## ES5代码规范

## 1. Naming Conventions

1.1 Avoid single letter names. Be descriptive with your naming.

// bad

function q() {

// ...stuff...

}

// good

function query() {

// ..stuff..

}

1.2 Use camelCase when naming objects, functions, and instances

// bad

var OBJEcttsssss = {};

var this\_is\_my\_object = {};

var o = {};

function c() {}

// good

var thisIsMyObject = {};

var person = new Person();

function thisIsMyFunction() {}

1.3 Use PascalCase when naming constructors or classes.

// bad

function user(options) {

this.name = options.name;

}

var bad = new user({

name: 'nope'

});

// good

function User(options) {

this.name = options.name;

}

var good = new User({

name: 'yup'

});

1.4 Use a leading underscore \_ when naming private properties

// bad

this.\_\_firstName\_\_ = 'Panda';

this.firstName\_ = 'Panda';

// good

this.\_firstName = 'Panda';

1.5 When saving a reference to this use \_this

// bad

function() {

var self = this;

return function() {

console.log(self);

};

}

// bad

function() {

var that = this;

return function() {

console.log(that);

};

}

// good

function() {

var \_this = this;

return function() {

console.log(\_this);

};

}

## 2. Strings

2.1 Use single quotes '' for strings.

// bad

var name = "Bob Parr";

// good

var name = 'Bob Parr';

## 3. Comparison Operators & Equality

3.1 Use === and !== over == and !=

3.2 Conditional statements

- **Objects** evaluate to **true**

- **Undefined** evaluates to **false**

- **Null** evaluates to **false**

- **Booleans** evaluate to the value of the boolean

- **Numbers** evaluate to false if **+0, -0, or NaN, otherwise true**

- **Strings** evaluate to false if an empty string '', otherwise true

3.3 Use shortcuts.

// bad

if (name !== '') {

// ...stuff...

}

// good

if (name) {

// ...stuff...

}

// bad

if (collection.length > 0) {

// ...stuff...

}

// good

if (collection.length) {

// ...stuff...

}

## 4. Comments

4.1 Use /\*\* ... \*/ for multi-line comments. Include a description, specify types and values for all parameters and return values.

// bad

// make() returns a new element

// based on the passed in tag name

//

// @param {String} tag

// @return {Element} element

function make(tag) {

// ...stuff...

return element;

}

// good

/\*\*

\* make() returns a new element

\* based on the passed in tag name

\*

\* @param {String} tag

\* @return {Element} element

\*/

function make(tag) {

// ...stuff...

return element;

}

4.2 Use // for single line comments. Place single line comments on a newline above the subject of the comment. Put an empty line before the comment.

// bad

var active = true; // is current tab

// good

// is current tab

var active = true;

// bad

function getType() {

console.log('fetching type...');

// set the default type to 'no type'

var type = this.\_type || 'no type';

return type;

}

// good

function getType() {

console.log('fetching type...');

// set the default type to 'no type'

var type = this.\_type || 'no type';

return type;

}

4.3 Prefixing your comments with FIXME or TODO

// FIXME: to annotate problems.

function Calculator() {

// FIXME: shouldn't use a global here

total = 0;

return this;

}

Use // TODO: to annotate solutions to problems.

function Calculator() {

// TODO: total should be configurable by an options param

this.total = 0;

return this;

}

## 5. 示例代码:







