

# XML Events

# Überblick

- Motivation
- DOM2 Events
- HTML Events
- XML Events Module
- Aufbau

# Motivation

- Die durch XML definierten Elemente sind statisch.
- Erwünscht:

## Möglichkeiten

- den durch XML definierten Elementen Verhalten zu verleihen
- dadurch Interaktion dieser Elemente zu ermöglichen, in dem sie auf aufgetretene Ereignisse reagieren

# Motivation

- Ereignis:  
Repräsentation einer asynchronen Erscheinung, die mit einem Element (target) in einem XML-Dokument in Beziehung gesetzt wird.
- Die Interagierbarkeit der durch XML definierten Elemente wird durch DOM2 Eventmodell realisiert.

# Motivation

## **XML Events Modul**

wurde konzipiert

um einen generischen Mechanismus zu bieten,  
für verschiedenen XML-basierten Sprachen  
DOM2 Listener zu definieren

# DOM 2 Events

# Document Object Model (DOM)

- Bildet die Struktur eines XML-Dokuments im Arbeitsspeicher
- Ermöglicht Zugriff auf die Struktur des Dokuments und den Inhalt einzelner Elemente über Schnittstelle API
- Anwendungen werden in Skriptsprachen oder in objektorientierten Sprachen verfasst

# DOM Levels

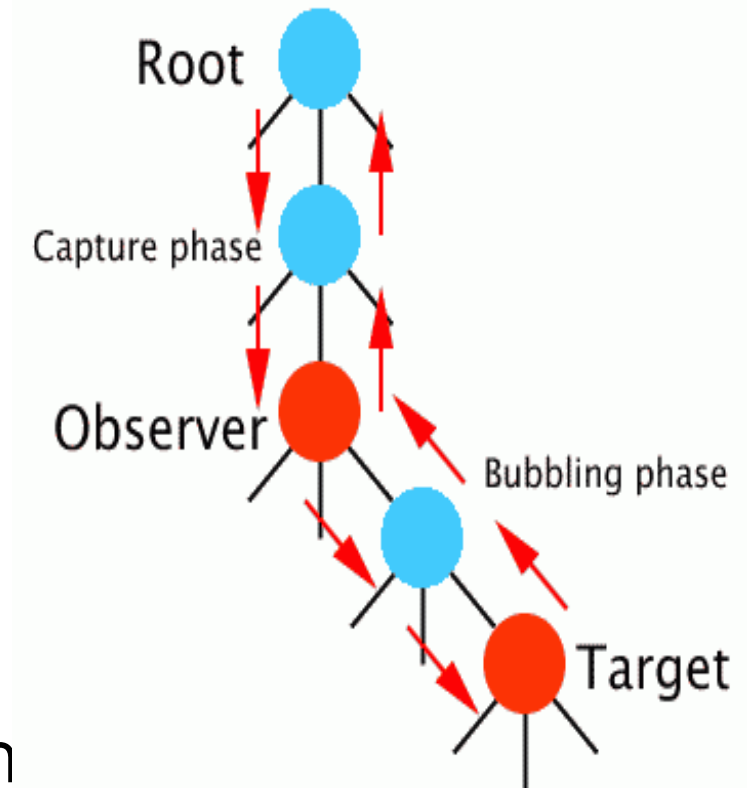
- DOM Level 1
  - Application programming interface (API)  
für XML (und HTML) Dokumente
  - Baumstruktur-
- DOM Level 2
  - Zusätzlich Schnittstelle für Dokumentnavigation,  
CSS und Manipulation der eventbasierten XML  
Dokumente

# Eigenschaften des DOM2 Events

- Ein allgemeines Ereignissystem
- Möglichkeiten, um Ereignis-Listener und -Handler zu registrieren
- Möglichkeiten, um Ereignisse durch die Baumstruktur zu leiten
- Zugang zu Kontextinformationen für jedes Ereignis
- Definition des Ereignisflusses

# Ereignis-Fluss

- **Capture-Phase**  
Ereignis wandert vom Root zum Target-Element
- **Bubbling-Phase**  
Ereignis kehrt zurück
- Bei jedem Observer auf diesem Pfad kann auf Ereignis reagiert werden



# HTML Events

# HTML Events vs XML Events

XML Events verwendet gleichen Event-Mechanismen wie HTML

```
<input type = "submit,, onclick="validate();return true;">
```

Gleiche Effekte

```
<input type="submit">  
  <script ev:event="DOMActivate" type="text/javascript">  
    validate();  
  </script>  
</input>
```

# HTML Events vs XML Events

- Einschränkungen von HTML Events
  - Fest spezifizierte Event-Namen
  - Nur eine Skriptsprache für Event-Behandlung
  - Untrennbare Event-Behandlung von Markup
- Vorteile von XML Events
  - ermöglicht neue Ereignistypen zu definieren
  - erlaubt mehrere Handler für verschiedene Skriptsprachen
  - Erlaubt einen Handler für mehrere Ereignisse

# HTML Event vs XML Event

```
<input type="submit">  
  <script ev:event="DOMActivate"  
          type="text/javascript"> ... </script>  
  <script ev:event="DOMActivate" type="text/vbs"> ...<script>  
</input>
```

```
<input type="submit">  
  <script ev:event="DOMActivate" type="text/javascript"> ... </script>  
  <script ev:event="DOMFocusIn" type="text/javascript"> ... </script>  
</input>
```

# XML Events Module

# XML-Events Module

- W3C Empfehlung zur Ereignissyntax für XML
- Verleiht das Verhalten oder Reaktion eines bestimmten Elements
- Verbindet einheitlich Event-Listener und Event-Handler mit Ereignisschnittstellen des DOM Level 2

# Ziele

- Das DOM-Ereignismodell syntaktisch auf ein XML-Dokument ausweiten
- Neue Ereignistypen bereitstellen ohne Modifikation des DOM oder der DTD
- Die Integration in andere XML-Sprachen gestatten

# Elemente und Attribute

## XML-Events Modul

Element	Attribute	Minimales Inhaltsmodell
listener	event ( NMTOKEN), observer (IDREF), target (IDREF), handler (URI), phase ("capture"   "default"*), propagate ("stop"   continue"*), defaultAction ("cancel"   perform"*), id (ID)	EMPTY

# Das Element listener

- Unterstützt die Schnittstelle **EventListener** des DOM
- Es wird verwendet, um EventListener zu deklarieren und sie mit bestimmten Knoten im DOM zu registrieren.

# Attributen von Listener

**event** – erforderlich -

Ereignistyp, für den der Listener registriert wird

**observer** - optional -

**id** des Elements, zu dem der EventListener registriert werden muss. Ist dieses Attribut nicht verfügbar, ist der *observer* das Element, für das das **event**-Attribut angegeben ist

**target** - optional -

gibt **id** des Zielelements des Ereignisses an

# Attributen von Listener

**handler** - optional-

gibt den URI-Verweis der Quelle an, welche die Aktion definiert, die ausgeführt werden soll, wenn das Ereignis den Observer erreicht

**phase** - optional –

gibt an, wann (zu welcher DOM 2-Ereignisverlaufphase) der Listener vom gewünschten Ereignis aktiviert wird

**capture, default**

# Attributen von Listener

**propagate** - optional –

gibt an, ob dem Ereignis nach der Verarbeitung aller Listener am aktuellen Knoten gestattet wird, seinen Weg fortzusetzen

**stop, continue (default)**

**defaultAction** – optional –

gibt an, ob nach der Verarbeitung aller Listener für das Ereignis, die voreingestellte Aktion für das Ereignis (sofern vorhanden) ausgeführt werden soll oder nicht.

**cancel, perform (default)**

# Attributen von Listener

**id** - optional-

ist eine dokumentweit einzigartiger Identifier. Der Wert dieses Identifiers wird oft dazu verwendet, das Element über eine DOM-Schnittstelle zu manipulieren.

# Aufbau

Konformität

Syntax

# Konformität der Dokumente

- XML Events ist kein eigenständiger Dokumenttyp
- Das Dokument muss hinsichtlich der Schema-Implementierung oder DTD-Implementierungen konform zu den Beschränkungen sein
- Das Dokument muss eine **xmlns**-Deklaration für den Namensraum von XML-Events enthalten

<http://www.w3.org/2001/xml-events>

# Syntax

- Spezifizieren
  - listener : spezifiziert  
event, handler, observer
  - Observer- Attribute spezifizieren  
Event, Handler
  - Handler – Attributes spezifizieren  
Event, Observer

# (Event, Observer, Handler ) explizit spezifizieren

```
<listener event = "activate"  
    observer = "button1 " handler = "#doit " />
```

- Wenn Ereignis *activate* stattfindet
- am Element *Id=button1* oder dessen Kindknoten
- rufe Handler *#doit*

# Event-Propagation stoppen

```
<listener propagate = " stop"  
    event = " activate" observer = "block"  
    handler = "popup " phase = "capture" />
```

- Rufe Handler *popup*
- Wenn das Ereignis *activate* den *block* erreicht
- Dann stoppe die weitere Ereigniswanderung

# Attribute einem Observer-Element direkt zuordnen

- Ereignis kann direkt am Observer  
spezifiziert werden

```
<anyelement ev:event="ev:click"  
  ev:handler = "#clicker1"/>
```

```
< a href = "doc.html"  ev:defaultAction = "cancel"  
  ev:event = "activate" ev:handler = "#popper">  
  The Document  
</a>
```

# Attribute einem Handler-Element direkt zuordnen

Event-Attribute können beim Handler gesetzt werden

```
<script type = ...  
    ev.event= " submit"    ev.observer = " form1 ">  
    return docheck( event );  
</script>
```

- Deklariert script handler für Ereignis submit,
- das am Element id=form1 ausgelöst wird

# Event Handler

Häufigen Methoden zur Definition von  
Handlern:

- Skripte
- Deklaratives Markup wie Element Action  
in XForm 1.0

# Voreinstellungen für Observer- und Handler-Attribute

Auswirkungen von weggelassenen Observer- und Handler-Attributen

	Handler vorhanden	Handler weggelassen
Observer vorhanden	(wie deklariert)	Element ist Handler
Observer weggelassen	Element ist Observer	Element ist Handler Elternteil ist Observer

# Beispiel

- Beispiel mit XForm



Event Beispiel

# Beispiel

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<html xmlns=http://www.w3.org/1999/xhtml
```

```
xmlns:ev="http://www.w3.org/2001/xml-events"
```

```
xmlns:xforms="http://www.w3.org/2002/xforms">
```

# Beispiel

```
<body>
  <xforms:select1 ref="cpu" appearance="full">
    <xforms:label>CPU</xforms:label>
    <xforms:item>
      <xforms:label>Pentium 4 2.53Ghz -
      $220</xforms:label>
      <xforms:value>2.5</xforms:value>
      <xforms:message level="ephemeral"
      ev:event="xforms-select">1.te
      Selektion</xforms:message>
    </xforms:item>
    .....
  </body>
```

# Anwendungen

- XHTML2( in Vorbereitung )
- XForms
- XHTML+Voice (X+V)
- SVG1.2 ( in Vorbereitung)

Ende

Vielen Dank!