Introduction to Firefox Extensions
Section 2: Adding Visible Browser Elements

CS493 – University of Virginia

The full guide will lead to the creation of an extension that lets a user e-mail text that has been highlighted on a webpage. In the last section, we set up the skeleton code for an extension. We will now further develop the overlay so that the extension is added to the browser's right-click and Tools menus.

Return to the code for `example.xul`. We will now insert the code for the right click menu. Beneath the old status bar code (but still within the `<overlay>` tags), add:

```
<popup id="contentAreaContextMenu">
    <menuitem id="helloworld" label="Example E-mail" accesskey="E"
insertafter="context-stop" oncommand="alert('hello');" />
</popup>
```

The popup ID `contentAreaContextMenu` is not unique to this code segment. This is the same ID that the browser overlay uses for the normal right-click menu. When `example.xul` is loaded, the two definitions are merged together. The `menuitem` should, however, receive an ID of your own conception. Regarding the `menuitem` fields: the `label` field is the text that will appear in the menu; `insertafter` specifies where the label should be inserted into the menu (consult the XUL documentation to see the alternatives); and `oncommand` says what should occur when the label is clicked. With this code, a standard alert dialog that says "Hello" will pop up.

After that, we will insert the code for the Tools menu. After the right click menu code, add:

```
<menupopup id="menu_ToolsPopup">
    <menuitem insertafter="devToolsSeparator" label="Example E-mail"
accesskey="E" oncommand="alert('hello');" />
</menupopup>
```

This code works similarly to the right click menu. However, if the JavaScript were longer than simple alert code, this style of scripting would be redundant and unwieldy. To solve this problem, we can remove the actual JavaScript actions from inside the `menuitem` to a separate JavaScript code section. Like in normal XHTML documents, JavaScript functions can be defined in `<script>` tags within the overlay. To demonstrate this, change the `alert('hello');` code to `pop();`. This new function `pop()` will contain the actual JavaScript code. Place `pop()`'s function definition inside `<script>` tags like:

```
<script type="application/x-javascript">
    function pop() { alert('hello'); }
</script>
```

At the end of this section, the extension can be accessed from the Tools or right click menu. Choosing the extension from the menu will generate a "hello" alert box. Your `example.xul` file at this stage should look like:

```
<?xml version="1.0"?>
```

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

    <statusbar id="status-bar">
        <statusbarpanel id="my-panel" label="Example Application" />
    </statusbar>

    <popup id="contentAreaContextMenu">
        <menuitem id="helloworld" label="Example E-mail" accesskey="E"
insertafter="context-stop" oncommand="pop();" />
    </popup>

    <menupopup id="menu_ToolsPopup">
        <menuitem insertafter="devToolsSeparator" label="Example E-mail"
accesskey="E" oncommand="pop();" />
    </menupopup>

    <script type="application/x-javascript">
        function pop() { alert('hello'); }
    </script>

</overlay>
```