SSP files contain a reference to site.js.

Site.js manages three views: Cart, Login/Register and Checkout.

When site.js (startup function) loads, it checks which view it’s on. This information is set by the SSP on the page as a JavaScript object: N.search.config.view.

Then the appropriate controller is started.

Models

Models talk to the services and are shared by the 3 controllers.

All models subclass NSDataModel.

All load, save, get functions are asynchronous, therefore you need to pass success and failure callbacks to the function.

Example:NSCreditCardModel

\_nsCreditCardModel.load(mp);

load: function (mp) {

var model = this;

var serviceParams = {

method: "getAll" // method passed to the service (.SS files)

};

this.comm(serviceParams, mp, {

success: function (result) {

// update the model with the results that came back from the successful service call

model.creditCards = result.creditcards;

}

});

}

mp can consist of args and callbacks.

args is a map of parameters needed by the function.

callbacks consists of optional success and failure arrays. This allows the one who is invoking the model to specify which function to run once the model was either successful or failed.

Calling the model:

\_nsCreditCardModel.save({

args: {

input: input

},

callbacks: {

success: saveCreditCardCallback.bind(\_ctx),

failure: saveCreditCardFailureCallback.bind(\_ctx)

}

});

If the save is successful saveCreditCardCallback will be called, otherwise in the failure case saveCreditCardFailureCallback will be called. You don’t need to do your own error checking, it is all taken care of in the NSDataModel (see comm function).

Controllers

Controllers invoke functions on the models. There are three controllers:

\_cartController

\_loginController

\_checkoutController

Each controller has a startup function for initialization code.

Observing Events

1. In site.js add the event handling code

BDK.observe(eventName, eventHandler, context, goFirst, debugString);

Example:

BDK.observe("emailCart", showEmailCartBridge, this, false, this.name + ":emailCartBridge");

2. In the template or other parts of site.js you can fire the event (e.g. "emailCart")

Example:

<button class="button2" name="sendButton" onclick='BDK.fire("emailCartToCustomer");return false;'>Send</button>

function showEmailCartBridge(evt)

The evt parameter contains the parameters which were passed from the template or other code that fired the event.

Examples:

1.

var id = "100";

BDK.fire("removeAddress");

function myRemoveAddress(evt) {

var id = evt.memo; // evt.memo contains the parameter

2.

var complex = { id:"100", type:"shipping"};

BDK.fire("removeAddress");

function myRemoveAddress(evt) {

var id = evt.memo.id;

var type = evt.memo.type;

Customizing Checkout

Checkout Code (see \_checkoutController)

\_checkoutController defines the steps in your checkout.

\_steps are an array of id and templates (defined in the File Cabinet with a proper tag (e.g. checkout\_tmpl\_shipping))

Reordering the array items, reorders the checkout flow

\_dataLoadingFuncs

Each step has a function for loading all the data it needs

More on loading data:

load is a utility function

load(modelToLoad, {

renderFunc: renderFuncObj.shippingMethods

/\* shippingMethods is the function which we want called for rendering once the data has loaded asynchronously \*/

});

Example:

load(\_nsShippingModel, {

renderFunc: renderFuncObj.shippingMethods

});

\_renderFuncs

Each step defines how to render its step's view and its summary view

Rendering templates:

Associate a template with a target element

BDK.templates.tmpl(template\_tag, optional\_target);

Examples:

checkoutTmpl: BDK.templates.tmpl("checkout\_tmpl", "main"),

addressesTmpl: BDK.templates.tmpl("checkout\_tmpl\_addresses", "shippingAddressesDiv");

Now when you call render on addressesTmpl, the target (shippingAddressesDiv) is updated with the results of binding model.addresses to the checkout\_tmpl\_addresses template.

addressesTmpl.render({addresses:model.addresses});

If the target is not specified, you can still call render.

Example:

var content = addressesTmpl.render({addresses:model.addresses});

$("someDiv").update(content);

Calling code from templates:

See \_templateHelpers in site.js.

These functions are accessible from the templates.

Example:

<b>

{if \_m.isAddressSelected(address, type)}<h1>Selected</h1>

</b>

Validation

Once a form is submitted or a button is clicked, form validation can take place.

Example:

var formId = "guestCheckoutEmailForm”;

var context = \_ctx;

var callback = saveGuestEmail; /\* call this function if the form validation was successful \*/

validateForm(formId, context, callback);

Error Handling

reportError(“Invalid Credit Card Number”);

function reportError(msg) {

var errorArea = $("checkoutError"); /\* checkoutError is a div above the current step the shopper is on \*/

if(errorArea) {

errorArea.update(msg);

}

}

Google Analytics

See \_analytics object. For checkout, every time the shopper moves between steps we track their URLs.

\_analytics.trackUrl("/checkout/" + \_currentStep.id);

In the Google Analytics admin, the above code would get logged as the following URLs /checkout/shipping or /checkout/billing depending on what checkout step the shopper was on.