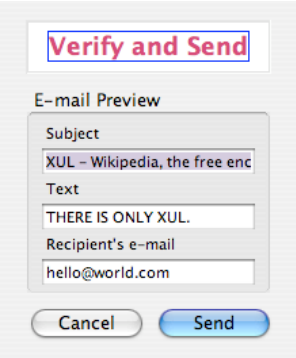
Introduction to Firefox Extensions
Section 4: Accessing the DOM
CS493 – University of Virginia

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 made a pop-up dialog appear. We will now access the Document Object Model (DOM) to pre-seed the dialog's fields.



The subject field will be initialized to the title of the active browser page, and the text field (the body of the e-mail) will be initialized to any text that the user has highlighted on the page. These values will be fetched from the DOM and passed from `example.xul` to the dialog box as arguments in the dialog's constructor. The JavaScript in `menu.js` will fetch the arguments and dynamically set the Subject and Text values.

The Document Object Model represents the structure and layout of an XHTML page. JavaScript sees HTML documents in terms of their DOM structures, and dynamic page changes to a page are done with DOM references. HTML elements can be accessed directly by ID (e.g., `<div id="foo">`); indirectly by relationship (e.g., parent); or by a standardized name (e.g., `document.title` will always refer to a page's title without the HTML programmer assigning an ID).

We need to access the DOM in `example.xul` to do two things: to get the page title (for the Subject field) and to get the text that the user has highlighted and wants to send. These will be input arguments to the dialog box. In `example.xul`, the `pop()` function will now be:

```
function pop() {
        window.openDialog(
            'chrome://example/content/menubox.xul',
            'showmore',
            'chrome',
            window.content.getSelection().toString(),
            document.title);
}
```

The first argument is `window.content.getSelect().toString()`, which gets the highlighted text from the page and converts it to a string. The second argument is `document.title`. The JavaScript in `menu.js` will now retrieve and make use of these arguments. Replace the contents of `menu.js` with:

```
1   var txt = window.arguments[0];
2   var sub = window.arguments[1];
3
4   document.getElementById('user_txt').value = txt;
5   document.getElementById('user_sub').value = sub;
6
7   function onSend() {
8       alert("Pretend like this e-mail was sent");
9       return true;
```

```
10  }
11
12  function onCancel() {
13      alert("Guess you changed your mind");
14      return true;
15  }
```

Lines 1 and 2 get the highlighted text string and document title passed in from `example.xul`. Lines 4 and 5 dynamically set the `value` fields of `user_txt` and `user_sub` in `menubox.xul`:

```
<textbox id="user_sub" />
...
<textbox id="user_txt" />
```

These two textboxes will now display the highlighted text string and document title as their default values for the user to approve and edit before clicking "Send" or "Cancel."

If this extension were to be completed, the `onSend()` function would need to be edited to actually send an e-mail. However, that is beyond the scope of this tutorial on general Firefox extension skills. Instead, the next section will explain how to package an extension in an installable format for normal distribution to users.