统一前端平台代码规范

V1.2.2-20181015

命名规范

1、 JS 代码中：变量、方法/函数 命名

代码中变量的命令统一使用驼峰格式，并且含量包含业务含义。

```
var a, b, c, myFunc;   // ✗ avoid  命名无业务含义,循环索引 i/j 不在此列
var cust_id;  // ✗ avoid  命名非驼峰
var custId, groupId;     // ✓ ok
```

代码中 this 指针暂存赋值，统一使用**_this ; 如：var _this = this;**

抵制使用： me、self、that 等等其它。统一使用_this;

2、 JS 代码中，定义为可继承的方法/函数，命名以首字母大写的驼峰格式

如何区分，简单来讲，当我们定义的 function 是需要 new 操作符来使用的，请把 function 的首字母大写。否则请参考上述 1 命名规则即可。

3、 HTML、相关自定义 DOM 属性（含 ID/CLASS/自定义属性）、自定义组件标签命名

由于 HTML/CSS 相关代码不区分大小写，故此类命名统一使用 kebab-case (短横线分隔) 命名。如：

<div id="**yu-container-left**" class="**yu-container-left**" **v-seed-id**="111">

4、 功能文件夹命名

文件夹命名原则上**最优先使用小写+横线**分隔方式命名。Url 使用多数情况不区分大小写。

但是，驼峰格式命名暂且也可以。项目组开始阶段，最好统一使用一种规范。

组件定义

业务组件（示例工程/实施项目）

自定义组件统一放置于：custom/widgets/ 目录下

自定义组件以 kebab-case (短横线分隔) 命名（包含文件名与组件名），

结构： yu-xx1-xx2-xx3

xx1 表示项目标识，如 base (框架基础设施)、cib (兴业银行)

xx2 表示组件含义，如 header、container

xx3 表示组件含义（可选），当 xx2 一个单词无法表达时，则可以有多个 xxn 表示

框架基础设施组件，统一用 base 作为 2 级标识，如：

首页：custom/widgets/yu-base-index.js

菜单：custom/widgets/yu-base-menu.js

页签：custom/widgets/yu-base-tabs.js

主页：custom/widgets/yu-base-dashboard.js

项目组自定义业务组件，统一用项目标识（如兴业银行 cib）作为 2 级标识，如

产品树： custom/widgets/yu-cib-product-tree.js

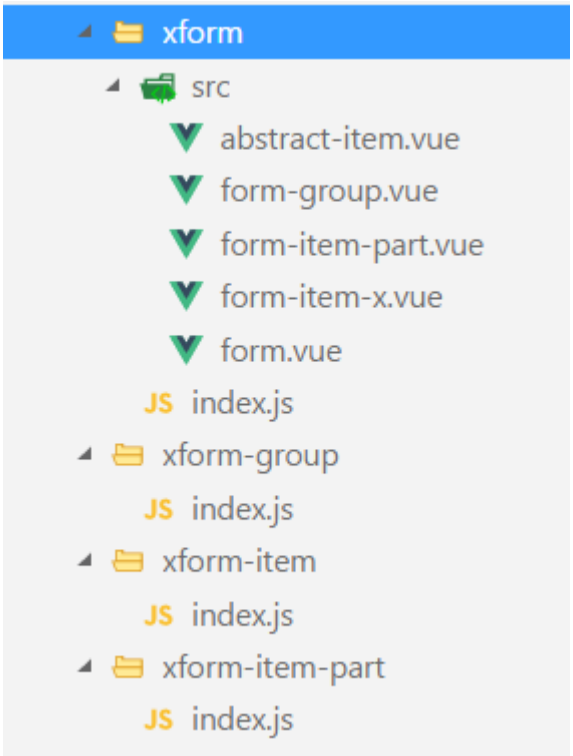
客户群： custom/widgets/yu-cib-customer-group.js

基础组件（组件工程/平台研发）

注：此小节平台实施项目无需关注。

1、 要对外发布的组件统一放置于组件工程 packages/目录下以组件名（短横线分隔命名）为目录

2、 相关子组件实现代码文件全部放置于父级组件 src 目录下，子组件只需定义 index.js 发布即可，如下图 xform-group、xform-item 等



3、 组件定义时，统一指定 name 属性，xtype 属性，其它 name 属性用于兼容老版本注册命名，xtype 用于指定最新命名，如下图

```
export default {
  name: 'ElDialogX',
  xtype: 'YuXdialog',
```

4、 新增业务组件统一使用 X 开头，如 Xform、Xtable 等，统一定义为 X 系列组件。

注释规范

文件首行说明注释

文件头注释都包含:
@created by 创建者、创建日期
@updated by 更新者、更新日期、更新内容（版本稳定后，每更新一次，增加一条@updated by 记录）
@description 文件功能简单描述

1. HTML

```
<!--
  @created by yourEmail Y-m-d
  @updated by
  @description 空白模板
-->
```

说明：
@created by 创建者、创建日期
@updated by 更新者、更新日期、更新内容（版本稳定后，每更新一次，增加一条@updated by 记录）
@description 文件功能简单描述

示例 1：

```
<!--
  @created by helin3 2018-01-16
  @updated by
  @description 空白模板
-->
```

示例 2：

```
<!--
  @created by helin3 2018-01-16
  @updated by zhangkun 2018-01-20 空白模板结构调整
  @description 空白模板
-->
```

2. JavaScript

```
/**
 * @created by yourEmail Y-m-d
 * @updated by
 * @description Description
 */
```

说明：参见 HTML

3. CSS

```
/**
 * @created by yourEmail Y-m-d
 * @updated by
 * @description Description
 */
```

说明：参见 HTML

方法（函数）申明注释

包含三部分：

1、 方法简要描述

2、 @param 参数名 参数说明

3、 @return 返回值名 返回值说明

```
/**
 * 页面加载完成时触发
 * @param hashCode 路由 ID
 * @param data 传递数据对象
 * @param cite 页面站点信息
 * @return vm 返回实例
 */
var readyFn = function (hashCode, data, cite) {
    ...
};
```

成员（属性）申明注释

可单行注释、也可多行注释

包含：

1、 成员类型

2、 成员说明

关键代码逻辑（不易于理解）注释

单行注释/多行注释，必须添加注释，便于提供代码可读性

无意义注释，必须删除

1、 无具体业务含义，或者说了等于白说的，如：//循环

2、 复制别人业务功能进行开发，注释掉的代码，或者后期由于功能改造注释原有代码，特别强调这部分代码直接删除即可，我们有版本管理工具的，若以后还需要，直接去SVN/GIT 取即可。

代码规范

强制执行 ESLINT

1. 使用两个空格进行缩进

eslint: indent

```
function hello (name) {
  console.log('hi', name)
}
```

2. 除需要转义的情况外，字符串统一使用单引号

eslint: quotes

```
console.log('hello there')
$('<div class="box">')
```

3. 不要定义未使用的变量

eslint: no-unused-vars

```
function myFunction () {
  var result = something()   // ✗ avoid
}
```

4. 关键字后面加空格

eslint: keyword-spacing

```
if (condition) { ... }   // ✓ ok
if(condition) { ... }    // ✗ avoid
```

5. 函数声明时括号与函数名间加空格

eslint: space-before-function-paren

```
function name (arg) { ... }   // ✓ ok
function name(arg) { ... }    // ✗ avoid

run(function () { ... })      // ✓ ok
run(function() { ... })       // ✗ avoid
```

6. 字符串拼接操作符 (Infix operators) 之间要留空格

eslint: space-infix-ops

```
// ✓ ok
var x = 2
var message = 'hello, ' + name + '!'

// X avoid
var x=2
var message = 'hello, '+name+'!'
```

7. 逗号后面加空格

eslint: comma-spacing

```
// ✓ ok
var list = [1, 2, 3, 4]
function greet (name, options) { ... }

// X avoid
var list = [1,2,3,4]
function greet (name,options) { ... }
```

8. else 关键字要与花括号保持在同一行

eslint: brace-style

```
// ✓ ok
if (condition) {
  // ...
} else {
  // ...
}

// X avoid
if (condition)
{
  // ...
}
else
{
```

```
  // ...
}
```

9. 多行 if 语句的**的括号不能省,**（建议所有 if 语句都必须有括号）

eslint: curly

```
// ✓ ok
if (options.quiet !== true) console.log('done')
// ✓ ok
if (options.quiet !== true) {
  console.log('done')
}
// X avoid
if (options.quiet !== true)
  console.log('done')
```

10.不要丢掉**异常处理中 err 参数**

eslint: handle-callback-err

```
// ✓ ok
run(function (err) {
  if (err) throw err
  window.alert('done')
})

// X avoid
run(function (err) {
  window.alert('done')
})
```

11.不允许有连续多行空行

eslint: no-multiple-empty-lines

```
// ✓ ok
var value = 'hello world'
console.log(value)
```

```
// X avoid
var value = 'hello world'


console.log(value)
```

12.对于三元运算符 ? 和 : 与他们所负责的代码处于同一行

eslint: operator-linebreak

```
// ✓ ok
var location = env.development ? 'localhost' : 'www.api.com'

// ✓ ok
var location = env.development
  ? 'localhost'
  : 'www.api.com'

// X avoid
var location = env.development ?
  'localhost' :
  'www.api.com'
```

13.条件语句中赋值语句使用括号包起来。这样使得代码更加清晰可读，而不会认为是将条件判断语句的全等号（===）错写成了等号（=）

eslint: no-cond-assign

```
// ✓ ok
while ((m = text.match(expr))) {
  // ...
}

// X avoid
while (m = text.match(expr)) {
  // ...
}
```

14.单行代码块两边加空格

eslint: block-spacing

```
function foo () {return true}    // ✗ avoid
  function foo () { return true }  // ✓ ok
```

15.对于变量和函数名统一使用驼峰命名法

eslint: camelcase

```
function my_function () { }    // ✗ avoid
  function myFunction () { }     // ✓ ok

  var my_var = 'hello'           // X avoid
  var myVar = 'hello'            // ✓ ok
```

16.不允许有多余的行末逗号

eslint: comma-dangle

```
var obj = {
    message: 'hello',   // ✗ avoid
  }
```

17.始终将逗号置于行末

eslint: comma-style

```
var obj = {
    foo: 'foo'
    ,bar: 'bar'   // X avoid
  }

  var obj = {
    foo: 'foo',
    bar: 'bar'   // ✓ ok
  }
```

18.点号操作符须与属性需在同一行

eslint: dot-location

```
console.
    log('hello')  // X avoid

  console
    .log('hello') // ✓ ok
```

19.函数调用时标识符与括号间不留间隔

eslint: func-call-spacing

```
console.log ('hello') // ✗ avoid
console.log('hello')  // ✓ ok
```

20.键值对当中冒号与值之间要留空白

eslint: key-spacing

```
var obj = { 'key' : 'value' }    // ✗ avoid
var obj = { 'key' :'value' }     // X avoid
var obj = { 'key':'value' }      // X avoid
var obj = { 'key': 'value' }     // ✓ ok
```

21.构造函数要以大写字母开头

eslint: new-cap

```
function animal () {}
var dog = new animal()    // X avoid

function Animal () {}
var dog = new Animal()    // ✓ ok
```

22.无参的构造函数调用时要带上括号

eslint: new-parens

```
function Animal () {}
var dog = new Animal    // X avoid
var dog = new Animal()  // ✓ ok
```

23.子类的构造器中一定要调用 super

eslint: constructor-super

```
class Dog {
  constructor () {
    super()   // X avoid
  }
}

class Dog extends Mammal {
  constructor () {
    super()   // ✓ ok
  }
}
```

24.使用数组字面量而不是构造器

eslint: no-array-constructor

```
var nums = new Array(1, 2, 3)   // ✗ avoid
var nums = [1, 2, 3]           // ✓ ok
```

25.避免使用 arguments.callee 和 arguments.caller

eslint: no-caller

```
function foo (n) {
  if (n <= 0) return

  arguments.callee(n - 1)   // X avoid
}

function foo (n) {
  if (n <= 0) return

  foo(n - 1)
}
```

26.避免对类名重新赋值

eslint: no-class-assign

```
class Dog {}
Dog = 'Fido'    // ✗ avoid
```

27.避免使用常量作为条件表达式的条件（循环语句除外）

eslint: no-constant-condition

```
if (false) {    // ✗ avoid
  // ...
}

if (x === 0) {  // ✓ ok
  // ...
}

while (true) {  // ✓ ok
  // ...
}
```

28.正则中不要使用控制符

eslint: no-control-regex

```
var pattern = /\x1f/    // ✗ avoid
var pattern = /\x20/    // ✓ ok
```

29.不要使用 debugger（代码提交时）

eslint: no-debugger

```
function sum (a, b) {
  debugger      // ✗ avoid
  return a + b
}
```

30.不要对变量使用 delete 操作

eslint: no-delete-var

```
var name
delete name     // ✗ avoid
```

31.不要定义冗余的函数参数

eslint: no-dupe-args

```
function sum (a, b, a) {  // ✗ avoid
  // ...
}

function sum (a, b, c) {  // ✓ ok
  // ...
}
```

32.类中不要定义冗余的属性

eslint: no-dupe-class-members

```
class Dog {
  bark () {}
  bark () {}    // ✗ avoid
}
```

33.对象字面量中不要定义重复的属性

eslint: no-dupe-keys

```
var user = {
  name: 'Jane Doe',
  name: 'John Doe'    // ✗ avoid
}
```

34.switch 语句中不要定义重复的 case 分支。

eslint: no-duplicate-case

```
switch (id) {
```

```
  case 1:
    // ...
  case 1:     // ✗ avoid
}
```

35.正则中不要使用空字符

eslint: no-empty-character-class

```
const myRegex = /^abc[]/      // ✗ avoid
const myRegex = /^abc[a-z]/   // ✓ ok
```

36.不要使用 eval()

eslint: no-eval

```
eval( "var result = user." + propName ) // ✗ avoid
var result = user[propName]            // ✓ ok
```

37.catch 中不要对错误重新赋值

eslint: no-ex-assign

```
try {
  // ...
} catch (e) {
  e = 'new value'            // X avoid
}

try {
  // ...
} catch (e) {
  const newVal = 'new value'  // ✓ ok
}
```

38.不要扩展原生对象

eslint: no-extend-native

```
Object.prototype.age = 21     // ✗ avoid
```

39.避免多余的函数上下文绑定

eslint: no-extra-bind

```
const name = function () {
  getName()
}.bind(user)    // X avoid

const name = function () {
  this.getName()
}.bind(user)    // ✓ ok
```

40.避免不必要的布尔转换

eslint: no-extra-boolean-cast

```
const result = true
if (!!result) {   // X avoid
  // ...
}

const result = true
if (result) {     // ✓ ok
  // ...
}
```

41.不要使用多余的括号包裹函数

eslint: no-extra-parens

```
const myFunc = (function () { })   // ✗ avoid
const myFunc = function () { }     // ✓ ok
```

42.switch 一定要使用 break 来将条件分支正常中断

eslint: no-fallthrough

```
switch (filter) {
  case 1:
    doSomething()    // X avoid
  case 2:
    doSomethingElse()
}
```

```
switch (filter) {
  case 1:
    doSomething()
    break           // ✓ ok
  case 2:
    doSomethingElse()
}

switch (filter) {
  case 1:
    doSomething()
    // fallthrough  // ✓ ok
  case 2:
    doSomethingElse()
}
```

43.不要省去小数点前面的 0

eslint: no-floating-decimal

```
const discount = .5      // ✗ avoid
const discount = 0.5     // ✓ ok
```

44.避免对声明过的函数重新赋值

eslint: no-func-assign

```
function myFunc () { }
myFunc = myOtherFunc    // ✗ avoid
```

45.不要对全局只读对象重新赋值

eslint: no-global-assign

```
window = {}     // ✗ avoid
```

46. 注意隐式的 eval()

eslint: no-implied-eval

```
setTimeout("alert('Hello world')")                  // ✗ avoid
```

```
setTimeout(function () { alert('Hello world') })     // ✓ ok
```

47.嵌套的代码块中禁止再定义函数

eslint: no-inner-declarations

```
if (authenticated) {
  function setAuthUser () {}    // ✗ avoid
}
```

48.不要向 RegExp 构造器传入非法的正则表达式

eslint: no-invalid-regexp

```
RegExp('[a-z')    // ✗ avoid
RegExp('[a-z]')   // ✓ ok
```

49.外部变量不要与对象属性重名

eslint: no-label-var

```
var score = 100
function game () {
  score: 50         // ✗ avoid
}
```

50.不要使用标签语句

eslint: no-labels

```
label:
  while (true) {
    break label     // ✗ avoid
  }
```

51.不要书写不必要的嵌套代码块

eslint: no-lone-blocks

```
function myFunc () {
  {                 // ✗ avoid
```

```
    myOtherFunc()
  }
}

function myFunc () {
  myOtherFunc()      // ✓ ok
}
```

52.不要混合使用空格与制表符作为缩进

eslint: no-mixed-spaces-and-tabs

53.除了缩进，不要使用多个空格

eslint: no-multi-spaces

```
const id =    1234    // ✗ avoid
const id = 1234       // ✓ ok
```

54.new 创建对象实例后需要赋值给变量。

eslint: no-new

```
new Character()                    // ✗ avoid
const character = new Character()   // ✓ ok
```

55.禁止使用 Function 构造器

eslint: no-new-func

```
var sum = new Function('a', 'b', 'return a + b')    // ✗ avoid
```

56.禁止使用 Object 构造器

eslint: no-new-object

```
let config = new Object()   // ✗ avoid
```

57.禁止使用原始包装器

eslint: no-new-wrappers

```
const message = new String('hello')   // ✗ avoid
```

58.不要将全局对象的属性作为函数调用

eslint: no-obj-calls

```
const math = Math()   // ✗ avoid
```

59.不要使用八进制字面量

eslint: no-octal

```
const num = 042     // ✗ avoid
const num = '042'   // ✓ ok
```

60.字符串字面量中也不要使用八进制转义字符

eslint: no-octal-escape

```
const copyright = 'Copyright \251'  // ✗ avoid
```

61.不要重复声明变量

eslint: no-redeclare

```
let name = 'John'
let name = 'Jane'     // X avoid

let name = 'John'
name = 'Jane'         // ✓ ok
```

62.正则中避免使用多个空格

eslint: no-regex-spaces

```
const regexp = /test   value/   // ✗ avoid
```

```
const regexp = /test {3}value/  // ✓ ok
const regexp = /test value/     // ✓ ok
```

63.return 语句中的赋值必需有括号包裹

eslint: no-return-assign

```
function sum (a, b) {
  return result = a + b     // X avoid
}

function sum (a, b) {
  return (result = a + b)   // ✓ ok
}
```

64.避免将变量赋值给自己

eslint: no-self-assign

```
name = name   // ✗ avoid
```

65.避免将变量与自己进行比较操作

esint: no-self-compare

```
if (score === score) {}   // ✗ avoid
```

66.避免使用逗号操作符

eslint: no-sequences

```
if (doSomething(), !!test) {}   // ✗ avoid
```

67.不要随意更改关键字的值

eslint: no-shadow-restricted-names

```
let undefined = 'value'     // ✗ avoid
```

68.禁止使用稀疏数组（Sparse arrays）

eslint: no-sparse-arrays

```
let fruits = ['apple',, 'orange']      // ✗ avoid
```

69.不要使用制表符

eslint: no-tabs

70.使用 this 前请确保 super() 已调用

eslint: no-this-before-super

```
class Dog extends Animal {
  constructor () {
    this.legs = 4     // X avoid
    super()
  }
}
```

71.用 throw 抛错时，抛出 Error 对象而不是字符串

eslint: no-throw-literal

```
throw 'error'               // ✗ avoid
throw new Error('error')    // ✓ ok
```

72.行末不留空格

eslint: no-trailing-spaces

73.不要使用 undefined 来初始化变量

eslint: no-undef-init

```
let name = undefined    // ✗ avoid

let name
name = 'value'          // ✓ ok
```

74.循环语句中注意更新循环变量

eslint: no-unmodified-loop-condition

```
for (let i = 0; i < items.length; j++) {...}    // ✗ avoid
for (let i = 0; i < items.length; i++) {...}    // ✓ ok
```

75.如果有更好的实现，尽量不要使用三元表达式。

eslint: no-unneeded-ternary

```
let score = val ? val : 0     // ✗ avoid
let score = val || 0          // ✓ ok
```

76.return， throw， continue 和 break 后不要再跟代码

eslint: no-unreachable

```
function doSomething () {
  return true
  console.log('never called')     // ✗ avoid
}
```

77.finally 代码块中不要再改变程序执行流程

eslint: no-unsafe-finally

```
try {
  // ...
} catch (e) {
  // ...
} finally {
  return 42     // ✗ avoid
}
```

78.关系运算符的左值不要做取反操作

eslint: no-unsafe-negation

```
if (!key in obj) {}       // ✗ avoid
```

79.避免不必要的 .call() 和 .apply()

eslint: no-useless-call

```
sum.call(null, 1, 2, 3)   // ✗ avoid
```

80.避免使用不必要的计算值作对象属性

eslint: no-useless-computed-key

```
const user = { ['name']: 'John Doe' }   // ✗ avoid
const user = { name: 'John Doe' }       // ✓ ok
```

81.禁止多余的构造器

eslint: no-useless-constructor

```
class Car {
  constructor () {      // ✗ avoid
  }
}
```

82.禁止不必要的转义

eslint: no-useless-escape

```
let message = 'Hell\o'  // ✗ avoid
```

83.属性前面不要加空格

eslint: no-whitespace-before-property

```
user .name      // ✗ avoid
user.name       // ✓ ok
```

84.禁止使用 with

eslint: no-with

```
with (val) {...}    // ✗ avoid
```

85.对象属性换行时注意统一代码风格

eslint: object-property-newline

```
const user = {
  name: 'Jane Doe', age: 30,
  username: 'jdoe86'           // ✗ avoid
}

const user = { name: 'Jane Doe', age: 30, username: 'jdoe86' }    // ✓ ok

const user = {
  name: 'Jane Doe',
  age: 30,
  username: 'jdoe86'
}
```

86.代码块中避免多余留白

eslint: padded-blocks

```
if (user) {
                          // ✗ avoid
  const name = getName()

}

if (user) {
  const name = getName()    // ✓ ok
}
```

87.展开运算符与它的表达式间不要留空白

eslint: rest-spread-spacing

```
fn(... args)    // ✗ avoid
fn(...args)     // ✓ ok
```

88.遇到分号时空格要后留前不留

eslint: semi-spacing

```
for (let i = 0 ;i < items.length ;i++) {...}    // ✗ avoid
for (let i = 0; i < items.length; i++) {...}    // ✓ ok
```

89.代码块首尾留空格

eslint: space-before-blocks

```
if (admin){...}     // ✗ avoid
if (admin) {...}    // ✓ ok
```

90.圆括号间不留空格

eslint: space-in-parens

```
getName( name )     // ✗ avoid
getName(name)       // ✓ ok
```

91.一元运算符后面跟一个空格

eslint: space-unary-ops

```
typeof!admin        // ✗ avoid
typeof !admin        // ✓ ok
```

92.注释首尾留空格

eslint: spaced-comment

```
//comment           // ✗ avoid
// comment          // ✓ ok

/*comment*/         // ✗ avoid
/* comment */       // ✓ ok
```

93.模板字符串中变量前后不加空格

eslint: template-curly-spacing

```
const message = `Hello, ${ name }`    // ✗ avoid
const message = `Hello, ${name}`      // ✓ ok
```

94.检查 NaN 的正确姿势是使用 isNaN()

eslint: use-isnan

```
if (price === NaN) { }      // ✗ avoid
if (isNaN(price)) { }       // ✓ ok
```

95.用合法的字符串跟 typeof 进行比较操作

eslint: valid-typeof

```
typeof name === 'undefimed'     // ✗ avoid
typeof name === 'undefined'     // ✓ ok
```

96.请书写优雅的条件语句（avoid Yoda conditions）

eslint: yoda

```
if (42 === age) { }    // ✗ avoid
if (age === 42) { }    // ✓ ok
```

97.不要解构空值

eslint: no-empty-pattern

```
const { a: {} } = foo         // ✗ avoid
const { a: { b } } = foo      // ✓ ok
```

98.对象中定义了存值器，一定要对应的定义取值器

eslint: accessor-pairs

```
var person = {
```

```
  set name (value) {    // X avoid
    this.name = value
  }
}

var person = {
  set name (value) {
    this.name = value
  },
  get name () {         // ✓ ok
    return this.name
  }
}
```

99.避免修改使用 const 声明的变量（避免使用 const/let, IE9 不支持）。

eslint: no-const-assign

```
const score = 100
score = 125       // ✗ avoid
```

100. 同一模块有多个导入时一次性写完（注: ES6 语法）

eslint: no-duplicate-imports

```
import { myFunc1 } from 'module'
import { myFunc2 } from 'module'          // X avoid

import { myFunc1, myFunc2 } from 'module' // ✓ ok
```

101. 不要使用非法的空白符

eslint: no-irregular-whitespace

```
function myFunc () /*<NBSP>*/{}   // ✗ avoid
```

102. 禁止使用 __iterator__

eslint: no-iterator

```
Foo.prototype.__iterator__ = function () {}   // ✗ avoid
```

103. 禁止使用 new require

eslint: no-new-require

```
const myModule = new require('my-module')    // ✗ avoid
```

104. 禁止使用 Symbol 构造器

eslint: no-new-symbol

```
const foo = new Symbol('foo')   // ✗ avoid
```

105. 使用 __dirname 和 __filename 时尽量避免使用字符串拼接

eslint: no-path-concat

```
const pathToFile = __dirname + '/app.js'           // ✗ avoid
const pathToFile = path.join(__dirname, 'app.js')   // ✓ ok
```

106. 使用 getPrototypeOf 来替代 __proto__

eslint: no-proto

```
const foo = obj.__proto__               // ✗ avoid
const foo = Object.getPrototypeOf(obj)  // ✓ ok
```

107. 正确使用 ES6 中的字符串模板

eslint: no-template-curly-in-string

```
const message = 'Hello ${name}'   // ✗ avoid
const message = `Hello ${name}`   // ✓ ok
```

108. import, export 和解构操作中，禁止赋值到同名变量

eslint: no-useless-rename

```
import { config as config } from './config'     // ✗ avoid
import { config } from './config'              // ✓ ok
```

109. 自调用匿名函数（IIFEs）使用括号包裹

eslint: wrap-iife

```
const getName = function () { }()     // ✗ avoid

const getName = (function () { }())   // ✓ ok
const getName = (function () { })()   // ✓ ok
```

110. yield * 中的 * 前后都要有空格

eslint: yield-star-spacing

```
yield* increment()    // ✗ avoid
yield * increment()   // ✓ ok
```

建议执行

111. 代码中凡是需要使用 this 变量转换的地方，统一转换为_this

```
var self = this;    // ✗ avoid
var that = this;    // ✗ avoid
var me = this;      // ✗ avoid
var _this = this;   // ✓ ok
```

112. 不要使用 alert（代码提交时）

no-alert

```
alert(1);    // ✗ avoid
```

113. 不要使用 console（代码提交时）

no-console

```
cosole.log(1);    // ✗ avoid
```

114. 使用浏览器全局变量时加上 window. 前缀

no-undef

```
window.alert('hi')   // ✓ ok
```

ESLINTRC.JSON 配置文件附件

.eslintrc.json

```
{
  "env": {
    "browser": true,
    "node": true
  },
  "globals": {
    "yufp": true,
    "Vue": true,
        "Mock":true,
    "backend": true,
    "expect": true,
    "sinon": true,
    "$": true
  },
  "parserOptions": {
    "ecmaVersion": 5,
    "sourceType": "script",
    "ecmaFeatures": {}
  },
  "rules": {
    "indent-legacy":  ["error", 2],
    "indent":  ["error", 2],
    "quotes": ["error", "single"],
    "no-unused-vars": ["error", { "args": "none" }],
    "keyword-spacing": 2,
    "space-before-function-paren": 2,
    "space-infix-ops": 2,
    "comma-spacing": 2,
    "brace-style": 2,
    "curly": 2,
    "handle-callback-err": 2,
```

```
    "no-multiple-empty-lines": 2,
    "operator-linebreak": 2,
    "no-cond-assign": 2,
    "block-spacing": 2,
    "camelcase": 2,
    "comma-dangle": 2,
    "comma-style": 2,
    "dot-location": 2,
    "func-call-spacing": 2,
    "key-spacing": 2,
    "new-cap": 2,
    "new-parens": 2,
    "constructor-super": 2,
    "no-array-constructor": 2,
    "no-caller": 2,
    "no-class-assign": 2,
    "no-constant-condition": 2,
    "no-control-regex": 2,
    "no-delete-var": 2,
    "no-dupe-args": 2,
    "no-dupe-class-members": 2,
    "no-dupe-keys": 2,
    "no-duplicate-case": 2,
    "no-empty-character-class": 2,
    "no-eval": 2,
    "no-ex-assign": 2,
    "no-extend-native": 2,
    "no-extra-bind": 2,
    "no-extra-boolean-cast": 2,
    "no-extra-parens": ["error", "all", {"conditionalAssign":false,
"returnAssign": false ,"nestedBinaryExpressions": false }],
    "no-fallthrough": 2,
    "no-floating-decimal": 2,
    "no-func-assign": 2,
    "no-global-assign": 2,
    "no-implied-eval": 2,
    "no-invalid-regexp": 2,
    "no-label-var": 2,
    "no-labels": 2,
    "no-lone-blocks": 2,
    "no-mixed-operators": 2,
    "no-multi-spaces": 2,
    "no-new": 2,
    "no-new-func": 2,
```

```
    "no-new-object": 2,
    "no-new-wrappers": 2,
    "no-obj-calls": 2,
    "no-octal": 2,
    "no-octal-escape": 2,
    "no-regex-spaces": 2,
    "no-return-assign": 2,
    "no-self-assign": 2,
    "no-self-compare": 2,
    "no-sequences": 2,
    "no-shadow-restricted-names": 2,
    "no-sparse-arrays": 2,
    "no-tabs": 2,
    "no-this-before-super": 2,
    "no-throw-literal": 2,
    "no-trailing-spaces": 2,
    "no-undef-init": 2,
    "no-unmodified-loop-condition": 2,
    "no-unneeded-ternary": 2,
    "no-unreachable": 2,
    "no-unsafe-finally": 2,
    "no-unsafe-negation": 2,
    "no-useless-call": 2,
    "no-useless-computed-key": 2,
    "no-useless-constructor": 2,
    "no-useless-escape": 2,
    "no-whitespace-before-property": 2,
    "no-with": 2,
    "object-property-newline":
["error",{"allowAllPropertiesOnSameLine":true}],
    "padded-blocks": ["error", "never"],
    "rest-spread-spacing": 2,
        "semi": ["error", "always", {"omitLastInOneLineBlock":true }],
    "semi-spacing": 2,
    "space-before-blocks": 2,
    "space-in-parens": 2,
    "space-unary-ops": 2,
    "spaced-comment": 2,
    "template-curly-spacing": 2,
    "use-isnan": 2,
    "valid-typeof": 2,
    "yoda": 2,
    "no-empty-pattern": 2,
    "accessor-pairs": 2,
```

```
    "no-const-assign": 2,
    "no-duplicate-imports": 2,
    "no-irregular-whitespace": 2,
    "no-iterator": 2,
    "no-new-require": 2,
    "no-new-symbol": 2,
    "no-path-concat": 2,
    "no-proto": 2,
    "no-template-curly-in-string": 2,
    "no-useless-rename": 2,
    "wrap-iife": 2,
    "yield-star-spacing": 2,
    "no-undef": 2,
    "no-redeclare": 0,
    "no-inner-declarations": 1,
    "no-debugger": 1,
    "no-alert": 1,
    "no-console": 2,
    "eqeqeq": 0,
    "eol-last": 0,
    "no-multi-str": 0,
    "one-var": 0
  },
  "env": {
    "browser": true,
    "amd": true
  }
}
```