Naming Conventions

Subject	Training Program
Date	07 October 2020
Author	Tarun Verma
Version	1.0
Status	[Status]
Category	Training and Development
Last Modified	

Yukti Solutions Private Limited
1001 C, 11th Floor, Westend Mall, District Center, Janakpuri,
New Delhi – 110058.
Web. https://yuktisolutions.com
Email. info@yuktisolutions.com
Phone. +91 11 47074063

Contents

Introduction

In computer programming, a naming convention is a set of rules for choosing the character sequence to be used to define variables, types, functions, classes and other entities name in source code.

Reasons for using a naming convention:

- To reduce the effort needed to read and understand source code
- To focus on more important issues than arguing over syntax and naming standards.

Why do we need to follow the naming conventions?

Removing spaces between words

In programming, we often remove the spaces between words because programs of different sorts reserve the space (' ') character for special purposes. Because the space character is reserved, we cannot use it to represent a concept that we express in our human language with multiple words.

No additional comments required

If an element's or function name is defined properly then there is no need to add any special comment for that. A good name itself tell you the purpose of it e.g. GetUserDetail, SetUserDetail, DeleteUserAccount, RemoveAllUsersAccount.

We must use speaking names only

We must always use speaking names only. For example, if one stored procedure in SQL server calculate the salary of an empty then its name and parameters name should be something like Exec_CalculateEmployeeSalary(@employeeId,@month,@year). This way one can easily understand that this stored procedure is being written to calculate the salary of an employee for a particular month and year based on its employee id.

Some more example could be

- DeleteUserAccount
- SaveCompanyDetails

We must have to use complete names and not use short names e.g. Del, Reg. etc. Short names might give confusing meanings and might not convey the right purpose. While a complete name conveys the exact purpose. For example:

- Bad example: DelCompDt(@cmpId)
- Good example: DeleteCompanyDetails(@companyId)

Naming conventions

There are many other naming conventions but we will only make use of following four naming conventions.

- **P**ascal**C**ase
- **c**amel**C**ase
- snake**_**case
- kebab**-**case

PascalCase

Pascal**C**ase is a naming convention in which the first letter of each word in a compound word is capitalized

Some examples

- **N**ew**O**bject;
- **L**ong**F**unction**N**ame()

camelCase

camel**C**ase is a naming convention in which the first letter of each word in a compound word is capitalized, except for the first word.

Some examples

- **n**ew**S**tring;
- **g**et**N**ew**S**tring()

snake_case

snake_case combines words by replacing each space with an underscore (_). You can use either small letter or capital letters but not combination of both.

Some examples

- user_login_count or USER_LOGIN_COUNT

kebab-case

kebab-case combines words by replacing each space with a dash (-). You can use either small letter or capital letters but not combination of both.

Some examples

- user-login-count or USER**-**LOGIN**-**COUNT

Which one is the best?

There is no best method of combining words. The main thing is to be consistent with the convention used, and, if you're in a team, to come to an agreement on the convention together.

- All the following cases are valid
  - **w**hich**I**s**B**est

- **W**hich**I**s**B**est
- which_is_best or WHICH_IS_BEST
- which-is-best or WHICH-IS-BEST

Areas must cover

We can apply these naming conventions to various areas but we must apply these practices to following areas

- JavaScript and jQuery
  - Variables and function names
- CSS and SCSS
  - Class names
- HTML
  - Id and Class attribute values
- SQL
  - Database, tables, views, functions, stored procedure, triggers, indexes, relationships names
- C#
  - Namespaces, classes, variables, properties, methods, constants names
- MVC
  - Project, Controllers, Actions, Models, Views, Partial Views, Folder and Files names.

JavaScript and jQuery

In JavaScript we can have variables and functions but at different scope. We will use the naming conventions based on variable scope here as follows

A sample program

```
<script type="text/javascript">
    var ProjectName = 'CRM';
    function GetEmployeeName(employeeId) {
        var employee_name = 'Tarun Verma';
        return employee_name;
    }

    var Company = {
        completeName: 'Yukti Solutions Private Limited',
        phoneNumber: '9718292704',
        getCompanyDetails: function(companyId) {
            var company_detail = {};
            return this;
        }
    };
</script>
```

- **P**ascal**C**ase
  - Public variables: which are defined outside a JavaScript object e.g. **P**roject**N**ame.

- Public functions: which are defined outside a JavaScript object e.g. **G**et**E**mployee**N**ame

- **c**amel**C**ase
  - Function parameters for example GetEmployeeName(**e**mployee**I**d), getCompanyDetail(**c**ompany**I**d)
  - Object's inner properties for example, **c**omplete**N**ame, **p**hone**N**umber etc.
  - Object's inner methods for example **g**et**C**ompany**D**etail(companyId)
- snake_case
  - Function scope variables which can be used within this function only e.g. employee_name, company_detail. These variables cannot be used outside the function scope.

CSS and SCSS

In CSS and SCSS, we have just class names here with kebab-case only.

- CSS examples
  - `.user-grid{ /* some style definition here..*/ }`
  - `.user-grid .row { /* some style definition here..*/ }`
- SCSS examples
  - `.user-grid { .row { /* some style definition here..*/ } }`

Please note that we can also make use of a simple case here e.g. "**.row"** but whenever there are two words combined we always use kebal-case e.g. "**.user-grid"**.

Important things to remember

We only use kabab-case here but still we must take care of some important things here.

Define class for the container div or element only

Always define a class at a container div or other html element which can then further define the style against its children. For example:

```
<div class="user-grid">
    <table>
        <tr><th>Title</th></tr>
        <tr><td>Text</td></tr>
    </table>
</div>
```

```
.user-grid {
    table {
        tr {
            // style definition of a row
            th {
                    // style definition of a th
                td {
                    // style definition of a td
                }
            }
        }
    }
}
```

As you can see, we only defined one class name but still it can cover the definition of each children inside it e.g. table, tr, th, td etc.

Use specific class names only when required

Give names to children only when there are multiple children and it is difficult to style one of those without a specific name. For example

```
<div class="user-grid">
    <table>
        <tr><th>Title</th></tr>
        <tr><td>Text</td></tr>
        <tr class="alt-row"><td>Text Alternate</td></tr>
    </table>
</div>
.user-grid {
    table {
        tr {
            // style definition of a row
            th {
                // style definition of a th
                td {
                    // style definition of a td
                }
            }

            &.alt-row {
                // style definition of an alternate row
            }
        }
    }
}
```

As you can see, we had to define the alt-row class here to give a special appearance to alternate rows within a table.

HTML

In case of HTML we just need to take care of id and class attribute values.

- For an id attribute we use the **P**ascal**C**ase e.g. **U**ser**L**ist, **C**ompany**L**ist etc. These are unique throughout your html page.
- For a class attribute we use the kebab-case e.g cs**-**list, js**-**list etc. These can be duplicate on the same page.

A sample code block

```
<div id="UserList" class="cs-list js-list"></div>
<div id="CompanyList" class="cs-list js-list"></div>
```

Important things to remember

- Id attribute must be used only when we need to identify an html element uniquely within the page and get or set its value through JavaScript.
- class attribute must be used when there is a possibility of duplicate html elements on the same or other pages.
    - when we define a class attribute, we also need to take case of following two cases.

- Prefix **cs-*** with custom identifier but only when class name is a style identifier For example: <div class="**cs-list** js-list"></div>
- Prefix **js-*** with custom identifier but only when class name is a JavaScript identifier. For example: <div class="cs-list **js-list**"></div>

Using these prefixes, we can easily distinguish between a designer scope and programmer scope. cs-* prefix is only for the designer while js-* is only for the programmer. This way one element can be managed in two different ways without interfering into another person work or any standard library code.

For designer

```
<style type="text/css">
    .cs-list {
        color: #fff;
        background-color: #000;
    }
</style>
```

For developer

```
<script type="text/javascript">
    $.each('.js-list', function (index, element) {
        // write your code here..
    })
</script>
```

SQL

In SQL server we have a long list of entities. Database, tables, views, functions, stored procedure, triggers, indexes, relationships etc.

- PascalCase case be used for
  - Database name: **W**eb**P**ortal**D**b
  - Tables name: **C**ompany**U**sers, **U**serRoles
  - Views name: **V**iew**C**ompanyList, **V**iew**U**ser**L**ist
  - Function name: **G**et**C**ompany**D**etail, **S**et**U**ser**D**etail
- CamelCase and snake_case can be used for
  - Procedure name: **E**xec_**C**ompanyRegistration, **E**xec_**U**ser**R**egistration
  - Trigger name: **T**rg_**C**ompany_**U**pdate, **T**rg_**U**ser_**D**eletion
  - Index name: **IX**_**C**ompany**S**earch
  - RelationShip: **R**elation_**C**ompany**T**o**U**ser
- Use only camelCase in following cases
  - Function or procedure parameter names e.g. @**u**ser**I**d, @**c**ompany**I**d, @**s**tart**D**ate, @**e**nd**D**ate
  - Procedures names which are not called directly but within other procedures e.g. **e**xec_**C**ompany**M**eetings, **e**xec_**U**ser**S**alary**F**or**T**he**M**onth.

C#

In C# we can define names for namespaces, classes, variables, properties, methods, constants, enums etc.

- We should use PacalCase for
  - Namespace e.g. **C**rm.**W**eb, **C**rm.**W**eb.**S**ervice, **C**rm.**W**eb.**A**pi, **C**rm.**D**ata**S**tore
  - Class e.g. **C**ompany**D**etail
  - Public Variables or Properties e.g. **C**ompany**N**ame, **C**ompany**A**ddress
  - Public Methods e.g. **G**et**C**ompany**D**etail(…), **S**et**C**ompany**D**etail(…)
  - Enums e.g. enum **U**ser**T**ypes { **A**dmin, **O**perator, **G**uest, **C**ompany**A**dmin }
- We should use camelCase for
  - Private Variables e.g. **f**ull**N**ame
  - Private Methods e.g. **g**et**U**ser**D**etail**B**y**I**d(…)
  - Method parameter names e.g. GetCompanyList(int **p**age**I**ndex, string **s**ort**C**olumn, string **s**ort**O**rder);
- We should use snake_case for
  - Variables which are defined within a function scope and cannot be used outside the function e.g. public string GetUserNameById(string userId){var **u**ser_**n**ame =null; user_name = "Tarun Verma"; return user_name;}
- Constants e.g. const string DEFAULT_DATE_FORMAT = "dd/MM/yyyy";

MVC

In MVC we can define naming conventions for Project, Controllers, Actions, Models, Views, Partial Views, Folder and Files names.

Pascal**C**ase should be used for

- ProjectName e.g. **W**eb**P**ortal
- Controllers e.g. **A**ccount**C**ontroller, **H**ome**C**ontroller
- Actions e.g. **I**ndex, **C**ompany**L**ist
- Models e.g. **C**ompany**V**iew**M**odel, **A**ccount**V**iew**M**odel
- Views e.g. **L**ogin.cshtml, **U**ser**R**egistration.cshtml
- Folder e.g. **U**ploaded**F**iles, **I**mported**F**iles
- Files e.g. **B**undle**C**onfig.json, **W**eb**C**ompiler.json, **C**ompany.js, **U**ser**L**ist.js

camel**C**ase should be used for partial views along with a underscope as prefix to distinguish it with a normal view file e.g. _**c**ompany**D**etails.cshtml, _**h**eader.cshtml,