Программирование в Mozilla

По материалам сайта

http://www.xulplanet.com/tutorials/xultu/

Добавление Javascript в XUL-страницы

Добавление страниц с текстом на *Javascript* происходит точно так же, как и в HTML-страницах – с помощью тега `<script>`.

```
<window title="Find Files" orient="horizontal"

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
   <script type="text/x-javascript">
      var myFileName = null;
   </script>
   <script src="chrome://findfiles/content/findfiles.js"/>
   ...
</window>
```

В качестве типа скрипта часто используется "text/x-javascript" – это «расширенный» Javascript. Не рекомендуется вставлять тексты скриптов непосредственно, лучше использовать отдельные файлы.

Механизм обработки реакции на события

Определим реакцию на щелчок мышью внутри некоторого элемента:

```
<window ...>
  <script src="..."/>
  <hbox>
    <label value="Click here" onclick="doSomething();"/>
  </hbox>
</window>
```

При возникновении события его обработка проходит следующие фазы:

1. Capturing: событие распространяется от объектов window и document вниз до того элемента, внутри которого возникло событие;
2. Bubbling: после обработки в каждом из элементов событие начинает распространяться обратно вверх вплоть до объекта window.

При обработке события в каждом из элементов сначала исполняется реакция, определенная программистом, а затем реакция «по умолчанию», определяемая типом и природой элемента.

Можно управлять тем, на какой из фаз распространения происходит обработка события, продолжать ли распространение и сохранить ли реакцию «по умолчанию».

Пример определения реакции

Событие command – это событие нажатия на кнопку, выбор элемента меню, выбор радио-кнопки и т.п.

```
<?xml version="1.0" encoding="windows-1251" ?>

<window id="example-window" title="Пример определения реакции"
        xmlns:html="http://www.w3.org/1999/xhtml"

xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

  <vbox oncommand="alert(event.target.tagName);">
    <button label="OK"/>
    <checkbox label="Показать"/>
  </vbox>

</window>
```

test

Событие обрабатывается на фазе bubbling и покажет тег элемента, на котором произошло событие (сделали щелчок мышью или другим способом активизировали элемент).

Добавим реакцию на кнопку «Отмена» в нашем диалоге поиска файлов:

```
<button id="cancel-button" label="Отмена"
    oncommand="window.close();"/>
```

findfiles

Определение реакции на событие в js-файле

Определим файл, содержащий весь *javascript*-код: `findfiles.js` и привяжем его к нашему диалогу.

```
<window ...>
  <script src="findfiles.js"/>
   ...
  <button id="cancel-button" label="Отмена"/>
</window>


window.addEventListener('load', FF_OnWindowLoad, false);

function FF_OnWindowLoad() {
  var cancelButton =
      document.getElementById('cancel-button');
  cancelButton.addEventListener(
      'command', FF_CloseDialog, true);
}

function FF_CloseDialog() {
  window.close();
}
```

findfiles-1

Свойства и методы события

В функцию, определяющую реакцию на событие, всегда передается объект `event`. Он, в частности, имеет следующие атрибуты и методы:

- `target` – элемент, на котором возникло событие;
- `currentTarget` – элемент, на котором возникло событие;
- `stopPropagation()` – остановить процесс распространения события;
- `preventDefault()` – не выполнять реакцию «по умолчанию»;

Если событие принадлежит определенному классу (событие от мыши, например), то дополнительно имеются атрибуты и методы, характеризующие этот конкретный класс событий.

Mouse events:

- `click` – событие щелчка;
- `dblclick` – событие двойного щелчка;
- `mousedown, mouseup` – событие нажатия и отжатия кнопки;
- `mousemove` – событие изменения позиции курсора;
- `mouseover, mouseout` – событие изменения позиции курсора относительно выбранного элемента;

Свойства события от мыши и клавиатуры

Если произошедшее событие – это событие от мыши, то объект `event` имеет следующие атрибуты:

- `screenX, screenY` – координаты курсора в точках от начала экрана;
- `clientX, clientY` – координаты курсора от текущего документа;

В примере показано, как значения этих атрибутов используются для вычисления текущего положения курсора относительно окна и того элемента, внутри которого обрабатывается событие.

mouse

Keyboard events:

- `keypress` – событие нажатия клавиши;
- `keydown, keyup` – событие нажатия и отжатия клавиши;

События от клавиатуры возникают на элементе, имеющем фокус. Если такого элемента нет, а клавиша нажата внутри активного окна, то событие возникает на всем документе. Соответствующую реакцию можно определять для элемента `<window>`.

На самом деле для определения глобальных событий от клавиатуры обычно используется другой механизм.

Определение функциональных клавиш

«Глобальную» клавишу можно определить с помощью специального элемента <key>, расположенного в составе набора <keyset>.

```
<keyset>
  <key id="copy-key" modifiers="accel“
       key="C" onkeypress="doCopy();"/>
</keyset>
```

Ссылка на такую клавишу обычно делается из элемента меню или кнопки и это приводит к появлению текста клавиши в метке.

```
<menu id="edit-menu" label="Edit" accesskey="e">
  <menupopup id="edit-popup">
    <menuitem id="copy-menuitem" accesskey="c" key="copy-key“
              label="Copy" oncommand="doCopy();"/>
  </menupopup>
</menu>
```

findfiles-2

Исполнение кода команд

Аналогично, вместо того, чтобы определять один и тот же код в разных элементах, можно определить его один раз в элементе <command>.

```
<commandset>
  <command id="cmd-copy" oncommand="doCopy();"/>
</commandset>
  ...
<menuitem id="copy-command" accesskey="c" key="copy-key“
              label="Copy" command="cmd-copy"/>
```

Преимущество такого выделения команд в отдельный элемент – можно приписывать им свойства (например, disabled), не делая этого по отдельности для всех элементов, исполняющих эту команду.

Помимо приписывания свойств, элемент <command> может исполнить свою команду при вызове метода doCommand();

Используя эту методику, получим новый диалог findfiles: findfiles-3

Отделение кода команд

Так же, как и в случае HTML, практически весь код можно отделить от основного файла и поместить его в отдельный js-файл.

```
<script src="findfiles.js" type="text-javascript"/>

<commandset>
  <command id="cmd-search" label="Поиск" accesskey="и"/>
  <command id="cmd-cancel" label="Отмена" accesskey="м"/>
</commandset>

window.addEventListener('load', init, false);

function init(event) {
  document.getElementById('cmd-search').addEventListener(
      'command', doSearch, false);
}

function doSearch(event) {
  document.getElementById('progmeter').style.display = 'inherit';
}
```

findfiles-4