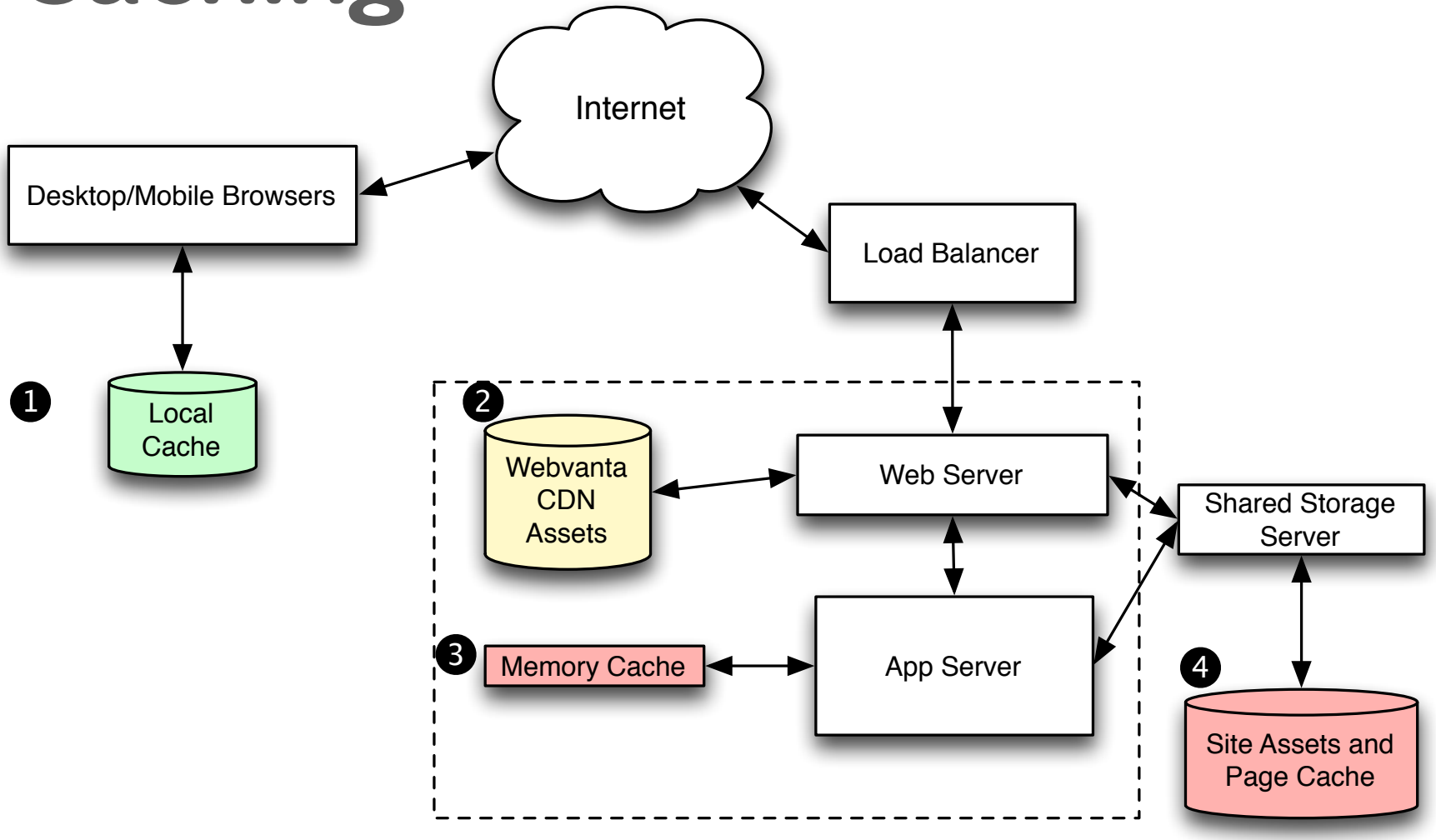
Minimizing Image File Size

- Right-size your images
  - Upload only the largest size that will be needed (not necessarily the true original)
- Use the correct rendition for the job
  - leverage the Webvanta asset rendition generator
  - specify the minimum required set of image sizes
  - WebvantaScript: <w:asset name=”logo.png” rendition=”NAME” />
  - Faster: /rendition.NAME/path/to/logo.png

Pick the Right Image Format

- Using the right format for image content can increase performance significantly
  - JPEG (many-colored, continuous-tone photos)
  - PNG (average palette, transparency)
  - GIF (small palette)
- Remove as much metadata as you can
  - Full EXIF + XMP can add 20-30K per image even on *small images*

Caching


Internet
Desktop/Mobile Browsers
Load Balancer
1
Local Cache
2
Webvanta CDN Assets
Web Server
Shared Storage Server
3
Memory Cache
App Server
4
Site Assets and Page Cache

Browser Caching

- Browsers maintain a local cache of recently accessed files (HTML, CSS, JS, images)
- Sharing resources across pages and sites dramatically reduces load time
  - e.g., JavaScript libraries (across pages and sites), images (across a site)
  - A good reason to make JS and CSS external if shared
- Always clear browser cache manually to see what the “cold load” performance is

Many Sources of Content

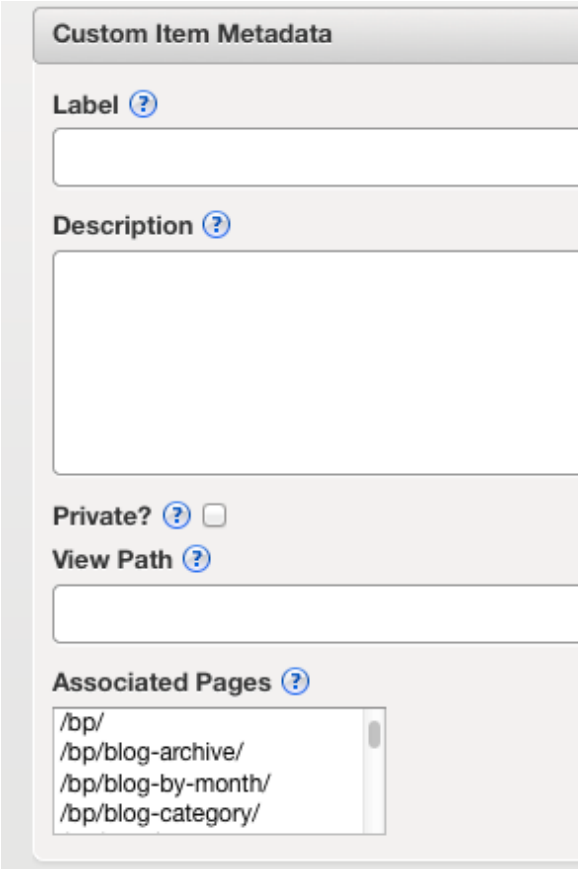
- Wide range of speed (fast to slow)
  - External Content Distribution Network (CDN)
    - e.g. jQuery and other libraries from Google
  - Webvanta CDN (e.g. our plugins)
  - Account Files (whatever you upload to Files)
  - Webvanta Memory Cache (data about your account)
  - Webvanta Page Cache (your rendered pages, including HTML, JavaScript, CSS, XML)
  - Uncached Pages or pages not yet rendered

Keeping the Caches Full

- Pages not in the server cache need to be created dynamically and will always be relatively slow
  - With proper design, vast majority of accesses are to cached content
- Webvanta Page Cache is cleared:
  - Completely, when you do so manually
  - Selectively, and automatically, when content updates
- If you don't specify which pages need to be cleared from cache when a database item changes, then ***everything*** will be cleared

Optimizing Webvanta Caching

- Set Associated Pages for every database item type
- Put common JavaScript and CSS files into Files, not under Structure
- Follow WebvantaScript performance best practice



Summary

- Some simple habits will make your sites faster
  - Put JS in the bottom of the body
  - Consolidate CSS files
  - Be careful about CSS and JS in snippets
  - Construct WebvantaScript loops carefully
  - Always set associated pages for database item types
- Some techniques are extra-cost but pay off
  - CSS sprites
  - Combining and minifying JavaScript

Useful References

- “High Performance Web-Sites” by Steve Souders
- “Even Faster Web-Sites” by Steve Souders
- http://www.webstockbox.com/css/10-free-online-tools-for-compressing-css-code/
- http://www.w3schools.com/css/css_image_sprites.asp
- https://github.com/mishoo/UglifyJS
- http://stevesouders.com/cuzillion/
- http://code.google.com/apis/libraries/