

Code Convention

Html, CSS, JS

Code Convention Html, CSS, JS

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| No | Date | Content | Author |
| 1 | 26.10.2018 | First version | Toan |
| 2 |  |  |  |

# Approvals

|  |  |  |  |
| --- | --- | --- | --- |
| No | Date | Role | Approved By |
|  |  |  |  |

# **Table of Contents**

Html, CSS, JS

[Revision History 1](#_Toc528328386)

[Approvals 1](#_Toc528328387)

[**Table of Contents** 2](#_Toc528328388)

[**Table of Figures** 2](#_Toc528328389)

[Introduction 3](#_Toc528328390)

[Purpose 3](#_Toc528328391)

[Related Documents 3](#_Toc528328392)

[Html, CSS, JS 4](#_Toc528328393)

[1 HTML 4](#_Toc528328394)

[**Creating new page** 4](#_Toc528328395)

[**Naming element ID** 4](#_Toc528328396)

[CSS 5](#_Toc528328397)

[**Naming class, id** 5](#_Toc528328398)

[**Styles must be nested in a root** 5](#_Toc528328399)

[**Same CSS styles** 5](#_Toc528328400)

[JS 6](#_Toc528328401)

[**Package JS** 6](#_Toc528328402)

[**Using URL/Links** 6](#_Toc528328403)

[**Using translated captions** 6](#_Toc528328404)

# **Table of Figures**

No table of figures entries found.

# Introduction

## Purpose

The purpose of this document is describing about our code conventions

## Related Documents

-

# **Html, CSS, JS**

## HTML

|  |  |  |  |
| --- | --- | --- | --- |
| No | As a | I want to | So that |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| No | Given | When | Then |
|  |  |  |  |

### Creating new page

When creating a new page (html, cshtml), remember to create the extra files: CSS, JS.

+ E.g:

EquipmentList.cshtml ; EquipmentList.css ; EquipmentList.js

EquipmentDetail.cshtml ; EquipmentDetail.css ; EquipmentDetail.js

+ Note: If the page is too simple/small such as an introduction dialog, we can contain all together in one page (but still divide each section)

### Naming element ID

All HTML element id of pages should be unique so that we can quickly find the HTML page in the source code exactly.

+ E.g:

<div id="pageEquimentDetail">

...

</div>

## CSS

|  |  |  |  |
| --- | --- | --- | --- |
| No | As a | I want to | So that |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| No | Given | When | Then |
|  |  |  |  |

### Naming class, id

Naming for HTML element id, class.

+ Element Id such as: equipmentDetail ; producInfo ;

+ Element class such as: equipment-detail ; product-info ;

E.g:

<div class="equipment-info">

...

</div>

### Styles must be nested in a root

All the CSS styles on a page should have the root id so that we can prevent having bad layouts (conflict CSS) when switching between pages.

+ E.g:

<style>

#equipmentList .icon { ... }

#equipmentList .item .description { ... }

</style>

### Same CSS styles

If there are the same CSS styles:

+ They are used on a few pages, we can duplicate the CSS styles so that it would be easy to adjust each in the future.

+ They are standard (stable or ensure they won't be changed) and would be used on many pages, we should put them into the Common Styles file.

## JS

|  |  |  |  |
| --- | --- | --- | --- |
| No | As a | I want to | So that |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| No | Given | When | Then |
|  |  |  |  |

### Package JS

The javascript should be wrapped into an object to manage easily and have an 'init' function:

var EquipmentDetailJS = { //the JS package for the Equipment Detail page.

init: function () {

//do something here.

},

totalValue: 0, //variable.

calculateTotal: function () { //some actions

},

};

function calculateTotal() {

//Note: We should **not** define drift/float functions like this on a page.

};

### Using URL/Links

In JS section, use this global JS method:

gGetUrl("ActionName", "ControllerName")

+ E.g:

<script>

var urlGetEquip = gGetUrl("GetEquipment", "Equipment");

//instead of C# method:

var urlGetEquip = '@Url.Action("GetEquipment", "Equipment")';

</script>

### Using translated captions

We should define the translated captions on page and separate in order to re-use easily in the JS section or file.

+ E.g:

<script src="~/Scripts/pages/Equipment/EquipmentDetail.js")"></script>

<script>

var EquipmentDetailCaptions = {

SerialNum: '@Html.CustomDisplay(Forms.Equipment, "SerialNum")',

SelectProd: '@Html.CustomDisplay(Forms.Equipment, "SelectProd")',

};

EquipmentDetail.init();

</script>

### Comment

Give a general comment in the JS method to summarize the content and some parameters if it has.

+ E.g:  
<script>

function selectItemByKey(controlElement, itemObject, primaryKeyName) {

///<summary>Select/Open and scroll to a specific item in list.</summary>

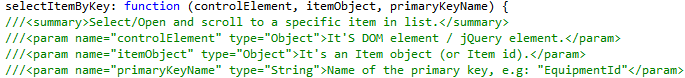
///<param name="controlElement" type="Object">It’s DOM element / jQuery element.</param>

///<param name="itemObject" type="Object">It’s an Item object (or an Item id).</param>

///<param name="primaryKeyName" type="String">Name of the primary key, e.g: "EquipmentId"</param>

</script>

When it is placed in the JS object, it would be like this:



## HTML

|  |  |  |  |
| --- | --- | --- | --- |
| No | As a | I want to | So that |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| No | Given | When | Then |
|  |  |  |  |

**Naming convention**

Use breakline for multiple-option tag:

E.g:

<div [attr.id]="filter.UniqueId" class="item-content flex-grow-1"

[ngbPopover]="popoverContent"

placement="left"

[autoClose]="false"

#popover="ngbPopover"

popoverClass="personnel-popover edit-item"

container="body"

triggers="manual"

(click)="TriggerPopover(popover, filter)"></div>

## SaSS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No |  | As a | I want to | So that |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| No | Given | When | Then |
|  |  |  |  |

**Naming convention**

Using kebab-case for naming class, id

E.g: #product-item, .personnel-details….

Sass style must be 2-nested level max

Use short hand for multiple CSS properties

Use hex colors.

**Do not use inline CSS;**

E.g:

.personnel-list {

background: #fff url(‘../images/demo.png’) no-repeat top left;

.personnel-item {

.name {

Border: 1px solid #000;

}

.address {

}

}

}

## Typescript

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No |  | As a | I want to | So that |
|  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| No | Given | When | Then |
|  |  |  |  |

**Naming convention**

Class, Interface…. Must follow Pacal Case

Use prefix I for interface

E.g: IChildView

Use suffix Model for model class

E.g: SetupFieldModel

Non-private variables must follow Camel Case convention

Use prefix is for Boolean variables.

E.g: columnFields, isLoading…

Use underscore for private variables

E.x: \_listResult

**Layout**

class MyClass {

if (something) {

// ...

} else if (somethingElse) {

// ...

} else {

// ...

}

}

Braces are required around the statements for a conditional.

if (condition) {

//…

}

**Comment**

Add comment on the top of function

Function should follow Camel Case convention

Add history to comment after fixing bugs: // Refer to bug #123, fixed by Cuong Le

E.g

/\*\*

\* Sort data by field

\* @param sortField field to sort

\* @param ascending ascending or descending

\*/

// Refer to bug #123, fixed by Cuong Le

// Refer to bug #111, fixed by Cuong Le

sortByColumn(sortField: string, ascending: boolean) {

}