Программирование в Mozilla

По материалам сайта

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

XUL и Chrome

Цель: научиться программировать расширения (add-ons, extensions), работающие в среде *Mozilla Firefox*.

Программы на *JavaScript*, работающие внутри страниц, имеют много ограничений на доступ к системе.
С другой стороны, универсальные программы не имеют прямого доступа к содержимому страниц и управлению браузером.
Промежуточный подход: Chrome-пространство, имеющее доступ к содержимому браузера и загруженным страницам, и имеющее богатый набор функций для работы с различными компонентами системы.

XUL – XML-oriented User interface Language – язык для определения элементов диалога с пользователем. XUL-диалоги в Chrome-пространстве имеют XML-структуру, что позволяет управлять ими с помощью обычных *JavaScript*-программ. Весь браузер *Firefox* построен в виде большого XUL-диалога (Chrome-документа).

`Overlay` – это средство, с помощью которого можно «расширить» стандартный браузер *Firefox*, добавив в него новые элементы управления.

Построение первого extension’а

Чтобы установить свое расширение (extension), необходимо создать каталоги специальной структуры и некоторые специальные файлы.

1. Создаем каталог `firstextension/chrome/content/`.
2. Внутри каталога `firstextension/` создаем два текстовых файла: `install.rdf` и `chrome.manifest`.
3. Создаем приложение внутри `firstextension/chrome/content/`.
4. Упаковываем всю структуру `firstextension/` в `zip`-архив.
5. Присваиваем архиву расширение `xpi`, запускаем *firefox* и открываем в нем наше расширение.

Наш extension будет установлен как дополнение, и информацию о нем можно будет просмотреть в add-ons менеджере.

sample

Создание XUL-диалогов

Расширение, как правило, определяет элементы интерфейса с пользователем (новые пункты меню, кнопки, диалоги) и определяет программы на Javascript, работающие в ответ на действия с элементами.

Элементы интерфейса принято описывать не на HTML, а на специальном XML-языке, называемом XUL.

В качестве примера определим диалог для поиска файлов на языке XUL и добавим новый пункт меню для его вызова.

Создаем диалог `findfiles.xul` в файле со следующим содержанием:

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window id="findfile-window"
        title="Find Files"
        orient="horizontal"

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

Это пока просто пустое окно, которое будет доступно по адресу `chrome://findfiles/content/`

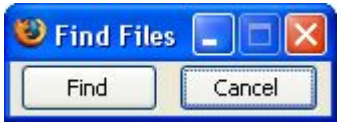
findfiles

Добавление элементов интерфейса в окно

Добавим пару кнопок в наше пустое окно. Кнопки описываются практически так же, как в языке HTML:

```
<window ...>
  <button id="find-button" label="Find"/>
  <button id="cancel-button" label="Cancel"/>
</window>
```

Наш диалог будет выглядеть примерно так:



Надписи можно добавить в наш диалог, используя элементы `<label>` и `<description>`. Фактически эти элементы ничем друг от друга не отличаются, но принято немного по-разному их использовать.
Еще в XUL (в отличие от HTML) можно задавать простое текстовое поле ввода с помощью элемента `<textbox>`.

Добавим поле для ввода имени файла для поиска:

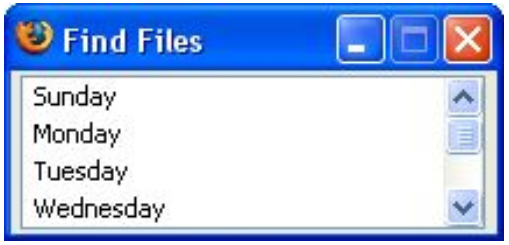
```
<label value="Search for:" control="find-text"/>
<textbox id="find-text"/>
```

Задание других элементов интерфейса

```
<description>
  Этот текст будет переноситься по словам, если не помещается
  в отведенное для него место.
</description>

<img src="chrome://findfiles/skin/myimage.jpg"/>
  или лучше:
<img id="my-image"/>
  а в CSS-файле
#my-image {
  list-style-image: url("chrome://findfiles/skin/myimage.jpg");
}

<textbox id="t-id" value="default" type="password" maxlength="8"/>

<checkbox id="c-id" checked="true" label="Флажок"/>

<radiogroup>
  <radio id="play" selected="true" label="Play"/>
  <radio id="work" label="Work"/>
</radiogroup>
```

Списки.

```
<listbox rows="4">
  <listitem value="0" label="Sunday"/>
  <listitem value="1" label="Monday"/>
  <listitem value="2" label="Tuesday"/>
  <listitem value="3" label="Wednesday"/>
  <listitem value="4" label="Thursday"/>
  <listitem value="5" label="Friday"/>
  <listitem value="6" label="Saturday"/>
</listbox>
```

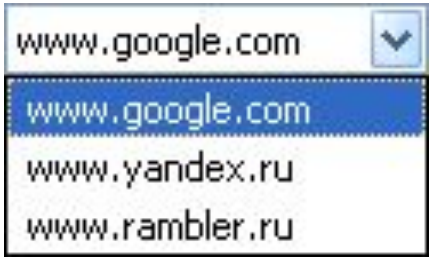



```
<listbox>
  <listhead>
    <listheader label="Name"/>
    <listheader label="Occupation"/>
  </listhead>

  <listcols><listcol/><listcol
flex="1"/></listcols>
  <listitem>
    <listcell label="Alex"/>
    <listcell label="Engineer"/></listitem>
  <listitem>
    <listcell label="Peter"/>
    <listcell label="Artist"/></listitem>
</listbox>
```

Выпадающие меню и указатели прогресса.

Для создания выпадающего меню используются три элемента: `menulist`, `menupopup` и `menuitem`.

```
<menulist editable="true">
  <menupopup>
    <menuitem label="www.google.com"
selected="true"/>
    <menuitem label="www.yandex.ru"/>
    <menuitem label="www.rambler.ru"/>
  </menupopup>
</menulist>
```



Счетчики прогресса:

```
<progressmeter
    value="50%"
    mode="determined"/>
```

Расширим наш диалог дополнительными элементами управления и разместим их, пользуясь элементами <vbox> и <hbox>.

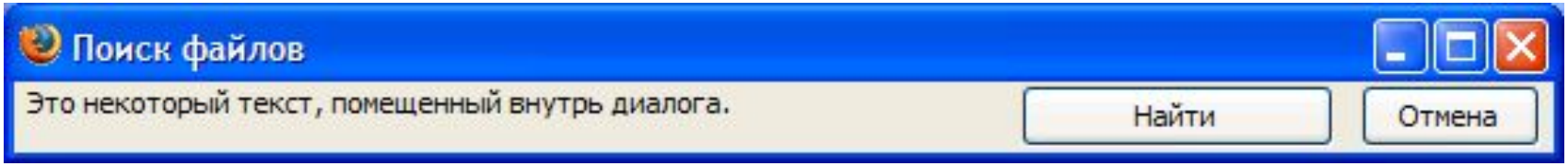
findfiles\chrome\content

Гибкие размеры элементов.

Часто бывает необходимо разместить свободное пространство в определенных местах диалога. Специально предназначенный для этого элемент называется `<spacer>`.

Регулировать размеры элементов можно явным указанием ширины и высоты с помощью атрибутов `width=` и `height=` или с помощью языка CSS, а можно указанием атрибута «гибкости» `flex`, например:

```
<hbox flex="1">
  <description>
    Это некоторый текст, помещенный внутрь
диалога.
  </description>
  <spacer flex="2"/>
  <button label="Найти" flex="1"/>
  <button label="Отмена"/>
</hbox>
```



Дополнительные возможности кнопок.

Можно добавлять картинки (иконки) на кнопки.

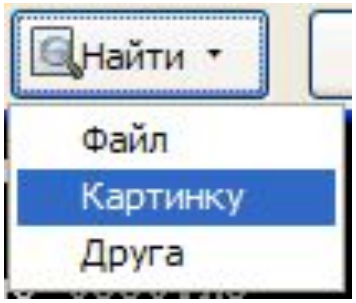
```
<button id="find-button" label="Найти" />

#find-button {
  list-style-image:
url("chrome://findfiles/skin/find.png");
}
```

Или можно полностью сформировать содержимое кнопки:

```
<button id="find-button">
  <description>Найти</description>
  <image
src="chrome://findfiles/skin/find.png"/>
  <description>файл</description>
</button>
```

Кнопка может содержать выпадающее меню:

```
<button id=“find-button" label="Найти"
type="menu">
  <menupopup>
    <menuitem label="Файл"/>
    <menuitem label="Картинку"/>
    <menuitem label="Друга"/>
  </menupopup>
</button>
```



Box-модель.

Основой всех элементов является элемент <box>. Все прочие элементы являются лишь частными случаями этого.

<vbox> эквивалентно <box orient="vertical">
<hbox> эквивалентно <box orient="horizontal">.

Размер элемента обычно определяется внутренним содержанием. Дополнительно можно указать размеры в атрибутах (только в точках) или с помощью привязки файла стилей на языке CSS.

width="300" – задание в виде атрибута элемента (в CSS – width:300px;)

height="100" – задание в виде атрибута элемента (в CSS – height:100px;)

minwidth="10" – задание в виде атрибута элемента (в CSS – min-width:10px;)

maxheight="80" – задание в виде атрибута элемента (в CSS – max-height:80px;)

Размещение элементов внутри box'а.

Кроме размера можно управлять размещением элементов внутри box'а, если он сам гибкий, а его внутренние элементы – нет. Это делается с помощью атрибутов box'а `pack` и `align`.

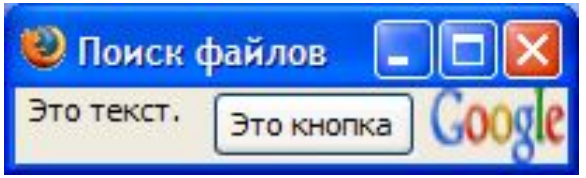
Возможные значения атрибута `pack`:

`start, center, end`

Возможные значения атрибута `align`:

`start, center, end, baseline, stretch`

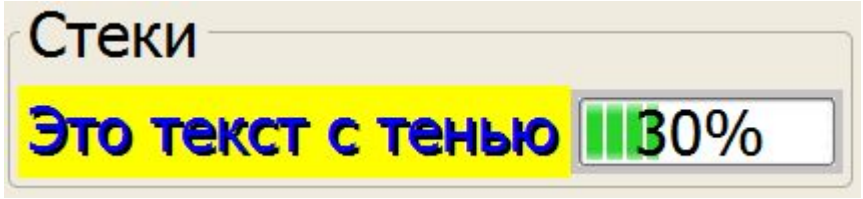
```
<hbox flex="1" pack="" align="">
  <description>
    Это текст.
  </description>
  <button label="Это кнопка"/>
  <image src="chrome://findfiles/skin/google.png"/>
</hbox>
```



Stack и Deck – специальные контейнеры.

Обычно элементы внутри контейнеров располагаются в ряд (по вертикали или по горизонтали). Можно их наложить друг на друга. Для этого используются специальные контейнеры – `Stack` и `Deck`.

Элементы стека располагаются один поверх другого, причем все они растягиваются так, чтобы иметь размер максимального из них.

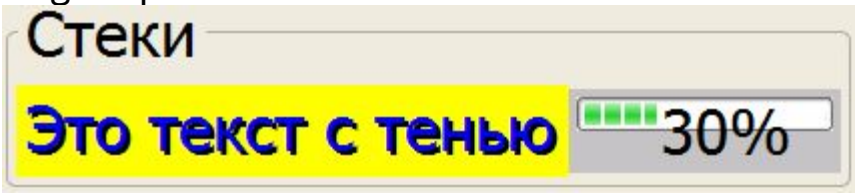



```
<groupbox orient="horizontal" align="center">
   <caption label="Стеки"/>
   <stack style="background-color: yellow;">
     <description style="padding-top: 1px; padding-left: 1px;">
        Это текст с тенью</description>
     <description style="color: blue;">Это текст с
тенью</description>
   </stack>
   <stack style="background-color: silver;" orient="horizontal">
     <progressmeter value="30%" style="margin: 4px;"/>
     <label value="30%" style="margin-left: 1em;"/>
   </stack>
 </groupbox>
```

Stack и Deck – специальные контейнеры.

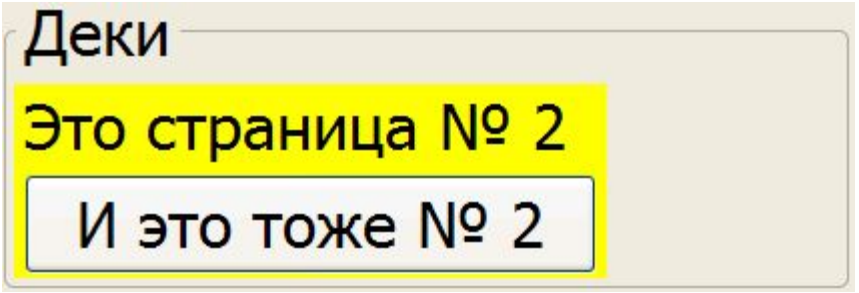
Элементы внутри стека можно и не растягивать, если для каждого из них задать позицию внутри контейнера. Например, текст с тенью и шкалу прогресса из предыдущего слайда можно оформить и по-другому.

```
 <groupbox orient="horizontal" align="center">
    <caption label="Стеки"/>
    <stack style="background-color: yellow;">
      <description left="1" top="1">Это текст с тенью</description>
      <description style="color: blue;">Это текст с
тенью</description>
    </stack>
    <stack style="background-color: silver;" orient="horizontal">
      <progressmeter value="30%" style="margin: 4px;" top="0"/>
      <label value="30%" left="40"/>
    </stack>
  </groupbox>
```



Stack и Deck – специальные контейнеры.

Элементы дека располагаются все в одном и том же месте, причем виден всегда только один из них.




```
  <groupbox>
    <caption label="Деки"/>
    <deck selectedIndex="2" style="background-color:
yellow;">
      <description value="Это страница № 0"/>
      <button label="Это страница № 1"/>
      <vbox>
        <description value="Это страница № 2"/>
        <button label="И это тоже № 2"/>
      </vbox>
    </deck>
  </groupbox>
```

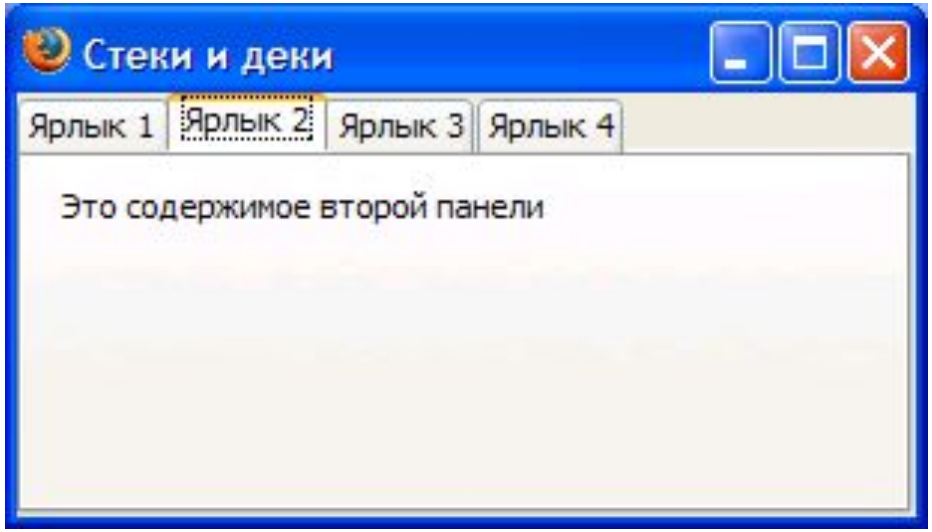
Значение атрибута `selectedIndex` можно менять динамически из программ на *JavaScript*, чтобы в разное время показывать разное содержание. Например, менять содержимое панели с ярлыками.

Панели с ярлыками.

Имеется несколько взаимосвязанных элементов, образующих все вместе такой важный элемент управления, как «панель с ярлыками». Это:

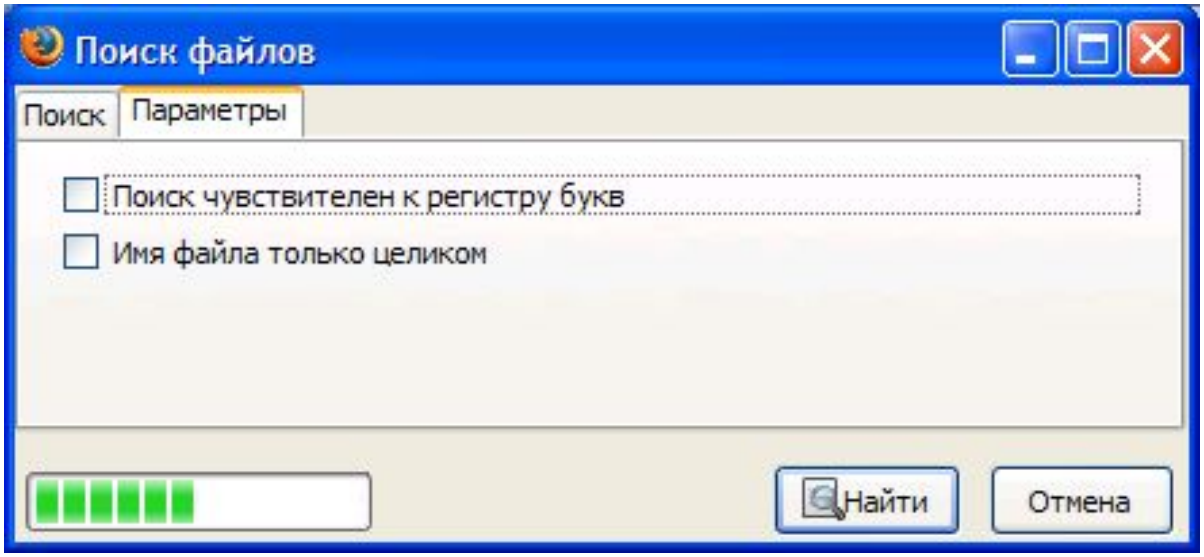
`tabbox, tabs, tab, tabpanels, tabpanel`

```
<tabbox width="400">
   <tabs>
     <tab label="Ярлык 1"/>
     <tab label="Ярлык 2"/>
     <tab label="Ярлык 3"/>
     <tab label="Ярлык 4"/>
   </tabs>
   <tabpanels height="100">
     <tabpanel>...</tabpanel>
     <tabpanel>...</tabpanel>
     <tabpanel>...</tabpanel>
     <tabpanel>...</tabpanel>
   </tabpanels>
 </tabbox>
```



Панели с ярлыками.

Введем панели в наш диалог поиска файлов:



Вторая панель диалога выглядит так:

```
<tabpanel id="optionspanel" orient="vertical">
    <checkbox id="casecheck"
              label="Поиск чувствителен к регистру букв"/>
    <checkbox id="wordscheck"
              label="Имя файла только целиком"/>
</tabpanel>
```