Code

WEBSCRP

612596
University of Portsmouth

admin/add-product.php

```
<!-- BEGIN #products -->
<div id="add-product">
    <h2>Add Product</h2>

    <section id="fields">
        <input type="text" id="single-title" placeholder="* Product Name" />
        <div id="title-message"></div>
        <textarea id="single-content" placeholder="* Product Description"></textarea>
        <div id="content-message"></div>
        <input type="text" id="single-price" placeholder="* Price" required />
        <div id="price-message"></div>
        <input type="text" id="single-sale" placeholder="Sale Price" />
        <div id="sale-message"></div>
        <input type="text" id="single-colour" placeholder="Colour" />

        <div class="thumbnail-container">
            <!-- For uploading a product thumbnail -->
            <input type="file" id="single-thumbnail" />
        </div>

        <div class="product-category">
            <h3>Pick a Category</h3>
            <ul id="single-category-list"></ul>
            <div id="category-message"></div>
        </div>
    </section>

    <div class="single-product-options">
        <select id="single-status">
            <option value="publish" selected>Publish</option>
            <option value="draft">Draft</option>
            <option value="trash">Trash</option>
        </select>

        <a href="product/" id="add-product-button">Add Product</a>
    </div>

    <div class="single-products">
        <div id="values-and-stocks">
            <input type="text" class="single-values" placeholder="Size (optional)" />
            <input type="text" class="single-stocks" placeholder="* Stock" />
            <div id="stock-message"></div>
        </div>

        <a href="" id="add-value-stock">Add another size</a>
    </div>

</div>
<!-- END #products -->
```

admin/products.php

```
<!-- BEGIN #products -->
<div id="products">
    <div class="clearfix">
        <h2>Product Management</h2>
        <a href="add-product" id="add-product-section">Add Product</a>
    </div>
    <div class="filter">
        FILTER BY:
        <a href="products/page=1&filter=publish" class="filter-table">PUBLISH</a> /
        <a href="products/page=1&filter=draft" class="filter-table">DRAFT</a> /
        <a href="products/page=1&filter=trash" class="filter-table">TRASH</a>
    </div>
    <div class="view">
        VIEW: <a href="" id="ten">10</a> / <a href="" id="all">ALL</a>
    </div>
    <ul id="products_list"></ul>
    <ul id="pagination"></ul>
</div>
<!-- END #products -->
```

admin/categories.php

```
<!-- BEGIN #categories -->
<div id="categories">
    <h2>Category Management</h2>

    <section id="cat-fields">
        <h3>Add Category</h3>
        <input type="text" id="cat-name" placeholder="* Category Name" />
        <input type="text" id="cat-menu-order" placeholder="* Menu Order (number)" />
        <div class="clearfix">
            <select id="cat-status">
                <option value="publish" selected>Publish</option>
                <option value="draft">Draft</option>
                <option value="trash">Trash</option>
            </select>
        </div>
        <p class="note"><em>Note. Setting -1 removes it from the menu</em></p>
        <a href="" id="add-cat-button">Add Category</a>
    </section>

    <section class="current-cats">
        <div class="filter">
            FILTER BY:
            <a href="categories/filter=publish" class="filter-table">PUBLISH</a> /
            <a href="categories/filter=draft" class="filter-table">DRAFT</a> /
            <a href="categories/filter=trash" class="filter-table">TRASH</a>
        </div>
        <ul id="category-list"></ul>
    </section>

</div>
<!-- END #categories -->
```

admin/settings.php

```
<!-- BEGIN #settings -->
<div id="settings">
    <h2>Settings</h2>
    <section class="site-information">
        <h3>Set Company Name</h3>
        <input type="text" value="" id="site-name" placeholder="Company Name" />
        <a href="" id="site-name-button">Set Site Name</a>
    </section>

    <section class="thumbnail-container">
        <h3>Set Default Picture</h3>
        <!-- For uploading a product thumbnail -->
        <input type="file" id="default-picture" />
        <div id="single-thumbnail-preview"></div>
        <a href="settings" id="set-default-picture">Set Default Picture</a>
    </section>

    <section class="danger-zone">
        <h3>Danger Zone</h3>
        <a href="" id="reset-database">Reset Database</a>
    </section>
</div>
<!-- END #settings -->
```

index.php

```
<?php
$href = dirname(__FILE__);
$arr = explode("\\", $href);
// If $arr has been split it will be bigger than 1,
// if not, the system will have to split it a different
// way. Mainly for use on OS X/Linux
if(count($arr) == 1)
    $arr = explode("/", $href);
$base = end($arr);
// Sets up the database, if it has not already been done.
// Mainly for new installations.
include('inc/api_config.php'); ?>
<!DOCTYPE html>
<html>
    <head>
        <title><?php echo $base; ?></title>
        <base href="/<?php echo $base; ?>/" />
        <!-- BEGIN meta -->
        <meta charset="utf8" />
        <!-- BEGIN stylesheets -->
        <link rel="stylesheet" type="text/css" href="css/style.css" />
    </head>
    <!-- BEGIN body -->
    <body>
        <div id="loader-container">
            <div id="loader">
                <div class="load-circle" id="load-rotate-01"></div>
                <div class="load-circle" id="load-rotate-02"></div>
                <div class="load-circle" id="load-rotate-03"></div>
                <div class="load-circle" id="load-rotate-04"></div>
                <div class="load-circle" id="load-rotate-05"></div>
                <div class="load-circle" id="load-rotate-06"></div>
                <div class="load-circle" id="load-rotate-07"></div>
                <div class="load-circle" id="load-rotate-08"></div>
            </div>
        </div>
        <!-- BEGIN #jay-z (main wrapper) -->
        <section id="jay-z">

            <!-- BEGIN header -->
            <header>
                <div class="titlebar clearfix">
                    <h2 id="site-name"></h2>
                    <div class="basket">
                        <a href="basket" id="basket-link">
                            <div id="basket-items"></div>
                            <div id="basket-value"></div>
                        </a>
                    </div>
                </div>
                <!-- BEGIN nav -->
                <nav class="clearfix">
                    <ul id="navigation"></ul>
                    <input type="text" id="search-products" placeholder="Search Products" />
                <!-- END nav -->
                </nav>
                <div id="global-message"></div>
            <!-- END header -->
            </header>
            <section id="content"></section>
        <!-- END #jay-z -->
        </section>
        <!-- BEGIN scripts -->
        <script src="js/functions.js" async></script>
        <script src="js/basket.js" async></script>
        <script src="js/eventHandler.js" async></script>
        <script src="js/load.js" defer></script>
        <!-- END scripts -->
    <!-- END body -->
    </body>
</html>
```

/api/v.1

WEBSCRP

612596
University of Portsmouth

add/product.php

```
<?php

/**
 * The header content.
 */
header("Content-type: application/json");

/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;

if($db || isset($db)) {

    /* Unescaped parameters passed in */
    $unsafe_title = $_POST['title'];
    $unsafe_content = $_POST['content'];
    $unsafe_price = $_POST['price'];
    $unsafe_sale_price = $_POST['sale'];
    $unsafe_colour = $_POST['colour'];
    $unsafe_status = $_POST['status'];
    $unsafe_category_id = $_POST['category_id'];

    $unsafe_values = json_decode($_POST['values']); // Array
    $unsafe_stocks = json_decode($_POST['stocks']); // Array

    /* safe to inject parameters */
    $title = $db->real_escape_string($unsafe_title);
    $content = $db->real_escape_string($unsafe_content);
    $price = $db->real_escape_string($unsafe_price);
    $sale_price = $db->real_escape_string($unsafe_sale_price);
    $colour = $db->real_escape_string($unsafe_colour);
    $status = $db->real_escape_string($unsafe_status);
    $category_id = $db->real_escape_string($unsafe_category_id);

    /* Sale Validation */
    if($sale_price === null)
        $sale_price = '0.00';

    $db->insert("INSERT INTO product_group (title, content, price, sale_price, colour, post_status,
post_date, categoryID) VALUES ('$title', '$content', '$price', '$sale_price', '$colour', '$status',
NOW(), '$category_id')");

    // Get the recently inserted sku number
    $sku = $db->insert_id;

    // Thumbnail checks and validation.
    if(isset($_FILES['thumbnail'])) {
        function findExts($filename)  {
            $filename = strtolower($filename);
            $exts = explode(".", $filename);
            $exts = end($exts);
            return $exts;
        }
        //This applies the function to our file
        $exts = findExts($_FILES['thumbnail']['name']);
        $filename = $sku.'.'.$exts;
        $target = "../../../media/".$filename;

        // Allowed extensions
        $allowedExts = array("gif", "jpeg", "jpg", "png", "pjpeg");
        // Check to make sure file type is picture, and size is < 2mb
        if ((($_FILES["thumbnail"]["type"] == "image/gif") || ($_FILES["thumbnail"]["type"] ==
"image/jpeg")
        || ($_FILES["thumbnail"]["type"] == "image/jpg") || ($_FILES["thumbnail"]["type"] ==
"image/png")
        || ($_FILES["thumbnail"]["type"] == "image/pjpeg")) && ($_FILES["thumbnail"]["size"] <
200000)
        && in_array($exts, $allowedExts)) {
            // Has the file got an error?
            if ($_FILES["thumbnail"]["error"] > 0) {
```

```
                // Give the product default picture.
                $filename = "default.jpg";
            } else {
                // Move uploaded file into media directory.
                move_uploaded_file($_FILES['thumbnail']['tmp_name'], $target);
            }

        } else {
            // Give the product default picture.
            $filename = "default.jpg";
        }
    } else {
        // Give the product default picture.
        $filename = "default.jpg";
    }

    $db->update("UPDATE product_group SET thumbnail='media/$filename' WHERE sku='$sku'");

    /* Add new records for values and stock in */
    for($i=0, $len=count($unsafe_stocks); $i<$len; $i++) {
        /* Make sure the parameters are safe to inject */
        $value = $db->real_escape_string($unsafe_values[$i]);
        $stock = $db->real_escape_string($unsafe_stocks[$i]);
        $db->insert("INSERT INTO product (sku, value, stock) VALUES('$sku','$value','$stock')");
    }

    /* Data returned by the API */
    if($db->error_thrown) {
        $response['error']['thrown'] = true;
        $response['report']['status'] = $db->error_message;
    } else {
        $response['error']['thrown'] = false;
        $response['report']['inserted_id'] = $sku;
        $response['report']['status'] = "Added product successfully";
    }

} else {
    $response['error']['thrown'] = true;
    $response['error']['message'] = "Unable to connect to the database.";
}

echo json_encode($response);
?>
```

add/confirm-order.php

```
<?php

/**
 * The header content.
 */
header("Content-type: application/json");

/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;

if($db || isset($db)) {

    /* The JSON object with the products in */
    $product_ids = json_decode($_POST['id']);
    $product_quantities = json_decode($_POST['quantity']);
    $product_prices = json_decode($_POST['price']);
    $unsafe_customer_name = $_POST['customer_name'];

    $customer_name = $db->real_escape_string($unsafe_customer_name);

    $db->insert("INSERT INTO $db->db_name.order (customer, purchase_date, mail_type, dispatched)
VALUES ('$customer_name', NOW(), 'standard', 'false')");

    // Get the recently inserted sku number
    $order_id = $db->insert_id;

    /* Add new records for values and stock in */
    for($i=0, $len=count($product_ids); $i<$len; $i++) {
        $quantity = $product_quantities[$i];
        $price = $product_prices[$i];
        $product_id = $product_ids[$i];
        $db->insert("INSERT INTO $db->db_name.product_order (ID, quantity, price, productID)
VALUES('$order_id', '$quantity', '$price', '$product_id')");
    }

    $response['error']['thrown'] = false;
    $response['report']['message'] = "Thank you for your purchase!";

} else {
    $response['error']['thrown'] = true;
    $response['error']['message'] = "Unable to connect to the database.";
}

echo json_encode($response);
?>
```

edit/product.php

```
<?php

/**
 * The header content.
 */
header("Content-type: application/json");

/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;

if($db || isset($db)) {

    /* Unescaped parameters passed in */
    $sku = $_POST['sku'];
    $unsafe_title = $_POST['title'];
    $unsafe_content = $_POST['content'];
    $unsafe_price = $_POST['price'];
```

```
    $unsafe_sale_price = $_POST['sale'];
    $unsafe_colour = $_POST['colour'];
    $unsafe_status = $_POST['status'];
    $unsafe_category_id = $_POST['category_id'];

    $unsafe_values = json_decode($_POST['values']); // Array
    $unsafe_stocks = json_decode($_POST['stocks']); // Array

    /* Safe to inject parameters */
    $title = $db->real_escape_string($unsafe_title);
    $content = $db->real_escape_string($unsafe_content);
    $price = $db->real_escape_string($unsafe_price);
    $sale_price = $db->real_escape_string($unsafe_sale_price);
    $colour = $db->real_escape_string($unsafe_colour);
    $status = $db->real_escape_string($unsafe_status);
    $category_id = $db->real_escape_string($unsafe_category_id);

    // Get the extension of a file.
    function findExts($filename)  {
        $filename = strtolower($filename);
        $exts = explode(".", $filename);
        $exts = end($exts);
        return $exts;
    }

    // Thumbnail checks and validation.
    if(isset($_FILES['thumbnail'])) {
        //This applies the function to our file
        $exts = findExts($_FILES['thumbnail']['name']);
        $filename = $sku.'.'.$exts;
        $target = "../../../media/".$filename;

        // Allowed extensions
        $allowedExts = array("gif", "jpeg", "jpg", "png", "pjpeg");
        // Check to make sure file type is picture, and size is < 2mb
        if ((($_FILES["thumbnail"]["type"] == "image/gif") || ($_FILES["thumbnail"]["type"] ==
"image/jpeg")
        || ($_FILES["thumbnail"]["type"] == "image/jpg") || ($_FILES["thumbnail"]["type"] ==
"image/png")
        || ($_FILES["thumbnail"]["type"] == "image/pjpeg")) && ($_FILES["thumbnail"]["size"] <
200000)
        && in_array($exts, $allowedExts)) {
            // Has the file got an error?
            if ($_FILES["thumbnail"]["error"] > 0) {
                // Give the product default picture.
                $filename = "default.jpg";
            } else {
                // Move uploaded file into media directory.
                move_uploaded_file($_FILES['thumbnail']['tmp_name'], $target);
            }
        } else {
            // Give the product default picture.
            $filename = "default.jpg";
        }
    } else {
        // Give the product default picture.
        // Get current thumbnail from database.
        $result = $db->select("SELECT thumbnail FROM product_group WHERE sku='$sku'");
        $result = $result->fetch_assoc();
        // Explode media from the path.
        $old_file = explode("media/", $result['thumbnail']);
        // Get the filename.
        $old_file = $old_file[1];

        if(file_exists('../../../media/'.$old_file)) {
            $filename = $old_file;
        } else {
            $filename = 'default.jpg';
        }
    }

    /* Sale Validation */
    if($sale_price === null)
        $sale_price = '0.00';
```

```
    /* Make the update the the product group */
    $db->update("UPDATE product_group SET title='$title', content='$content', price='$price',
sale_price='$sale_price', colour='$colour', thumbnail='media/$filename', post_status='$status',
post_modified=NOW(), categoryID='$category_id' WHERE sku='$sku'");

    /* Reset product table */
    $db->delete("DELETE FROM product WHERE sku='$sku'");

    /* Add new records for values and stock in */
    for($i=0, $len=count($unsafe_stocks); $i<$len; $i++) {
        /* Make sure the parameters are safe to inject */
        $value = $db->real_escape_string($unsafe_values[$i]);
        $stock = $db->real_escape_string($unsafe_stocks[$i]);
        $db->insert("INSERT INTO product (sku, value, stock) VALUES('$sku','$value','$stock')");
    }

    if($db->error_thrown) {
        $response['error']['thrown'] = true;
        $response['report']['status'] = $db->error_message;
    } else {
        $response['error']['thrown'] = false;
        $response['report']['status'] = "Update successful";
    }

} else {
    $response['error']['thrown'] = true;
    $response['error']['message'] = "Unable to connect to the database.";
}

echo json_encode($response);
?>
```

edit/category-status.php

```
<?php
/**
 * The header content.
 */
header("Content-type: application/json");

/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;

if($db || isset($db)) {
    /* Unescaped parameters passed in */
    $id = $_POST['id'];
    $unsafe_status = $_POST['status'];

    $status = $db->real_escape_string($unsafe_status);

    /* Make the update the the product group */
    $db->update("UPDATE category SET post_status='$status' WHERE id='$id'");

    if($db->error_thrown) {
        $response['error']['thrown'] = true;
        $response['report']['status'] = $db->error_message;
    } else {
        $response['error']['thrown'] = false;
        $response['report']['status'] = "Updated Category Status";
    }
} else {
    $response['error']['thrown'] = true;
    $response['report']['status'] = "Unable to connect to the database.";
}

echo json_encode($response);
?>
```

search/products.php

```
<?php
/**
 * Settings the HTTP header
 */
header("Content-type: application/json");
/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;
if($db || isset($db)) {

    /**
     * Parse all information.
     */
    $unescaped_q = $_GET['q']; // The query.

    /**
     * Escape the query string of any SQL harmful
     * characters.
     */
    $q = $db->real_escape_string($unescaped_q);
    /**
     * Run the query.
     */
    $result = $db->select("SELECT * FROM product_group WHERE post_status = 'publish' AND (title
LIKE '%$q%' OR sku LIKE '$q')");

        /**
         * Loop through all records retrieved from the query.
         */
        $i = 0;
        while($item = $result->fetch_assoc()) {
            $id = $item['sku'];
            $response['product_group'][$i]['sku'] = $item['sku'];
            $response['product_group'][$i]['title'] = $item['title'];
            $response['product_group'][$i]['content'] = $item['content'];
            $response['product_group'][$i]['price'] = $item['price'];
            $response['product_group'][$i]['sale_price'] = $item['sale_price'];
            $response['product_group'][$i]['colour'] = $item['colour'];
            $response['product_group'][$i]['thumbnail'] = $item['thumbnail'];
            $response['product_group'][$i]['post_status'] = $item['post_status'];
            $response['product_group'][$i]['post_date'] = $item['post_date'];
            $response['product_group'][$i]['post_modified'] = $item['post_modified'];
            $response['product_group'][$i]['categoryID'] = $item['categoryID'];
            $products = $db->select("SELECT * FROM product WHERE sku = '$id'");
            $j = 0;
            while($product = $products->fetch_assoc()) {
                $response['product_group'][$i]['product'][$j]['value'] = $product['value'];
                $response['product_group'][$i]['product'][$j]['stock'] = $product['stock'];
                $j++;
            }
            $i++;
        }

    if(empty($response)) {
        $response['error']['thrown'] = true;
        $response['error']['message'] = 'No results found.';
    } else {
        $response['error']['thrown'] = false;
    }

} else {
    $response['error']['thrown'] = true;
    $response['error']['message'] = 'Unable to connect to the database.';
}

/**
 * Encode the JSON obj and return it.
 */
echo json_encode($response);
?>
```

search/single-result.php

```
<?php
/**
 * The header content.
 */
header("Content-type: application/json");

/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;

if($db || isset($db)) {

    /**
     * The unsafe variables sent with the request
     */
    $unescaped_table = $_GET['table'];
    $id = $_GET['id'];

    /**
     * The variables after they have been cleaned up
     */
    $table = $db->real_escape_string($unescaped_table);

    if(isset($_GET['status'])) {
        $unescaped_status = $_GET['status'];
        $status = $db->real_escape_string($unescaped_status);
    }

    if($table === 'product_group' && isset($status)) {
       $result = $db->select("SELECT * FROM product_group WHERE sku = '$id' AND post_status =
'$status'");
    } else if($table === 'product_group') {
        $result = $db->select("SELECT * FROM product_group WHERE sku = '$id'");
    } else {
       $result = $db->select("SELECT * FROM $table WHERE ID = '$id'");
    }

    $item = $result->fetch_assoc();

    if(empty($item)) {
        $item['error']['thrown'] = true;
      $item['error']['message'] = 'No results found';
    } else {
        $unformattedTime = strtotime($item['post_date']);
        $item['post_date'] = date('d F, Y', $unformattedTime);

        $unformattedTime = strtotime($item['post_modified']);
        $item['post_modified'] = date('d F, Y', $unformattedTime);

        $product_values = $db->select("SELECT * FROM product WHERE sku = '$id'");
        $i = 0;
        while($product = $product_values->fetch_assoc()) {
            $item['product'][$i]['id'] = $product['ID'];
            $item['product'][$i]['sku'] = $product['sku'];
            $item['product'][$i]['value'] = $product['value'];
            $item['product'][$i]['stock'] = $product['stock'];
            $i++;
        }

        $item['error']['thrown'] = false;
        $item['error']['message'] = '';
    }

} else {
    $item['error']['thrown'] = true;
    $item['error']['message'] = "Unable to connect to the database.";
}

echo json_encode($item);
?>
```

view/categories.php

```
<?php
/**
 * Settings the HTTP header
 */

header("Content-type: application/json");
/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;

if($db || isset($db)) {

    /**
     * Parse all information.
     */
    $unescaped_status = $_GET['status']; // The status.
    $status = $db->real_escape_string($unescaped_status);

    if(isset($_GET['menu'])) {
        $unescaped_menu = $_GET['menu']; // Whether it's a menu
        $menu = $db->real_escape_string($unescaped_menu);
    }

    if(!isset($status) || $status === 'not-trash')
        $status = 'publish';

    if(isset($menu) && $menu == true) {
        $result = $db->select("SELECT * FROM category WHERE post_status = '$status' AND
menu_order != '-1' ORDER BY menu_order, name ASC");
    } else {
        $result = $db->select("SELECT * FROM category WHERE post_status = '$status' ORDER BY
menu_order, name ASC");
    }

    /**
     * Loop through all records retrieved from the query.
     */
    $i = 0;
    while($item = $result->fetch_assoc()) {
        $response['category'][$i]['id'] = $item['ID'];
        $response['category'][$i]['name'] = $item['name'];
        $response['category'][$i]['slug'] = $item['slug'];
        $response['category'][$i]['menu_order'] = $item['menu_order'];
        $response['category'][$i]['post_status'] = $item['post_status'];
        $i++;
    }

    if(empty($response)) {
        $response['error']['thrown'] = true;
        $response['error']['message'] = 'No results found.';
    } else {
        $response['error']['thrown'] = false;
    }

} else {
    $response['error']['thrown'] = true;
    $response['error']['message'] = 'Unable to connect to the database.';
}

/**
 * Encode the JSON obj and return it.
 */
echo json_encode($response);
?>
```

view/count.php

```
<?php
// The header content.
header("Content-type: application/json");

// Configure the API.
require_once('../../../inc/api_config.php');
global $db;
if($db || isset($db)) {
    /**
     * The table to count.
     */
    $show = $_GET['show'];
    $status = $_GET['status'];
    // Check to see if the user wants to specify a
    // category to view.
    if(isset($_GET['category'])) {
        $category = $_GET['category'];
        $category = "AND categoryID='".$category."'";
    } else {
        $category = '';
    }
    /**
     * Validate which table to search. Default is 'product_group'.
     */
    switch($show) {
        case 'category':
            $table = 'category';
            $id = 'ID'; break;
        default:
            $table = 'product_group';
            $id = 'sku'; break;
    }
    /**
     * Return on results the user has specified. Default is all.
     */
    switch($status) {
        case 'publish':
            $products_query = $db->select("SELECT COUNT($id) FROM $table WHERE post_status =
'publish' $category"); break;
        case 'trash':
            $products_query = $db->select("SELECT COUNT($id) FROM $table WHERE post_status =
'trash' $category"); break;
        case 'draft':
            $products_query = $db->select("SELECT COUNT($id) FROM $table WHERE post_status =
'draft' $category"); break;
        case 'not-trash':
            $products_query = $db->select("SELECT COUNT($id) FROM $table WHERE post_status <>
'trash' $category"); break;
        default:
            $products_query = $db->select("SELECT COUNT($id) FROM $table"); break;
    }
    /**
     * Calculate the number of pages requred.
     */
    $count = $products_query->fetch_assoc();
    /**
     * Generate the response array.
     */
    $response['count'] = $count['COUNT('.$id.')'];
    $response['show'] = $table;
    $response['status'] = ($status === null ? 'all' : $status);
} else {
    $response['error']['thrown'] = true;
    $response['error']['message'] = 'Unable to connect to the database.';
}
/**
 * Return the encoded JSON object.
 */
echo json_encode($response);
?>
```

view/range.php

```
<?php
/**
 * The header content.
 */
header("Content-type: application/json");

/**
 * Configure the API.
 */
require_once('../../../inc/api_config.php');
global $db;

if($db || isset($db)) {

    /**
     * The variables for the range. Start from and
     * how many to show per page.
     */
    $status = $_GET['status'];
    $start = (int) $_GET['start'];
    $show = (int) $_GET['show'];
    // Check to see if the user wants to specify a
    // category to view.
    if(isset($_GET['category'])) {
        $category = $_GET['category'];
        $category = "AND categoryID='".$category."'";
    } else {
        $category = '';
    }

    /**
     * Return on results the user has specified. Default is all.
     */
    switch($status) {
        case 'publish':
            $products = $db->select("SELECT * FROM product_group WHERE post_status = 'publish'
$category ORDER BY sku DESC LIMIT $start, $show"); break;
        case 'trash':
            $products = $db->select("SELECT * FROM product_group WHERE post_status = 'trash'
$category ORDER BY sku DESC LIMIT $start, $show"); break;
        case 'draft':
            $products = $db->select("SELECT * FROM product_group WHERE post_status = 'draft'
$category ORDER BY sku DESC LIMIT $start, $show"); break;
        case 'not-trash':
            $products = $db->select("SELECT * FROM product_group WHERE post_status <> 'trash'
$category ORDER BY sku DESC LIMIT $start, $show"); break;
        default:
            $products = $db->select("SELECT * FROM product_group ORDER BY sku DESC LIMIT $start,
$show"); break;
    }

    /**
     * Populate the products array.
     */
    $i = 0;
    while($item = $products->fetch_assoc()) {
        $id = $item['sku'];
        $response['product_group'][$i]['sku'] = $id;
        $response['product_group'][$i]['title'] = $item['title'];
        $response['product_group'][$i]['content'] = $item['content'];
        $response['product_group'][$i]['price'] = $item['price'];
        $response['product_group'][$i]['sale_price'] = $item['sale_price'];
        $response['product_group'][$i]['colour'] = $item['colour'];
        $response['product_group'][$i]['thumbnail'] = $item['thumbnail'];
        $response['product_group'][$i]['post_status'] = $item['post_status'];
        $response['product_group'][$i]['categoryID'] = $item['categoryID'];
        $product_values = $db->select("SELECT * FROM product WHERE sku = '$id'");
        $j = 0;
        while($product_value = $product_values->fetch_assoc()) {
            $response['product_group'][$i]['product'][$j]['value'] = $product_value['value'];
            $response['product_group'][$i]['product'][$j]['stock'] = $product_value['stock'];
            $j++;
```

```
        }
        $i++;
    }

    /**
     * Nothing was added to the array.
     */
    if(empty($response) || $response['product_group'][0]['sku'] == null) {
        $response['error']['thrown'] = true;
        $response['error']['message'] = 'No results found.';
    } else {
        $response['error']['thrown'] = false;
        $response['error']['message'] = '';
    }

    /**
     * Return the start number and how many shown.
     */
    $response['start'] = $start;
    $response['show'] = $show;

} else {
    $response['error']['thrown'] = true;
    $response['error']['message'] = 'Unable to connect to the database.';
}

echo json_encode($response);

?>
```

admin/add/default-thumb.php

```
<?php
// The header content.
header("Content-type: application/json");
$successful = false;
// Gets the extension on the file.
function findExts($filename)  {
    $filename = strtolower($filename);
    $exts = explode(".", $filename);
    $exts = end($exts);
    return $exts;
}
// Thumbnail checks and validation.
if(isset($_FILES['thumbnail'])) {
    //This applies the function to our file
    $exts = findExts($_FILES['thumbnail']['name']);
    $target = "../../../../media/default.jpg";
    // Allowed extensions
    $allowedExts = array("jpeg", "jpg", "png", "pjpeg");
    // Check to make sure file type is picture, and size is < 2mb
    if ((($_FILES["thumbnail"]["type"] == "image/jpeg") || ($_FILES["thumbnail"]["type"] == "image/
jpg")
    || ($_FILES["thumbnail"]["type"] == "image/png") || ($_FILES["thumbnail"]["type"] == "image/
pjpeg"))
    && ($_FILES["thumbnail"]["size"] < 200000) && in_array($exts, $allowedExts)) {
        // Has the file got an error?
        if ($_FILES["thumbnail"]["error"] < 1) {
            // Move uploaded file into media directory.
            move_uploaded_file($_FILES['thumbnail']['tmp_name'], $target);
            $successful = true;
        }
    }
}
/* Data returned by the API */
if($successful) {
    $response['error']['thrown'] = false;
    $response['report']['status'] = "Default thumbnail changed";
} else {
    $response['error']['thrown'] = true;
    $response['report']['status'] = "Default thumbnail not changed";
}
echo json_encode($response); ?>
```

admin/delete/empty.db.php

```
<?php
/**
 * The header content.
 */
header("Content-type: application/json");

/**
 * Configure the API.
 */
require_once('../../../../inc/api_config.php');
global $db;

if($db || isset($db)) {

    $db->truncate();

    $response['error']['thrown'] = false;
    $response['report']['message'] = "Database reset successfully.";
} else {
    $response['error']['thrown'] = true;
    $response['report']['message'] = 'Unable to connect to the database.';
}

/**
 * Return the encoded JSON object.
 */
echo json_encode($response);
?>
```

admin/edit/dispatched.php

```
<?php
/**
 * The header content.
 */
header("Content-type: application/json");
/**
 * Configure the API.
 */
require_once('../../../../inc/api_config.php');
global $db;
if($db || isset($db)) {
    /**
     * Initialise the variables sent through $_POST.
     */
    $id = $_POST['orderID'];

    /**
     * Execute the queries to update the settings.
     */
    $db->update("UPDATE $db->db_name.order SET dispatched='true' WHERE ID='$id'");

    /**
     * The array to be returned. As long as an error
     * has not occurred.
     */
    if($db->error_thrown) {
        $response['error']['thrown'] = true;
        $response['report']['status'] = $db->error_message;
    } else {
        $response['error']['thrown'] = false;
    }
} else {
    $response['error']['thrown'] = true;
    $response['error']['status'] = 'Unable to connect to the database.';
}
/**
 * The data returned, as a JSON object.
 */
echo json_encode($response);
?>
```

/inc

WEBSCRP

612596

University of Portsmouth

config.php

```
<?php

// Get array of location.
$site_address1 = explode("/", dirname(__FILE__));
// Find index of root (this DIR is /inc, parent DIR required).
$index = count($site_address1)-2;
$site_address = $site_address1[$index];

/**
 * Define global constants.
 */
define('VERSION', 1.0);

define('SITE_ADDRESS', '/'.$site_address.'/', false);
define('ROOT', dirname(__FILE__), false);
define('INC', 'inc', false);
define('SITE', 'site', false);
define('ADMIN', 'admin', false);

/**
 * Database constants.
 */
define('DB_HOST', 'localhost', false);
define('DB_PORT', '8889', false);
define('DB_USER', 'root', false);
define('DB_PASS', 'root', false);
define('DB_NAME', 'up612596', false);
define('DB_CHARSET', 'utf8', false);

define('DIR_SEPERATOR', '/', false);

?>
```

api_config.php

```
<?php
/**
 * Require the config file. If not found, throw error.
 */
$config_path = dirname(__FILE__);
require_once($config_path . '/config.php');

/**
 * Check required file 'class.database.php' exists. Else, throw error.
 * Required to connect to the database.
 */
require_once($config_path . '/class.db.php');

/**
 * Instantiate the database class, declare the new
 * object to a variable and give it global scope.
 *
 * @global $db Database Object.
 */
function init_db() {
    global $db;

    if(isset($db)) // If true, don't instantiate the obj.
        return;
    else
        $db = new db(DB_HOST, DB_USER, DB_PASS, DB_NAME);
}

/**
 * Call to funtion init_db();
 *
 * @see funct. init_db()
 */
init_db();
?>
```

Database Class

class.db.php

```
<?php

class db {

    /**
     * Whether the database queries are ready to start executing.
     *
     * @access public
     * @var bool
     */
    var $connection;

    /**
     * The name of the database.
     *
     * @access private
     * @var string
     */
    var $db_name;

    /**
     * The last inserted if from the last query.
     *
     * @access public
     * @var int
     */
    var $insert_id;

    /**
     * The value of rows affected from the last query.
     *
     * @access public
     * @var int
     */
    var $affected_rows;

    /**
     * Whether the database queries are ready to start executing.
     *
     * @access private
     * @var bool
     */
    var $ready = false;

    /**
     * The saved results from the last query.
     *
     * @access private
     * @var array|null
     */
    var $last_result;

    /**
     * Whether an error has been thrown from the last query.
     *
     * @access private
     * @var int
     */
    var $error_thrown = false;

    /**
     * The reason behind the last error.
     *
     * @access private
     * @var int
     */
    var $error_message;

    /**
```

```
     * PHP5 constructor.
     *
     * @param string $db_host. The database host.
     * @param string $db_user. The database user.
     * @param string $db_pass. The database password.
     * @param string $db_name. The database name.
     */
    function __construct($db_host, $db_user, $db_pass, $db_name) {
        $this->db_name = $db_name;
        // Instantiates the mysqli object.
        $this->connection = new mysqli($db_host, $db_user, $db_pass);
        if($this->connection)
            $this->ready = true;
        // initialise the database if it cannot be selected.
        if(!$this->connection->select_db($db_name))
            $this->init_db();
        // Echo any errors thrown.
        if(mysqli_connect_errno())
            die("Database connect Error : " . mysqli_connect_error($this->connection));
    }

    private function create_table($query) {
        $this->query($query);
    }

    /**
     * Store the mysqli object.
     *
     * @return object $connection. The stored mysqli object.
     */
    function connect() {
        return $this->connection;
    }

    /**
     * Resets all the variables.
     */
    private function flush() {
        $this->insert_id = null;
        $this->affected_rows = null;
        $this->last_result = null;
        $this->error_thrown = false;
        $this->error_message = "";
    }

    /**
     * The error handling function
     *
     * @param string $message. The error message. Default "Unknown error".
     */
    private function errorHandler($message = "Unknown error.") {
        $this->error_thrown = true;
        $this->error_message = $message;
    }

    /**
     * Escapes special characters in $string for use in the query.
     *
     * @param string $unescaped_string. The unescaped string.
     * @return string. The same string, safe to be used in SQL statements.
     */
    function real_escape_string($unescaped_string) {
        return $this->connection->real_escape_string($unescaped_string);
    }

    /**
     * Query the database.
     *
     * @uses mysqli::query()
     * @param string $query. The string query to be executed.
     */
    private function query($query) {
        // Check the connection is ready to be used.
        if(!$this->ready)
```

```
        return null;
    // Reset variables before new query.
    $this->flush();
    // Makes the query to the database.
    $this->last_result = $this->connection->query($query);
    // If the query was an insert, set variable of last row added.
    $this->insert_id = $this->connection->insert_id;
    // If query was insert, update, replace or delete, set variable to
    // the number affected from the last query.
    $this->affected_rows = $this->connection->affected_rows;
}

/**
 * Function used when data is to be selected from the database.
 *
 * @param string $query. The query to be executed.
 * @return array|null. The result set of the query.
 */
function select($query) {
    // Runs the query.
    $this->query($query);
    return $this->last_result;
}

/**
 * Function used when data is to be selected from the database.
 *
 * @param string $query. The query to be executed.
 * @return array|null. An associative array of results.
 */
function select_row($query) {
    // Runs the query.
    $this->query($query);
    if($this->last_result->num_rows == 1) // Make sure there is just one result.
        return $this->last_result->fetch_assoc();
    // Nothing has happened.
    return null;
}

/**
 * Function used when data is to be inserted into the database.
 *
 * @param string $query. The query to be executed.
 * @return null. Returns null if the insert was unsuccessful.
 */
function insert($query) {
    $this->query($query);
    if($this->insert_id == null) {
        $this->errorHandler("No rows inserted.");
        return null;
    }
}

/**
 * Function used when data is to be modifying the database.
 *
 * @param string $query. The query to be executed.
 * @return null. Returns null if the update was unsuccessful.
 */
function update($query) {
    $this->last_result = $this->query($query);
    if($this->affected_rows === -1) { // No rows where affected.
        $this->errorHandler("No rows were affected.");
        return null;
    }
}

/**
 * Function used when data is to be modifying the database.
 *
 * @param string $query. The query to be executed.
 * @return null. Returns null if the update was unsuccessful.
 */
function empty_trash($table) {
```

```
        if($table === 'product')
            $id = 'sku';
        else
            $id = 'id';
        $this->query("DELETE FROM $table WHERE $id IN (SELECT id FROM $table WHERE post_status =
'trash')");
        if($this->affected_rows === -1) { // No rows where affected.
            $this->errorHandler("No rows were affected.");
            return null;
        }
    }

    /**
     * Function to reset all tables to an empty state.
     *
     * @param string $table. The table to be executed. Default is all tables.
     */
    function truncate() {
        // Drop all Foreign Key Constraints
        $this->query("ALTER TABLE $this->db_name.product_order DROP FOREIGN KEY
product_order_order_fk");
        $this->query("ALTER TABLE $this->db_name.product_order DROP FOREIGN KEY
product_order_product_fk");
        $this->query("ALTER TABLE $this->db_name.product_group DROP FOREIGN KEY
product_group_category_fk");
        $this->query("ALTER TABLE $this->db_name.product DROP FOREIGN KEY
product_product_group_fk");

        // Truncate all tables
        $this->query("TRUNCATE TABLE $this->db_name.product_order");
        $this->query("TRUNCATE TABLE $this->db_name.product");
        $this->query("TRUNCATE TABLE $this->db_name.product_group");
        $this->query("TRUNCATE TABLE $this->db_name.order");
        $this->query("TRUNCATE TABLE $this->db_name.settings");
        $this->query("TRUNCATE TABLE $this->db_name.category");

        // Add all Foreign key constraints back
        $this->query("ALTER TABLE $this->db_name.product_order ADD CONSTRAINT
product_order_order_fk FOREIGN KEY (ID) REFERENCES $this->db_name.order (ID)");
        $this->query("ALTER TABLE $this->db_name.product_order ADD CONSTRAINT
product_order_product_fk FOREIGN KEY (productID) REFERENCES $this->db_name.product (ID)");
        $this->query("ALTER TABLE $this->db_name.product_group ADD CONSTRAINT
product_group_category_fk FOREIGN KEY (categoryID) REFERENCES $this->db_name.category (ID)");
        $this->query("ALTER TABLE $this->db_name.product ADD CONSTRAINT product_product_group_fk
FOREIGN KEY (sku) REFERENCES $this->db_name.product_group (sku)");

        // Insert basic information.
        $this->insert("INSERT INTO $this->db_name.settings (name, value) VALUES('site_url',
'".SITE_ADDRESS."')");
        $this->insert("INSERT INTO $this->db_name.settings (name, value) VALUES('site_name', 'This
needs setting')");
        $this->insert("INSERT INTO $this->db_name.category (name, slug, menu_order, post_status)
VALUES('Uncategorized', 'uncategorized', -1, 'publish')");
    }

    function delete($query) {
        $this->query($query);
    }

    /**
     * Create the database.
     *
     * @uses db::query()
     * @uses db::create_table()
     * @uses db::default_table_entries()
     */
    private function init_db() {
        $this->query("CREATE DATABASE $this->db_name COLLATE utf8_general_ci");

        $this->create_table("CREATE TABLE IF NOT EXISTS $this->db_name.category
                            (ID int NOT NULL AUTO_INCREMENT, name varchar(50), slug varchar(50),
                            menu_order int, post_status varchar(50),
                            CONSTRAINT category_pk PRIMARY KEY(ID))"
                           );
```

```
        $this->insert("INSERT INTO $this->db_name.category (name, slug, menu_order, post_status)
VALUES('Uncategorized', 'uncategorized', -1, 'publish')");
        $this->insert("INSERT INTO $this->db_name.category (name, slug, menu_order, post_status)
VALUES('Toys', 'toys', 1, 'publish')");
        $this->insert("INSERT INTO $this->db_name.category (name, slug, menu_order, post_status)
VALUES('Cuddly Toys', 'cuddly-toys', 2, 'publish')");
        $this->insert("INSERT INTO $this->db_name.category (name, slug, menu_order, post_status)
VALUES('Clothes', 'clothes', 3, 'publish')");

        $this->create_table("CREATE TABLE IF NOT EXISTS $this->db_name.settings
                             (ID int NOT NULL AUTO_INCREMENT,
                             name varchar(50),
                             value longtext,
                             CONSTRAINT settings_pk PRIMARY KEY(ID))"
                            );
        $this->insert("INSERT INTO $this->db_name.settings (name, value) VALUES('site_url',
'".SITE_ADDRESS."')");
        $this->insert("INSERT INTO $this->db_name.settings (name, value) VALUES('site_name',
'Shop')");

        $this->create_table("CREATE TABLE IF NOT EXISTS $this->db_name.order
                             (ID int NOT NULL AUTO_INCREMENT,
                             customer varchar(50),
                             purchase_date datetime, mail_type varchar(50),
                             dispatched varchar(6) NOT NULL DEFAULT 'false',
                             CONSTRAINT order_pk PRIMARY KEY(ID))"
                            );
        $this->create_table("CREATE TABLE IF NOT EXISTS $this->db_name.product_group
                             (sku int NOT NULL AUTO_INCREMENT, title varchar(150), content text,
price double(10,2),
                             sale_price double(10,2), colour varchar(50), thumbnail varchar(150),
post_status varchar(50),
                             post_date datetime, post_modified datetime, categoryID int,
                             CONSTRAINT product_group_pk PRIMARY KEY(sku),
                             CONSTRAINT product_group_category_fk FOREIGN KEY (categoryID)
REFERENCES $this->db_name.category (ID))"
                            );
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
VALUES('Aeroplane','This is a dummy product.','9.99','4.99','','media/
default.jpg','publish',NOW(),'2')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
VALUES('Bear','This is a dummy product.','9.99','0.00','Brown','media/
default.jpg','publish',NOW(),'3')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
VALUES('Cat','This is a dummy product.','9.99','4.99','Ginger','media/
default.jpg','draft',NOW(),'3')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
VALUES('Elephant','This is a dummy product.','9.99','0.00','Grey','media/
default.jpg','trash',NOW(),'3')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
VALUES('Radio','This is a dummy product.','9.99','0.00','Red','media/
default.jpg','draft',NOW(),'1')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
VALUES('Shirt','This is a dummy product.','9.99','4.99','Orange','media/
default.jpg','draft',NOW(),'4')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID) VALUES('T-
Shirt','This is a dummy product.','9.99','4.99','Navy','media/default.jpg','publish',NOW(),'4')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID) VALUES('T-
Shirt','This is a dummy product.','9.99','0.00','Dark Green','media/
default.jpg','publish',NOW(),'4')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
VALUES('Umbrella','This is a dummy product.','9.99','4.99','','media/
default.jpg','publish',NOW(),'1')");
        $this->insert("INSERT INTO $this->db_name.product_group
(title,content,price,sale_price,colour,thumbnail,post_status,post_date,categoryID)
```

```
VALUES('Zebra','This is a dummy product.','9.99','0.00','Black & White','media/
default.jpg','publish',NOW(),'3')");

        $this->create_table("CREATE TABLE IF NOT EXISTS $this->db_name.product
                            (ID int NOT NULL AUTO_INCREMENT, sku int NOT NULL, value varchar(50),
stock int,
                            CONSTRAINT product_pk PRIMARY KEY(ID),
                            CONSTRAINT product_product_group_fk FOREIGN KEY (sku) REFERENCES $this-
>db_name.product_group (sku))"
                           );
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock) VALUES('1','','10')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('2','Small','10')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('2','Medium','11')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('2','Large','12')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('3','Small','8')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('3','Medium','7')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('3','Large','5')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock) VALUES('4','','14')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock) VALUES('5','','16')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('6','Small','14')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('6','Medium','14')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('6','Large','5')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('6','XLarge','12')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('6','XXLarge','42')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('7','Small','43')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('7','Medium','5')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('7','Large','12')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('8','Small','42')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('8','Medium','43')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock)
VALUES('8','Large','12')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock) VALUES('9','','12')");
        $this->insert("INSERT INTO $this->db_name.product (sku,value,stock) VALUES('10','','23')");

        $this->create_table("CREATE TABLE IF NOT EXISTS $this->db_name.product_order
                            (ID int,
                            quantity int, price double(10,2), productID int,
                            CONSTRAINT product_order_pk PRIMARY KEY(ID, productID),
                            CONSTRAINT product_order_order_fk FOREIGN KEY (ID) REFERENCES $this-
>db_name.order (ID),
                            CONSTRAINT product_order_product_fk FOREIGN KEY (productID) REFERENCES
$this->db_name.product (ID))"
                           );
    }
}

?>
```

/js

WEBSCRP

612596

University of Portsmouth

load.js

```
// load.js

// If browser understands the .addEventListener function
if(window.addEventListener) {
    // Listen to when the window has loaded.
    window.addEventListener('load', function () {
        init();
    }, false);

    // Listen to when popstate is fired
    window.addEventListener('popstate', function () {
        init();
    }, false);

    function init () {
        // Set event listeners to elements ALWAYS present
        var siteName = document.getElementById('site-name'),
            adminNavigation = document.querySelectorAll('nav a'),
            searchBox = document.getElementById('search-products'),
            basketItems = document.getElementById('basket-items'),
            basketValue = document.getElementById('basket-value'),
            basketLink = document.getElementById('basket-link'),
            navigation = document.getElementById('navigation'),
            totalPrice = 0.00;

        // Loop through all navigation links, adding the listener
        for(var i=0, len=adminNavigation.length; i<len; i++) {
            if(adminNavigation[i].id != 'site') {
                addListenerGetPage(adminNavigation[i]);
                loadPage(document.URL);
            }
        }

        // Add event listener for the search box
        if(searchBox) {
            searchBox.addEventListener('keydown', function (e) {
                if(e.keyCode === 13 && searchBox.value !== '') {
                    href = searchBox.baseURI+'search/q='+searchBox.value;
                    history.pushState(null, null, href);
                    loadPage(href);
                }
            }, false);
        }

        // Add event listener for when user views basket
        if(basketLink) {
            addListenerGetPage(basketLink);
            // Update the basket information in the header.
            updateBasketInformation();
        }

        // If the navigation.
        if(navigation) {
            getSiteTitle();
            getNavigation();
        }
    }
}
```

functions.js

```
// Takes a hypertext reference, breaks down the string
// and loads the relevant page.
function loadPage (_href) {
    // Set local variables.
    var filename = getFilenameAsString(_href),
        xhr = new XMLHttpRequest(), url,

    // On success, add request response to content.
    // Then call dynamic content.
    success = function () {
        var response = xhr.responseText,
            content = document.getElementById('content');
        // Display content.
        content.innerHTML = response;
        // Add the page content.
        addDynamicContent();
    },
    // Once ready state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage('Status '+xhr.status+' returned.', 'error'); break;
            }
        }
    };

    url = filename+'.php';

    // Open & send request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

// Adding dynamic content and event listeners
function addDynamicContent () {
    var home = document.getElementById('home'),
        category = document.getElementById('category'),
        product = document.getElementById('product'),
        search = document.getElementById('search'),
        basket = document.getElementById('basket'), hold;

    if(home) {
        displayRecentProducts();
    }

    // If a category page.
    if(category) {
        // Hold execution, just to ensure the DOM is loaded.
        hold = setTimeout(function() {
            addEventListeners('products');
            showProducts('publish', getEntityInformation(document.URL, 'category'),
getViewCriteria(document.URL), getCategory(document.URL));
        }, 100);
    }

    // If an individual product page.
    if(product) {
        getProduct(getEntityInformation(document.URL, 'product'));
    }

    // If user has search for something
    if(search) {
        searchProducts(getEntityInformation(document.URL, 'search'));
    }

    // If the user requests to see the basket
    if(basket) {
        displayBasket();
```

```
    }
}

// Ruturns the file to load, from a hypertext reference.
function getFilenameAsString (_href) {
    // Split and splice href, removing the host etc.
    var href = _href.split("/"), href = href.splice(4,4), filename;
    // href should now be a non-empty array.
    if(href[0] === '')
        filename = 'home';
    else if(href[0] === 'product' || href[0] === 'basket' || href[0] === 'search' || href[0] ===
'home') {
        filename = href[0];
    } else {
        filename = 'category';
    }
    // Return the page.
    return filename;
}

// Ruturns the file to load, from a hypertext reference.
function getCategory (_href) {
    // Split and splice href, removing the host etc.
    var href = _href.split("/"), href = href.splice(4,4), category;
    // href should now be a non-empty array.
    if(href[0] === '')
        category = 'home';
    else
        category = href[0];
    // Return the page.
    return category;
}

// Returns the page number (if any), from a hypertext reference.
function getEntityInformation (_href, entity) {
    // Split and splice href, removing the host, admin & page.
    var href = _href.split("/"), href = href.splice(5,5), info;
    // If nothing is left.
    if(href.length < 1 && entity !== 'category') {
        info = '';
    } else {
        switch(entity) {
            case 'product':
                info = href[0]; break;
            case 'basket':
                info = href[0].split("id=").pop(); break;
            case 'search':
                info = href[0].split("q=").pop(); break;
            case 'category':
                info = getPageAsString(document.URL); break;
        }
    }
    // Return the category.
    return info;
}

// Returns the page number (if any), from a hypertext reference.
function getPageAsString (_href) {
    // Split and splice href, removing the host, admin & page.
    var href = _href.split("/"), href = href.splice(5,5), pageNumber;
    // If nothing is left.
    if(href.length < 1) {
        pageNumber = 1;
    } else {
        pageNumber = href[0].split("&");
        pageNumber = pageNumber[0].split("page=").pop();
    }
    // Return the page number.
    return pageNumber;
}

// Returns the view criteria (if any), from a hypertext reference.
// If none set, returns 10.
function getViewCriteria (_href) {
```

```
    // Split and splice href, removing the host, admin & page.
    var href = _href.split("/"), href = href.splice(5,5), view = undefined,
        viewCriteria, _view;
    // If nothing is left.
    if(href.length < 1) {
        viewCriteria = 16;
    } else {
        view = href[0].split('&');
        for(var i in view) {
            _view = view[i].split('=');
            if(_view[0] === 'view')
                viewCriteria = _view[1];
        }
        if(viewCriteria === undefined)
            viewCriteria = 16;
    }
    // Return the view information.
    return viewCriteria;
}

// Function closure
// Adding a listener to an element (on click),
// to get a page.
function addListenerGetPage (elem) {
    elem.addEventListener('click', function (e) {
        history.pushState(null, null, elem.href);
        loadPage(elem.href);
        e.preventDefault();
    }, false);
}

// Display a message to the document.
function showMessage (message, classname) {
    // Get message element from document.
    var requestMessage = document.getElementById("global-message"), timeout;
    if(classname === undefined)
        classname = "success";
    // Set the classification of the message.
    requestMessage.setAttribute("class", classname);
    // Set the message.
    requestMessage.innerHTML = message;
    // Display the message.
    requestMessage.style.display = "block";
    // Reset the message after 3 seconds.
    timeout = setTimeout(function () {
        requestMessage.style.display = "none";
        requestMessage.innerHTML = "";
    }, 3000);
}

// Sets the loader's visibility (boolean).
function loader (visible) {
    // Get loader container from the document.
    var loader = document.getElementById("loader-container");
    // If visible is true, display loader, else hide.
    if(visible)
        loader.style.display = 'block';
    else
        loader.style.display = 'none'; // Hide loader.
}

function getSiteTitle () {
    var url, xhr = new XMLHttpRequest(),

    success = function () {
        var response = JSON.parse(xhr.responseText),
            siteTitle = document.getElementById('site-name'),
            anchor;

        if(response.error.thrown) {
            showMessage(response.error.message);
        } else {
            siteTitle.innerHTML = '<a href="home" id="site-name-anchor">'+response.site_name+'</
a>';
```

```
        }

        anchor = document.getElementById('site-name-anchor');

        addListenerGetPage(anchor);
    },

    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error");
                    return null; break;
            }
        }
    };

    url = 'api/v.1/view/settings.php';

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}


// Get the list of categories for the menu.
function getNavigation () {
    var url, xhr = new XMLHttpRequest(),

    success = function () {
        var response = JSON.parse(xhr.responseText), li = '',
            content = document.getElementById("navigation");

        if(response.error.thrown) {
            content.innerHTML = response.report.message;
        } else {
            // Loop through each result.
            for(i in response.category) {
                category = response.category[i];
                // Set up table body.
                li += '<li class="nav-list-item"><a href="'+category.slug+'" class="nav-list-link"
id="category-'+category.slug+'" data-id="'+category.id+'">' + category.name + '</a></li>';
            }
        }
        // Fill content with new data.
        content.innerHTML = li;
        // Now the navigation has loaded, load the page.
        loadPage(document.URL);
        // add event listeners
        addEventListeners('navigation-links');
    },

    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };

    url = 'api/v.1/view/categories.php?status=publish&menu=true';

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}


// API call for products based on query.
```

```
function searchProducts (query) {
    var url = 'api/v.1/search/products.php?q='+query,
        xhr = new XMLHttpRequest(),

    success = function () {
        var response = JSON.parse(xhr.responseText), li = '',
            container = document.getElementById("search"),
            content = document.getElementById("products-list"),
            product_group, colourString;
        // If API returns error.
        if(response.error.thrown) {
            container.innerHTML = "No products matching your search '" + query + "'";
        } else {
            for(var i in response.product_group) {
                colourString = '';
                product_group = response.product_group[i];
                li += '<div class="products-list-item" data-sku="'+product_group.sku+'">';
                li += '<div class="product-image">';
                li += '<img src="'+product_group.thumbnail+'" alt="'+product_group.title+'" />';
                li += '</div>';
                if(product_group.colour !== '') {
                    colourString = ' <em>('+product_group.colour+')</em>';
                }
                li += '<div class="product-name">'+product_group.title+colourString+'</div>';
                li += '<div class="price-group">';
                if(product_group.sale_price === '0.00' || product_group.sale_price == null) {
                    li += '<div class="product-price">£'+product_group.price+'</div>';
                } else {
                    li += '<div class="product-sale-price">£'+product_group.sale_price+'</div>';
                    li += '<div class="product-price"><del>£'+product_group.price+'</del></div>';
                }
                li += '</div>';
                li += '</div>';
            }
        }
        // Hide loader before content is filled.
        loader(false);
        // Paint to the document.
        content.innerHTML = li;
        // add events
        addEventListeners('product-items');
    }

    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    }

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

// Display the 5 most recently added products
function displayRecentProducts () {
    var xhr = new XMLHttpRequest(), url,
    // Request was successful, display retrieved data.
    success = function () {
        // Parse JSON array.
        var products = JSON.parse(xhr.responseText),
            product, div,
            container = document.getElementById('recent-products');

        loader(false);

        div = '<h3>Recently Added</h3>'

        for(var i=0, len=products.product_group.length; i<len; i++) {
```

```
            product = products.product_group[i];
            div += '<div class="products-list-item" data-sku="'+product.sku+'">';

            div += '<img src="'+product.thumbnail+'" alt="'+product.title+'" />';
            div += '<div class="title">'+product.title+'</div>';
            if(product.sale_price != '0.00') {
                div += '<div class="price">£'+product.sale_price+'</div>';
                div += '<div class="price"><del>was £'+product.price+'</del></div>';
            } else {
                div += '<div class="price">£'+product.price+'</div>';
            }

            div += '</div>';
        }

        container.innerHTML = div;
        // Add events for added products.
        addEventListeners("product-items");
    },

    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };

    // Show loader.
    loader(true);
    // Set url of API, with parameters.
    url = 'api/v.1/search/recent-products.php';
    // Open & send the request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;

}


// Function get and display the product
function getProduct (sku) {
    var url, xhr = new XMLHttpRequest(),
    // Request was successful, display retrieved data.
    success = function () {
        // Parse JSON array.
        var product_group = JSON.parse(xhr.responseText),
            thumbnail = document.getElementById("product-image"),
            title = document.getElementById("product-title"),
            sku = document.getElementById("product-sku"),
            price = document.getElementById("product-price"),
            colour = document.getElementById("product-colour"),
            content = document.getElementById("product-content"),
            size = document.getElementById("product-sizes"),
            stock = document.getElementById("product-stock"),
            basket = document.getElementById("product-basket"),
            quantity = document.getElementById("product-quantity"),
            addToBasket = document.getElementById("product-add-basket"),
            product, sizeElem, sizeElemLabel, j = 0, li = '', checked,
            disabled, outOfStock, stockLevel;

        loader(false);

        if(product_group.error.thrown) {
            title.innerHTML = "No product found."
        } else {

            // Get thumbnail image.
            thumbnail.innerHTML = '<img src="'+product_group.thumbnail+'"
alt="'+product_group.title+'" />';
```

```
            // Populate title.
            title.innerHTML = '<h2>'+product_group.title+'</h2>';
            // Put the sku number.
            sku.innerHTML = 'Product code: '+product_group.sku;
            // Display the colour, if there is a colour set.
            if(product_group.colour.length > 0)
                colour.innerHTML = 'Colour: '+product_group.colour;
            // Display the sale price, if one is set, else just display the normal price.
            if(product_group.sale_price !== "0.00") {
                price.innerHTML += 'Price: £'+product_group.sale_price;
                price.innerHTML += ' <del>was £'+product_group.price+'</del>';
            } else {
                price.innerHTML = 'Price: £'+product_group.price;
            }
            // Put the content in.
            content.innerHTML = '<h3>Description</h3><p>'+product_group.content+'</p>';

            // SIZES.
            if(product_group.product.length > 0) {
                for(var i in product_group.product) {
                    // Get individual product.
                    product = product_group.product[i],
                    checked = '';
                    disabled = '';
                    outOfStock = '';
                    // Checks to see if there is a value set (size set)
                    if(product.value.length != '') {
                        // If the item comes in different sizes...
                        if(j == 0 && product.stock != 0) {
                            checked = 'checked';
                            if(product.stock > 10) {
                                for(var j=0; j<10; j++) {
                                    quantity.innerHTML += '<option value="'+(j+1)+'">'+(j+1)+'</
option>';
                                }
                            } else {
                                for(var j=0, jen=product.stock; j<jen; j++) {
                                    quantity.innerHTML += '<option value="'+(j+1)+'">'+(j+1)+'</
option>';
                                }
                            }
                            addToBasket.dataset.id = product.id;
                            j++;
                        }
                        if(product.stock == 0) {
                            disabled = ' disabled';
                            outOfStock = ' <em>Out of Stock</em>';
                        }

                        li += '<li class="product-size">';

                        li += '<div class="product-size-radio">';
                        li += '<input type="radio" class="size-radio" data-id="'+product.id+'"
name="size" '+checked+disabled+' />';
                        li += '</div>';

                        li += '<div class="size-radio-label">'+product.value+outOfStock+'</div>';

                        li += '</li>';
                    } else {
                        // If the item does not come in different sizes.
                        // Set up the quantity select boxes.
                        // If stock level is greater than 10, set limit to 10, else set limit to
max stock level.
                        if(product.stock > 10) {
                            for(var j=0; j<10; j++) {
                                quantity.innerHTML += '<option value="'+(j+1)+'">'+(j+1)+'</
option>';
                            }
                        } else {
                            for(var j=0, jen=product.stock; j<jen; j++) {
                                quantity.innerHTML += '<option value="'+(j+1)+'">'+(j+1)+'</
option>';
                            }
```

```
                        }
                        addToBasket.dataset.id = product.id;
                    }
                }
                addToBasket.dataset.sku = product_group.sku;
                size.innerHTML = li;
            }

            // Setting the number of options in the quantity, depending on what option is selected.
            addEventListeners("size-quantity-option");
        }

        addEventListeners("product-item");
    },

    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };

    // Show loader.
    loader(true);
    // Set url of API, with parameters.
    url = 'api/v.1/search/single-result.php?table=product_group&id='+sku+'&status=publish';
    // Open & send the request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

// The function called to display both a list of products and
// the pagination links.
function showProducts (status, pageNo, perPage, category) {
    // Now, load the list of products.
    showProductList(status, pageNo, perPage, category);
    //Now, load the pagination
    showPagination(status, pageNo, perPage, category);
}


// Show product list.
function showProductList (status, pageNo, perPage, category) {
    var start = '', url, xhr = new XMLHttpRequest(),
        categoryID = document.getElementById("category-"+category),

    // On success, add request response to content.
    success = function () {
        var response = JSON.parse(xhr.responseText), li = '',
            content = document.getElementById("products-list"),
            product_group, colourString;
        // If API returns error.
        if(response.error.thrown) {
            showMessage(response.error.message, "error");
        } else {
            for(var i in response.product_group) {
                colourString = '';
                product_group = response.product_group[i];
                li += '<div class="products-list-item" data-sku="'+product_group.sku+'">';
                li += '<div class="product-image">';
                li += '<img src="'+product_group.thumbnail+'" alt="'+product_group.title+'" />';
                li += '</div>';
                if(product_group.colour !== '') {
                    colourString = ' <em>('+product_group.colour+')</em>';
                }
                li += '<div class="product-name">'+product_group.title+colourString+'</div>';
                li += '<div class="price-group">';
                if(product_group.sale_price === '0.00' || product_group.sale_price == null) {
```

```
                    li += '<div class="product-price">£'+product_group.price+'</div>';
                } else {
                    li += '<div class="product-sale-price">£'+product_group.sale_price+'</div>';
                    li += '<div class="product-price"><del>£'+product_group.price+'</del></div>';
                }
                li += '</div>';
                li += '</div>';
            }
        }
        // Hide loader before content is filled.
        loader(false);
        // Paint to the document.
        content.innerHTML = li;
        // add events
        addEventListeners('product-items');
    },
    // Once ready state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Display loader will request is sent and retrieved.
    loader(true);
    // Initialise where the API should start getting records from.
    start = (pageNo - 1) * perPage;
    // Get category ID from dataset.
    categoryID = parseInt(categoryID.dataset.id);

    url = 'api/v.1/view/range.php';
    url += '?status='+status+'&start='+start+'&show='+perPage+'&category='+categoryID;

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

// Display the pagination for a category.
function showPagination (status, pageNo, perPage, category) {
    var pageNo = parseInt(pageNo),
        url, xhr = new XMLHttpRequest(),
        categoryID = document.getElementById("category-"+category),

    success = function () {
        var response = JSON.parse(xhr.responseText),
            pagNav = '', filter = '',
            ul = document.getElementById('pagination'),
            pages = Math.ceil(response.count / perPage);

        // Request complete, hide loader.
        loader(false);

        // If page number is 1, do NOT apply anchors to first 2 elements.
        if (pageNo === 1) {
            pagNav += '<li class="pagItemInactive">&laquo;</li>';
            pagNav += '<li class="pagItemInactive">&#139;</li>';
        } else {
            pagNav += '<li><a href="'+category+'/page=1" class="pagItem" data-pageNo="1" data-
perPage="'+perPage+'">&laquo;</a></li>';
            pagNav += '<li><a href="'+category+'/page='+(pageNo-1)+'" class="pagItem" data-
pageNo="'+(pageNo-1)+'" data-perPage="'+perPage+'">&#139;</a></li>';
        }
        // Loop through pages, adding numeric anchors.
        for(var i=0; i<pages; i++) {
            var currentPage = (i+1), active = '';
            // Set the current page classification.
            if (currentPage === pageNo)
                active = 'active';
            // Display ONLY 3 numbers before and after the current page.
```

```
            if (currentPage >= (pageNo-3) && currentPage <= pageNo || currentPage <= (pageNo+3) &&
currentPage >= pageNo)
                pagNav += '<li><a href="'+category+'/page='+currentPage+'" class="pagItem '+active
+'" data-pageNo="'+currentPage+'" data-perPage="'+perPage+'">' + currentPage + '</a></li>';
        }
        // If page last page, do NOT apply anchors to last 2 elements.
        if(pageNo === pages) {
            pagNav += '<li class="pagItemInactive">&#155;</li>';
            pagNav += '<li class="pagItemInactive">&raquo;</li>';
        } else {
            pagNav += '<li><a href="'+category+'/page='+(pageNo+1)+'" class="pagItem" data-
pageNo="'+(pageNo+1)+'" data-perPage="'+perPage+'">&#155;</a></li>';
            pagNav += '<li><a href="'+category+'/page='+pages+'" class="pagItem" data-
pageNo="'+pages+'" data-perPage="'+perPage+'">&raquo;</a></li>';
        }
        // Add the new list elements to the document.
        ul.innerHTML = pagNav;
        // Display the pagination block after it has loaded,
        // Without this a dark grey, empty box displays quickly
        // before the content is added. Very annoying.
        ul.style.display = 'block';
        // Add event listeners for these newly added elements.
        addEventListeners('products-pagination');
    }

    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    }

    // Get category ID from dataset.
    categoryID = parseInt(categoryID.dataset.id);

    url = 'api/v.1/view/count.php';
    url += '?show=product_group&status='+status+'&category='+categoryID;

    // Display loader.
    loader(true);
    // Open & send request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}
```

eventHandler.js

```
// Function to add the appropriate event listeners
function addEventListeners (eventsFor) {
    // If events are for product pagination, apply those.
    switch(eventsFor) {
        // Events for the main navigation.
        case 'navigation-links':
            var links = document.getElementsByClassName('nav-list-link');
            // Loop through navigation links
            for(var i=0, len=links.length; i<len; i++) {
                addListenerGetPage(links[i]);
            }
            break;

        // The events for the individual product items.
        case 'product-items':
            var productItems = document.getElementsByClassName('products-list-item');
            // Add event listener to all.
            function addProductLink (elem) {
                elem.addEventListener('click', function (e) {
                    // As the div is not an anchor, href has to be created,
                    // elem.baseURI gets the elements base uri,
                    // e.g. http://localhost/[name]/
                    href = elem.baseURI+'product/'+elem.dataset.sku;
                    history.pushState(null, null, href);
                    // Load the page.
                    loadPage(href);
                    e.preventDefault();
                }, false);
            }
            // Loop through products.
            for(var i=0, len=productItems.length; i<len; i++) {
                addProductLink(productItems[i]);
            }
            break;

        // Events for the pagination for products.
        case 'products-pagination':
            var pagItems = document.getElementsByClassName('pagItem');
            function addPagLink (elem) {
                elem.addEventListener('click', function (e) {
                    history.pushState(null, null, elem.href);
                    loadPage(elem.href);
                    e.preventDefault();
                })
            }
            for(var i=0, len=pagItems.length; i<len; i++) {
                addPagLink(pagItems[i]);
            }
            break;

        // The events for a single product item
        case 'product-item':
            // When size option is changed, fire event to get stock for selected item.
            var addBasket = document.getElementById('product-add-basket');
            // Need to do event listener for adding to basket.
            addBasket.addEventListener('click', function (e) {
                addToBasket(addBasket.dataset.id, addBasket.dataset.sku);
                e.preventDefault();
            }, false);
            break;

        // The events that occur when a size option is changed
        case 'size-quantity-option':
            var productSize = document.getElementsByClassName('size-radio'),
                quantity = document.getElementById('product-quantity'),
                addBasket = document.getElementById('product-add-basket');
            // Loop through list.
            for(var i=0, len=productSize.length; i<len; i++) {
                onCheckedEvent(productSize[i]);
            }
            // Add event listener for on checked.
            function onCheckedEvent (elem) {
```

```
                elem.addEventListener('click', function () {
                    var url, xhr = new XMLHttpRequest(),
                    success = function () {
                        var response = JSON.parse(xhr.responseText);
                        // Reset options
                        quantity.innerHTML = '';

                        if(response.stock > 10) {
                            for(var j=0; j<10; j++) {
                                quantity.innerHTML += '<option value="'+(j+1)+'">'+(j+1)+'</
option>';
                            }
                        } else {
                            for(var j=0, jen=response.stock; j<jen; j++) {
                                quantity.innerHTML += '<option value="'+(j+1)+'">'+(j+1)+'</
option>';
                            }
                        }
                    },

                    stateChanged = function () {
                        if(xhr.readyState === 4) {
                            switch(xhr.status) {
                                case 200:
                                    success(); break;
                                default:
                                    showMessage("Status "+xhr.status+" returned.", "error"); break;
                            }
                        }
                    };

                    addBasket.dataset.id = elem.dataset.id;

                    url = 'api/v.1/search/stock.php?id='+elem.dataset.id;
                    xhr.open("GET", url, true);
                    xhr.send(null);
                    xhr.onreadystatechange = stateChanged;
                }, false);
            }
            break;

        case 'basket':
            var removeChecks = document.getElementsByClassName('remove'),
                confirmOrderBtn = document.getElementById('confirm-order'),
                continueShopping = document.getElementById('continue-shopping'),
                customerName = document.getElementById('customer-name');

            for(var i=0, len=removeChecks.length; i<len; i++) {
                addCheckedEventListener(removeChecks[i], i);
            }
            function addCheckedEventListener(elem, index) {
                elem.addEventListener('click', function () {
                    removeProduct(sessionStorage.key((index-1)));
                }, false);
            }
            if(confirmOrderBtn) {
                confirmOrderBtn.addEventListener('click', function (e) {
                    if(customerName.value != '') {
                        history.pushState(null, null, confirmOrderBtn.href);
                        confirmOrder();
                    } else {
                        customerName.style.border = '1px solid red';
                    }
                    e.preventDefault();
                }, false);
            }
            if(continueShopping) {
                addListenerGetPage(continueShopping);
            }
            break;

        default: break;
    }
}
```

basket.js

```
// Function to add a product to the session.
function addToBasket (sessionID, productCode) {
    // Retrieve data for purchase for product from elements on site
    var xhr = new XMLHttpRequest(), url,
        _productQuantity = document.getElementById('product-quantity'),
        productQuantity = parseInt(_productQuantity.options[_productQuantity.selectedIndex].value),
        _productSize = document.getElementsByClassName('size-radio'), productSizeID,
        success, stateChanged;

    // Check stock.

    if(_productSize) {
        for (var i=0, len=_productSize.length; i<len; i++) {
            if (_productSize[i].checked) {
                productSizeID = _productSize[i].dataset.id;
            }
        }
    }

    // Contain information about the product.
    product = new Object();
    product.id = sessionID;
    product.sku = productCode;
    product.quantity = parseInt(productQuantity);

    function addToSessionStorage () {
        // Try and find exisitng product in basket.
        var existingProduct = JSON.parse(sessionStorage.getItem(sessionID));

        // Check to see if the product is already there.
        if(!existingProduct) {
            // Not in basket, add product
            sessionStorage.setItem(sessionID, JSON.stringify(product));
        } else {
            // Already there! Increase quantity
            product.quantity = existingProduct.quantity + productQuantity;
            // Remove old
            sessionStorage.removeItem(sessionID);
            // Add new product to the basket
            sessionStorage.setItem(sessionID, JSON.stringify(product));
        }
        // Update the basket content in the header.
        updateBasketInformation();

        // Update the stock on the database.
        updateStock(sessionID, product.quantity, 'minus');

        showMessage('Successfully added to basket', 'success');
    }

    // On request success
    success = function () {
        // Add more info into the object
        var response = JSON.parse(xhr.responseText);

        // Add more information to the basket
        product.name = response.title;
        product.colour = response.colour;
        // Add correct price
        if(response.sale_price === '0.00' || response.sale_price === '') {
            product.price = response.price;
        } else {
            product.price = response.sale_price;
        }

        for(var i=0, len=response.product.length; i<len; i++) {
            if(response.product[i].id === productSizeID) {
                product.size = response.product[i].value;
            }
        }

        // Then add to storage.
```

```
        addToSessionStorage();
    },

    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };

    url = 'api/v.1/search/single-result.php?table=product_group&id='+productCode;

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

function displayBasket () {
    var totalPrice = 0.00, ul = '', price = '',
        basket = document.getElementById('basket-table'),
        subprice = document.getElementById('subprice'),
        confirmOrderBtn = document.createElement('a'),
        name = document.createElement('input');

    // Start basket table.
    ul += '<li class="thead">';
    ul += '<div class="sku">SKU</div>';
    ul += '<div class="name">Name</div>';
    ul += '<div class="size">Size</div>';
    ul += '<div class="quantity">Quantity</div>';
    ul += '<div class="price">Price</div>';
    ul += '<div class="remove">Delete</div>';
    ul += '</li>';

    // Lopp through objects in session
    for(var i=0, len=sessionStorage.length; i<len; i++) {
        var product = JSON.parse(sessionStorage.getItem(sessionStorage.key(i))),
        colour = '';

        totalPrice += parseFloat(product.price) * product.quantity;

        if(product.colour != '')
            colour = ' <em>('+product.colour+')</em>';

        ul += '<li class="tbody">';
        ul += '<div class="sku">'+product.sku+'</div>';
        ul += '<div class="name"><a href="product/'+product.sku+'">'+product.name+colour+'</a></
div>';
        if(product.size != undefined)
            ul += '<div class="size">'+product.size+'</div>';
        else
            ul += '<div class="size"></div>';
        ul += '<div class="quantity">'+product.quantity+'</div>';
        ul += '<div class="price">£'+product.price+'</div>';
        ul += '<div class="remove"><input type="checkbox" class="remove-item-basket" /></div>';
        ul += '</li>';
    }

    price += 'Subtotal £'+totalPrice.toFixed(2);

    // Display total for order.
    basket.innerHTML = ul;
    subprice.innerHTML = price;

    if(sessionStorage.length > 0) {
        name.type = 'text';
        name.placeholder = 'Please enter your name';
        name.id = 'customer-name';
        insertAfter(subprice, name);
```

```
        // Create to confirm order button to make the order.
        confirmOrderBtn.innerHTML = 'Confirm Order';
        confirmOrderBtn.href = 'basket';
        confirmOrderBtn.id = 'confirm-order';
        // Add it in after the last element on the site.
        insertAfter(name, confirmOrderBtn);
    } else {
        // Create to confirm order button to make the order.
        confirmOrderBtn.innerHTML = 'Continue Shopping';
        confirmOrderBtn.href = 'home';
        confirmOrderBtn.id = 'continue-shopping';
        // Add it in after the last element on the site.
        insertAfter(subprice, confirmOrderBtn);
    }

    // Inserts an element after a given element.
    function insertAfter(referenceNode, newNode) {
        referenceNode.parentNode.insertBefore(newNode, referenceNode.nextSibling);
    }

    addEventListeners('basket');
}

// Remove single product

function removeProduct(sessionKey) {
    var item = JSON.parse(sessionStorage.getItem(sessionKey));
    // Update stock levels in database to reflect.
    updateStock(sessionKey, item.quantity, 'add');
    // Remove from session storage.
    sessionStorage.removeItem(sessionKey);
    //Update the basket information in the header.
    updateBasketInformation();
    // Reload page.
    loadPage(document.URL);
}

// Update the stock (remove and add).
function updateStock (productID, quantity, operation) {
    var xhr = new XMLHttpRequest(), url, param;
    // Set up url and parameters
    url = 'api/v.1/edit/stock.php';
    param = 'productID='+productID+'&quantity='+quantity+'&operation='+operation;
    // Open & post request.
    xhr.open("POST", url, true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.send(param);
}

// Confirm order and save to the database.
function confirmOrder () {
    // Loop through objects in session
    // adding them to the order table in the database
    var url, param, xhr = new XMLHttpRequest(),
        id = [], quantity = [], price = [],
        currentProduct, content = document.getElementById('basket'),
        customerName = document.getElementById('customer-name'),

    // Request successful, display response
    success = function () {
        var response = JSON.parse(xhr.responseText);
        // Hide loader
        loader(false);
        // if error thrown
        if(response.error.thrown) {
            // Show message of update failure
            content.innerHTML = "<h2>There was a problem with your order. Please try again.</h2>";
        } else {
            // Clear session storage
            // Display congrates
            content.innerHTML = "<h2>"+response.report.message+"</h2>";
            // Clear the basket
            sessionStorage.clear();
            // Update the basket information in the header
```

```
            updateBasketInformation();
        }
    },
    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Show loader before request is sent
    loader(true);

    // Add values to arrays
    for(var i=0, len=sessionStorage.length; i<len; i++) {
        // Get the current product from the session using the key
        currentProduct = JSON.parse(sessionStorage.getItem(sessionStorage.key(i)));
        // Push the data to the array
        id.push(currentProduct.id);
        quantity.push(currentProduct.quantity);
        price.push(currentProduct.price);
    }

    // Stringify the arrays to allow them to be passed through AJAX.
    id = JSON.stringify(id);
    quantity = JSON.stringify(quantity);
    price = JSON.stringify(price);

    // Request parameters
    param = new FormData();
    param.append('id', id);
    param.append('quantity', quantity);
    param.append('price', price);
    param.append('customer_name', customerName.value);

    // Set URL and parameters to be sent
    url = 'api/v.1/add/confirm-order.php';

    // Open, set headers & post request
    xhr.open("POST", url, true);
    xhr.send(param);
    xhr.onreadystatechange = stateChanged;
}

// Updates the basket content in the header.
function updateBasketInformation () {
    var basketItems = document.getElementById('basket-items'),
        basketValue = document.getElementById('basket-value'),
        totalPrice = 0.00, totalQuantity = 0;

    // Set the total price
    for(var i=0, len=sessionStorage.length; i<len; i++) {
        var currentId = sessionStorage.key(i),
            currentProduct = JSON.parse(sessionStorage.getItem(currentId));
        totalPrice += parseFloat(currentProduct.price) * currentProduct.quantity;
        totalQuantity += currentProduct.quantity
    }
    basketItems.innerHTML = totalQuantity + ' items';
    basketValue.innerHTML = '£' + totalPrice.toFixed(2);
}
```

admin/functions.js

```
// Takes a hypertext reference, breaks down the string
// and loads the relevant page.
function loadPage (_href) {
    // Set local variables.
    var filename = getFilenameAsString(_href),
        xhr = new XMLHttpRequest(), url,
    // On success, add request response to content.
    // Then call dynamic content.
    success = function () {
        var response = xhr.responseText,
            content = document.getElementById('content');
        content.innerHTML = response;
        addDynamicContent();
    },
    // Once ready state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage('Status '+xhr.status+' returned.', 'error'); break;
            }
        }
    };
    // Set url.
    url = './'+filename+'.php';
    // Open & send request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

// Adding dynamic content and event listeners
function addDynamicContent () {
    // The page container elements.
    var products = document.getElementById("products"),
        categories = document.getElementById("categories"),
        tags = document.getElementById("tags"),
        editProduct = document.getElementById("edit-product"),
        addProduct = document.getElementById("add-product"),
        editTag = document.getElementById("edit-tag"),
        addTag = document.getElementById("add-tag"),
        settings = document.getElementById("settings"),
        orders = document.getElementById('orders'),
        pageNumber, status, view, productID;

    // If the page is the Products page.
    if(products) {
        // Get url criteria.
        pageNo = getPageNumberAsString(document.URL);
        status = getFilterCriteria(document.URL);
        view = getViewCriteria(document.URL);
        // Show products.
        showProducts(status, pageNo, view);
        addEventListeners('products');
    }

    if(editProduct) {
        // Populate list of categories
        getCategories();
        // Get products from id set in url.
        getProduct(getPageNumberAsString(document.URL));
    }

    if(addProduct) {
        // Populate list of categories
        getCategories();
        // Add listeners
        addEventListeners('add-product');
    }
```

```
    if(categories) {
        // Display list of current categories
        // Add event listeners.
        addEventListeners('add-category');
        status = getFilterCriteria(document.URL);
        showCategoryList(status);
    }

    if(orders) {
        undispatchedOrders();
    }

    if(settings) {
        getCompanyName();
        getDefaultPicture();
        addEventListeners('settings');
    }
}

// Ruturns the file to load, from a hypertext reference.
function getFilenameAsString (_href) {
    // Split and splice href, removing the host etc.
    var href = _href.split("/"), href = href.splice(4,4), filename;
    // href should now be a non-empty array.
    // Example: ["admin", "products", "page=1&filter=publish"]
    // Check admin area.
    if(href[0] === 'admin') {
        filename = href[1];
    }
    else {
        filename = href[0];
    }
    if(filename === '')
        filename = 'home';
    // Return the page.
    return filename;
}

// Returns the page number (if any), from a hypertext reference.
function getPageNumberAsString (_href) {
    // Split and splice href, removing the host, admin & page.
    var href = _href.split("/"), href = href.splice(6,6), pageNumber;
    // If nothing is left.
    if(href.length < 1) {
        pageNumber = 1;
    } else {
        pageNumber = href[0].split("&");
        pageNumber = pageNumber[0].split("page=").pop();
    }
    // Return the page number.
    return pageNumber;
}

// Returns the filter criteria (if any), from a hypertext reference.
// If none set, returns 'not-trash'.
function getFilterCriteria (_href) {
    // Split and splice href, removing the host, admin & page.
    var href = _href.split("/"), href = href.splice(6,6), filter = undefined,
        filterCriteria, _filter;
    // If nothing is left.
    if(href.length < 1) {
        filterCriteria = 'not-trash';
    } else {
        filter = href[0].split('&');
        for(var i in filter) {
            _filter = filter[i].split('=');
            if(_filter[0] === 'filter')
                filterCriteria = _filter[1];
        }
        if(filterCriteria === undefined)
            filterCriteria = 'not-trash';
    }
    // Return the filter information.
```

```
    return filterCriteria;
}

// Returns the view criteria (if any), from a hypertext reference.
// If none set, returns 10.
function getViewCriteria (_href) {
    // Split and splice href, removing the host, admin & page.
    var href = _href.split("/"), href = href.splice(6,6), view = undefined,
        viewCriteria, _view;
    // If nothing is left.
    if(href.length < 1) {
        viewCriteria = 10;
    } else {
        view = href[0].split('&');
        for(var i in view) {
            _view = view[i].split('=');
            if(_view[0] === 'view')
                viewCriteria = _view[1];
        }
        if(viewCriteria === undefined)
            viewCriteria = 10;
    }
    // Return the view information.
    return viewCriteria;
}

// Function closure
// Adding a listener to an element (on click),
// to get a page.
function addListenerGetPage (elem) {
    elem.addEventListener('click', function (e) {
        history.pushState(null, null, elem.href);
        loadPage(elem.href);
        e.preventDefault();
    }, false);
}

// Add the value ans stock inputs to the document.
function addValueStockInputs () {
    var inputValue = document.createElement("input"),
        inputStock = document.createElement("input"),
        addValueStockInput = document.getElementById("values-and-stocks");

    // Input for values and sizes
    inputValue.type = "text";
    inputValue.className = "single-values";
    inputValue.placeholder = "Size (optional)";
    // Input for stock
    inputStock.type = "text";
    inputStock.className = "single-stocks";
    inputStock.placeholder = "Stock";
    // Append these to the window.
    addValueStockInput.appendChild(inputValue);
    addValueStockInput.appendChild(inputStock);
}

// Display a message to the document.
function showMessage (message, classname) {
    // Get message element from document.
    var requestMessage = document.getElementById("request-message"), timeout;
    if(classname === undefined)
        classname = "success";
    // Set the classification of the message.
    requestMessage.setAttribute("class", classname);
    // Set the message.
    requestMessage.innerHTML = message;
    // Display the message.
    requestMessage.style.display = "block";
    // Reset the message after 3 seconds.
    timeout = setTimeout(function() {
        requestMessage.style.display = "none";
        requestMessage.innerHTML = "";
    }, 3000);
}
```

```
// Sets the loader's visibility (boolean).
function loader (visible) {
    // Get loader container from the document.
    var loader = document.getElementById("loader-container");
    // If visible is true, display loader, else hide.
    if(visible)
        loader.style.display = 'block';
    else
        loader.style.display = 'none'; // Hide loader.
}

// Function to validate product field input.
function validateProductInput () {
    var isValid = 0, checked,
        titleInput = document.getElementById("single-title"),
        contentInput = document.getElementById("single-content"),
        priceInput = document.getElementById("single-price"),
        saleInput = document.getElementById("single-sale"),
        stockInputs = document.getElementsByClassName("single-stocks"),
        categorySection = document.getElementsByClassName('product-category'),
        categoryRadio = document.getElementsByClassName("category-radio");

    categorySection[0].style.border = '1px solid rgba(150,150,150,0.2)';
    titleInput.style.border = '';
    contentInput.style.border = '';
    priceInput.style.border = '';
    saleInput.style.border = '';
    stockInputs[0].style.border = '';

    for(var i=0, len=categoryRadio.length; i<len; i++) {
        if(categoryRadio[i].checked) {
            checked = true;
        }
    }

    // Check a catgeory is selected.
    if(!checked) {
        categorySection[0].style.border = '1px solid red';
        isValid++;
    }

    // Check required fields are not empty.
    if(!titleInput.value) {
        titleInput.style.border = '1px solid red';
        isValid++;
    }


    if(!contentInput.value) {
        contentInput.style.border = '1px solid red';
        isValid++;
    }

    if(!priceInput.value) {
        priceInput.style.border = '1px solid red';
        isValid++;
    }

    if(!stockInputs[0].value) {
        stockInputs[0].style.border = '1px solid red';
        isValid++;
    }

    // Check the user has not entered a '£' before.
    if(isNaN(priceInput.value)) {
        priceInput.style.border = '1px solid red';
        showMessage("Please do not add symbols, such as '£'", 'error');
        isValid++;
    }
    if(isNaN(saleInput.value)) {
        saleInput.style.border = '1px solid red';
        showMessage("Please do not add symbols, such as '£'", 'error');
        isValid++;
```

```
    }

    // Check the sale price is not bigger than the original price.
    if(parseInt(priceInput.value) < parseInt(saleInput.value)) {
        saleInput.style.border = '1px solid red';
        isValid++;
    }

    // Check the stock is positive.
    for(var i=0, len=stockInputs.length; i<len; i++) {
        if(stockInputs[i].value < 0) {
            isValid++;
        }
        if(isNaN(stockInputs[i].value)) {
            isValid++;
        }
    }

    // If isValid equals 0, nothing has been flagged. Return True.
    if(isValid === 0) { return true; }
    else { return false; }
}

// Function to validate product field input.
function validateCategoryInput () {
    var isValid = 0,
        name = document.getElementById("cat-name"),
        menuOrder = document.getElementById("cat-menu-order");
    console.log(name, menuOrder);

    // Initialise borders
    name.style.border = '';
    menuOrder.style.border = '';

    // Check required fields are not empty.
    if(!name.value) {
        name.style.border = '1px solid red';
        isValid++;
    }
    if(name.value.toLowerCase() == 'admin') {
        name.style.border = '1px solid red';
        showMessage("'Admin' is a reserved word", "error");
        isValid++;
    }
    if(!menuOrder.value) {
        menuOrder.style.border = '1px solid red';
        isValid++;
    }
    // Check the value is an integer (including negative.) - NEEDS IMPROVING: '1w' is accepted.
    if(isNaN(parseInt(menuOrder.value))) {
        menuOrder.style.border = '1px solid red';
        isValid++;
    }
    // If isValid equals 0, nothing has been flagged. Return True.
    if(isValid === 0) { return true; }
    else { return false; }
}


// SETTINGS

function getCompanyName () {
    var url, xhr = new XMLHttpRequest(),
        companyName = document.getElementById('site-name');

    // Request successful, display response.
    success = function () {
        var response = JSON.parse(xhr.responseText);
        // Hide loader.
        loader(false);
        // if error thrown.
        if(!response.error.thrown) {
            // Show message of update failure.
            companyName.value = response.site_name;
```

```
        }
    },

    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };

    // Show loader before request is sent.
    loader(true);

    // Set URL and parameters to be sent.
    url = '../api/v.1/admin/view/settings.php';

    // Open, set headers & post request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

// Function to set the company name
function setCompanyName () {
    var url, param, xhr = new XMLHttpRequest(),
        companyName = document.getElementById('site-name').value;

    // Request successful, display response.
    success = function () {
        var response = JSON.parse(xhr.responseText);
        // Hide loader.
        loader(false);
        // if error thrown.
        if(response.error.thrown) {
            // Show message of update failure.
            showMessage(response.report.status, "error");
        } else {
            // Show message of update success.
            showMessage(response.report.status, "success");
        }
    },

    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };

    // Show loader before request is sent.
    loader(true);

    // Request parameters.
    param = new FormData();
    param.append('site_name', companyName);

    // Set URL and parameters to be sent.
    url = '../api/v.1/admin/edit/site-name.php';

    // Open, set headers & post request.
    xhr.open("POST", url, true);
    xhr.send(param);
    xhr.onreadystatechange = stateChanged;
}
```

```
// Function to set the company name
function getDefaultPicture () {
    var url, param, xhr = new XMLHttpRequest(),
        thumbnailPreview = document.getElementById('single-thumbnail-preview');

    thumbnailPreview.innerHTML = '<img src="../media/default.jpg" alt="The default picture set for
products." />';
}

// Function to set the default picture.
function setDefaultPicture () {
    var url, param, xhr = new XMLHttpRequest(), thumbnail,
        thumbnailElem = document.getElementById("default-picture"),

    // Request successful, display response.
    success = function () {
        var response = JSON.parse(xhr.responseText),
            setDefault = document.getElementById("set-default-picture");
        // Hide loader.
        loader(false);
        // if error thrown.
        if(response.error.thrown) {
            // Show message of update failure.
            showMessage(response.report.status, "error");
        } else {
            // Show message of update success.
            showMessage(response.report.status, "success");
            history.pushState(null, null, setDefault.href);
            loadPage(setDefault.href);
        }
    },

    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };

    // Show loader before request is sent.
    loader(true);

    // Takes the file and posts it.
    thumbnail = thumbnailElem.files[0];
    // Request parameters.
    param = new FormData();
    param.append('thumbnail', thumbnail);

    // Set URL and parameters to be sent.
    url = '../api/v.1/admin/add/default-thumb.php';

    // Open, set headers & post request.
    xhr.open("POST", url, true);
    xhr.send(param);
    xhr.onreadystatechange = stateChanged;
}

// Ajax call to empty database.
function truncateDbTables () {
    var success, stateChanged, url,
        xhr = new XMLHttpRequest(),

    success = function () {
        var response = JSON.parse(xhr.responseText);
        // If API returns error.
        if(response.error.thrown) {
            showMessage(response.report.message, 'error');
        } else {
```

```
            showMessage(response.report.message, 'success');
        }
    },

    stateChanged = function() {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    failed("Status "+xhr.status+" returned."); break;
            }
        }
    };

    url = '../api/v.1/admin/delete/empty.db.php';

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

// Display all undispatched orders.
function undispatchedOrders () {
    var success, stateChanged, url,
        xhr = new XMLHttpRequest(),

    success = function () {
        var response = JSON.parse(xhr.responseText), li = '',
            undispatchedOrders = document.getElementById('undispatched-orders');

        // If API returns error.
        if(response.error.thrown) {
            showMessage(response.error.message, "error");
        } else {
            li += '<li class="order-list thead">';
            li += '<div class="order-id">Sku</div>';
            li += '<div class="customer-name">Customer</div>';
            li += '<div class="order-products">Products</div>';
            li += '<div class="mark-dispatched">Dispatched</div>';
            li += '</li>';
            // Loop through each result.
            for(var i in response.order) {
                order = response.order[i];

                // Set up table body.
                li += '<li class="order-list '+order.id+'">';

                li += '<div class="order-id">' + order.id + '</div>';

                li += '<div class="customer-name">' + order.customer + '</a></div>';

                li += '<div class="order-products"><ul>';
                for(var i in order.product_order) {
                    var product = order.product_order[i];
                    li += '<li>Product: <a href="product/'+product.sku+'" class="product-item"
data-id="'+product.sku+'">'+product.productID+'</a> Quantity: '+product.quantity+'</li>';
                }
                li += '</ul></div>';

                li += '<div class="mark-dispatched">';
                li += '<input type="checkbox" id="mark-dispatched" data-id="'+order.id+'" />'
                li += '</div>'

                li += '</li>';
            }
        }

        undispatchedOrders.innerHTML = li;

        addEventListeners('orders');
    },

    stateChanged = function() {
```

```
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    failed("Status "+xhr.status+" returned."); break;
            }
        }
    };

    url = '../api/v.1/view/orders.php';

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
}

function markOrderAsDispatched (orderID) {
    var success, stateChanged, url, param = 'orderID='+orderID;
        xhr = new XMLHttpRequest(),

    success = function () {
        var response = JSON.parse(xhr.responseText);
        // If API returns error.
        if(response.error.thrown) {
            showMessage(response.error.message, "error");
        } else {
            loadPage(document.URL);
        }
    },

    stateChanged = function() {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    failed("Status "+xhr.status+" returned."); break;
            }
        }
    };

    url = '../api/v.1/admin/edit/dispatched.php';

    xhr.open("POST", url, true);
    xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
    xhr.send(param);
    xhr.onreadystatechange = stateChanged;
}
```

admin/products.js

```
// products.js
var

// Get a product, with a given sku.
getProduct = function(sku) {
    var xhr = new XMLHttpRequest(), url,

    // Request was successful, display retrieved data.
    success = function () {
        // Parse JSON array.
        var product_group = JSON.parse(xhr.responseText),
            updateButton = document.getElementById("update-product-button"),
            title = document.getElementById("single-title"),
            content = document.getElementById("single-content"),
            price = document.getElementById("single-price"),
            sale = document.getElementById("single-sale"),
            colour = document.getElementById("single-colour"),
            thumbnail = document.getElementById("single-thumbnail"),
            thumbnailPreview = document.getElementById('single-thumbnail-preview'),
            valuesStocks = document.getElementById("values-and-stocks"),
            valueInputs = document.getElementsByClassName("single-values"),
            stockInputs = document.getElementsByClassName("single-stocks"),
            status = document.getElementById("single-status"),
            postDate = document.getElementById("date-posted"),
            categoryRadio = document.getElementsByClassName("category-radio"),
            categoryRadioArray;

        // Hide loader.
        loader(false);
        // If API returns error.
        if(product_group.error.thrown) {
            showMessage(product_group.error.message, "error");
        } else { // Otherwise.
            // Set variables as information from request.
            title.value = product_group.title;
            content.innerHTML = product_group.content;
            price.value = product_group.price;
            // Leave blank if not set.
            if(product_group.sale_price != 0.00) {
                sale.value = product_group.sale_price;
            }
            colour.value = product_group.colour;

            thumbnailPreview.innerHTML = '<img src="../'+product_group.thumbnail+'"
alt="'+product_group.title+'" />';

            // loop through all products.
            for(var i in product_group.product) {
                valuesStocks.innerHTML += '<input type="text" class="single-values"
placeholder="Size (optional)" />';
                valuesStocks.innerHTML += '<input type="text" class="single-stocks"
placeholder="Stock" />';
            }
            for(var i in product_group.product) {
                product = product_group.product[i];
                valueInputs[i].value = product.value;
                stockInputs[i].value = product.stock;
            }
            // Set the option that should be selected.
            if(product_group.post_status === 'publish') {
                status.selectedIndex = 0;
            } else if(product_group.post_status === 'draft') {
                status.selectedIndex = 1;
            } else if(product_group.post_status === 'trash') {
                status.selectedIndex = 2;
            }
            // Timeout set here to fix bug of loading radio buttons
            // once they are set on the DOM.
            timeout = setTimeout(function() {
                for(var i=0, len=categoryRadio.length; i<len; i++) {
                    if(categoryRadio[i].dataset.id == product_group.categoryID) {
                        categoryRadio[i].checked = true;
```

```
                    }
                }
            }, 50);
            // Set date to the date original product was created.
            postDate.innerHTML = 'Date created: <em>' + product_group.post_date + '</em>';
            // Update the data-sku for the update button to reflect
            // the current product.
            updateButton.dataset.sku = product_group.sku;
            // Add event listeners to new elements.
            addEventListeners('update-product');
        }
    },
    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Show loader.
    loader(true);
    // Set url of API, with parameters.
    url = '../api/v.1/search/single-result.php?table=product_group&id='+sku;
    // Open & send the request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
},

// Call to add product.
addProduct = function () {
    var url, param, xhr = new XMLHttpRequest(),
        valuesArray = [], stocksArray = [],
        values, stocks, categoryId, thumbnail,
        title = document.getElementById("single-title").value,
        content = document.getElementById("single-content").value,
        price = document.getElementById("single-price").value,
        sale = document.getElementById("single-sale").value,
        colour = document.getElementById("single-colour").value,
        thumbnailElem = document.getElementById("single-thumbnail"),
        valueInputs = document.getElementsByClassName("single-values"),
        stockInputs = document.getElementsByClassName("single-stocks"),
        statusList = document.getElementById("single-status"),
        statusValue = statusList.options[statusList.selectedIndex].value,
        categoryRadio = document.getElementsByClassName("category-radio"),

    // Request successful, display response.
    success = function () {
        var response = JSON.parse(xhr.responseText),
            addProductButton = document.getElementById("add-product-button");
        // Hide loader.
        loader(false);
        // if error thrown.
        if(response.error.thrown) {
            // Show message of update failure.
            showMessage(response.report.status, "error");
        } else {
            // Show message of update success.
            showMessage(response.report.status, "success");
            // Calculate the new 'edit product' sku and send the user to that page.
            addProductButton.href = addProductButton.href + response.report.inserted_id;
            history.pushState(null, null, addProductButton.href);
            loadPage(addProductButton.href);
        }
    },
    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
```

```
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Show loader before request is sent.
    loader(true);

    // Get selected category.
    for(var i=0, len=categoryRadio.length; i<len; i++) {
        if(categoryRadio[i].checked) {
            categoryId = categoryRadio[i].dataset.id;
        }
    }
    // Populate stocks array with stocks from page
    for(var i=0, len=stockInputs.length; i<len; i++) {
        if(stockInputs[i].value != "") {
            valuesArray[i] = valueInputs[i].value;
            stocksArray[i] = stockInputs[i].value;
        }
    }
    // Stringify array ready for parameter.
    values = JSON.stringify(valuesArray);
    stocks = JSON.stringify(stocksArray);

    // Takes the file and posts it.
    thumbnail = thumbnailElem.files[0];
    // Request parameters.
    param = new FormData();
    param.append('title', title);
    param.append('content', content);
    param.append('price', price);
    param.append('sale', sale);
    param.append('colour', colour);
    param.append('thumbnail', thumbnail);
    param.append('values', values);
    param.append('stocks', stocks);
    param.append('status', statusValue);
    param.append('category_id', categoryId);

    // Set URL and parameters to be sent.
    url = '../api/v.1/add/product.php';

    // Open, set headers & post request.
    xhr.open("POST", url, true);
    xhr.send(param);
    xhr.onreadystatechange = stateChanged;
},

// Call to update a product, with the given sku.
updateProduct = function (sku) {
    var url, param, xhr = new XMLHttpRequest(),
        valuesArray = [], stocksArray = [],
        values, stocks, categoryId, thumbnail,
        title = document.getElementById("single-title").value,
        content = document.getElementById("single-content").value,
        price = document.getElementById("single-price").value,
        sale = document.getElementById("single-sale").value,
        colour = document.getElementById("single-colour").value,
        thumbnailElem = document.getElementById("single-thumbnail"),
        valueInputs = document.getElementsByClassName("single-values"),
        stockInputs = document.getElementsByClassName("single-stocks"),
        statusList = document.getElementById("single-status"),
        statusValue = statusList.options[statusList.selectedIndex].value,
        categoryRadio = document.getElementsByClassName("category-radio");

    // Request was successful, fill content.
    success = function () {
        // The retrieved data.
        var response = JSON.parse(xhr.responseText);
        // Hide the loader.
        loader(false);
        // if error thrown.
```

```
        if(response.error.thrown) {
            // Show message of update failure.
            showMessage(response.report.status, "error");
        } else {
            // Show message of update success.
            showMessage(response.report.status, "success");
        }

        loadPage(document.URL);
    },
    // Once state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Show loader.
    loader(true);

    // Get selected category.
    for(var i=0, len=categoryRadio.length; i<len; i++) {
        if(categoryRadio[i].checked) {
            categoryId = categoryRadio[i].dataset.id;
        }
    }

    // Populate values array with values from document.
    for(var i=0, len=stockInputs.length; i<len; i++) {
        // Because stock is required, check it isn't empty.
        if(stockInputs[i].value != "") {
            valuesArray[i] = valueInputs[i].value;
            stocksArray[i] = stockInputs[i].value;
        }
    }
    // Stringify array to send in parameter.
    values = JSON.stringify(valuesArray);
    stocks = JSON.stringify(stocksArray);

    // Takes the file and posts it.
    thumbnail = thumbnailElem.files[0];
    // Request parameters.
    param = new FormData();
    param.append('sku', sku);
    param.append('title', title);
    param.append('content', content);
    param.append('price', price);
    param.append('sale', sale);
    param.append('colour', colour);
    param.append('thumbnail', thumbnail);
    param.append('values', values);
    param.append('stocks', stocks);
    param.append('status', statusValue);
    param.append('category_id', categoryId);

    // Request url.
    url = '../api/v.1/edit/product.php';

    // Open, set header & post request.
    xhr.open("POST", url, true);
    xhr.send(param);
    xhr.onreadystatechange = stateChanged;
},
// Quick update the status of a specific product.
updateProductStatus = function (elem) {
    var url = '../api/v.1/edit/product-status.php',
        param = 'sku='+elem.dataset.sku+'&status='+elem.options[elem.selectedIndex].value,
        xhr = new XMLHttpRequest(),
    // If the request is successful.
    success = function () {
```

```
        // Response from the request.
        var response = JSON.parse(xhr.responseText);
        // Hide loader as request was successful.
        loader(false);
        // Display a message to the user signalling update
        // was successful.
        showMessage(response.report.status, "success");
        // Reload the page to update the list.
        loadPage(document.URL);
    },
    // Once ready state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Show loader before request is sent.
    loader(true);
    // Open, set header & post request.
    xhr.open("POST", url, true);
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.send(param);
    xhr.onreadystatechange = stateChanged;
},

// Display products and pagination.
showProducts = function (status, pageNo, perPage) {
    // Load the paginaton for the products to be displayed.
    showProductPagination(status, pageNo, perPage);
    // Now, load the list of products.
    showProductList(status, pageNo, perPage);
},
// Show product pagination.
showProductPagination = function (status, pageNo, perPage) {
    var pageNo = parseInt(pageNo),
        url = '../api/v.1/view/count.php';
        url += '?show=product_group&status='+status;
        xhr = new XMLHttpRequest(),

    // If the request is successful.
    success = function () {
        var response = JSON.parse(xhr.responseText),
            pagNav = '', filter = '',
            ul = document.getElementById('pagination'),
            pages = Math.ceil(response.count / perPage);

        // Request complete, hide loader.
        loader(false);

        // Set the filter, (if any).
        if(status !== 'not-trash')
            filter = '&filter='+status;
        // If page number is 1, do NOT apply anchors to first 2 elements.
        if (pageNo === 1) {
            pagNav += '<li class="pagItemInactive">&laquo;</li>';
            pagNav += '<li class="pagItemInactive">&#139;</li>';
        } else {
            pagNav += '<li><a href="products/page=1'+filter+'" class="pagItem" data-pageNo="1"
data-perPage="'+perPage+'">&laquo;</a></li>';
            pagNav += '<li><a href="products/page='+(pageNo-1)+filter+'" class="pagItem" data-
pageNo="'+(pageNo-1)+'" data-perPage="'+perPage+'">&#139;</a></li>';
        }
        // Loop through pages, adding numeric anchors.
        for(var i=0; i<pages; i++) {
            var currentPage = (i+1), active = '';
            // Set the current page classification.
            if (currentPage === pageNo)
                active = 'active';
            // Display ONLY 3 numbers before and after the current page.
```

```
            if (currentPage >= (pageNo-3) && currentPage <= pageNo || currentPage <= (pageNo+3) &&
currentPage >= pageNo)
                pagNav += '<li><a href="products/page='+currentPage+filter+'" class="pagItem
'+active+'" data-pageNo="'+currentPage+'" data-perPage="'+perPage+'">' + currentPage + '</a></li>';
        }
        // If page last page, do NOT apply anchors to last 2 elements.
        if(pageNo === pages) {
            pagNav += '<li class="pagItemInactive">&#155;</li>';
            pagNav += '<li class="pagItemInactive">&raquo;</li>';
        } else {
            pagNav += '<li><a href="products/page='+(pageNo+1)+filter+'" class="pagItem" data-
pageNo="'+(pageNo+1)+'" data-perPage="'+perPage+'">&#155;</a></li>';
            pagNav += '<li><a href="products/page='+pages+filter+'" class="pagItem" data-
pageNo="'+pages+'" data-perPage="'+perPage+'">&raquo;</a></li>';
        }
        // Add the new list elements to the document.
        ul.innerHTML = pagNav;
        // Add event listeners for these newly added elements.
        addEventListeners('products-pagination');
    },
    // Once ready state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Before request is sent, display loader.
    loader(true);
    // Open & send request.
    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
},
// Show product list.
showProductList = function (status, pageNo, perPage) {
    var start = '', url, xhr = new XMLHttpRequest(),
    // On success, add request response to content.
    success = function () {
        var response = JSON.parse(xhr.responseText), li = '', colourString,
            content = document.getElementById("products_list"),
            publishOption, draftOption, trashOption, product_group;
        // If API returns error.
        if(response.error.thrown) {
            showMessage(response.error.message, "error");
        } else {
            li += '<li class="product-list thead">';
            li += '<div class="product-sku">Sku</div>';
            li += '<div class="product-title">Title</div>';
            li += '<div class="product-quantity">Quantity</div>';
            li += '<div class="product-post-status">Status</div>';
            li += '</li>';
            // Loop through each result.
            for(i in response.product_group) {
                product_group = response.product_group[i];
                publishOption = ""; draftOption = ""; trashOption = "";
                // Set up table body.
                li += '<li class="product-list '+product_group.sku+'">';
                li += '<div class="product-sku">' + product_group.sku + '</div>';
                colourString = product_group.colour !== "" ? ' <em>(' + product_group.colour + ')</
em>' : '';
                li += '<div class="product-title"><a href="product/'+product_group.sku+'"
class="product-item" data-sku="' + product_group.sku + '">' + product_group.title + '</
a>'+colourString+'</div>';
                li += '<div class="product-quantity"><ul>';
                for(i in product_group.product) {
                    var product = product_group.product[i],
                        lowClass = '';
                    if(product.stock<10)
                        lowClass = 'low';
```

```
                    li += '<li class="'+lowClass+'">'+ product.stock + ' <em>' + product.value +
'</em></li>';
                }
                li += '</ul></div>';
                li += '<div class="product-post-status"><select class="products-status" data-sku="'
+ product_group.sku + '">';
                // Find which option should be selected.
                switch(product_group.post_status) {
                    case 'publish':
                        publishOption = "selected"; break;
                    case 'draft':
                        draftOption = "selected"; break;
                    case 'trash':
                        trashOption = "selected"; break;
                    default:
                        publishOption = "selected"; break;
                }
                li += '<option value="publish" ' + publishOption + '>Publish</option>';
                li += '<option value="draft" ' + draftOption + '>Draft</option>';
                li += '<option value="trash"' + trashOption + '>Trash</option>';
                li += '</select></div>'
                li += '</li>';
            }
        }
        // Hide loader before content is filled.
        loader(false);
        // Fill content with new data.
        content.innerHTML = li;
        // Add event listeners for these newly added elements.
        addEventListeners('products-list');
    },
    // Once ready state is changed, check status.
    stateChanged = function () {
        if(xhr.readyState === 4) {
            switch(xhr.status) {
                case 200:
                    success(); break;
                default:
                    showMessage("Status "+xhr.status+" returned.", "error"); break;
            }
        }
    };
    // Display loader will request is sent and retrieved.
    loader(true);
    // Initialise where the API should start getting records from.
    start = (pageNo - 1) * perPage;

    url = '../api/v.1/view/range.php';
    url += '?status='+status+'&start='+start+'&show='+perPage;

    xhr.open("GET", url, true);
    xhr.send(null);
    xhr.onreadystatechange = stateChanged;
};
```

admin/eventHandler.js

```
// Function to add relevant event listeners
function addEventListeners (eventsFor) {
    // If events are for product pagination, apply those.
    switch(eventsFor) {
        case 'products':
            // The new elements to add listeners to.
            var ten = document.getElementById('ten'),
                all = document.getElementById('all'),
                addProductSection = document.getElementById('add-product-section'),
                filterButtons = document.getElementsByClassName("filter-table");
            // Loop through the filter buttons, adding listeners.
            for(var i=0, len=filterButtons.length; i<len; i++) {
                addListenerGetPage(filterButtons[i]);
            }
            // Add event listeners to the other buttons on the page.
            ten.addEventListener('click', function (e) {
                showProducts(getFilterCriteria(document.URL), 1, 10);
                e.preventDefault();
            }, false);
            all.addEventListener('click', function (e) {
                showProducts(getFilterCriteria(document.URL), 1, 9999);
                e.preventDefault();
            }, false);
            addListenerGetPage(addProductSection);
            break;

        case 'products-pagination':
            var pagItems = document.getElementsByClassName('pagItem');
            for(var i=0, len=pagItems.length; i<len; i++) {
                addListenerGetPage(pagItems[i]);
            }
            break;

        case 'products-list':
            var productItems = document.getElementsByClassName('product-item'),
                productsStatus = document.getElementsByClassName('products-status');
            // SINGLE PRODUCT LINKS
            for(var i=0, len=productItems.length; i<len; i++) {
                addListenerGetPage(productItems[i]);
            }
            // EVENT FOR CHANGING OPTION ON DROPDOWN
            for(var i=0, len=productsStatus.length; i<len; i++) {
                addProductChangeEvent(productsStatus[i]);
            }
            function addProductChangeEvent (elem) {
                elem.addEventListener('change', function (e) {
                    updateProductStatus(elem);
                }, false);
            }
            break;

        case 'update-product':
            var updateProductButton = document.getElementById("update-product-button"),
                addValueStockButton = document.getElementById("add-value-stock");
            // When the user specifies the product is to be updated,
            // check the data being posted is valid, and then make the request.
            updateProductButton.addEventListener('click', function (e) {
                var valid = validateProductInput();
                if(valid)
                    updateProduct(updateProductButton.dataset.sku);
                e.preventDefault();
            }, false);
            // Add two new inputs, one for a new size and one for a new stock,
            // by calling a function to do the heavy work.
            addValueStockButton.addEventListener("click", function (e) {
                addValueStockInputs();
                e.preventDefault();
            }, false);
            break;

        case 'add-product':
            var addProductButton = document.getElementById("add-product-button"),
```

```
        addValueStockButton = document.getElementById("add-value-stock");
    // When the user specifier the product is to be added,
    // check the data being posted is valid, and then make the request.
    addProductButton.addEventListener("click", function (e) {
        var valid = validateProductInput();
        if(valid)
            addProduct();
        e.preventDefault();
    }, false);
    // Add two new inputs, one for a new size and one for a new stock,
    // by calling a function to do the heavy work.
    addValueStockButton.addEventListener("click", function (e) {
        addValueStockInputs();
        e.preventDefault();
    }, false);
    break;

case 'categories':
    var categoryFilter = document.getElementsByClassName("filter-table"),
        status = document.getElementsByClassName("category-status");

    for(var i=0, len=categoryFilter.length; i<len; i++) {
        addListenerGetPage(categoryFilter[i]);
    }
    for(var i=0, len=status.length; i<len; i++) {
        addCategoryChangeEvent(status[i]);
    }
    function addCategoryChangeEvent (elem) {
        elem.addEventListener("change", function (e) {
            updateCategoryStatus(elem);
        }, false);
    }
    break;

case 'add-category':
    var addCategoryButton = document.getElementById("add-cat-button");

    // Event listener to listen for when the user wants to add a category.
    addCategoryButton.addEventListener('click', function (e) {
        var valid = validateCategoryInput();
        if(valid)
            addCategory();
        e.preventDefault();
    }, false);
    break;

case 'orders':
    var markDispatched = document.getElementById('mark-dispatched');
    // Mark the order as dispatched.
    markDispatched.addEventListener('click', function () {
        markOrderAsDispatched(markDispatched.dataset.id);
    }, false);
    break;

case 'settings':
    var companyName = document.getElementById('site-name'),
        companyNameBtn = document.getElementById('site-name-button'),
        resetDB = document.getElementById("reset-database"),
        setDefault = document.getElementById("set-default-picture");

    companyName.addEventListener('keydown', function (e) {
        if(e.keyCode === 13 && companyName.value !== '') {
            setCompanyName();
        }
    }, false);

    companyNameBtn.addEventListener('click', function (e) {
        setCompanyName();
        e.preventDefault();
    }, false);

    // Event to fire when default picture is pressed.
    setDefault.addEventListener('click', function (e) {
        setDefaultPicture();
```

```
                e.preventDefault();
            }, false);

            // The event to fire when the user requests
            resetDB.addEventListener('click', function (e) {
                if (confirm('You are about to wipe the database clean, are you sure?')) {
                    truncateDbTables();
                }
                e.preventDefault();
            }, false);

        default: break;
    }
}
```