Canon

Component Model for Multimodal Web Applications

How to combine object oriented concepts with markup for event driven applications

Dave Raggett, W3C/Canon
MWeb Workshop, Sophia Antipolis, July 2004

Goals

- Break Web applications out of the browser!
- Freedom to build wider variety of applications
- Reduced costs and increased flexibility
- Easy adaptation to wide range of devices
- Multimodal User Interface
- Author defined controls
- Ability to mix novel and standard markup

Theming

- Zinf – one application, three themes

Theses can also be applied across applications, vis Gnome desktop themes

Break free out of the Browser

- Non-rectangular windows
- No window chrome
- Treat like regular applications
  - Launched from application menus or desktop icons
  - Can be bound to media types, e.g. audio/mpeg

Application Model

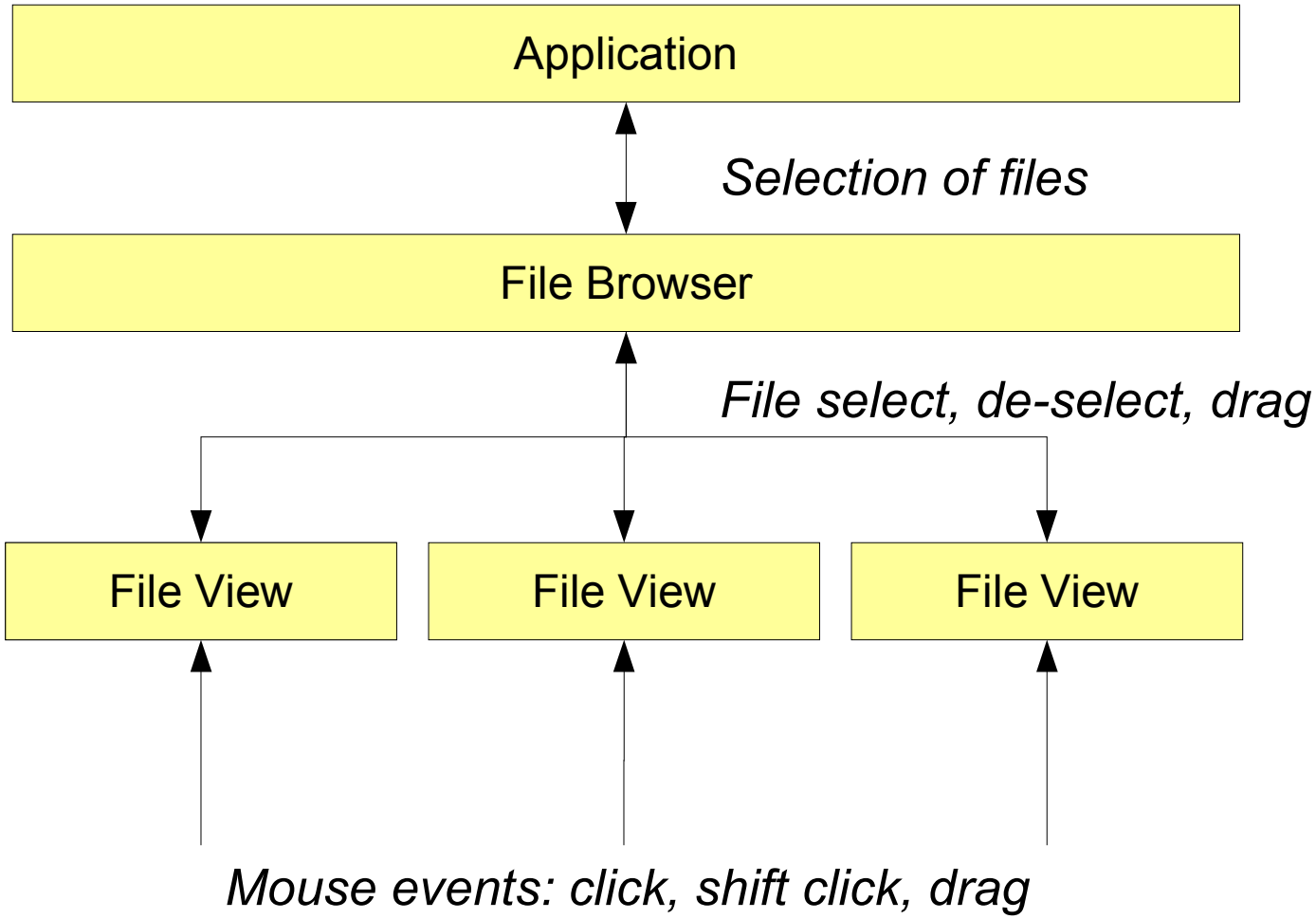
- Describe object model in XML
  - Properties
    - Preferences
    - Play-list
    - Current track
      - Duration, current position, name, artist, ...
  - Methods
    - volume, pause, rewind, fast forward, previous/next track, quit
  - Events
    - Playback error, end of track, end of play-list, ...

Themes and Intentions

- Split user interface into
  - Abstract UI controls and layout intentions
  - Theme = presentation and behavior for controls
- XForms range control as scalar value
  - Themed as rotary dial or thermometer
  - Role of SVG and XBL for defining such controls
- Layout intentions
  - Vertical, horizontal, grid, ...
  - Delegate size/position decisions to layout manager
  - Detailed appearance determined by current theme

File Browser Application

Components and Events



Declarative Treatment of Behavior

- Simple event binding
  - dom:activate on button invokes application *pause* method
  - xf:xforms-value-changed on dial sets app *volume*
- State transition models
  - Event driven transitions between named states
  - Represented in XML and XPath
  - Transitions invoke methods, update data, raise events
  - Nested states and concurrent execution
  - Run on device or server, or on both
    - Distributed execution model

Input Modalities

- Mouse emulation events
  - Click, double click, down, up, drag
- Pointer trace data
  - InkML as XML format for trace data
  - Interpreted by component as gesture, drawing etc.
- Constrained text input
  - Provided with keystrokes, speech or handwriting
  - Regular expressions, CFGs or data types
- Semantic events
  - Platform specific binding to keys, gestures or speech

Interpreted Input

- Treat all input uniformly as events
  - Audio, speech, keystrokes and stylus
- Use of grammars for recognition and extracting application semantics
- EMMA – extensible multi-modal annotations
  - XML language for interpreted input
    - Application specific + standard annotations
      - Time stamps, confidence scores, N-best interpretations, ...
    - Role of XPath for dealing with EMMA

Output Modalities

- Speech and Audio
  - Speech Synthesis Markup Language (SSML)
- Visual
  - XHTML
  - SVG
  - XForms
- Coordinated output with SMIL
- Promise of Natural Language Generation

Multimodal Interaction

- Enable user to choose which mode to use
  - Constrained text input and semantic events
- Deictic References and Ellipsis
  - Behavior is defined by the application component
    - *In context of file browser application: "Print this"*
      - Ask file browser for current selection
      - If necessary prompt user to make selection
    - For map application *"Zoom in here"* + *tap with pen*
      - *Map interprets mouse click event as map location*
- Complementary use of output modes
  - Combine speech with highlighting of visual objects
    - This hotel is conveniently close to the opera house

Natural Language Understanding

- Context free grammars in XML
  - Speech Recognition Grammar Specification (SRGS)
- Functional approach to semantic interpretation
  - W3C Semantic Interpretation specification
    - Semantic expressions as annotations on grammar rules
    - *"Pepsi Cola"* is mapped to <beverage>pepsi</beverage>
    - *"July 20th"* is mapped to <date>2004-07-20</date>
    - Ripples up parse tree to define semantics of utterance
- No match, or No input events
- Tapered prompts and traffic lights model

Natural Language Generation

- VoiceXML uses variables and expressions
  - Simple templates are sufficient in many cases
  - Dynamic generation of VoiceXML on web server
- Promise of richer NLG
  - Improvements in speech synthesis
  - More flexible use of natural language
  - Dynamic generation in user's preferred language
- Potential for NLG engines "realizers"
  - Driven by XML representation of language independent communication acts
  - Realizer deals with sentence planning and word selection

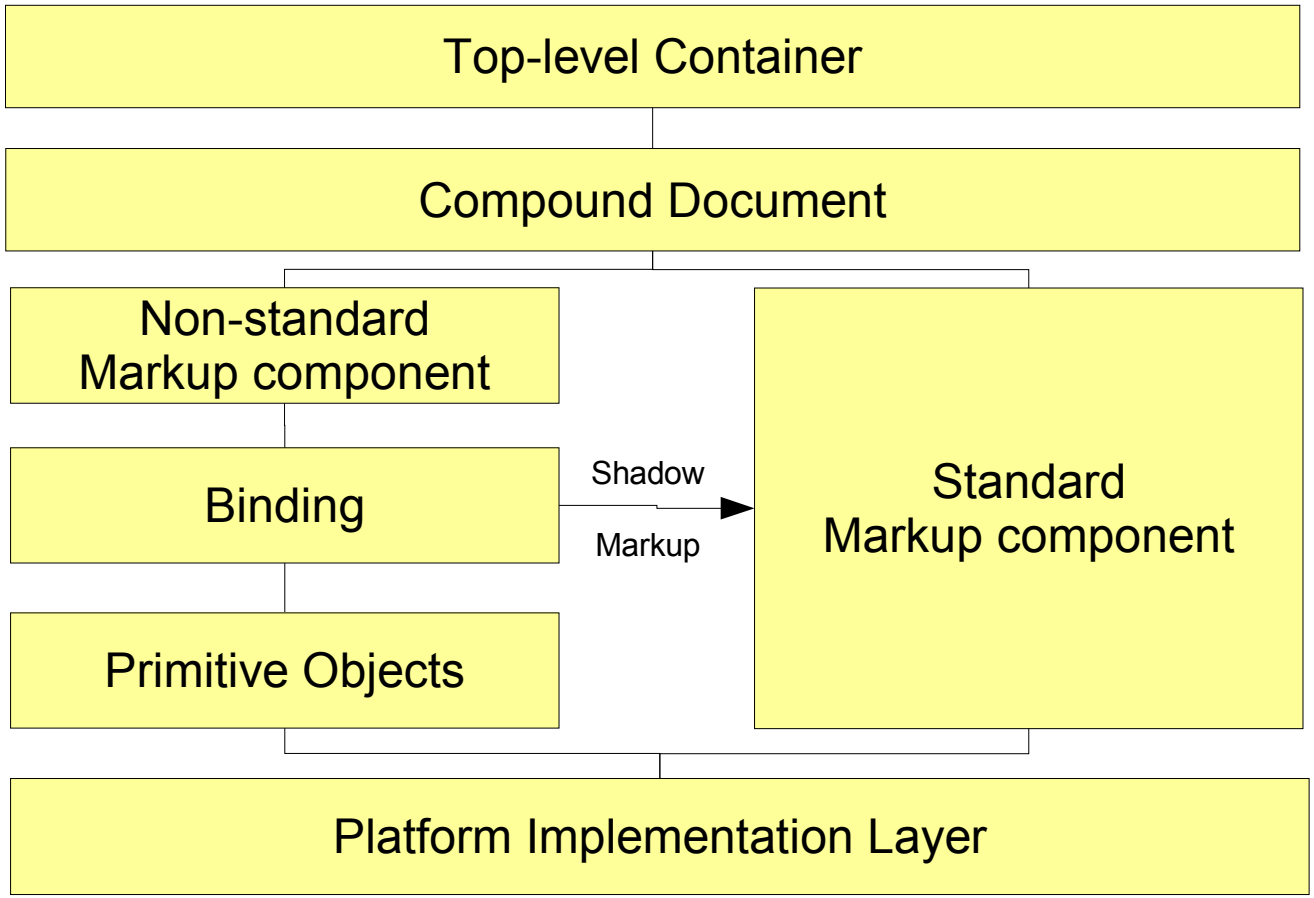
Adapting to Current Conditions

- How to enable applications to adapt to conditions on a moment to moment basis?
  - Portrait to Landscape flip
  - User mutes audio
  - Low battery alert
- System and Environment Framework
  - Access to device capabilities, user preferences and environmental conditions
  - Exposed as XPath expressions or JavaScript
  - Shields application from lower level interfaces

Mixing Novel and Standard Markup

- Treat XML as instructions for creating a composition of objects
- Bind elements to objects
  - Implicit binding for standards-based markup
  - Objects defined in XML or in code
- Interoperability depends on
  - Binding mechanism, and object language
    - e.g. XBL and JavaScript
  - Libraries and interfaces these objects depend on

Markup Components



Compound document consists of sequences of nestable markup components. Non-standard markup components are bound to object models in terms of primitive objects using a mixture of declarative and imperative definitions. Type constraints determine which compositions of markup components are valid.

XBL example

```
<?xml version="1.0"?>
<?xml-stylesheet href="notes.css"?>
<bindings xmlns="http://www.mozilla.org/xbl"
  xmlns:html="http://www.w3.org/1999/xhtml">
  <binding id="board" styleexplicitcontent="true">
    <implementation>
      <property name="dragging"> null </property>
      <property name="currX"> 0 </property>
      <property name="currY"> 0 </property>
    </implementation>
    <handlers>
      <handler event="mousedown">
        if (event.originalTarget.parentNode.className == 'caption') {
          this.dragging = event.originalTarget.parentNode.parentNode;
          this.currX = event.clientX;
          this.currY = event.clientY;
        }
    </handler>
    <handler event="mouseup">
      this.dragging = null;
    </handler>
    <handler event="mousemove">
        this.currY = event.clientY;
        ...
    </handler>
  </handlers>
</binding>
```

Canon

XBL Syntax Summary

```
<?xml version="1.0"?>
<bindings xmlns="http://www.mozilla.org/xbl">
  <binding id="name">
    <implementation>
      <property name="name" inherit="name">
         ... initial value ...
      </property>
      <constructor> ... </constructor>
      <destructor> ... </destructor>
      <method name="name"> ... </method>
      ...
    </implementation>
    <handlers>
      <handler event="name">
        ... script for event handler ...
      </handler>
      ...
    </handlers>
    <content>
      ... anonymous content ...
    </content>
  </binding>
</bindings>
```

Task-based Interaction

- Example of value of novel markup
  - Define your own markup for task dependency trees
    - Partial ordering of tasks and sub-tasks
  - Implement your plan engine in JavaScript
  - Bind behavior with XBL
- Enables developers to innovate on richer ideas for multimodal web applications

Summary

- Use Object Oriented approach
  - XML for describing objects and their behaviors
  - Separate application from user interface
  - Shield from platform specific interfaces
- Treat input in terms of events
  - Mouse events, text events, semantic events ...
- Dynamic adaptation to changing conditions
  - Avoid locking applications to specific platforms
  - Complying with user preferences/impairments
  - Applications with multiple devices/people