네이버 코딩 컨벤션 (JavaScript)

Ver 0.3.1

저작권

목차

1. 들여쓰기 (Indentation)

1.1 메소드 체인이 길어지는 경우 적절히 사용한다.

나쁜 예.

```
$("#items").find(".openLayer").slideUp().end().find(".closeLayer").slideDown();
```

좋은 예.

```
$("#items")
 .find(".openLayer")
  .slideUp()
  .end()
 .find(".closeLayer")
  .slideDown();
```

2. 중괄호 (Brace)

중괄호는 클래스, 메서드, 제어문의 블럭을 구분한다.

2.1 줄의 끝에서 중괄호 시작 (requireSpacesInFunction)

중괄호는 클래스 선언, 메서드 선언, 조건/반복문/제어문,줄의 마지막에서 시작한다.

나쁜 예.

```
var Empty = function()
{
}
```

좋은 예.

```
var Empty = function() {

}

switch (type) {
 case 0 :
  break;
 case 1 : {
  break;
 }
 default :
  common();
}

if (true) {
 return;
} else if (false) {
 return;
} else {
}
```

2.2 조건/반복문/제어문에 중괄호 사용

조건/반복문/제어문이 한줄로 끝이라도 중괄호를 활용한다.

나쁜 예.

```
if (exp == null) return false;

for (var i in obj) if ( i === "key" ) return obj[i];
```

좋은 예.

```
if (exp == null) {
 return false;
}
```

```
for (var i in obj) {
 if ( i === "stop" ) {
  return obj[i];
 }
}
```

3. 줄바꿈 (Line Wrapping)

줄 바꿈은 작성한 명령어가 줄 너비를 초과했을 경우 코드 가독성을 위해서 강제로 줄을 바꾸는 것을 말한다. 명령문을 길게 작성하면 소스 코드 분석이 어려워 진다.

3.1 최대 줄 너비는 100

고해상도 모니터(해상도 1280*1024)사용이 보편화 됨에 따라, 최대 줄 사용 너비는 100자까지 가능하다.

3.2 줄

- 변수, 파라미터 등의 경우에는 쉼표(,) 다음에 줄 바꿈을 한다.
- 메서드 체이닝을 하여 줄바꿈을 할 때 마침표(.) 다음에 줄 바꿈을 한다.
- 연산식의 경우에는 연산자 후에 줄 바꿈을 한다.
- 시작 소괄호(( )의 경우에는 시작 소괄호(( ) 전에 줄 바꿈을 한다.
- 줄 바꿈 후에는 가독성을 위하여 자동 들여쓰기를 한다.
- 상위 레벨의 깊이에 맞게 들여쓰기를 한다

좋은 예.

```
other.bar(100, 200, 300, 400,
500, 600, 700, 800, 900);

var sum = 100 + 200 + 300 + 400 +
500 + 600 + 700 + 800;

var arr = [100, 200, 300, 400,
500, 600, 700, 800, 900];
```

3.3 긴 문자의 경우 줄바꿈시 escape 문자 금지(₩)

escape문자 대신 +연산사를 사용한다.

나쁜 예.

```
var text = "Hello\
World";
```

좋은 예.

```
var text = "Hello" +
 "World";
```

4. 주석(Comments)

4.1 복수행 주석

/** … */을 사용하고 함수의 설명과 모든 매개 변수와 반환 값에 대한 형식과 값을 설명한다.

나쁜 예.

```
// extend는 Class를 상속할 때 사용합니다.
// @static
// @method eg.Class.extend
// @param {eg.Class} 상속하려는 클래스
// @param {Object} 리터럴 형태의 클래스 정의부
// @return {Class}
// @example
//   var Some = eg.Class.extend(eg.Component,{
//   "some": function(){}
//  });
```

좋은 예.

```
/**
 * extend는 Class를 상속할 때 사용합니다.
 * @static
 * @method eg.Class.extend
 * @param {eg.Class} 상속하려는 클래스
 * @param {Object} 리터럴 형태의 클래스 정의부
 * @return {Class}
 * @example
  var Some = eg.Class.extend(eg.Component,{
  "some": function(){}
  });
 */
```

4.2 한줄 주석

복수행을 제외한 상황에서는 "//" 한줄 주석을 사용한다. 코드 끝엔 한 줄 주석 지양한다.

나쁜 예.

```
/* super 속성을 지원하지 않습니다. */
extendClass.$super = oSuperClass.prototype;
```

좋은 예.

```
// super 속성을 지원하지 않습니다.
extendClass.$super = oSuperClass.prototype;
```

5. 빈 줄(Blank Lines)

빈 줄은 명령문 그룹의 영역을 표시하기 위하여 사용한다. 특정 로직을 처리하는 수 많은 명령문이 구분 없이 혼재하여 있거나 제어문과 명령문 사이의 영역을 구분하지 않으면 소스 코드 분석이 어려워진다.

5.1 함수 선언 후 빈 줄 사용 지양

함수를 선언한 경우에는 다음에 빈 줄을 삽입하지 않는다.

나쁜 예.

```
function _getXHR(){

 if (window.XMLHttpRequest) {
  return new XMLHttpRequest();
 } else if (ActiveXObject) {
  ...
 }

}
```

좋은 예.

```
function _getXHR() {
 if (window.XMLHttpRequest) { /* (빈 줄 위치) 메소드 선언 후 다음 줄에 빈 줄을 삽입하지 않는다. */
  return new XMLHttpRequest();
 } else if (ActiveXObject) {
  ...
 }
}
```

5.2 함수 선언 간, 변수 선언 후 빈 줄 사용 준수

나쁜 예.

```
var panel; var options; var children;
var padding = options.previewPadding.concat(),
 prefix = options.prefix,
 horizontal = options.horizontal;
function foo() {

}
function bar() {

}
```

좋은 예.

```
var panel;
var options;
var children;
var padding = options.previewPadding.concat();
var prefix = options.prefix;
```

```
var horizontal = options.horizontal;

function foo() {

}

function bar() {

}
```

5.3 명령문, 제어문간 빈 줄 사용 준수

좋은 예.

```
req.open(opt.method.toUpperCase(), this._url, true);
req.setRequestHeader("charset", "utf-8");
/* (빈 줄 위치) 명령문과 제어문 사이에는 빈 줄을 삽입한다. */
for (var x in this._headers) {
 if (typeof this._headers[x] == "function") {
  continue;
 }
 /* (빈 줄 위치) 명령문과 제어문 사이에는 빈 줄을 삽입한다. */
 req.setRequestHeader(x, this._headers[x]);
}
/* (빈 줄 위치) 명령문과 제어문 사이에는 빈 줄을 삽입한다. */
if (typeof req.onload != "undefined") {
 req.onload = function(rq) {
  clearTimeout(_timer);
  t._onload(rq)
 };
} else {
 req.onreadystatechange = function(rq) {
  clearTimeout(_timer);
  t._onload(rq)
 };
}
```

6. 새 줄(New Lines)

6.1 오브젝트 리터럴은 키와 값을 한 쌍으로 새 줄을 삽입한다.

나쁜 예.

```
var option = {
 min: [ 0, 0 ], max: [100,100],
 margin: 0, circular: false,
 easing: options.panelEffect,
 deceleration: options.deceleration
};
```

좋은 예.

```
var option = {
 min: [ 0, 0 ],
 max: [100,100],
 margin: 0,
 circular: false,
 easing: options.panelEffect,
 deceleration: options.deceleration
};
```

6.2 else if, else, catch, finally, do 문 새 줄 사용 지양

나쁜 예.

```
if (true) {
 return;
}
else if (false) {
 return;
}
else {
 return;
}
```

좋은 예.

```
if (true) {
 return;
} else if (false) {
 return;
} else {
 return;
}
```

6.3 한 줄 명령문 새 줄 사용 준수

6.4 함수의 빈 블록, 빈 몸체 새 줄 사용 준수(for, do..)

나쁜 예.

```
function Empty() {}
```

좋은 예.

```
function Empty() {
}
function Example() {
 var listener = new Listener({});
 foo = function() {
  do {
  } while(false);
  for(;;) {
  }
 }
}
```

6.5 파라미터, 닫는 괄호간 새 줄 사용 지양

나쁜 예.

```
var newArray = array.filter(function(v, i, o) { return v > 50; } );
```

좋은 예.

```
var newArray = array.filter(function(v, i, o) {
 return v > 50;
});
```

7. 공백 (White spaces)

7.1 공백은 탭을 사용한다.

나쁜 예.

```
function() {
####var name;
}

function() {
#var name;
}
```

좋은 예.

```
function() {
 var name;
}
```

7.2 중괄호({})의 앞에 공백을 하나 넣는다.

나쁜 예.

```
function(){

}
var obj ={

};
```

좋은 예.

```
function() {

}
var obj = {

};
```

7.3 단항 연산자(!, ++..)와 피연산자 사이에 공백을 두지 않는다.

나쁜 예.

```
! isUp;
++ i;
i ++;
```

좋은 예.

```
!isUp;
++i;
i++;
```

7.4 산술 연산자, 비교 연산자 앞,뒤에 공백을 추가한다.(requireSpaceBeforeBinaryOperators, requireSpaceAfterBinaryOperators)

나쁜 예.

```
a+b;
a==b;
```

좋은 예.

```
a + b;
a == b;
```

7.5 종료 구분자(;) 앞에는 공백 사용 지양.

나쁜 예.

```
var obj = []#;
```

좋은 예.

```
var obj = [];
```

7.6 콤마(,)은 뒤에 공백을 삽입한다.

나쁜 예.

```
var arr = ["a","b"];
```

좋은 예.

```
var arr = ["a", "b"];
```

7.7 콜론(:)을 사용하는 경우에는 반드시 뒤에 공백을 삽입한다.(disallowSpaceAfterObjectKeys)

나쁜 예.

```
var obj = {
 "foo":"a",
 "bar":"b"
};
```

좋은 예.

```
var obj = {
 "foo": "a",
 "bar": "b"
}
```

7.8 괄호 안에 공백을 삽입하지 않는다.(disallowSpacesInsideArrayBrackets, disallowSpacesInsideObjectBrackets, disallowSpacesInsideParentheses)

나쁜 예.

```
var obj = { };
var obj = { "a": 2 };
var arr = [ ];
var arr = [ 1, 2 ];
function foo( a, b ) {

}
```

좋은 예.

```
var obj = {};
var obj = {"a": 2};
var arr = [];
var arr = [1, 2];
function foo(a, b) {

}
```

7.9 특정 키워드의 경우 뒤에 공백을 삽입한다. (requireSpaceAfterKeywords)

특정 키워드 는 `do,for,if,else,switch,case,try,catch,void,while,with,return,typeof`이다.

나쁜 예.

```
if(condition){

}
try{

}catch(e){

}
```

좋은 예.

```
if (condition) {

}
try {

} catch(e) {

}
```

8. 이름(Names)

8.1 공통 규칙

영문 사용

소스의 변수명, 클래스명 등에는 영문 이외의 언어를 사용하지 않는다.

한글 발음 사용 금지

한글 발음을 그대로 사용하지 않는다.

''무형자산''이라는 의미의 변수를 예로 들면 아래와 같다.

- 나쁜 예 : `moohyungJasan`
- 좋은 예 : `intangibleAssets`

특수 문자 사용 금지

클래스, 메소드 등의 이름에는 특수 문자를 사용하지 않는다. jQuery변수의 경우 $을 사용하는 것은 예외사항으로 한다.

나쁜 예.

```
funtion $some() {

}
```

네임스페이스, 오브젝트, 함수, 그리고 인스턴스로는 camel case를 사용한다.

나쁜 예.

```
var ajax_history = new AjaxHistory(...);
```

좋은 예.

```
var ajaxHistory = new AjaxHistory(...);
```

8.2 네임스페이스 명명 규칙

소문자 사용 준수

나쁜 예.

```
naver.FOO.bar = function() {

}
```

좋은 예.

```
naver.foo.bar = function() {
```

```
}
```

8.3 클래스명

명사 사용 준수

Class와 생성자에는 Upper camel case(Pascal case)를 사용한다.

나쁜 예.

```
var eventDelegator = eg.Class(...);
```

좋은 예.

```
var EventDelegator = eg.Class(...);
```

8.4 메서드명(Methods)

동사 사용 준수

외부에서 접근할 수 있는 경우, 사용하면 안되는 메서드는 _을 사용한다.

나쁜 예.

```
eg.Class({
 "privateMethod": function() {

 },
 "_publicMethod": function() {

 }
});
```

좋은 예.

```
eg.Class({
 "_privateMethod": function() {

 },
 "publicMethod": function() {

 }
});
```

외부에서 접근할 수 없다면, 메서드에 _을 사용하지 않는다.

나쁜 예.

```
(function(){
 function _privateMethod() {

 }
```

```
 function publicMethod() {
  _privateMethod();
 }
})();
```

좋은 예.

```
(function() {
 function privateMethod() {

 }

 function publicMethod() {
  privateMethod();
 }
})();
```

8.5 변수명

명사 사용 준수

this의 참조를 저장할 때 self 를 사용한다. 깊이가 깊은 경우 적절하게 작명하지만 가능하면 $.proxy, bind을 사용한다.

나쁜 예.

```
var foo = {
 "some": function() {
  var _this = this;
  setTimeout(function() {
   _this.thing();
  },1000);
 },
 "thing": function() {
 }
}
```

좋은 예.

```
var foo = {
 "some": function() {
  var self = this;
  setTimeout(function() {
   self.thing();
  },1000);
 },
 "thing": function() {
 }
}
```

jQuery레퍼런스의 경우 $을 접미사로 일반적인 변수명에 준하여 작성한다.

나쁜 예.

```
var foo = {
 "some": function() {
  var base = jQuery("#base");
 }
}
```

좋은 예.

```
var foo = {
 "some": function() {
  var $base = jQuery("#base");
 }
}
```

8.6 속성명

외부에서 접근할 수 있는 경우, 사용하면 안되는 속성은 _을 사용한다.

나쁜 예.

```
eg.Class({
 "privateState": true,
 "_publicState": false
});
```

좋은 예.

```
eg.Class({
 "_privateState": true,
 "publicState": false
});
```

외부에서 접근할 수 없다면, 변수에 _을 사용하지 않는다.

나쁜 예.

```
(function(){
 var _privateState = true;

 function publicMethod() {
  _privateState;
 }
})();
```

좋은 예.

```
(function(){
 var privateState = true;

 function publicMethod() {
  privateState;
 }
})();
```

오브젝트에 속성으로 접근할 때는 .(점)을 사용한다.

나쁜 예.

```
var a = b["c"];
var a = b["while"];
```

좋은 예.

```
var a = b.c;
var a = b.while;
```

8.7 상수 이름

대문자 사용 준수

나쁜 예.

```
var firefox = 1;
var is_left = true;
```

좋은 예.

```
var FIREFOX = 1;
var IS_LEFT = true;
```

9. 변수 선언

9.1 변수 선언은 상단에 변수당 하나씩 사용한다. (disallowMultipleVarDecl)

Note

let, const의 경우는 블록 스코프이기 때문에 해당 블록상단에 하나씩 사용한다.

나쁜 예.

```
function foo() {
 var i = 0;
 if (i > 0) {
  var j = 0;
 }
}
```

좋은 예.

```
function foo() {
 var i = 0;
 var j = 0;
 if (i > 0) {
  j = 0;
 }
}
```

10. 따옴표 (validateQuoteMarks)

10.1 따옴표는 쌍따옴표를 사용한다. 이스케이프한 경우는 예외로 홑따옴표를 사용할 수 있다.

나쁜 예.

```
var key = 'naver';
var obj = {
 'key': '1'
}
```

좋은 예.

```
var key = "naver";
var obj = {
 "key": "1"
}
```

11. ECMAScript 6+ (ES 2015+) Styles

Note

ECMAScript 6+ style (이하 es6+ style) 챕터 밖 규칙의 '좋은 예'중에서 es6+ style 규칙에 어긋나는 코드는 es6+ style 을 사용하는 경우에만 '나쁜 예'가 되며 본 챕터내 규칙을 적용하여야 한다.

11.1 var 키워드를 사용하지 않는다. (disallowVar)

나쁜 예.

```
var foo;
```

좋은 예.

```
let foo;
const bar;
```

11.2 배열 구조분해를 사용한다. (requireArrayDestructuring)

나쁜 예.

```
var colors = ["red", "green", "blue"];
var red = colors[0];
```

좋은 예.

```
var colors = ["red", "green", "blue"];
var [ red ] = colors;
```

11.3 향상된 오브젝트 리터럴을 사용한다. (requireEnhancedObjectLiterals)

나쁜 예.

```
var bar = true;
var obj = {
 foo: function() {
 },
 bar: bar
};
```

좋은 예.

```
var bar = true;
var obj = {
 foo() {
 },
 bar
};
```

11.4 숫자 리터럴을 사용한다. (requireNumericLiterals)

parseInt()대신 2, 8, 16진수 숫자를 사용한다.

나쁜 예.

```
parseInt("111110111", 2) === 503;
parseInt("767", 8) === 503;
parseInt("1F7", 16) === 255;
```

좋은 예.

```
0b111110111 === 503;
0o767 === 503;
0x1F7 === 503;
```

11.5 속기 화살표 함수를 사용한다. (requireShorthandArrowFunctions)

단일 구문을 반환하는 표현식 함수는 화살표 함수를 사용한다.

나쁜 예.

```
[0, 1, 2].map(v => { return v + 1; });
```

좋은 예.

```
// 단일 구문
[0, 1, 2].map(v => v + 1);

// 다수 구문. 블록 필요
[0, 1, 2].map(v => {
 var s = v + 1;
 return s;
});
```

11.6 오브젝트 구조분해 할당에서 value앞에 공백을 삽입한다. (requireSpaceBeforeDestructuredValues)

나쁜 예.

```
var { foo:objectsFoo } = someObject;
```

좋은 예.

```
var { foo: objectsFoo } = someObject;
```

11.7 spread 연산자를 사용한다. (requireSpread)

spread 연산자를 써야할 자리에 Function.prototype.apply()를 쓰면 안된다.

나쁜 예.

```
var wrap = (f, g) => (...args) => g.apply(g, [f].concat(args))

instance.method.apply(instance, args);
```

좋은 예.

```
var wrap = (f, g) => (...args) => g(f, ...args)

instance.method(...args)
```

11.8 템플릿 문자열을 사용한다. (requireTemplateStrings)

일반 문자열 연쇄가 아닌 템플릿 문자열을 사용한다.

나쁜 예.

```
function sayHi(name) {
    return 'How are you, ' + name + '?';
}
"a" + (b + c)
"a" + a()
```

좋은 예.

```
function sayHi(name) {
    return `How are you, ${name}?`;
}
`a ${b + c}`
`a ${a()}`
```