selector.js

/\*

A visual selector builder script to be inserted into arbitrary pages. Once the

page loads, a set of controls is inserted into the page and these allow the

user to visually select an element, and the script generates a selector behind

the scenes. Once the user accepts their selection, a message is broadcasted

to the extension with the selector and the URL of the page in question and the tab is closed.

\*/

// Constants.

var OUTLINE\_CLASS = 'chrome\_page\_monitor\_outline';

var TEMP\_OUTLINE\_CLASS = 'chrome\_page\_monitor\_temp\_outline';

var ACTIVE\_CLASS = 'chrome\_page\_monitor\_active';

var DISABLED\_CLASS = 'chrome\_page\_monitor\_disabled';

var FRAME\_ID = 'chrome\_page\_monitor\_selector';

// Picking state.

var current\_element = null;

var current\_selector = '';

var pick\_mode = true;

// References to the controls.

var frame = null;

var pick\_button = null;

var parent\_button = null;

var done\_button = null;

var help\_button = null;

// Generates the controls that are inserted into the page. Returns a wrapped

// jQuery set that's ready to be used with appendTo and the like.

function generateControls() {

var template = '<div id="' + FRAME\_ID + '"> ' +

' <span title="Pick" class="' + ACTIVE\_CLASS + '">%pick%</span> ' +

' <span title="Parent" class="' + DISABLED\_CLASS + '">%parent%</span> ' +

' <input type="button" title="Done" value="%done%" disabled="disabled" /> ' +

' <input type="button" title="Help" value="%help%" /> ' +

'</div>';

var pick = chrome.i18n.getMessage('selector\_gui\_pick');

var parent = chrome.i18n.getMessage('selector\_gui\_parent');

var done = chrome.i18n.getMessage('selector\_gui\_done');

var help = chrome.i18n.getMessage('selector\_gui\_help');

var controls = template.replace('%pick%', pick)

.replace('%parent%', parent)

.replace('%done%', done)

.replace('%help%', help);

return $(controls);

}

// Updates current\_selector, the visual outline and the state of various buttons

// depending on the value of current\_element.

function currentElementChanged() {

$('\*').removeClass(TEMP\_OUTLINE\_CLASS).removeClass(OUTLINE\_CLASS);

done\_button.attr('disabled', !current\_element);

if (current\_element) {

$(current\_element).addClass(OUTLINE\_CLASS);

parent\_button.removeClass(DISABLED\_CLASS);

current\_selector = elementToSelector(current\_element);

} else {

parent\_button.addClass(DISABLED\_CLASS);

current\_selector = '';

}

}

// Takes an element and walks up its hierarchy constructing a selector which

// would match this element (and hopefully it alone). Stops as soon as it

// reaches an element with a defined ID attribute or when reaching the <body>.

// Ignores classes starting with chrome\_page\_monitor\_ (e.g. the outline class).

// Elements outside of <body> return null.

function elementToSelector(element) {

var path = [];

element = $(element);

if (element.is('body')) {

return 'body';

} else if (element.closest('body').length == 0) {

return null;

} else {

while (!(element.is('body') || element.attr('id'))) {

var tagname = element.get(0).tagName.toLowerCase();

var classname = element.get(0).className;

classname = classname.replace(/chrome\_page\_monitor\_\w+/g, '')

.replace(/^\s+|\s+$/g, '')

.replace(/\s+/g, '.');

var selector = classname ? (tagname + '.' + classname) : tagname;

if (element.siblings(selector).length > 0) {

selector += ':nth-child(' + (element.index() + 1) + ')';

}

path.push(selector);

element = element.parent();

}

if (element.attr('id')) {

path.push('#' + element.attr('id'));

} else {

path.push('');

}

path.reverse();

return path.join('>');

}

}

// Sets up the mousemove and click handlers for the <body> to highlight the

// element currently being hovered on with the chrome\_page\_monitor\_temp\_outline

// class and the selected one with chrome\_page\_monitor\_active. Also sets

// current\_element if one is clicked in pick mode, deactivates the pick button

// by removing its chrome\_page\_monitor\_active class and calls

// currentElementChanged() to update the selection. Elements inside the control

// block are ignored during selection.

function setUpBodyHandlers() {

$('body').mousemove(function(e) {

if (pick\_mode) {

$('\*').removeClass(TEMP\_OUTLINE\_CLASS);

$(e.target).addClass(TEMP\_OUTLINE\_CLASS);

}

});

$('body').click(function(e) {

if (pick\_mode) {

var element = e.target;

if (!($(element).is('body') ||

$(element).closest('#' + FRAME\_ID).length)) {

current\_element = element;

currentElementChanged();

pick\_mode = false;

pick\_button.removeClass(ACTIVE\_CLASS);

}

return false;

}

});

}

// Sets up the button handlers:

// 1. The pick button turns on pick mode and discards the current selection.

// 2. The parent button replaces the selection with its parent.

// 3. The done button sends the current selector and URL back to the extension

// and closes the window when a reply is received.

// 4. The help button display an instructions message.

function setUpButtonHandlers() {

pick\_button.click(function() {

pick\_mode = true;

current\_element = null;

currentElementChanged();

$(this).addClass(ACTIVE\_CLASS);

});

parent\_button.click(function() {

if (!$(this).hasClass(DISABLED\_CLASS) && current\_element) {

var parent = $(current\_element).parent();

if (parent.is('body')) {

parent\_button.addClass(DISABLED\_CLASS);

} else {

current\_element = parent.get(0);

currentElementChanged();

}

}

});

done\_button.click(function() {

if (current\_selector) {

chrome.extension.sendRequest({

selector: current\_selector,

url: window.location.href

}, window.close);

} else {

window.close();

}

});

help\_button.click(function() {

alert(chrome.i18n.getMessage('selector\_gui\_help\_text'));

});

}

// The main function. Inserts the controls, updates the global references to

// them, then sets up event handlers for everything by calling

// setUpBodyHandlers() and setUpButtonHandlers().

function initialize() {

generateControls().appendTo('body');

frame = $('#' + FRAME\_ID);

pick\_button = $('span[title=Pick]', frame);

parent\_button = $('span[title=Parent]', frame);

done\_button = $('input[type=button][title=Done]', frame);

help\_button = $('input[type=button][title=Help]', frame);

setUpButtonHandlers();

setUpBodyHandlers();

}