Developing Mobile Applications

Client side scripting
Client side scripting
Client side scripting

• Pros:
  – off-loads server
  – avoids access over high latency links

• Cons:
  – reduced language
  – limited access to runtime environment
  – interpreted, slow execution speed
  – limited memory
WMLScript

- Part of the WAP 1.x specification.
- A much reduced version of ECMAScript
  - JavaScript was developed by Sun/Netscape and standardized by ECMA
  - JScript is the MS version of JavaScript
  - JavaScript has nothing to do with Java
- Called from WML pages
  - no in-line code in WML pages
- Libraries give access to the browser environment (WAE).
ECMAscript MP (ESMP)

- Scripting used with XHTML-MP
  - a large subset of ECMAscript
  - and Document Object Model by W3C
- Only recently (2006) approved as a OMA standard and thus not available in the first generation of XHTML phones.
- Combine this with AJAX and you have an exiting platform for mobile applications.
  - AJAX: provides access to XML documents
The future

Presentation is adapted to user and terminal.

XML content in raw format

XHTML/ESMP
WML and WMLScript

- WMLScripts are stored in separate files and access from a WML page by a URL in for example:

  `<go href="foo.wmls#func(bar)"/>

- WML and WMLScript communicate with shared environment variables.
- WMLScripts will direct the browser to the next (previous or current) page.
WMLScript  calc.wml

<wml>
  <card>
    <p>
      3 + 4 =
      <a href="calc.wmls#add(3,4)">calc</a>
    </p>
  </card>
  <card id="result">
    <p>
      3 + 4 = $res
    </p>
  </card>
</wml>
WMLScript    calc.wmls

extern function add(x, y) {
    var xn = Lang.parseInt(x);
    var yn = Lang.parseInt(y);
    var zn = xn + yn;
    WMLBrowser.setVar(“res”, zn);
    WMLBrowser.go(“calc.wml#result”);
}
WMLScript calc2.wml

<wml>
  <card>
    <p>
      3 + 4 = $res
      <br/>
      <a href="calc.wmls#add(3,4)">calc</a>
    </p>
  </card>
</wml>
extern function add(x, y) {
    var xn = Lang.parseInt(x);
    var yn = Lang.parseInt(y);
    var zn = xn + yn;
    WMLBrowser.setVar("res", zn);
    WMLBrowser.refresh();
}
Browser interaction

Shared variables

WML

WMLscript

WMLscript

call

go, ...

Developing Mobile Applications 2G1722 Johan Montelius
WMLScript data types

- Strings: “foo”, 'bar', “a \n newline”
  - all browser variables are strings!
- Integers: 1, 2, 0xa
- Floats: 3.14  0.1e3
- Boolean: true, false
- Invalid: invalid
- No objects, structures nor arrays!
  - string can be used as arrays :-)

Developing Mobile Applications 2G1722  Johan Montelius
WMLScript syntax

- **expression**: 5, true, x, x + y, x < y, ....
- **statement**: x = y + 3; , foo(3,4); , ...
  - var x;
  - if ( expression ) { .... } else { .... };
  - for (x =1, x < 10, x++) { ... };
  - while (x < 10) { ... };
  - break; , continue;
- **function definition**:
  - function foo ( ) { ... return expression; }
WMLScript variables

- Variables must be declared before used.
- Variables have a scope only in the function where they are defined.
- Weakly typed
- Automatic conversion – warning!
- Explicit conversion by using library functions.
- No type casting.
A timer

<wml>
  <card ontimer="timer.wmls#tic()">
    <timer value="50"/>
    <p>Time: $tac s</p>
  </card>
</wml>
A timer

timer.wmls

extern function tic() {
    var ts = WMLBrowser.getVar("tac");
    var tn = Lang.parseInt(ts);
    tn = tn + 5;
    WMLBrowser.setVar("tac", tn);
    WMLBroser.refresh();
}
Libraries

- **Lang** - functions for data type manipulation, and calculations
- **Float** - the float library is optional
- **String** - string operations
- **URL** - a set of functions for URL construction
- **Dialogs** – prompting the user for input
- **WMLBrowser** – gives access to and directs the browser environment
WMLBrowser

- **Variables**
  - `getVar(variable)`, `setVar(variable, value)`
- **Navigating**
  - `go(page)`, `prev()`, `refresh()`
- `getCurrentCard()`
- `newContext()`
Crypto

- Signing of text strings by accessing the WAP Identity Module (WIM).
- WIM is implemented on the SIM card (SWIM card), the operator is thus involved!
- Will encrypt a text that can be sent to a server for verification.
- Check if this library is implemented on your phone and if it's supported by your operator.
usage

• It is hard to do advanced calculations since we lack support for complex data structures.
• Problematic to do advanced interaction using WML pages.
• Try to do sanity check on values that are sent to a server application – we do not want to wait 4 seconds to find out that we typed in the wrong information.
WMLscript byte code

- A terminal that supports WMLscript does often only support interpreting the byte code form of the compiled script.

- Advantages
  - less bytes in the air
  - much easier to execute, no parsing!

- Disadvantage
  - compilation done on the fly by the WAP gateway
  - what if there is no WAP gateway
ECMAscript MP

- A much richer language compared to WMLScript.
  - data structures
  - libraries
- New phones might have support only for ESMP even if WML is supported.
- Most interesting is how to work with the Document Object Model.
Interact through document objects

```html
<html>
<body>
  <script type="text/javascript">
    document.write("Hello World!"
  </script>
</body>
</html>

Dynamic HTML
Identify the browser

```html
<body>
<script type="text/javascript">
var browser= navigator.appName;
var version= navigator.appVersion;
document.write("Browser name: "+ browser)
document.write("<br />")
document.write("Browser version: "+ version)
</script>
</body>
```
And more

- Set and detect cookies all on the client side.
- Detect screen size, color, ...
- Current URL.
- Look at history.
- Catch and set events, timer,...
- XMLHttpRequest ....
  - this is where the fun starts
- XML parsing
  - this is the final piece
Summar

- Client side scripting is important:
  - it can hide latency in the network
  - it offloads the server, it scales
  - allows content to be adapted
- AJAX
  - client side scripting
  - Document Object Model
  - HTTP requests
  - XML parsing