

真实项目最佳实践

Best Practices in Real World Projects



<https://github.com/WisdomFusion>

{ TOC

- 统一代码风格
- 命名
- 布局和样式没那么简单
- 本应很熟练的 jQuery
- JavaScript 薄弱环节
- ES6 和 TypeScript
- 扩展运算符
- 解构赋值
- 泛型
- 代码逻辑折射解决思路
- 数据结构
- 数据库设计
- 数据查询与更新
- RESTful API 注意事项
- 数据类型及检测
- 要看的文档和要写的文档
- 代码复用
- 数据复用
- 重构与重写
- 代码审查
- Angular 窗体应用

统一代码风格

- 分号（语句结束符不能少）
- 运算符（前后加空格，让代码不要太拥挤）
- 代码行的长度（78 或 80/120 columns per row，断开长行又要保障可读性）
- 4 spaces: PHP, HTML, JavaScript, TypeScript; 2 spaces: CSS, SCSS
- **注释和段落**（代码分组，相关的变量或语句放一起）
- 垂直对齐（变量赋值，对象 value，代码分组的注释）

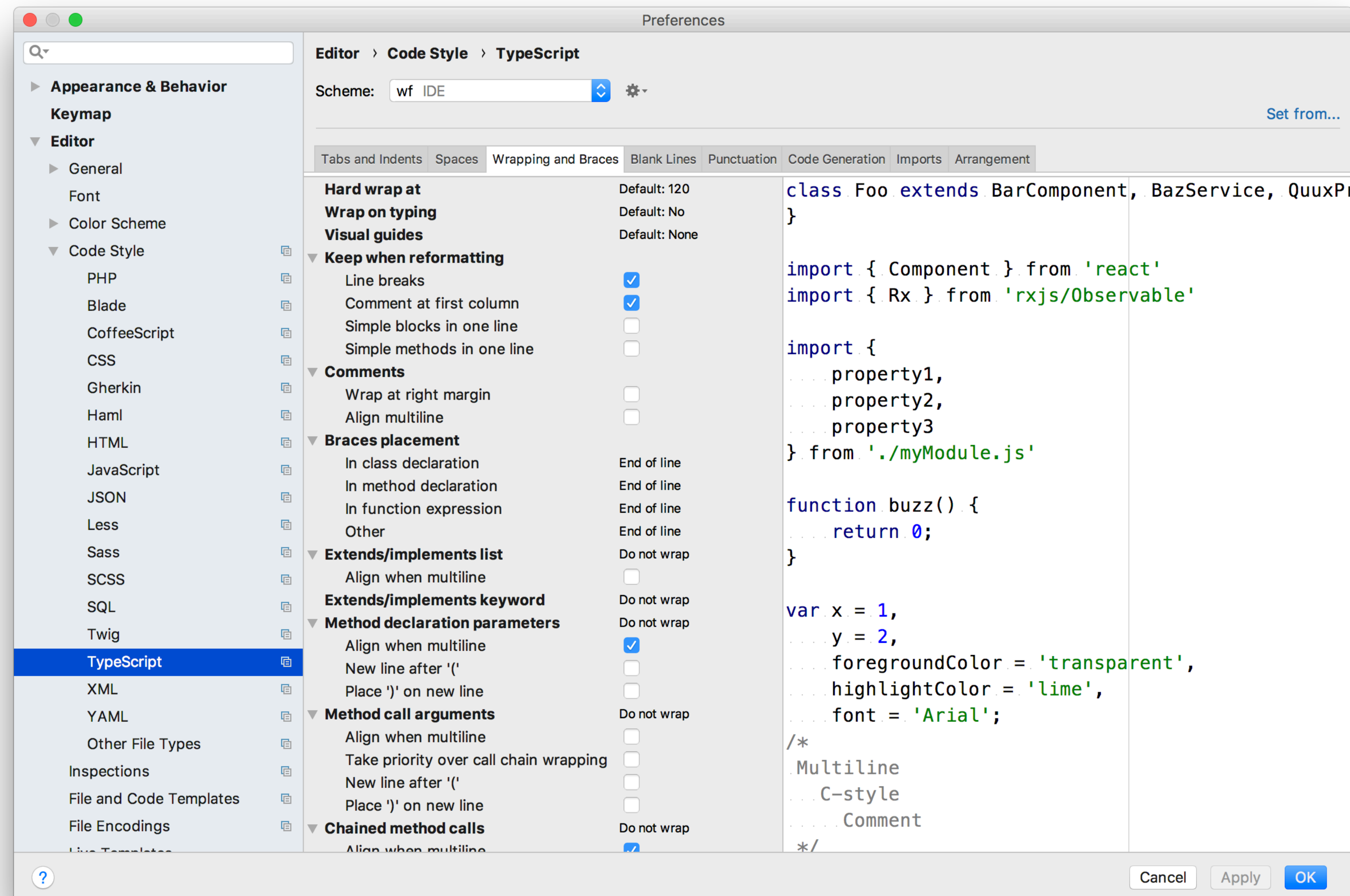
统一代码风格

- 代码要分组，相关的语句或函数要扎堆儿
- 杜绝同一个项目或同一个端出现迥异的代码风格
- 发现 **bad** 风格，及时格式化成 **good** 风格，不要嫌麻烦
- “将错就错”是不思进取的表现，视若无睹和无法察觉问题所在同样危险
- 只有新文件（新建的或生成的，或是需要大量重构及修改的）才能用全局的 Reformat Code，开发中或是修改维护的代码，**只格式化选中的区域**
- 为什么风格居然能成为老生长谈的问题？！

“风格的本质是可读性强则易改易维护

PhpStorm - Preferences - Editor - Code Style

PHP, JavaScript, TypeScript, CSS, SCSS



代码风格统一之 JetBrains 设定： PHP

Wrapping and Braces

- Chained method calls ✓ Align when multiline
- Assignment statement ✓ Align consecutive assignments
- Class field/constant groups ✓ Align constants
- Ternary operation ✓ Align when multiline
- Array initializer ✓ Align when multiline ✓ Align key-value pairs

代码风格统一之 JetBrains 设定：JavaScript

Spaces

- With ✓ Object literal braces ✓ ES6 import/export braces

Wrapping and Braces

- Chained method calls ✓ Align when multiline ✓ '.' on new line
- Ternary operation ✓ Align when multiline
- Objects - Align - On value
- Variable declarations - Align - When grouped

Punctuation

- Use single quotes in new code

代码风格统一之 JetBrains 设定: TypeScript

Spaces

- With ✓ Object literal type braces ✓ Object literal braces ✓ ES6 import/export braces

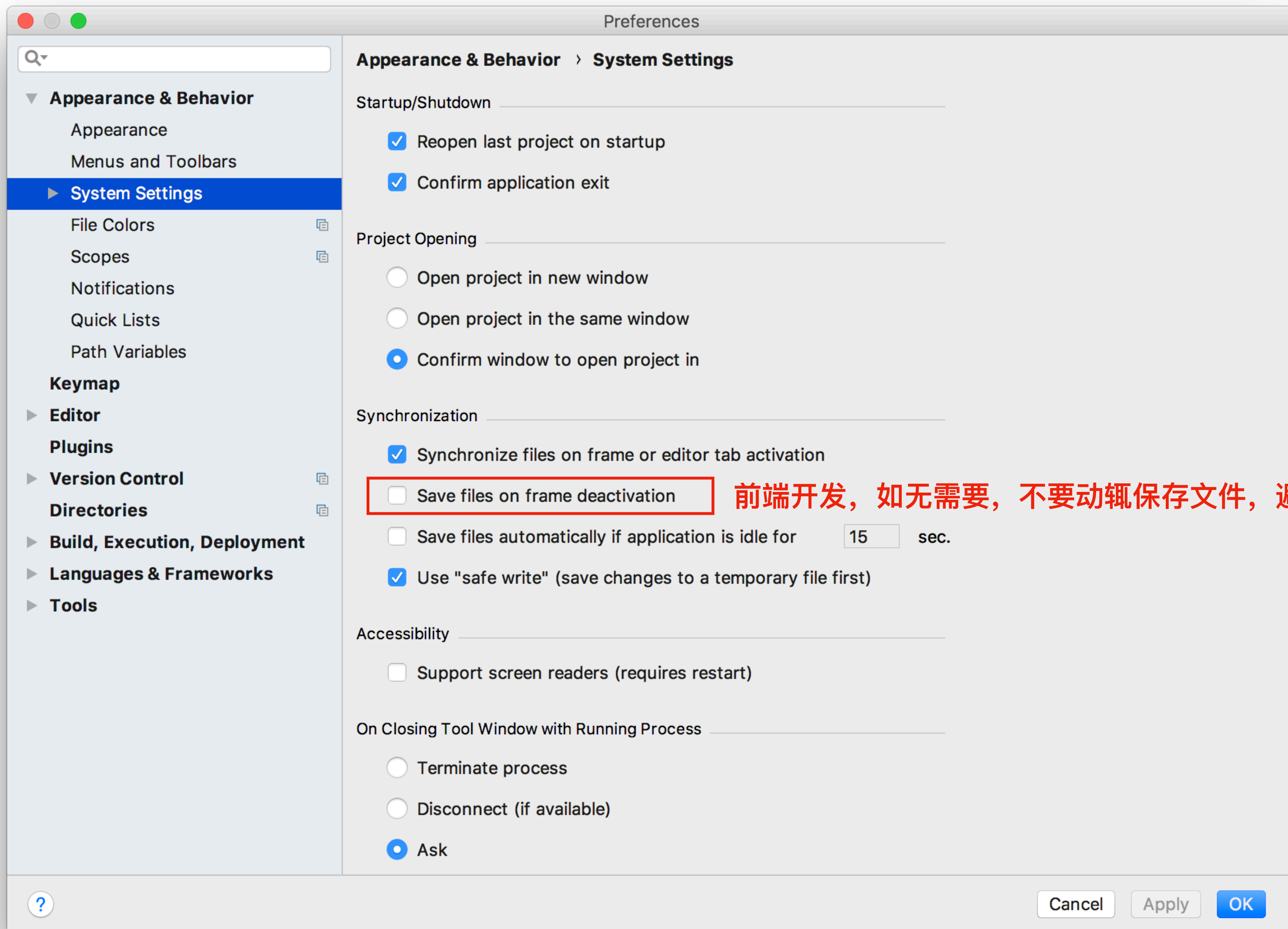
Wrapping and Braces

- Chained method calls ✓ Align when multiline ✓ '.' on new line
- Ternary operation ✓ Align when multiline
- Objects - Align - On value
- Variable declarations - Align - When grouped

Punctuation

- Use single quotes in new code

**“能让机器帮我们做的就不要手动
但代码逻辑、分组和命名等做不到自动化**



命名 (Naming)

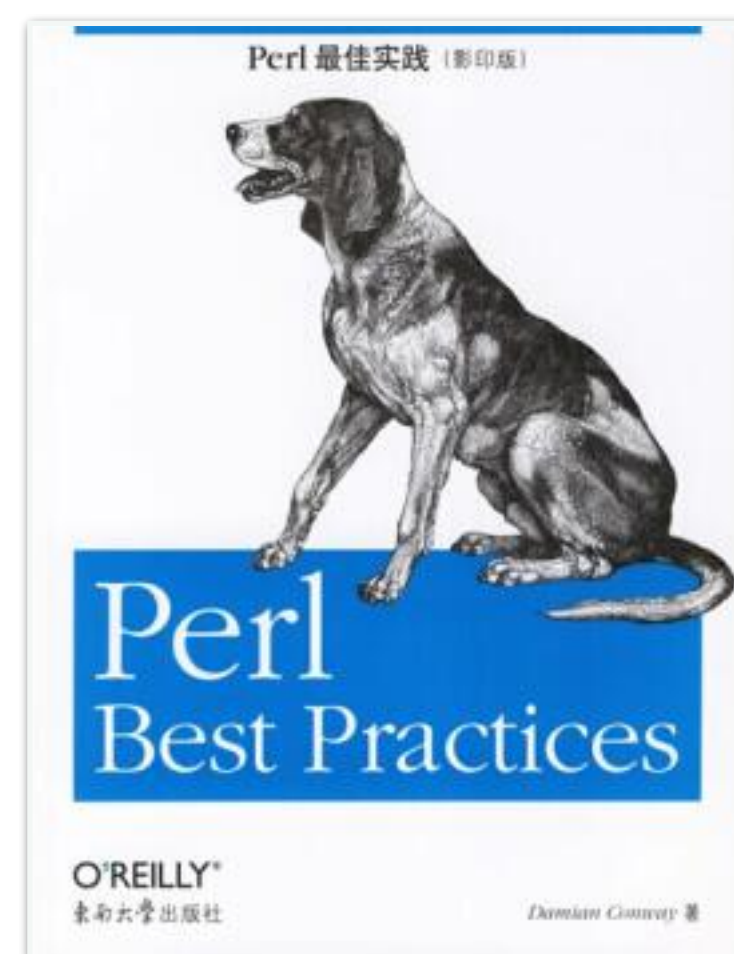
- 避开关键字
- 避免使用不合理的简写形式，尽量使用通识性的缩写
- 合理的英文词组，状态用形容词或动词的过去分词，可加 is 作前缀，有时需要用动名词 (finished, isVisibleAddPermission, loading)
- 命名除了表达清楚做什么事，有时候可以添加数据类型使之更明确 (courseForm, permissionGroupsFormArray, permissionGroupFormGroup)
- 先表达清楚，再考虑长短，不要嫌啰嗦

命名 (Naming)

- camelCase: PHP, JavaScript, TypeScript
- 不要用 snake_case, 更不能混用
- CSS类名: `class="course-attachment-list"`
- 不要瞎拷贝, 避免发生连注释都张冠李戴的笑话 (这可不是个例)
- 用业务相关的、具体的词, 不要直接拿控件名等作为变量名
- 学好英语



Rachel's English



布局和样式没那么简单

- 结构尽可能简单（能少一层容器就少一层），为了结构的健壮可以适当多加一层容器（多一层容器无妨）
- 注意数据有无对布局的影响（如坍塌、撑爆了，甚至整个页面面目全非）
- 响应式设计 vs. 自适应设计（Responsive Design vs. Adaptive Design）
- 先考虑多平台适配，最后考虑为不同平台单独编写代码
- 设计还原度不高，只有两种可能：要么设计没有讲清楚，要么没有用心观察
- 功能实现有偏差同上
- 注意见贤思齐，而不能有比上不足比下有余有心态

本应很熟练的 jQuery

- 应该尽可能复用同一个选择器的 DOM 对象
- 调用一次方法，传对象参数；而不是一个方法反复调用
- 链式方法调用要注意换行，不要“一链到底”
- 变量或对象属性前缀下划线很丑，除非语言本身强制要求，尽量少用
- “找祖先”和“找后代”不要 `parent()`, `children()` 或 `find()` 个没完，换个起点试试
- 大量样式的切换，先考虑切换 `class`，而不是动辄就用 jQuery 的 `css()` 方法

JavaScript 薄弱环节

- Vanilla JavaScript
- 模块化和组件化（样式表也有类似问题）
- 面向对象编程思想
- 类型检查
- webpack 工程化
- 引入了 jQuery 却又要自己造轮子
- HTML5 API



ES6 和 TypeScript

- 面向对象特性
- Promise 对象
- Observable 可观察者对象与 RxJS
- First-class Functions
- High-order Functions: forEach, map, filter, every, some, reduce
- ES5 可以通过 lodash 使用以上高阶函数，甚至更多

ES6 和 TypeScript

```
const myNumbers = [ 1, 2, 5 ];
```

```
const sum = myNumbers.reduce( function(prev, curr) {  
    return prev + curr;  
}, 0);
```

```
console.log(sum); // 8
```

```
const myWords = [ 'These', 'all', 'form', 'a', 'sentence' ];
```

```
const sentence = myWords.reduce( (prev, curr) => {  
    return prev + ' ' + curr;  
}); // the initial value is optional
```

```
console.log(sentence); // 'These all form a sentence'
```

扩展运算符

```
case CLASS_ADD:
  return {
    ...state,
    classes: [
      ...state.classes,
      action.classData
    ]
  };
case CLASS_UPDATE:
  return {
    ...state,
    classes: state.classes.map(
      item => (item.class_id === action.id) ? {...action.classData} : item
    )
  };
case CLASS_DELETE:
  return {
    ...state,
    classes: state.classes.filter(item => item.class_id !== action.id)
  };
```

扩展运算符

```
// 预处理需要提交的数据
prepareFormData(): object {
  let data = { ...this.roleDetailForm.value };
  data.disabled = data.disabled ? 1 : 0;

  // 提取权限的 ID 成为数组
  let permIds = [];
  if (data.permissionGroups) {
    data.permissionGroups.forEach(permissionGroup => {
      permissionGroup.permissions && permissionGroup.permissions.forEach(permission => {
        permission.checked && permIds.push(permission.id);
      });
    });
    delete data.permissionGroups;
  }

  permIds.length && (data['permIds'] = permIds);

  return data;
}
```

解构赋值

```
class ClassroomList extends React.Component {
  componentDidMount() {
    this.props.loadClassrooms(this.props.filter);
    this.props.loadSchoolList();
  }

  ...

  render() {
    const { classrooms, total } = this.props;
    const { filter: { page, per_page } } = this.props;

    const {
      schoolList,
      filter,
      filterClassrooms,
      loadClassroomList,
      loadClassrooms,
      setClassroomFilter
    } = this.props;
```

泛型 (Generics)

泛型数据类型

```
Array<T>
```

泛型函数

```
function identity<T>(arg: T): T {  
    return arg;  
}
```

泛型类

```
class GenericNumber<T> {  
    zeroValue: T;  
    add: (x: T, y: T) => T;  
}
```

```
let myGenericNumber = new GenericNumber<number>();  
myGenericNumber.zeroValue = 0;  
myGenericNumber.add = function(x, y) { return x + y; };
```

泛型 (Generics)

```
courses: Array<Course> = [];
```

```
get<T>(url: string, params?: object): Observable<T> {  
    return this.http.get<T>(url, { params: this.prepareParams(params) });  
}
```

```
get permissionGroups(): FormArray {  
    return <FormArray>this.roleDetailForm.get('permissionGroups');  
}
```

```
(<FormArray>permissionGroupForm.get('permissions')).push(permissionForm);
```


BAD #0001

```
function modal_tip(a) {
    setTimeout(function () {
        var text = a.text || '操作成功';
        var time = a.time || 2000;
        var ele_bg = $('<div class="modal_bg"></div>');
        var ele_tip = $('<div class="modal-tip"></div>');
        if(a.error){
            ele_tip = $('<div class="modal-tip error-bg"></div>');
        }
        $(document.body).append(ele_bg);
        $(document.body).append(ele_tip);
        ele_tip.html(text);
        ele_tip.animate({'marginTop':-50+'px','opacity':1},
200);

        setTimeout(function () {
            ele_tip.animate({'marginTop':-20+'px','opacity':0},
200,function () {
                ele_bg.remove();
                ele_tip.remove();
                if(a.fun){
                    a.fun();
                }
            });
        }, time)
    },300)
}
```

```
function responseMessage(response){
    if(response.status===422&&response.data){
        var text = '';

        $.each(response.data,function(k,v){
            if(response.data[k].length===1){
                text += response.data[k][0];
            }else{
                for(var i=0;i<response.data[k].length;i++){
                    text += response.data[k][i];
                }
            }
        });
        modal_tip({text : text,error:true});
    }else if(response.status === 422 && !response.data){
        modal_tip({text:response.message,error:true});
    }else{
        modal_tip({text:response.message,error:true});
    }
}

/*
例: modal_tip({text: '33333', time: 1000});
参数是一个对象{
    text:'提示文字',//这是提示文字,默认是 操作成功
    time:2000      //这是显示持续时间,默认是两秒后消失
}
*/
```

BAD #0002

```
$('.courselearn-container').on('input keydown', '.homework-explain', function (event) {
    var msgBox = $(this).siblings('.homework-message').find('.fl-l');
    if($(this).val().length>200){
        msgBox.html('文字不得超过200字');
        $(this).val($(this).val().substr(0,200))
    }else{
        msgBox.html('');
    }
    if($(this).val() === ''){
        if( $(this).parent().parent().find('.upload-img-list').find('.delete-img-btn').length > 0 ){
            $(this).parent().parent().find('.answer-edit-ok-button').css('color','#60acf3');
            $(this).parent().parent().find('.answer-edit-ok-button').css('cursor','pointer');
            $(this).parent().parent().find('.answer-edit-ok-button').css('pointer-events','visible');

        }else{
            $(this).parent().parent().find('.answer-edit-ok-button').css('color','#999');
            $(this).parent().parent().find('.answer-edit-ok-button').css('cursor','default');
            $(this).parent().parent().find('.answer-edit-ok-button').css('pointer-events','none');
        }
    }else{
        $(this).parent().parent().find('.answer-edit-ok-button').css('color','#60acf3');
        $(this).parent().parent().find('.answer-edit-ok-button').css('cursor','pointer');
        $(this).parent().parent().find('.answer-edit-ok-button').css('pointer-events','visible');
    }
})
})
```

BAD #0003a

```
import {Component, OnInit} from '@angular/core';
import {
    AbstractControl,
    FormBuilder,
    FormGroup,
    Validators
} from '@angular/forms';
import {CareerCourseService} from '../service/
career-course.service';
import {CareerHomeworkService} from '../service/
career-homework.service';
import {Configuration} from '../shared/
configuration';
import {CourseService} from '../service/
course.service';
import {NzMessageService, UploadFile} from 'ng-
zorro-antd';
import { Utils } from '../shared/utils';
import { Router, ActivatedRoute } from '@angular/
router';

@Component({
    selector: 'app-homework-details',
    templateUrl: './homework-
details.component.html',
```

```
    styleUrls: ['./homework-
details.component.scss']
})
export class HomeworkDetailsComponent implements
OnInit {
    apiUrl: string;
    selectCourseForm: FormGroup;
    ObjectiveForm: FormGroup;
    subjectiveForm: FormGroup;
    homeworkTitle: string;
    userId = 0;
    // attachments: Array<Attachment> = [];
    courseList = [];
    courseObject = {};
    baseImageUrl = 'http://files.pixseed.com/
pixseed/';
    courseName = '';
    homeworkName = '';
    modelOne = true;
    modelObjective = false;
    modelSubjective = false;
    addHomeworkStatus = false;
    addHomeworkStatus = false;
    imgButtonStatus = true;
    radioValue = '';
    textOrImage = 'text';
```

BAD #0003b

```
inputOptionArr = ['inputA', 'inputB', 'inputC', 'inputD'];
imageStatus = true;
homeWorkIndexStatus = -1;
subjectHomeworkStatus = -1;
// fileTypeArray = 'image/png,image/jpeg,image/gif,image/bmp';
fileAccept = '.jpg,.png,.jpeg,.gif,.bmp';
optionInputValue = {
    inputA: '',
    inputB: '',
    inputC: '',
    inputD: ''
};
homeworkData = {
    course_id: '',
    title: '',
    objective: [],
    subjective: []
};
objectiveJson = {
    name: '',
    attach: '',
    analysis: '',
    option: [
        ['A', '', '', 0],
        ['B', '', '', 0],
        ['C', '', '', 0],
        ['D', '', '', 0]
    ]
}
```

```
};
subjectiveJson = {
    name: '',
    attach: '',
    attach_name: '',
    analysis: ''
};
optionTypeObject = {
    A: 0,
    B: 1,
    C: 2,
    D: 3
};

optionTypeArray=['A', 'B', 'C', 'D'];

previewImage = '';
previewVisible = false;
fileList = [];
optionFile = {
    fileListA : [],
    fileListB : [],
    fileListC : [],
    fileListD : [],
};
fileListSubject = [];
```

BAD #0004

```
if (HomeworkSchedule::where('id', $id)->update($data = $request->only('admin_id', 'ended_at')) {  
    $cancelContent = '您的《' . $schedule->course->name . '》课' . $schedule->classes->name . '作业' .  
($schedule->section ? $schedule->section->title : '') . ' 点评教师身份已被取消。';  
    $newContent = '您已被设为《' . $schedule->course->name . '》课' . $schedule->classes->name . '作业'  
    . ($schedule->section ? $schedule->section->title : '') . ' 点评教师，学生的作业将会发送给您，请注意查看并及时批改。';  
    Message::sendNotifyToTeacher($schedule, $data['admin_id'], $cancelContent, $newContent);  
  
    return response()->success('', '修改成功!');  
} else {  
    return response()->errorInternal();  
}
```


GOOD #0001

```
$detailSubmit = QuestionAnswer::with(['questionOptions'])
->join('users', 'users.id', '=', 'question_answers.user_id')
->join('questions', 'questions.id', '=', 'question_answers.question_id')
->leftJoin('admins', 'admins.id', 'question_answers.admin_id')
->where('question_answers.homework_schedule_id', $homeworkScheduleId)
->where('question_answers.user_id', $userId)
->select('question_answers.question_id',
        'questions.type as question_type',
        'questions.desc as question_desc',
        'questions.analysis as question_analysis',
        'question_answers.upload',
        'question_answers.question_id',
        'question_answers.upload_name',
        'question_answers.type',
        'question_answers.answer',
        'question_answers.right',
        'question_answers.score',
        'question_answers.comment',
        'question_answers.commented_at',
        'question_answers.img',
        'question_answers.video',
        'question_answers.created_at as submit_time',
        'admins.name as teacher_name',
        'users.name as student_name'
    )
->get()->toArray();
```

GOOD #0002

```
res => {
  const permissionsData = [...res.data];

  // 创建权限列表表单:
  // 权限分组
  permissionsData.forEach(permissionGroup => {
    let permissionGroupForm = this.fb.group({
      id:      permissionGroup.id,
      name:    permissionGroup.name,
      description: permissionGroup.description,
      permissions: this.fb.array([])
    });

    // 某分组权限列表
    if (permissionGroup.permissions && permissionGroup.permissions.length) {
      permissionGroup.permissions.forEach(permission => {
        let permissionForm = this.fb.group({
          id:      permission.id,
          group_id: permission.group_id,
          name:    permission.name,
          display_name: permission.display_name,
          description: permission.description,
          checked: 0
        });
        (<FormArray>permissionGroupForm.get('permissions')).push(permissionForm);
      });
    }
    this.permissionGroups.push(permissionGroupForm);
  });
}
```

代码逻辑折射解决思路

- 混乱的思路产生的代码肯定更混乱
- 先厘清思路，再动手
- 先编写骨架代码，再填充血肉，避免血肉模糊
- 代码首先是给开发人员看的，然后才是机器执行，逻辑要清晰，可读性要好
- 重构，使代码逻辑更清晰

数据结构

- 数据结构应该和逻辑结构保持一致
- 数据从属关系要正确，需要有从属关系的，不要用并列关系
- 接口响应的数据为前端作**适度预处理**（过度处理前端可能还需要对数据进行解析或转换）
- 响应数据的**数据类型**应该准备且明确
- 响应数据不能少，但也不应返回过多冗余数据

数据库设计

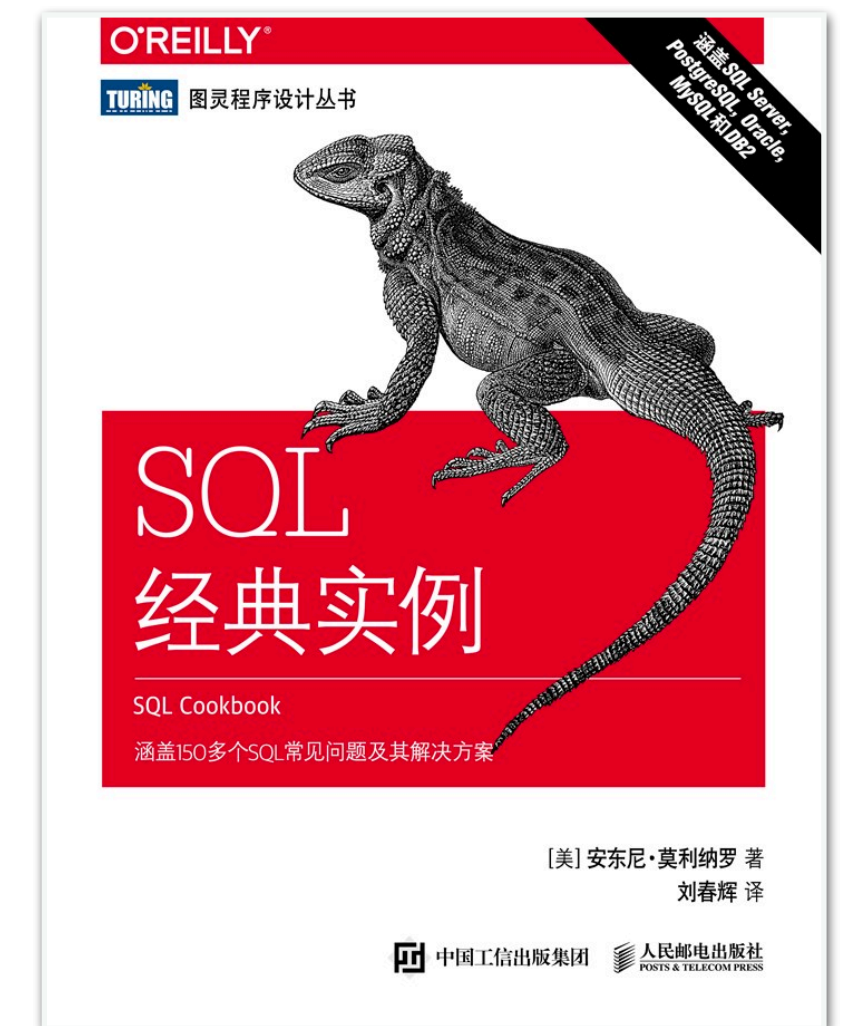
- 表名和字段名 snake_case, 有些单词不要拆分它们 (如 homework, feedback, classroom 等)
- 避免使用关键字作为字段名 (如 desc)
- 人物相关的名称用 name, 事物相关的名称或标题用 title
- 尽可能用通识类的英文表达, 不要再出现类似 is_single 的字段
- 避免大量冗余字段, 以免出现数据不一致, 为改进性能或方便聚合可视情况增加冗余
- 避免空值 (日期类字段除外, 日期类字段均用 <动词的过去分词>_at 形式)

数据库设计

- 表注释和关键字段注释不能少
- 修改表或字段相关人员甚至整个开发团队悉知
- 修改表或字段的同时要注意对现有业务的影响，并给出数据升级方案
- 避免长时间在生产环境准备数据，可事先在开发环境准备好，再迁移到线上
- 数据字典（字段参考，厘清关系）
- 数据库的版本控制

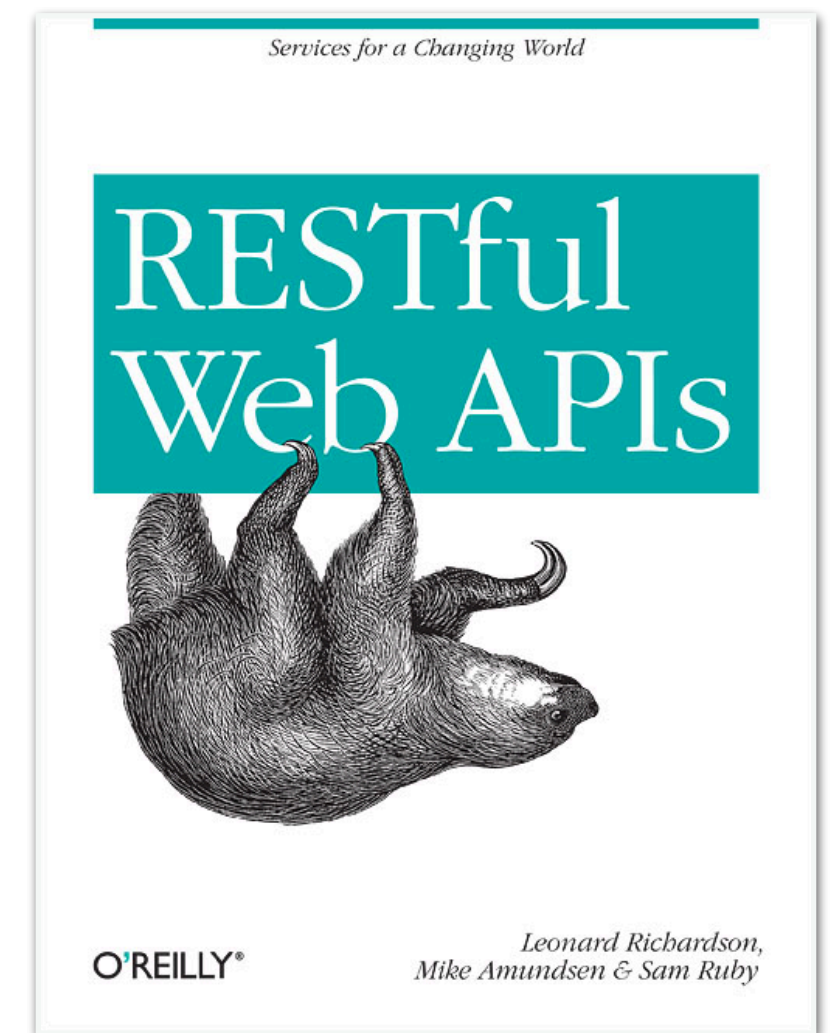
数据查询与更新

- 只查询或更新必要的表和数据
- 表的设计和业务逻辑要考虑数据一致性
- 业务逻辑和数据访问不要纠缠不清
- Eloquent ORM vs. Query Builder
- Eloquent Eager Loading
- 调试并优化关键查询，尤其是用户端无需登录的业务模块
- 查询和更新做好安全防范



RESTful API 注意事项

- 易于使用、便于更改、健壮
- 路由简明，美观，相关功能模块统一目录
- 合理的、明确的请求方法（method）
- 事先约定请求参数和响应参数
- 约定数据格式、状态码、错误类型
- 版本升级的过渡



GET	获取资源
POST	新增资源
PUT	更新已有资源
PATCH	更新部分资源
DELETE	删除资源

数据类型及检测

- 弱类型、动态类型语言更应该小心数据类型的陷阱
- 用 `===` 和 `!==` （编辑器把潜在的问题都高亮了，还视若无睹，真行！）
- 前端和后端不光要约定请求和响应字段、数据结构，JSON 中的具体字段的数据类型也要明确（给出示例输出即可）
- 使用之前要作空值等特殊值检测
- 对比之前要作必要的类型转换

要看的文档

- [PHP manual](#), [Laravel Docs](#), [MySQL Docs](#)
- JavaScript 高级编程
- [Learning JavaScript Design Patterns](#)
- [Angular](#), [NG-ZORRO](#), [Redux](#)
- [React](#), [antd](#), [Redux](#)
- [TypeScript](#), [ES6](#), [jQuery](#), [jQuery 1.12.x](#)
- 重构：改善既有代码的设计

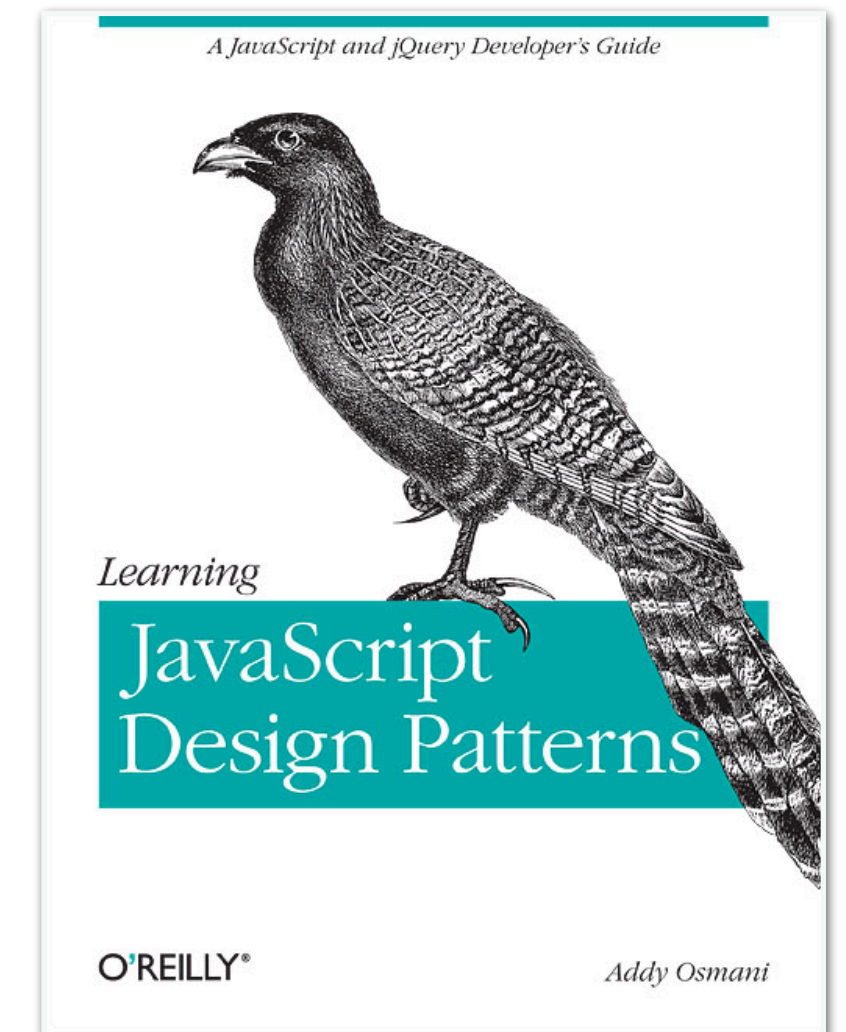
以上只是万分之一

要写的文档

- 注释注释注释
- 接口文档
- 公用代码库文档
- 数据字典和表关系图
- 日常笔记
- 系统详细设计文档（含项目说明）
- 关系业务流程图或 UML 图

代码复用

- Modular and Scalable CSS
- JavaScript: The Revealing Module Pattern
- 前端组件化开发（借助 jQuery 及其插件机制）
- Angular/React 细化组件
- PHP/Laravel: Repository 模式, ServiceProvider, Facades, Traits等等



数据复用

- 前端不同功能模块之间请求数据复用（公用）
- 不同业务模块的关联数据复用
- 中后台系统：sessionStorage, localStorage, and Redux
- 数据缓存

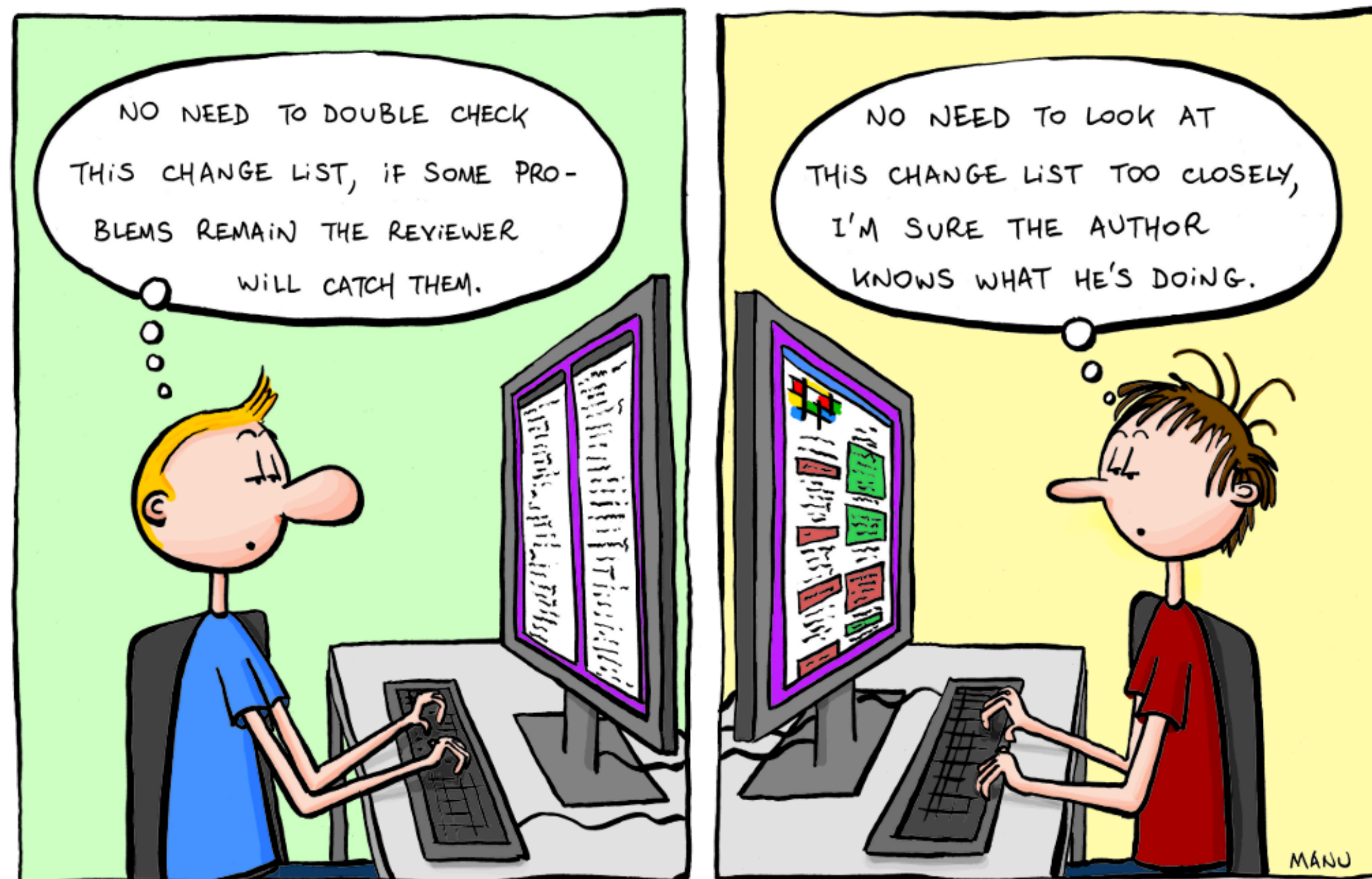
重构与重写

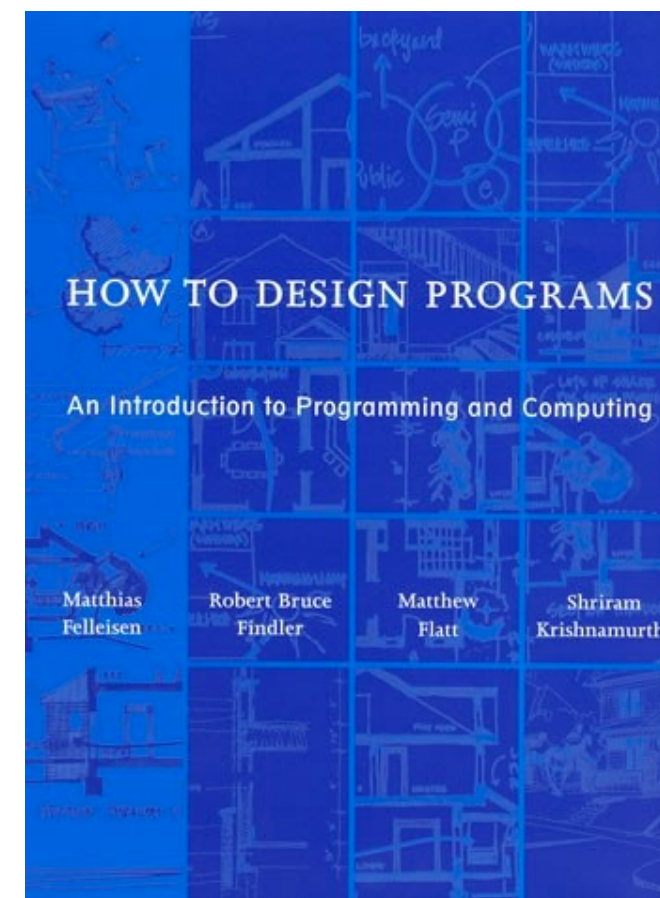
- 改进程序设计，更易理解，帮助找到缺陷
- 良好的设计是快速开发的根本
- 事不过三，三则重构
- 重构：添加功能时、修复 Bug 时、代码审查时
- 代码过于混乱，满是错误，则考虑重写
- 重构之前，代码必须在大部分情况下正常运作



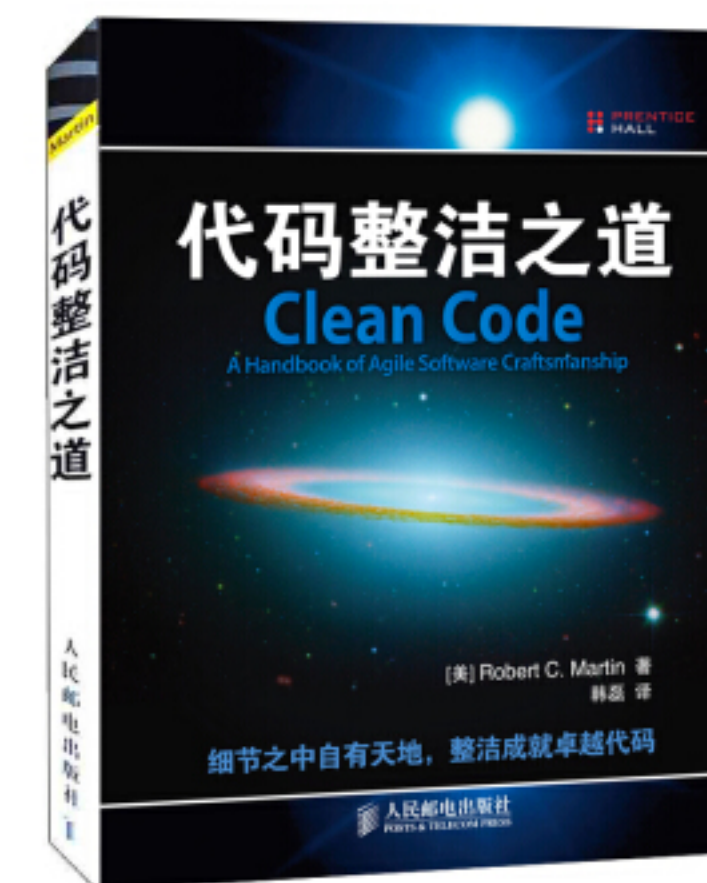
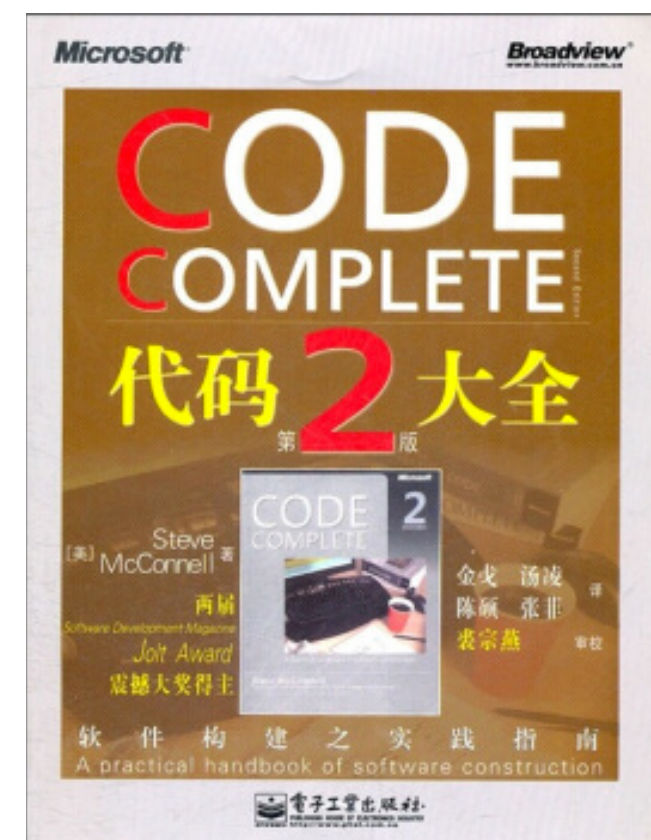
代码审查 (Code Review)

- 这不是结对编程 (pair-programming)
- Check List
- 控制每次审查的代码量
- 未经审查的代码不得进入代码库
- 写代码的人 (Author)
- 审查代码的人 (Reviewer)





How to Design Programs, 2nd Edition





Angular 表单应用

Angular Form Applications

Template-driven Forms

- 模板驱动表单和响应式表单不要同时存在（为什么？）
- 表单项添加 name 属性
- 用什么组件实现
- 特殊值处理（空值、布尔值等）
- 模板变量，使用 ViewChild 控制表单



Reactive Forms

- ① 先建立基本表单结构
- ② 渐近式添加其他表单结构
- ③ 添加必要的表单验证
- ④ 初始化表单控件数据
- ⑤ 维护表单数据
- ⑥ 整理表单数据
- ⑦ 提交数据
- ⑧ 善后（提示、跳转或对表单作处理）

Forms Case Study

表单结构分析

- RoleDetails
FormGroup -> FormArray -> FormGroup -> FormArray -> FormGroup
- ExerciseDetails
FormGroup -> FormArray -> FormGroup
- HomeworkDetails
FormGroup -> FormArray -> FormGroup -> FormArray -> FormGroup

“意识中本身就没有是怎么也挤不出料来的

“刻意练习 厚积薄发

Thanks. }