```
stop();
//
_global.CatalogData = function() {
    // do nothing now. just be a container
};
CatalogData.prototype.parseXML = function(CatalogRootNode) {
    if (CatalogRootNode.nodeName == "CatalogData") {
        // this is the correct starting node.
        var tempArray = CatalogRootNode.childNodes;
        for (var j = 0; j<tempArray.length; j++) {
            switch (tempArray[j].nodeName) {
            case "defaults" :
                //trace("case defaults");
                this.defaults = new ItemCollection();
                this.defaults.parseXML(tempArray[j]);
                break;
            case "thumbs" :
                //trace("case thumbs");
                this.thumbs = new ItemCollection();
                this.thumbs.parseXML(tempArray[j]);
                break;
            case "catalog" :
                //trace("case catalog");
                this.catalog = new ItemCollection();
                this.catalog.parseXML(tempArray[j]);
                break;
            default :
                // do nothing. some unexpected xml node name
                trace('unexpected xml node tempArray[j].nodeName='+tempArray[j].nodeName);
                break;
            }
        }
    } else {
        // do nothing
        trace('did not get the correct root node for the xml');
    }
};
_global.ItemCollection = function() {
    // do nothing now. fill later
    this.item = new Array();
};
ItemCollection.prototype.parseXML = function(itemCollectionNode) {
    var tempArray = itemCollectionNode.childNodes;
    for (var j = 0; j<tempArray.length; j++) {
        switch (tempArray[j].nodeName) {
        case "item" :
            //trace("case item");
            var nextItem = this.item.length;
            this.item[nextItem] = new Item();
            this.item[nextItem].parseXML(tempArray[j]);
            break;
        default :
            // do nothing. some unexpected xml node name
            trace('unexpected xml node tempArray[j].nodeName='+tempArray[j].nodeName);
            break;
        }
    }
};
```

```
_global.Item = function() {
    // do nothing now. fill later
};
Item.prototype.parseXML = function(itemNode) {
    var tempArray = itemNode.childNodes;
    for (var j = 0; j<tempArray.length; j++) {
        switch (tempArray[j].nodeName) {
        case "sku" :
            //trace("case sku");
            this.sku = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "price" :
            this.price = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "style" :
            this.style = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "color" :
            this.color = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "desc" :
            this.desc = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "memo" :
            this.memo = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "stylePhoto" :
            this.stylePhoto = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "itemPhoto" :
            this.itemPhoto = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "thumbPhoto" :
            this.thumbPhoto = unescape(tempArray[j].firstChild.nodeValue);
            break;
        case "headLine" :
            this.headLine = unescape(tempArray[j].firstChild.nodeValue);
            break;
        default :
            // do nothing. some unexpected xml node name
            trace('unexpected xml node tempArray[j].nodeName='+tempArray[j].nodeName);
            break;
        }
    }
};
Item.prototype.fillWithDefaults = function(itemObject) {
    if (!this.sku) {
        this.sku = itemObject.sku;
    }
    if (!this.headLine) {
        this.headLine = itemObject.headLine;
    }
    if (!this.price) {
        this.price = itemObject.price;
    }
    if (!this.style) {
        this.style = itemObject.style;
    }
```

```
    if (!this.color) {
        this.color = itemObject.color;
    }
    if (!this.desc) {
        this.desc = itemObject.desc;
    }
    if (!this.memo) {
        this.memo = itemObject.memo;
    }
    if (!this.stylePhoto) {
        this.stylePhoto = "inventry/"+this.sku+"s.jpg";
    }
    if (!this.itemPhoto) {
        this.itemPhoto = "inventry/"+this.sku+"f.jpg";
    }
    if (!this.thumbPhoto) {
        this.thumbPhoto = "inventry/"+this.sku+"t.jpg";
    }
};
function compareOrder(a, b) {
    return a.order-b.order;
}
function replace(originalString, stringToReplace, replacementString) {
    if (originalString.indexOf(stringToReplace) != -1) {
        var firstPart = originalString.substring(0, originalString.indexOf(stringToReplace));
        firstPart = firstPart.concat(replacementString);
        secondPart = originalString.substring(originalString.indexOf(stringToReplace)+stringToRepla
gth);
        var fixedString = firstPart.concat(secondPart);
        fixedString = replace(fixedString, stringToReplace, replacementString);
        return fixedString;
    } else {
        return originalString;
        // no cleaning needed
    }
}
function trimString(originalString, maxlength) {
    if (originalString.length>20) {
        return originalString.substring(0, maxlength);
    } else {
        return originalString;
    }
}
function removehtml(originalString) {
    if (originalString.indexOf("<") != -1) {
        var firstPart = originalString.substring(0, originalString.indexOf("<"));
        var secondPart = originalString.substring(originalString.indexOf(">")+1);
        var finalString = firstPart.concat(secondPart);
        finalString = removehtml(finalString);
        return finalString;
    } else {
        return originalString;
    }
}
function clean(originalString) {
    originalString = removehtml(originalString);
    //originalString = replace(originalString, "%", " percent");
    originalString = replace(originalString, "?", " ");
```

```
    //originalString = replace(originalString, "w/", "with ");
    //originalString = replace(originalString, "/", " or ");
    //originalString = replace(originalString, ":", " is ");
    //originalString = replace(originalString, ".", " ");
    originalString = replace(originalString, "  ", " ");
    //originalString = replace(originalString, ".", " ");
    //originalString = trimString(originalString, 25);
    return originalString;
}
function init() {
    myData = new CatalogData();
    myData.parseXML(myXML.firstChild);
    // try to get sku from URL Parameter below
    if (sku == null || sku == undefined) {
        // no parameter was provided in the url.
        if (myData.catalog.item.length > 0){
            //use the first sku.
            sku = myData.catalog.item[0].sku;
        } else {
            // there is no first sku then go to the home page.
            getURL("http://www.ekologic.com/index.html?tab=shop", "_self");
            }
    }
    // find index number in the xml catalog of sku specified in url
    currentCatalogIndex = "none";
    for (var i = 0; i<myData.catalog.item.length; i++) {
        if (myData.catalog.item[i].sku == sku) {
            currentCatalogIndex = i;
            trace("myData.catalog.item[i].sku="+myData.catalog.item[i].sku);
        }
    }
    if (currentCatalogIndex == "none") {
        // the sku did not match an item in the xml.
        // most likely it has been sold and removed from inventory. but someone bookmarked it wehn
 avaialable and ahs come back.
        // best choice is to go to a thumbnails page with note that sku was not found or sold.
        getURL("thumbs.html?page=sold", "_self");
    }
    // decide if we show the  next and previous arrows and what they link to.
    arrowsToShow = "neither";
    if (0<currentCatalogIndex<myData.catalog.item.length-2) {
        arrowsToShow = "both";
        nextSku = myData.catalog.item[currentCatalogIndex+1].sku;
        prevSku = myData.catalog.item[currentCatalogIndex-1].sku;
    }
    if (0 == currentCatalogIndex && myData.catalog.item.length>1) {
        arrowsToShow = "next";
        nextSku = myData.catalog.item[currentCatalogIndex+1].sku;
    }
    if (currentCatalogIndex == myData.catalog.item.length-1 && myData.catalog.item.length>1) {
        arrowsToShow = "previous";
        prevSku = myData.catalog.item[currentCatalogIndex-1].sku;
    }
    // fill in data for current item. use defaults or explicitly stated values.
    myItem = myData.catalog.item[currentCatalogIndex];
    myItem.fillWithDefaults(myData.defaults.item[0]);
    currentSku = sku;
    currentSkuText = "sku: "+myItem.sku;
```

```
    currentPhoto = "inventry/"+currentSku+"f.jpg";
    currentStylePhoto = "inventry/"+currentSku+"s.jpg";
    currentPrice = "$"+myItem.price;
    currentStyle = myItem.style;
    currentColor = myItem.color;
    currentDesc = myItem.desc;
    currentMemo = myItem.memo;
    currentHeadLine = myItem.headLine;
    itemNameForPayPal = escape(clean(currentStyle+': '+currentColor));
    addToCartCode = 'https://www.paypal.com/cgi-bin/webscr?cmd=_cart&add=1&business=info@ekologic.c
item_name='+itemNameForPayPal+'&item_number='+currentSkuText+'&amount='+currentPrice+'&on0=disclaim
er-color may vary &os0=due to monitor differences&on1=refund policy-return in&os1=10 days for credi
t or exchange&page_style=ekologic&return=http://www.ekologic.com/catalog/sold.html&cancel_return=ht
tp://www.ekologic.com/catalog/cancel.html&cn=AdditionalInstructions&currency_code=USD';
    //info for thumbnails page below
    thumbsPerPage = 15;
    currentPageIndex = Math.floor(currentCatalogIndex/thumbsPerPage);
    // page 0 is the first page.
    firstThumbIndexForPage = currentPageIndex*thumbsPerPage;
    pagesLength = Math.floor((myData.catalog.item.length-1)/thumbsPerPage)+1;
    PageArrowsToShow = "neither";
    if (0<currentPageIndex && currentPageIndex<pagesLength-1) {
        PageArrowsToShow = "both";
        nextPage = myData.catalog.item[(currentPageIndex+1)*thumbsPerPage].sku;
        prevPage = myData.catalog.item[currentPageIndex*thumbsPerPage-1].sku;
    }
    if (0 == currentPageIndex && pagesLength>1) {
        PageArrowsToShow = "next";
        nextPage = myData.catalog.item[(currentPageIndex+1)*thumbsPerPage].sku;
    }
    if (currentPageIndex+1 == pagesLength && pagesLength>1) {
        PageArrowsToShow = "previous";
        prevPage = myData.catalog.item[currentPageIndex*thumbsPerPage-1].sku;
    }
    currentPageNumberText = "page "+Number(currentPageIndex+1)+" of "+Number(pagesLength);
    xmlthumbIndex = (currentPageIndex)%(myData.thumbs.item.length);
    thumbStyle = myData.thumbs.item[xmlthumbIndex].style;
    thumbDesc = "<P ALIGN='RIGHT'><B>"+myData.thumbs.item[xmlthumbIndex].desc+"</B></P>";
    thumbHeadLine = myData.thumbs.item[xmlthumbIndex].headLine;
    thumbStylePhoto = myData.thumbs.item[xmlthumbIndex].stylePhoto;
    thumbMemo = myData.thumbs.item[xmlthumbIndex].memo;
    play();
}
// end of functions. start sequential code
var reloadHtmlOnTabClick = true;
var catalog, currentCatalogIndex, currentSku, currentSkuText, nextSku, prevSku, currentHeadLine, cu
rrentStylePhoto, currentPhoto, currentPrice, currentStyle, currentColor, currentColorForMouseOver,
itemNameForPayPal, currentDesc, currentMemo, addToCartCode, arrowsToShow;
var thumbsPerPage, currentPageIndex, pagesLength, firstThumbIndexForPage, PageArrowsToShow, prevPag
e, nextPage, currentPageNumberText, thumbStyle, thumbDesc, thumbHeadLine, thumbStylePhoto, thumbMem
o;
myXML = new XML();
myXML.ignoreWhite = true;
myXML.onLoad = init;
// this onload function populates the catalog array and reasigns all variables using the xml data a
nd the init function defined above.
myXML.load("inventry/edittext.xml");
// test to make sure file was found and loaded. if not jump to home page.
```