```
//create a new loader for loading in the images
var myLoader:Loader = new Loader();

/*
    This is an array of objects. Currently, there are three objects in the array, denoted by the { } a
    and the , seperating each array entry.

    Within the objects we have defined two members: name and url. These are both text based, but you c
    such as numbers, movieclips, or even other objects.

    An array of objects is immensely useful for when we need an array of several linked pieces of data
    allows you to keep them all together in an easily accessible way
*/

var stuffArr:Array = new Array( { name:"My first work", url:"image1.png" },
                                { name:"My second try", url:"image2.png" },
                                { name:"My last attempt", url:"image3.png" } );

//loading clip
var loading = new loadingClip();
//position the clip and make it invisible
loading.x = 100;
loading.y = 150;
loading.visible = false;
//add it to the stage
this.addChild( loading );

//this is a for-each loop that iterates through each item in the stuffArr array
for( var i in stuffArr ) {

    //for each item we create a new button
    var newButton = new nextb();

    /*
        then we create a property on the button equal to the current iteration of this loop
        this correspondes to the the items index in the array.
        we will be using this later on to reference the item from out array when the user clicks on th
    */
    newButton.num = i;
    //then we position the button, starting at 0, and moving down 70 pixels each iteration
    newButton.y = 70 * i;
    //add an event listener for this particular button
    newButton.addEventListener( MouseEvent.CLICK, showPicture );
    //finally we add it to the stage
    btns.addChild( newButton );

}

//called when the user clicks on a button
function showPicture( e:MouseEvent ):void {

    //create a new loader
    myLoader = new Loader();

    /*
        Create a new URL request using the url stored in our stuff array.
        The index of the array that stores the URL is referenced by the number property that
        we created on the target button in the imageMain function
    */
    var myURL:URLRequest = new URLRequest( stuffArr[ e.target.num ].url );
    //add an event for when the content is loaded
    myLoader.contentLoaderInfo.addEventListener( Event.COMPLETE, addPicture );
    //add event handler for while the content is still loading
    myLoader.contentLoaderInfo.addEventListener( ProgressEvent.PROGRESS, progressHandler );

    //create a variable to grab any image that is currently on the stage
    var toGo = this.getChildByName("currentGraphic");

    //if our variable has something in it
    if( toGo != null ) {
        //remove it from stage
        this.removeChild( toGo );
        //set it to be removed via actionscript
        toGo = null;
    }

    //have our loading bar be visible now
    loading.visible = true;

    //set the name of our image equal to the name stored in our stuff array
    //the same as how we got the url of our image
    txtName.text = stuffArr[ e.target.num ].name;

    //begin the load of the image
    myLoader.load( myURL );
}

//this is a standard progress handler for a loader bar
function progressHandler( e:ProgressEvent ):void {
    //get the percentage loaded
    var progressAmount:Number = e.target.bytesLoaded/e.target.bytesTotal;
    //set the width of the loading bar equal to our percentage loaded
    loading.lbar.scaleX = progressAmount;
}

//this function is called when the picture has been fully loaded, and needs to be aded to the screen
function addPicture( e:Event ):void {

    //position where the image will appear on the screen
    myLoader.x = 100;
    myLoader.y = 50;

    //name it so we can get ahold of it later
    myLoader.name = "currentGraphic";

    //make our loader bar invisible
    loading.visible = false;

    //and add the image we loaded to the stage
    this.addChild( myLoader );

}
```