

Developing Mobile Web Applications and Mobile Widgets

Alison Lee¹ and Andreas Girgensohn²

¹ Nokia Research Center – Palo Alto

² FX Palo Alto Laboratory – Palo Alto

<http://www.webcollab.com/course/>

Tutorial Topics

- Mashup data, functionality from Web
 - Mobile Web applications
 - Mobile widgets
 - Mobile widgets using Web RunTime (WRT)
- Issues related to mobile devices, mobility
 - Use opportunities
 - Concerns – battery, UI I/O, data plans, programming etc.
 - Benefits – sensors, haptics, telephony
 - Bridging Web and phone capabilities
- Code snippets and small experiments as illustrations

Mobility

- Mobility (being mobile) is common in everyday work
 - Macro-level: local travel down the hall, different floor, different building, different cities
 - Micro-level: necessary precursor for coordination and communication
- Mobile contexts are highly variable and different
 - Connectivity, access to information, unfamiliar settings
 - Instant on devices, degree of interruption
- Mobile work is different from desktop work
 - Social appropriateness
 - Amount of time
 - Access to resources, connectivity, people
- Tradeoffs with time, effort, deprivation, distraction

Mobility: New Opportunities

- Mobile information access – at finger-tip, actionable
 - Mashups – aggregation, projection, cross product of data and resources
 - E.g., remote content access
- Mobile communication – talk is central to work
 - Coordination, awareness, people proxy
 - E.g., meetings via mobile phone
- Mobility work
 - Adopt coping strategies, perform mobilization work
 - Availability time, attention, resources
 - E.g., read email, quickie tasks

Mobile Web

- Standards-based Web browser
 - S60 Web browser on S60 smartphones
 - Opera on S60 and other devices
 - Safari (WebKit-based) on iPhones
 - WebKit browsers on Android
 - Blackberry Bold, Curve, Storm
- Use of standards-based technologies
 - DHTML, CSS, JavaScript, XHR
- While mobile, provide integration of and access to:
 - Capabilities – e.g., communication
 - Data – i.e., mashups
 - Web and phone content

Findings : Informal Interview

- Planful opportunism – preparatory or just-in-case activities
 - Paper folders, printouts with information
 - Adding to calendar with events like car-service pickup
- Calendar as information corral
 - Email snippets, directions, phone #, URLs
 - Co-opt or overload calendar with information scraps
- Inability to use calendar data in context
 - Traditionally, a scheduler, time manager, reminder
 - When mobile, information use is different – take action with it
 - Call audio conference number
- Flexible appropriation, indexicality
 - Flexible, lightweight content, representation, usage, organization
 - Indexicality – meaning that only makes sense in a particular setting
 - Function indexed by calendar data – driving directions

Mobile Web Applications & Widgets

- Built using HTML/XHTML, CSS, JavaScript, DOM etc.
- Web applications are hosted by Web browser and accessed over the Internet
 - Full-featured, extensive but limited to Web content
 - E.g., Google's GMail, eBay's online auction, Facebook, etc.
- Widgets are hosted by a widget engine and installed on device
 - Small, task-specific but can integrate Web and phone content
 - E.g., accuweather, bloomberg

Examples of Mobile Web...

- Nokia Easy Meet – mobile Web application, mobile widget
 - Real-time data and voice sharing
 - Meetings from mobile device
 - <setup devices to be different participants>
- Widget examples
 - Accuweather
 - ...

Native vs. Web

- Nature of application – e.g., mashups, image processing
- Performance
- User experience
- Device heterogeneity
 - Capability and variability
- Reach to large user population
- Developer pool and skill set required
- Programming model ??
- Deployment – download, install, update
- Maintenance – scalability of implementation to different devices

Transition to Andreas for Issues related to Mobile Web Application (be it Web widget or Web application)

Q & A

Mobile Web Applications

- Many issues shared with desktop web applications
 - Features of HTTP protocol (e.g., statelessness)
 - Use of asynchronous requests (e.g., XMLHttpRequest)
 - Cross-domain issues for mash-ups
- Response time is a shared issue
 - Slower mobile network: 100–1000 Kbps vs. several Mbps
- Specific mobile issues and features
 - Less CPU power
 - Energy consumption
 - Small screen sizes
 - Limited input options and touch screens
 - Telephony features (making calls, sending SMS)
 - Sensors (e.g., GPS, accelerometer)

Mobile Communication Protocols

- 2G: EDGE
 - Published by AT&T: 75 Kbps – 135 Kbps
- 3G: HSDPA/UMTS
 - Published by AT&T: 700 Kbps – 1.7 Mbps
 - In practice, 3G is only about twice as fast as EDGE
 - UMTS frequency bands vary by country
 - UMTS2100 most widely used
- Faster protocols (e.g., HSPA+) in preparation

HTTP Protocol

- Uses TCP socket connection to server
 - Request header includes URL to be accessed, options such as cache control, and information about the browser
 - Response header describes content
 - Long headers in both directions (~0.25 KB each)
 - Connection may be reused for additional requests
- HTTP is stateless protocol
 - State represented as cookies
 - Adds to header size
 - Large cookies and cookies for static files increase header sizes

```
HTTP/1.1 200 OK
Date: Sun, 05 Apr 2009 20:55:52 GMT
Server: Apache/2.2.11 (Fedora)
Last-Modified: Sun, 05 Apr 2009 19:30:17 GMT
ETag: "6310277-d4c-466d3cfc88440"
Accept-Ranges: bytes
Content-Length: 3404
Connection: close
Content-Type: text/plain; charset=UTF-8
```

HTTP Proxy Servers

- Proxy server forwards and caches HTTP requests
 - May be added by mobile phone service provider
 - Check the user agent
Mozilla/5.0 (SymbianOS/9.2; U; Series60/3.1 NokiaN95/31.0.017; Profile/MIDP-2.0 Configuration/CLDC-1.1) AppleWebKit/413 (KHTML, like Gecko) Safari/413 **UP.Link/6.3.1.20.0**
 - Can compress content
 - Can re-author content
 - Remove links to advertisements
 - Can reduce image quality
 - Requires care when managing the cache
- No changes for HTTPS
 - Compression has to be done by the server

Improving Perceived Load Times

- Make effective use of browser and proxy caches
- Use compression if the browser supports it
- Keep images small
- Prefer static HTML over DOM manipulations
- Move scripts to the end of the page
 - CSS can go there, too, but causes page refresh
- Create small static pages that redirect to the real content
 - Set `window.location` in JavaScript
 - Use `<meta http-equiv="Refresh" ...>`
 - Reuse CSS and images cached on start pages

Managing Content Caches

- Content may be cached by the browser or proxy server
 - Avoids need for retrieving the content again
 - Can cause problems with changing content
- Strategies for not caching dynamic content
 - Header: `Cache-Control: no-cache`
 - Other headers should not be needed with well-behaved browsers
 - `Pragma: no-cache; Expires: 0; Cache-Control: max-age=0`
 - Use unique URL for every request by appending token in query string

Caching Content

- Keep static files in the cache
 - Mobile browser
 - Proxy server of service provider
- Cache is validated before cached copy is used
 - Makes use of "Last-modified" or "Etag" header
 - Default "Etag" header not suitable for load-balancing
 - Includes inode number
 - Custom tag could be a content hash
- HTTP headers can request revalidation through proxy server chain

Avoiding Frequent Cache Validation

- Cache validation requires HTTP request
 - No body is returned if cache is valid (status 304)
- Expiration dates in the future avoid validation
Expires: Tue, 06 Apr 2010 16:00:00 GMT
Cache-Control: max-age=31536000
 - Apache directive: ExpiresDefault "access plus 1 year"
- Problematic if files change before expiration
 - Rename file and refer to new name in HTML pages
 - Include SVN version in query string
 - All files are reloaded unless version is looked up for each file
 - "favicon.ico" needs shorter expiration (no renaming)

Caching Dynamic Content

- Dynamic content is generated from a database
 - Database can keep track of the last time the content was changed
 - Example: user address book
- Dynamic content can be of different types
 - HTML, JSON, XML, images
- Can be handled with "If-modified-since" header
 - Still requires HTTP request
- Alternatively, the main HTML page can include the data modification time as query parameter
 - Allows for expiration of the content in the far future

Compression During Transport

- Compression provides large benefits for some files
 - Text compresses well (HTML, JavaScript, CSS)
 - Images and PDF are already compressed
- Compression indicated by HTTP header
Content-Encoding: gzip
 - Add HTTP header to compressed script output
 - Use compressed size as content length

Benefit of Compression

- Download benchmark
 - 71.7 KB text file downloaded 10 times (no cache)
 - Compressed with gzip to 14.8 KB

Nokia N95 connected via EDGE		
	uncompressed	compressed
No proxy	73.9s	25.4s
AT&T proxy	40.1s	25.1s

- Uncompressed transfer rate: 78 Kbps
 - Some speed-up with compression provided by proxy
 - 2.9 times faster with compressing
 - But: compression factor is 4.8
 - Connection overhead accounts for the rest

Same Benchmark with iPod Touch

- iPod Touch is connected via Wi-Fi
 - Limiting factor is 512 Kbps connection of the web server

	uncompressed	compressed
iPod Touch	14.3s	3.6s
Desktop PC	13.8s	3.2s

- Uncompressed transfer rate: 401 - 415 Kbps
- iPod Touch has almost the same speed-up for compressed data as the desktop PC
 - Factor 4.0 vs. 4.3 (iPod vs. PC)
 - Decompression adds little overhead even for slow CPUs
- Compression offers benefits even at 3G speeds

Setting up Compression

- Store files compressed (e.g., test.js.gz)
Apache directive: AddEncoding x-gzip .gz
 - Conflicts with gzip content type — use test.gz.js or remove application/x-gzip from mime.types
- Compress files on-the-fly:
AddOutputFilterByType DEFLATE text/html text/css
 - Can be controlled by browser type, etc.
 - Somewhat more load for the server
- Compress script output (e.g., python)

```
import zlib  
body = '{"month": "April", "city": "Madrid"}'  
compressed = zlib.compress(body)
```

Reduce Image Sizes

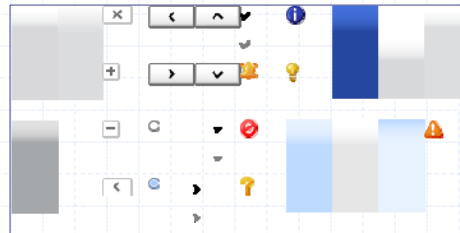
- Compare sizes of JPEG, PNG, GIF
 - GIF only offers 256 colors
- Reduce unnecessary color-depth in PNG and GIF
 - PNG alpha channel available in 32-bit image
 - Optional feature: alpha entries for color palette
 - Photoshop may not create smallest files
 - Dithering tends to increase the file size
- Determine appropriate JPEG quality

CSS Sprites

- Use CSS sprites to combine multiple images
 - Use offsets to access the appropriate tile
 - Avoids multiple HTTP requests
 - May reduce overall file size
- Tiles of larger image used for different backgrounds

```
{ background: transparent url(bg.jpg) 0 -80px no-repeat; }
{ background: transparent url(bg.jpg) -96px -80px no-repeat; }
```
- Potential problem if HTML tag exceeds tile dimensions

Yahoo! UI
Sprites



Combining Files to Reduce Requests

- Fewer, larger files reduce connection overhead
 - Fewer HTTP headers
 - Proxy server does not seem to reuse socket connections
 - Less benefit in combining large files
- Combined files should not include useless data
 - Different pages may need different scripts and CSS
 - User may never visit those other pages
 - Don't overwhelm welcome page with large files only needed elsewhere
- Combined files impact cache validity
 - Don't combine library with frequently changing code

Combining and Shrinking Files

- Multiple JavaScript and CSS files
 - Can each be concatenated into a single file
- JavaScript, JSON, XML, and CSS usually contain unnecessary whitespace
- Text files compress well
 - Content-encoding: gzip
 - Service provider may do that in proxy server
 - HTTP server can serve compressed version
- Combining, shrinking, and compressing files should be part of a build script

Shrinking JavaScript

- Whitespace and comments can be removed
 - Whitespace in strings, etc., is meaningful
 - Some comments should stay (e.g., copyright)
 - Same reduction applicable to CSS
- Variables can be renamed (obfuscation)
 - Can change public APIs
 - Problems with the use of eval, setTimeout, etc.
- Can reduce size of compressed version by 35%
 - Compression provides the most benefit (typical factor 3 to 7 even without prior shrinking)

Automatically Shrinking JavaScript

- **jsmin**
 - Only removes whitespace
 - Regular expressions may get confused by some JavaScript constructs
- **Obfuscation**
 - ShrinkSafe: uses Rhino as parser; renames variables
 - YUI Compressor: better variable renaming; handles eval
- **Transcoding JavaScript**
 - Packer: used eval to decode program on the client side
 - Decoding takes time
 - Now appears similar to YUI Compressor

Source (14.2K; gz 3.5K)	ShrinkSafe (8.2K; gz 2.7K)
<pre>function initImageSizes (min_size, max_size, step) { image_sizes = []; var size = min_size; var start_size = size; while (size <= max_size) { image_sizes.push (size); size += step; if (size == 2 * start_size) { step *= 2; start_size = size; } } }</pre>	<pre>function initImageSizes(_1,_2,_3){ image_sizes=[]; var _4=_1; var _5=_4; while(_4<=_2){ image_sizes.push(_4); _4+=_3; if(_4==2*_5){ _3*=2; _5=_4; } } };</pre>
jsmin (9.8K; gz 2.6K)	YUI Compressor (6.9K; gz 2.3K)
<pre>function initImageSizes(min_size,max_size,step){ image_sizes=[];var size=min_size; var start_size=size;while(size<=max_size){ image_sizes.push(size);size+=step; if(size==2*start_size){ step*=2;start_size=size;}}</pre>	<pre>function initImageSizes(d,e,c){ image_sizes=[];var b=d;var a=b;while(b<=e){ image_sizes.push(b);b+=c;if(b==2*a){ c*=2;a=b;}}</pre>

Problems with Obfuscation

- Can break public API if applied to public variables
- Problems with eval

```
function testEval (x) { var longname = "test";
eval("alert(longname)"); }
```

```
function testEval(x){ var _2="test";
eval("alert(longname)"); };
```

 - Handled by YUI Compressor
- Difficult to debug program
- Does not provide real protection of intellectual property
 - Easy to copy program code

Using AJAX

- Asynchronous JavaScript and XML
 - Used to make calls from JavaScript that can change parts of the page
- Simple approach (no XML): AJAST / JSONP
 - Add a JavaScript script tag to the DOM tree pointing to a script on the server that returns a JSON callback
 - Problematic in some browser when done repeatedly
 - Callback function is called with JSON parameter
 - Supports cross-domain access (under discussion)
 - No error handling

XMLHttpRequest

- Available in most browsers
 - IE 6 and earlier use ActiveX component
- Can only access the same domain
 - Cross-domain access for XMLHttpRequest may come
- Easy to use

```
var req = new XMLHttpRequest();
req.open("GET", url);
req.onreadystatechange = function () {
  if (req.readyState == 4 && req.status == 200)
    processData(req);
};
req.send(null); // null is the POST body
```
- Don't count on variable "this" to point to the request in the handler

XML vs. JSON with AJAX

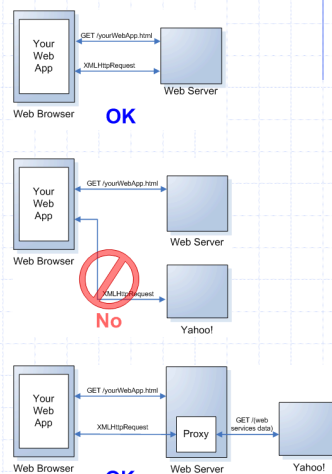
- No clear size advantages
 - XML rumored to be larger
 - Actually slightly smaller for Flickr API
 - Both compress about 1 to 7 for Flickr API
- No hard data on parse speed
- JSON JavaScript object may be easier to process than XML DOM tree
- No clear overall advantage but using compression is important
 - E.g., use zlib library as part of the server script
Content-Encoding: gzip

Cross-Domain Issues

- IFrames from different domains cannot access each others internal state
- XMLHttpRequest only works within the same domain
- Domain of an IFrame is determined by the HTML source, not the JavaScript source of DOM calls
 - Example: HTML from domain A includes JavaScript from domain B
 - JavaScript from domain B can manipulate HTML from domain A
 - JavaScript creates IFrame and adds content
 - IFrame is considered to be from domain A
 - Used by AJAST / JSONP approaches

Cross-Domain XMLHttpRequest

- XMLHttpRequest cannot access another domain
 - Port or protocol (http/https) is seen as difference
- Solution: set up a proxy server in your web server that forwards XMLHttpRequest
 - Don't make it too open to avoid abuse
- Yahoo library uses two proxies for cross-frame communication
- Firefox supports trusted scripts that can access other domains



from Yahoo! Developer Network

Hack for IFrame Communication

- IFrames can set each others location
 - However, can't read other locations
- Changing just the fragment ID (part after #) doesn't cause reload
- IFrames can periodically poll for their own location
 - Detect change of fragment ID
 - Change own state accordingly

Checking for Capabilities

- Checking the user agent is usually a bad idea
- Better to see if objects have certain properties
if (typeof(window.innerWidth) == "number")
- Use "if" statements to check for properties
 - false, undefined, null, 0, NaN, "" are false, everything else is true
 - A missing properties is undefined
 - Check with (typeof x.y) == "undefined" to be sure
 - (typeof null) returns "object"
- Use the "delete" operator to remove a property
x.y = null; (typeof x.y) => "object"
delete x.y; (typeof x.y) => "undefined"

Sample Capability Check

- Save to use simple boolean tests if properties can never have the value 0
 - E.g., document.body or document.body.clientWidth

```
if (typeof(window.innerWidth) == 'number') {
  win_width = window.innerWidth;
  win_height = window.innerHeight;
} else if (document.documentElement &&
  (document.documentElement.clientWidth ||
  document.documentElement.clientHeight)) {
  win_width = document.documentElement.clientWidth;
  win_height = document.documentElement.clientHeight;
} else if (document.body &&
  (document.body.clientWidth ||
  document.body.clientHeight)) {
  win_width = document.body.clientWidth;
  win_height = document.body.clientHeight;
}
```

Debugging Mobile Web Pages

- Avoid endless "alert" loops (press "cancel")
window.alert = function(s) {
 if (!confirm(s)) window.alert = function () {}; }
}
- Some mobile browsers have a console (most don't)
 - JavaScript fails silently without a console
- Use try-catch blocks in all JavaScript functions called from the outside (HTML, timeout, callback)
 - Major differences between browsers ("line", "lineNumber", "sourceURL", "fileName", or very little)
 - "onerror" event handler only available in IE and Firefox
- Test with Safari and hope for the best
 - WebKit is basis for Nokia and iPhone browsers
 - iPhone version is more modern
 - Firefox / Firebug for initial debugging

Console for Mobile Browsers

- Firebug for iPhone
 - Attempt to run a console on a PC in the same network as the mobile device
 - Uses a small python web server to facilitate communication between phone and PC
 - Doesn't work anymore due to cross-domain issue
- Console portion uses image requests
 - Can be made work without cross-domain issues
- Sending commands from PC to phone is difficult
 - XMLHttpRequest has cross-domain issue
 - Pending JavaScript requests can cause blocking
 - Image redirects are not reflected in image "src"
 - Console service can be integrated in web server

Implement Console in JavaScript

- Firebug lite
 - iPod Touch: displays; console works; buttons inoperable; issues with zooming
 - Nokia N95: doesn't display
- Even if it worked, console would use up small screen
 - Not enough room for Firebug lite controls
 - Even a console window may not show much

Energy Consumption

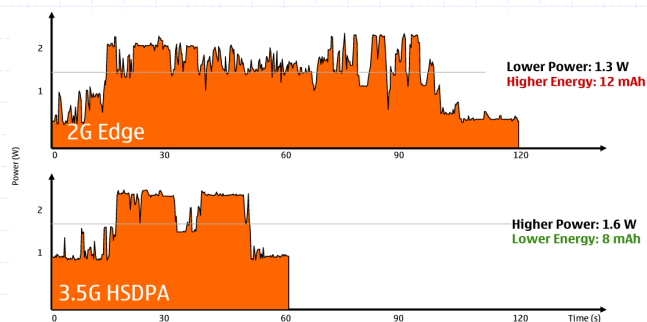
- Battery stores energy
 - Power is energy per time unit
- Energy consumption is a major issue
 - Users don't want to recharge their phones every day
- Power levels for Nokia N95 with EDGE

Screen dimmed	0.1W
Idle	0.25W
Scrolling	0.45W
Active connections	1 – 1.5W

- Important to consider energy consumption when making design decisions

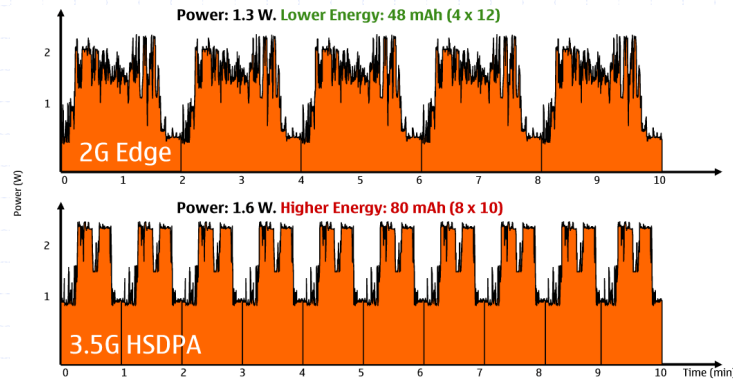
Monitoring Power Levels

- Nokia Energy Profiler (free download)
 - Charts power, CPU utilization, etc.
- Faster communication protocols need more power when accessing one web page but finish faster



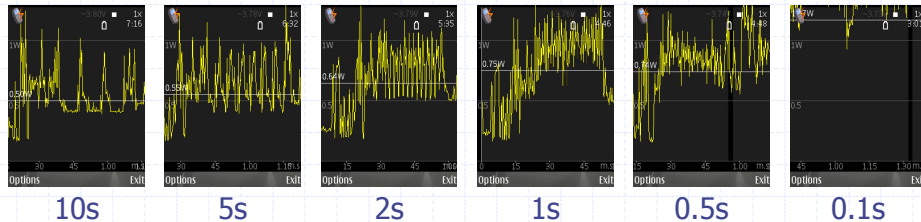
Browsing for 10 Minutes

- HSDPA loads twice as many pages and uses more energy



Periodic Polling and Power

- Periodically download small image in background
 - Change URL to get fresh copy
- Power consumption increases with frequency
 - Battery time between 7:16 and 3:01 (42%)
- Upper bound determined by CPU load and network
- Slightly higher consumption with XMLHttpRequest



Polling Strategies to Save Energy

- Keep phone on standby for long periods
 - Reduce polling as much as possible
 - Make sure that polling intervals don't overlap due to slow responses
 - Use compression to reduce response times
- Multiple parallel connections use extra power
 - Potentially less energy than serializing requests
- Benefits in batching up requests
 - 10 requests in 5 seconds followed by 5 seconds break better than one request per second
 - Combine multiple requests into a single request
 - Make server application work in bursts

Mobile Input and Output Constraints

- Typical screen aspect ratios
 - 1:1 (208x208), 4:3 (320x240 QVGA),
3:2 (480x320 iPhone), 16:9 (640x360 N97),
5:3 (800x480 N800), 1.76 (480x272 PSP)
- Phone may be locked into either landscape or portrait
- Phone may sense rotation and flip orientation
 - JavaScript event handler "onresize" can detect that
- Web pages have to work for different aspect ratios, orientations, and resolutions
 - Specify dimensions as percentages
 - Maybe use different versions for portrait / landscape

Determining Browser Window Size

- Browser window size not compatible across browsers
 - Most support window.innerWidth but IE doesn't
- Event handler can be used to detect changes to the window size
 - window.onresize = handleResize;
- Zooming can change window size
 - No event handler for that

```
if (typeof(window.innerWidth) == 'number') {  
  win_width = window.innerWidth;  
  win_height = window.innerHeight;  
} else if (document.documentElement &&  
  (document.documentElement.clientWidth ||  
  document.documentElement.clientHeight)) {  
  win_width = document.documentElement.clientWidth;  
  win_height = document.documentElement.clientHeight;  
} else if (document.body &&  
  (document.body.clientWidth ||  
  document.body.clientHeight)) {  
  win_width = document.body.clientWidth;  
  win_height = document.body.clientHeight;  
}
```

Polling for Window Size Changes

- Resize event works well when rotating phone
 - Doesn't work for zooming
- Poll for changed sizes periodically with setInterval
 - Checks the window size and only does something if size changed
 - No noticeable impact on energy consumption when polling four times per second
- Detects most zooms
 - Some iPod Touch results are not plausible

iPhone Screen Size

- iPhone screen: 320x480
- iPhone reports window width as 980px
 - Zoom factor 0.3265 for device width 320px
- Can be controlled with "viewport" meta tag

```
<meta name="viewport" content="initial-scale = 1.0">
<meta name="viewport" content="width = device-width">
```
- Zooming-out somewhat unpredictable when using this meta tag
 - Value provided by `window.innerWidth` not always plausible

Font Sizes for Different Devices

- Setting font sizes in pixels may not work well for different device resolutions
 - Percentage: relative to font size of parent element
 - 100% sometimes causes glitches, use 101%
 - Unit "em": relative to size of "M" in parent element
- Dimensions of HTML elements can be set relative to the font size
 - Unit "em": width of "M"
 - Unit "ex": height of "x" (less useful; often just 0.5em)

Key and Mouse Input

- Input – numeric keypad, full keypad, touch
 - Softkeys, cursor, numeric keys for purpose, ...
 - Cursor navigation, tab navigation, customized navigation
- Touch screens do not sense hovering
 - CSS class “:hover” not triggered
 - No tool tips for links
 - Link tool tips often not available on mobile devices with cursor control, either (N95, Android)

Capturing Key and Mouse Events

- Key events can be captured with event listeners
 - `document.onkeypress = handleKeyPress;`
 - Event object provides key code and character code
 - JavaScript key event codes vary across browser engines
 - Key events less useful for touch screens
 - Some keys can't be captured
 - Cursor keys, keys for zooming and searching
 - Nokia widget can capture cursor keys
 - `widget.setNavigationEnabled(false);`
 - Turns off mouse cursor control
- “mouseover” event not available for touch screens

Telephony Features

- Phone web browsers support two protocol handlers similar to "mailto:"
 - "tel:" is for making phone calls
 - "sms:" is for sending SMS messages

```
<a href="tel:6508424844">Call me</a>
```

```
<a href="sms:6507965392?body=SMS%20test">Send SMS</a>
```
- Clicking on either link brings up a dialog to complete the call or the SMS

Guidelines

- Reduce the size of downloaded data
 - Cache control
 - Combined files to reduce requests and HTTP headers
 - Shrunk and compressed data
 - Reduced color-depth or less quality for images
- Be aware of cross-domain issues for mash-ups
- Use effective means for debugging mobile applications
- Reduce energy consumption
 - Communicate in bursts
- Handle a variety of small screen sizes
- Address limited input options for mobile devices

Transition to Mobile Widget Development

Q & A

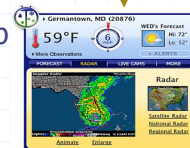
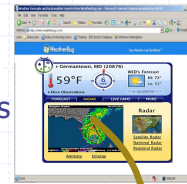
Mobile Mashups

- Aggregation, projection, cross product
 - Data and resources
 - XML or JSON Web service APIs
- Web services
 - Web service APIs for many services
 - www.programmableweb.com
- Traditional mashups – Internet content, functionality
- Mobile mashups – Internet mashups, phone mashups, or both

Widgets

- Lightweight, mini-applications
 - Accessory, information, application variants
- Runtime environment hosts widget
 - Built on top of browser engines with browser chrome or
 - Use browser components (e.g., interpreters)
- Variety – platforms and devices
 - Server-side widgets – hosted on servers, embedded in 3rd party Web pages e.g., Google gadgets,
 - Device widgets, including mobile and desktop (Dashboard, WidSets, S60 Widgets)
 - Implemented with Web technologies, Java
- Incompatibilities amongst vendors
- Effort in W3C to standardize widgets

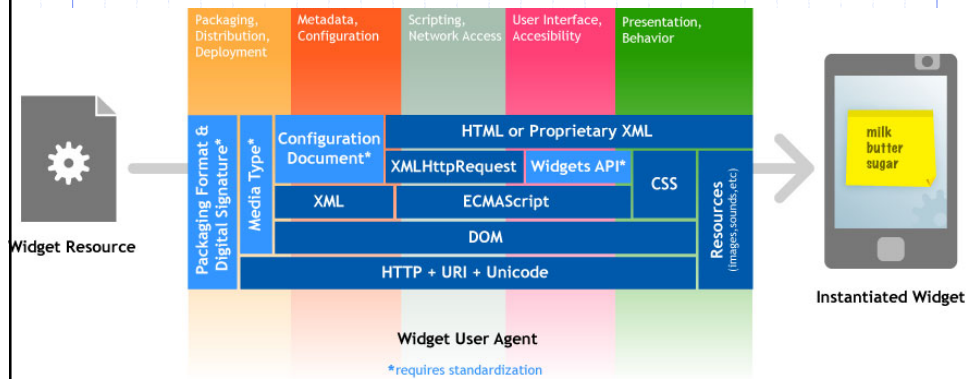
Web page without a browser UI



Widget Elements

- Runtime – hosts instantiated widget
- Image/icon – non-running state
- Resource – contents and media type
- Media type – links resource to runtime
- Packaging format of data – flat file or zip
- Resources – images, HTML, CSS, JS, sounds, etc.
- Configuration document – meta data and parameters for widget (typically in XML)
- Start file – resource when instantiated is the widget
- Bootstrap – finds start file for instantiated widget
- APIs – platform functionality

Widget Engine



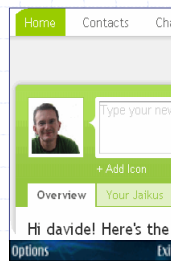
Mobile Web Widgets

- Standalone Web application
 - Looks, feels, acts like native application
 - Local on the handset & distributed like any other application
- Web page designed with specific purpose
 - Same technologies – HTML, CSS, XML, JavaScript, XHR
 - Developed in days – not months or years
 - Development to deployment in “clicks”
- Several vendors
 - Nokia S60 Web Run Time Widgets
 - Opera Widgets
 - Yahoo! Mobile Widgets (BluePrint language – XML, Xforms)

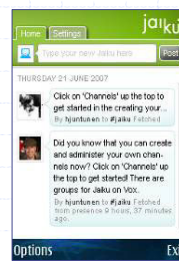
Web Application vs. Widget

- One click access to your favourite services
 - Less scrolling and typing
- Optimized web experience

Browser view on N95



Widget view



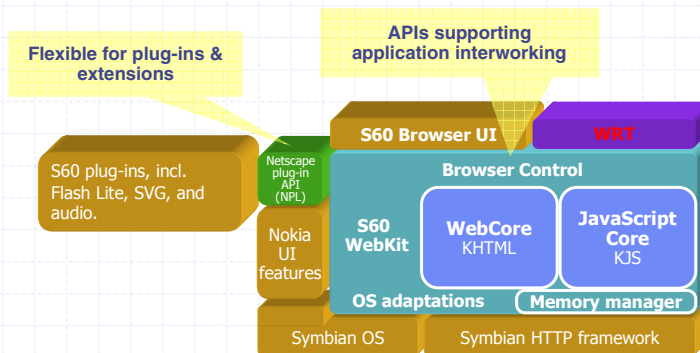
Web Application vs. Widget

- | | |
|---|--|
| <ul style="list-style-type: none">■ Runs under browser UI■ One browser instance■ More data over network■ Not for offline work■ No distribution channel■ Continually deliver as updated service | <ul style="list-style-type: none">■ Runs independent of browser UI■ Light-weight in use of resource■ Only data from server is downloaded■ Design for offline work■ Distributed via multiple channels■ Downloadable, installable package |
|---|--|

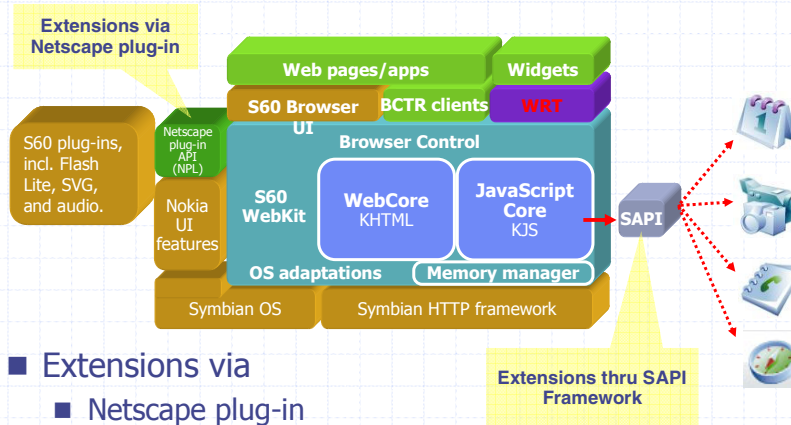
S60 Web Run Time (WRT)

- Enable Web applications to run independently from the S60 Web browser
- Supports standard-based Web technologies
 - HTML 4.01, XHTML 1.0 (basic, mobile profile)
 - CSS2.1
 - JavaScript 1.5 (standard ECMA-262)
 - XMLHttpRequest (XHR in AJAX)
 - DOM Level 2
- Includes built-in JavaScript extension libraries for widget's features (widget, menu, system info)
- Integrated into S60 3rd Edition FP 1+

S60 Web Browser



Out of S60 Web Browser Sandbox



- Extensions via
 - Netscape plug-in
 - SAPI – Service APIs

S60 Web Run Time Security

- Sandbox security model:
 - Need not be signed
 - All platform access is untrusted – requires user to grant permission
 - Widgets access network through S60 Web browser
 - Widgets access S60 platform services through scriptable plugins or JS service APIs

Runtime Security Manager

- Mobile device manufacturer determines access policy and not customizable by widget developer/user
- Default policy
 - A set of capabilities that are allowed automatically or granted by user via prompt
 - Duration (WRT 1.1) – onetime or while widget is launched (session)
- Security lifecycle
 - WRT1.1 files an access policy with security manager
 - Security manager registers a widget when widget installed
 - WRT1.1 starts a session with the security manager each time that it attempts to access a new platform service (by creating a new service object)
 - During a session, the security manger performs runtime access control to platform services (prompts user) according to policy
 - Security manager un-registers a widget when it is uninstalled.

S60 WRT Widgets

- Found in applications menu or folder with installed applications
- Full screen view (one and in foreground) and home screen (several, WRT 1.1) view
- When running, visible in Open Applications (hold down menu key)
- Look and feel like other applications
- Shortcuts to widgets in My Today Applications (active idle) or set key in active idle softkey

S60 Web Widget Components

- A widget consists of a bundle of files
 - info.plist (required)
 - [name].html (required)
 - icon.png (optional)
 - [name].css (optional)
 - [name].js (optional)
 - Resources (optional)
- Project is a file-system directory with component files
- Required widget components including optional icon.png MUST be located in the root directory
 - JavaScript, other images, and CSS can be at root or in subdirectories

Inside a S60 WRT Widget

- Create a widget as with Web page
- Existing tools can be used to create widgets

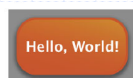
Technology	Purpose	File extension	Example
HTML	Structure	.html	HelloWorld.html
CSS	Design	.css	HelloWorld.css
JavaScript	Logic	.js	HelloWorld.js

```
<html>
<head>
<style>
  @import "HelloWorld.css";
</style>
</head>

<body>

  
  <span class="helloText">Hello, World!</span

</body>
</html>
```



Info.plist – Property of a Widget

- A manifest file in XML format, containing the property and configuration information of a widget

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Nokia//DTD PLIST 1.0//EN"
"http://www.nokia.com/NOKIA_COM_1/DTDs/plist-1.0.dtd">
<plist version="1.0">
  <dict>
    <key>DisplayName</key>
    <string>WidgetName</string>
    <key>Identifier</key>
    <string>com.company.widget.project</string>
    <key>MainHTML</key>
    <string>Main.html</string>
  </dict>
</plist>
```

WRT Supported Property

Name	Type	Status	Description
DisplayName	String	Required	Widget name listed in application
Identifier	String	Required	Unique string identifier – reverse domain format (e.g.,com.nokia.app)
MainHTML	String	Required	Main HTML page for widget
AllowNetworkAccess	Boolean	Optional	Whether network access is allowed
ShortVersionString	String	Optional	Release version of widget bundle
Version	Number	Optional	Build version of widget bundle

Icon.png

- Widget icon
- Image in Portable Network Graphics (.png) format
- Recommended size – 88x88 pixels
- If omitted from the widget installation package, a default S60 application icon is used

S60 Widget Package

- File format – compressed with zip
- File extension – widgetname.wgz
- MIME type –application/x-nokia-widget

Symbian App Code

Widget UI



Contents: HTML, JavaScript,
CSS, images, etc.

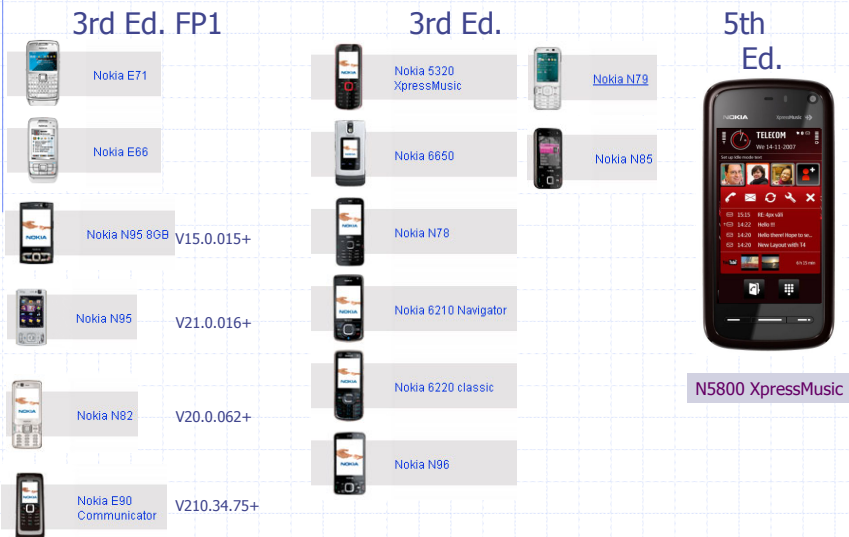
Symbian DLL

Widget-enabled browser control

Web Runtime – Versions, Devices

- WRT 1.0
 - First release of the S60 WRT
 - Support for widgets through built-in JS objects
 - S60 3rd edition FP2↑ devices, back-ported to S60 3rd edition, FP1 devices
- WRT 1.1
 - Support for S60 Platform Services through JavaScript Service APIs
 - Support for home screen widgets on S60 5th edition devices, such as the Nokia N97
 - S60 5th edition devices, back-ported to S60 3rd edition, FP2 devices

S60 Devices Supporting WRT



Web Run-Time V1.0

- Widgets leverage the Web
 - use internet services and Web APIs to access information
 - use XMLHttpRequest and AJAX techniques
 - access basic widget's specific features
 - secured inside the browser "sandbox"
- Widgets integrated into S60 user experience
 - managed the same as existing S60 applications
 - download and install like applications
 - can be assigned to Idle screen soft keys and shortcuts
 - appear in the Active Applications list
 - are registered as installed software in the Application Manager
 - display the icon in the Application Menu
 - are uninstalled from Application Menu and Application Manager
- Integrated (S60 3rd edition FP2), back ported (S60 3rd edition FP1)

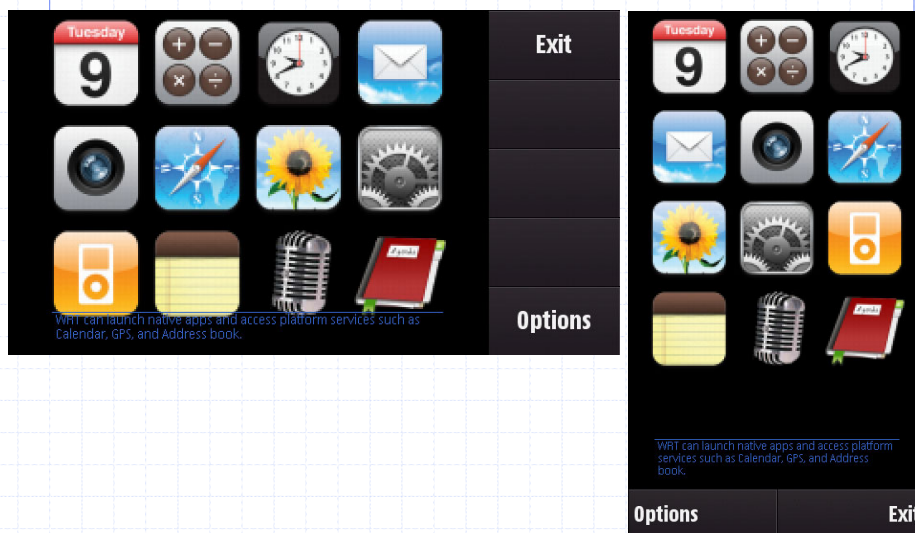
Web Run Time 1.0 Capability

- Widget Utility
 - Preferences, App Interactive, Orientation, Navigation Mode, Visibility Events
- Menu & MenuItem
 - Options Menu - Cascade Menu - Right Softkey
- System information (Scriptable Netscape plugin)
 - Power Strength, Network Strength, Memory Info, Storage Info, Language
 - Lights Control - Vibra Control - Beep Tones

Application Launcher Widget

- Used to demonstrate several APIs
 - Window size,
 - Menu, MenuItem, Softkeys, TabNavigation
 - Launch application
 - Battery level
 - Preferences to store history or frequency

Application Launcher Widget



Window Size, Icon Size

- Layout based on browser size

```
size = window_size(); // Array with width, height
```

- Icon size – 48x48 or 96x96 for regular, large sizes

```
if (width <= 320 && win_height <= 320)  
    setImageSize(true);
```

```
else  
    setImageSize(false);
```

```
// Set CSS class on img tag as regular or large
```

```
function setImageSize(regular) {
```

```
    var els = document.getElementsByTagName("img");
```

```
    for (var i = 0, len=elements.length; i < len; i++) {
```

```
        els[i].className = (regular) ? "regular":"large";
```

```
    }
```

```
}
```

```
img.regular {  
    width: 48px;  
    height: 48px;  
    padding: 0px;  
    margin: 0px;  
}
```

```
img.large {  
    width: 96px;  
    height: 96px;  
    padding: 0px;  
    margin: 0px;  
}
```

Layout – Compute & Position (X,Y)

```
var px = (typeof content_elements[0].style.left == "string") ? "px" : 0;
```

```
var x = margin, y = margin;
```

```
var max_width = 0, max_height = 0;
```

```
for (var i=0, len=ncontent_elements; i < len; i++) {
```

```
    var el = content_elements[i];
```

```
    var w = el.offsetWidth, h = el.offsetHeight;
```

```
    if (x > margin && x + w > width - margin) {
```

```
        x = margin; y = max_height + gap;
```

```
    }
```

```
    if (x + w > max_width) max_width = x + w;
```

```
    if (y + h > max_height) max_height = y + h;
```

```
    el.style.position = 'absolute';
```

```
    el.style.left = x + px;    el.style.top = y + px;
```

```
    x += w + gap;
```

```
}
```

```
container.style.width = (max_width + margin) + px;
```

```
container.style.height = (max_height + margin) + px;
```

Menu, MenuItem, Softkeys, TabNavigation

```
var CMD_CREDIT = 0, CMD_HIDE = 1, CMD_SHOW = 2;
var myMenu = {
  titles: [ 'Credits', 'Hide Notes', 'Show Notes' ],
  ids: [ 2001, 2002, 2003 ],
  items: [ null, null, null ]
};
// Create Menu items
for (var m=0, len<myMenu.titles.length; m<len; m++){
  // Adding items to menu
  myMenu.items[m] = new MenuItem(myMenu.titles[m], myMenu.ids[m]);

  // Setting handlers for main menu items
  myMenu.items[m].onSelect = menuEventHandler;

  // Append first 2 items to Menu
  if (m<len-1) window.menu.append(myMenu.items[m]);
}

window.menu.showSoftkeys();
window.widget.setNavigationEnabled(false);
```

MenuEventHandler

```
// Callback handler for menu_items
function menuEventHandler(id){
  var menuItem;
  switch (id) {
    case myMenu.ids[CMD_HIDE]:
      document.getElementById('note').style.display = "none";
      window.menu.remove(myMenu.items[1]);
      window.menu.append(myMenu.items[2]);
      break;
    case myMenu.ids[CMD_SHOW]:
      document.getElementById('note').style.display = "block";
      window.menu.remove(myMenu.items[2]);
      window.menu.append(myMenu.items[1]);
      break;
    case myMenu.ids[CMD_CREDITS]:
      document.getElementById('credit').style.display = "block";
      break;
  }
};
```

Preferences – State between Startup

```
var apps =[
  {idx: 0, img: "cal-48x48.png", appId:'0x10005901', count: 0 },
  ...
];

function savePrefs(){
  var str = "";
  for (var a=0, len=apps.length; a<len; a++){
    str += apps[a].count + ";";
  }
  window.widget.setPreferenceForKey(str, "frequency");
};

function fromPrefs(){
  var freq = window.widget.preferenceForKey("frequency");
  if (!freq) freq = savePrefs();
  var counts = freq.split(';');
  for (var a=0, len=counts.length; a<len; a++){
    apps[a].count = counts[a];
    order[a] = { idx: a, count: counts[a]};
  }
  order.sort( sortCount );
};

function sortCount(a, b) {
  return (a.count < b.count? 1 :
    ((a.count > b.count)? -1 :
    (a.idx<=b.idx)?-1:1));
};
```

Launch Application Given UID

```
var apps =[
  {idx: 0, img: "cal-48x48.png", appId:'0x10005901', count: 0 },
  ...
];

function launchApp3rdEd(uid){
  // UID must be an integer
  uid = parseInt(uid);
  window.widget.openApplication(uid, "");
}
```

Battery

```
<body onload = "init()">
  <embed type="application/x-systeminfo-widget" hidden="yes" />
</body>

var sysInfo;

function init() {
  // Obtain the SystemInfo object
  try {
    sysInfo = document.embeds[0];
  } catch (ex) {
    alert("No SystemInfo object");
    return;
  }
}

function checkSize() {
  var size = browser_size();
  ... // do layout

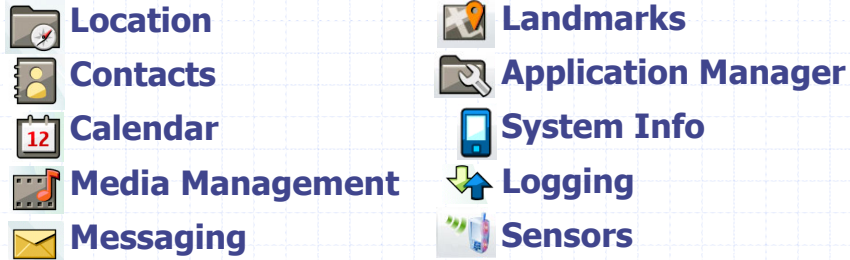
  if (sysInfo == undefined) return;
  recheck( sysInfo.chargeLevel);
}

function recheck(batteryLevel) {
  if (batteryLevel > 80)
    setTimeout( "checkSize()", 250);
  else if (batteryLevel > 40)
    setTimeout("checkSize()", 1);
  // ow, stop checking size
}
```

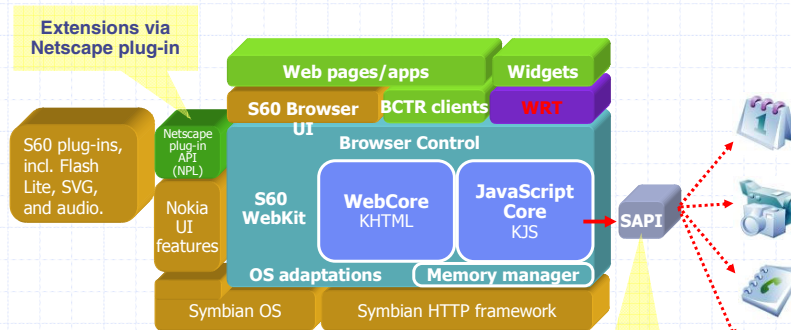
Web Run Time V1.1

- Widgets leverage the smart phone platform
 - combine information from Web with platform services
- jQuery compatible (partially)
- Extension and Fixes
 - Version information
 - Key events (virtual keyboard mode)
- Integrated (S60 5th edition), back ported (S60 3rd edition FP2)

Web Run Time 1.1 Capability



Out of S60 Web Browser Sandbox



- Language bindings for SAPI
 - JavaScript is the first available
 - Flash
- Permit access to phone data and functionality

Using WRT 1.1 APIs

- Create a service object

```
serviceObj = window.device.getServiceObject("Service.Location", "ILocation");
```

- Choose the desired method(s)

```
ILocation.GetLocation(...);
```

- Optionally define filter criteria for results

```
var criteria = {LocationInformationClass: "BasicLocationInformation"};
```

- Define functions to process the results

```
function showLocation(retVaue) {  
    alert("Lat. " + retValue.Latitude + ", Long. " + retValue.Longitude );  
}
```

- Retrieve the results

```
var result = serviceObj.ILocation.GetLocation(criteria);
```

- Process the results

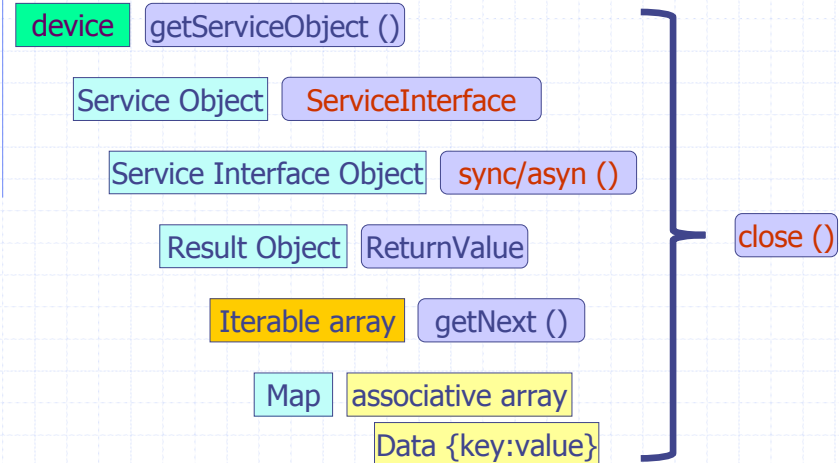
```
showLocation(result.returnValue);
```

Device Object, Service Object

- Supported since WRT 1.1
- An extension of window object
device or window.device
- An entry point to access platform services
- A service object is returned by the device object's method

```
so = window.device.getServiceObject(  
    Str:service, Str:interface  
)
```

Service Object Structure



Services Arguments

- JavaScript object as argument of service methods
- Argument can be a nested object
- Object construction

- Via new + dot notation syntax

```
var criteria = new Object();
criteria.Type = 'FileInfo';
```

```
// nested object
var filter = new Object();
filter.FileType = 'Sound'
var criteria = new Object();
criteria.Filter = filter;
```

- Via object literal syntax

```
var criteria = { 'Type': 'FileInfo', 'Filter': { 'FileType': 'Sound' } };
```

- Via new with array syntax

```
var criteria = new Object();
criteria ['Type'] = 'FileInfo';
```

```
// nested object
var filter = new Object();
filter['FileType'] = 'Sound'
var criteria = new Object();
criteria ['Filter'] = filter;
```


Platform Services

Service Type	Service Provider	Interface Identifier
Application Manager	Service.AppManager	IAppManager
Calendar	Service.Calendar	IDataSource
Location	Service.Location	ILocation
Logging	Service.Logging	IDataSource
System Information	Service.SysInfo	ISysInfo
Contacts	Service.Contact	IDataSource
Landmarks	Service.Landmarks	IDataSource
Media Management	Service.MediaManagement	IDataSource
Messaging	Service.Messaging	IMessaging
Sensors	Service.Sensor	ISensor

Service Interface

- Object extended from a service object and identified by service interface identifier

```
var si = serviceObject.ServiceIdentifier
```

- Object's methods are service dependent

```
ContactSI.Add() // add item to Contacts  
LocationSI.GetLocation() // get location
```

- Methods access service information

- Information returned in a Result object

```
var result = LocationSI.GetLocation()
```

- Provides both asynchronous and synchronous operations

```
LocationSI.GetLocation(criteria) // sync  
LocationSI.GetLocation(criteria, callback) // async
```

Result Object

- Result object property
 - ErrorCode: a predefined integer error code
 - ErrorMessage: a string that describes the error
 - ReturnValue: an iterative array of objects containing the information requested from the operation
- Result object as the returned value from a synchronous method

```
result = LocationSI.GetLocation(Obj:criteria)
```

- Result object as the parameter passed through callback method

```
LocationSI.GetLocation(Obj:criteria, Func:callback)  
callback(Int:transID, Int:eventCode, Obj:result)
```

Iterable Object Array

- An array of objects with the requested service information
- Length of the array is operation dependent
- Objects in the array can be mixed – number; string; boolean; object; array
- Methods
 - reset()
 - getNext()

Revisit Application Launcher Widget

- Coding to WRT 1.1 API
 - Launch application
 - Battery level

```
var serviceObj ;
function init(){
    if (window.menu == undefined) return;

    // Obtain the AppManager service object
    try {
        serviceObj = window.device.getServiceObject("Service.AppManager",
                                                    "IAppManager");
    }
    catch (ex) {
        serviceObj = null;
    }
}
```

Application Manager

iAppManager interface

- GetList:** Get the list of Applications and User Installed Packages
- LaunchApp:** Launch a given application based on UID
- LaunchDoc:** Launches the application based on the given document or mime
- Cancel:** Cancel an ongoing Async request.

Example

```
var so = device.getServiceObject("Service.AppManager", "IAppManager");
var criteria = { ApplicationID:"s60uid://0x101f4cce", // uid for contacts
                Option:{ Mode:"Standalone" } };
var result = so.IAppManager.LaunchApp(criteria);
```

Launch WRT 1.1 Application

```
var apps =[
  {idx: 0, img: "cal-48x48.png", appId:'0x10005901', count: 0 },
  ...
];

if (serviceObj == undefined)
  return; // No WRT
else if (serviceObj != null)
  launchApp5thEd(uid); //WRT 1.1
else
  launchApp3rdEd(uid); //WRT 1.0

function launchApp3rdEd(uid){
  // the UID must be an integer
  uid = parseInt(uid);
  widget.openApplication(uid, "");
}

function launchApp5thEd(uid){
  var criteria = { ApplicationID: "s60uid://" + uid };

  var result = serviceObj.IAppManager.LaunchApp(criteria);
  if (result.ErrorCode != 0) {
    alert(result.ErrorCode + ": " + result.ErrorMessage);
  }
}
```

Battery – Supporting WRT 1.0, 1.1

```
<body onload = "init()">
  ...
</body>

var sysInfo; // undefined

function init() {
  if (window.menu == undefined) return; // Check if WRT enabled

  // Try WRT 1.1 first by getting SysInfo as ServiceObject
  try {
    sysInfo = window.device.getServiceObject("Service.SysInfo", "ISysInfo");
  } catch (ex) {
    // Must be WRT 1.0, embed SystemInfo plugin
    sysInfo = document.createElement("embed");
    sysInfo.setAttribute("type", "application/x-systeminfo-widget");
    sysInfo.setAttribute("hidden", "yes");
    document.body.appendChild(sysInfo);
  }
}
```

Battery – WRT 1.1 Reading

```
function checkSize() {
    ...
    if (sysInfo == undefined) return;
    else if (typeof sysInfo.chargeLevel == "number")
        recheck( sysInfo.chargeLevel);
    else {
        var criteria = { Entity: "Battery", Key: "BatteryStrength" };
        try {
            var result = sysInfo.ISysInfo.GetInfo(criteria, batteryCallback);
        } catch (ex) { alert(ex); return; }
    }
}

function batteryCallback(transId, eventCode, result) {
    // On error situation, display the error message
    if (eventCode == 4) {
        alert("Error " + result.ErrorCode + ": " + result.ErrorMessage); return;
    }
    recheck(result.ReturnValue.Status);
}
```

System Info

iSysInfo Interface

- GetInfo/SetInfo: Read/modify System Attribute Values
- GetNotification: Register a callback function for listening to notifications
- Cancel: Cancel an ongoing Async request

Supported Features

- Battery: BatteryStrength, ChargeStatus
- Network: SignalStrength, CurrentNetwork, ...
- General: InputLanguage, VibraActive, ...
- Features: BlueTooth, CAMERA, Pen, ...
- Display: Brightness, ScreenSaverTimeout, ...
- Connectivity: BlueTooth, ActiveConnections, ...
- Memory: DriveInfo, CriticalMemory,
- Device: PlatformVersion, PhoneModel, IMEI, etc.

Example

```
var so = device.getServiceObject("Service.SysInfo", "ISysInfo");
var criteria = {entity:"General", key:"VibraActive"};
var result = so.ISysInfo.GetInfo(criteria);
```

Home Screen Widget

- New UI control in compatible 5th edition devices
 - Not a widget in the same sense used in this tutorial
 - First device to support this is N97
- Add up to 4 applications
 - Native application or WRT 1.1 widget
 - Landscape or portrait orientation
- Widget (implementing miniview) can be added
 - N97 – size is 309 x 85 pixels
 - Bitmap of the miniview div is displayed
 - Add to home screen if following added to info.plist file
 - `<key>MiniViewEnabled</key> & set to <true/>`

Home Screen Interactions w/ Widget

- Add widget to home screen – activate
 - Launch widget in background
 - Fire onload() and onshow() events
 - Prompt user permission
 - Send bitmap updates to home screen
- Select miniview to open widget to full view – select
 - Fire onshow() and onresize() events to widget
 - Starts timer for widget and suspends other home screen widgets
- Resume/bring home screen to foreground – resume
 - Start timers for all widgets in home screen
 - Fire onshow() and onresize() events to all widgets in home screen
- Suspend/send home screen to background – suspend
 - Fire onhide() event to widget which widget does not do anything with
 - Suspend timers and suspend bitmap updates of home screen
- Remove widget from home screen – deactivate
 - Shuts down widget

Enabling Widget for Home Screen

- Add MiniViewEnabled key to info.plist
- Publish miniview content
 - <div id="miniview"></div> with contents in main HTML
 - Stay within dimension of miniview
- Implement onload, onshow, onresize event handlers
 - onload() and onshow() when widget activated
 - onshow() and onresize() when widget selected
 - onshow() and onresize() when home screen resumes
- Full or mini view: use viewport changes (onresize)
- Determine policy for bitmap updates home screen
 - Use setInterval for widget to refresh content
 - Consider battery issues when doing this
 - Use onshow if content is not time but state dependent

Revisit Application Launcher Widget

- MiniView show the top-4 most used application
- Every click in AppLaunch increments use counter
- Frequency order changes, mini view is updated



Home Screen Widget & Web Widgets

- Home Screen widget
 - Notification
 - Accelerator to hosted widget
- Mobile Web widgets
 - Information – mashup of phone and Internet content
 - Accessory – stock quotes, weather, Facebook
 - Application – mashup of phone and Internet functionality
- Created with Web 2.0 technologies
 - For applications that are mashup-like
 - Development with short turnaround
 - Target for more users, developers, and mobile devices
 - Combine both Internet and mobile device data and capabilities

Good Practices

- Widget deployment model different – new updates?
 - Widget should check of updates
 - Offer user option to update to newer version
- Use templating wherever possible
 - DOM elements created using templates
 - Save JavaScript for dynamic elements
 - Templating – server-side (use build script), or client-side
 - Client-side options: mjt, ...
 - Server-side options: JSP, PHP, Kid, Django, webpy
- Test in Firefox with Firebug on desktop
 - Create test version
 - Use Aptana, emulator toolkits

Good Practices (Cont'd.)

- `if (window.menu !== undefined) {}`
- `try{ ... }catch(err){catchError("from text", err);}`

```
function catchError(title, err) {  
  var str = title + "\n";  
  for( var a in err) {  
    str+=a + ": " + err[a] + "\n";  
  }  
  if (window.console === undefined) alert(str);  
  else window.console.log(str);  
};
```

- With new touch UIs, fluid designs more important
- Input in absence of full keyboard or touch UI should be provided
 - Key shortcuts

Topics Not Covered

- Localization
- Security
- Accessibility
 - Point out Raman's work with AxsJAX and EyesFree
- Mobile UI designs especially with touch UIs
- Haptics and sensors for context and user experience
- Other mobile widget run times (e.g., Opera)

Summary

- Cache control – do not expire, use versioned URLs
- Compress, compress, compress
- Be conscious of energy in design of mobile services
- Handle heterogeneous mobile devices and environments
- Mobile Web widgets mashup Internet and phone data and capabilities
- Mobile Web widgets can do for Mobile 2.0 much like what AJAX did for Web 2.0
- Prototype and explore services for mobility and mobile work

Q & A