Clean code

هدی دانش پژوه

پاییز ۹۸

۱-به دست آوردن یک منطق خوب قبل از هر چیز

قبل از اینکه کورکورانه به آزمون و خطا کردن انتخاب‌های مختلف بپردازید، چند flow diagram که از دانش وتجربه های قبلی خود بدست آوردید تهیه کنید.اول، می‌تواند خیلی از شک‌ها یا ناامنی‌های ناشی از پیچیدگی عملیاتی کار را کاهش دهد؛ که نتیجه‌ی آن صرفه‌جویی قابل ملاحظه‌ای در زمان خواهد داشت. اما مهم‌تر ایت که کمک شایان توجهی به درک سریع‌تر کد و جلوگیری از آشفتگی آن می‌کند.

۲-استفاده از اسم‌گذاری استاندارد برای توابع و متغیرها

در نامگذاری متغیر ها از نام های معنی دار استفاده کنید. نامی را انتخاب کنید که چیزی را که درون متغیر میریزد و یا کاری که تابع انجام می دهد را به خوبی شرح دهد

```php
//Bad:
$a = $moment->format('y-m-d');
$behzad = $moment->format('y-m-d');
//واقعا یه دوستی داشتم همیشه اسم خودش رو روی کلاس ها و متغیرها و حتی کامیت هاش میزاشت



//Good:
$currentDate = $moment->format('y-m-d');
```

۳- نوشتن کامنت‌ها

این کار می‌تواند خیلی راحت، سریع و با اشاره‌ی کاملاً مستقیم به اصل مطلب باشد، چرا که درست در جای مناسب و مورد نیاز (روبروی کد) قرار دارد.

همچنین، کامنت‌ها به طور کارآمدی می‌توانند درست زمانی که شما مشغول به خواندن یک کد هستید، ابهامات آن را از بین ببرند. در ضمن، می‌توانید آنها را مجدداً مورد استفاده قرار دهید، که در توصیه بعدی باید به آن توجه گردد.

نمونه‌هایی از کامنت‌های مناسب:

مشخصات ناشر کد (مثلاً نوشته شده توسط علی در تاریخ دوازدهم نوامبر ۲۰۱۹)

توضیح عملکرد متد یا رویه‌ی مورد نظر (مثلاً این تابع صحت ورود افراد را به کمک بررسی ایمیل آن ها چک می‌کند)

قرار دادن کامنت‌های کوتاه یا برچسب‌هایی که مشخص کند در هر مرحله چه تغییری صورت گرفته است (مثلاً رویه‌ی بررسی صحت ایمیل اضافه شد)

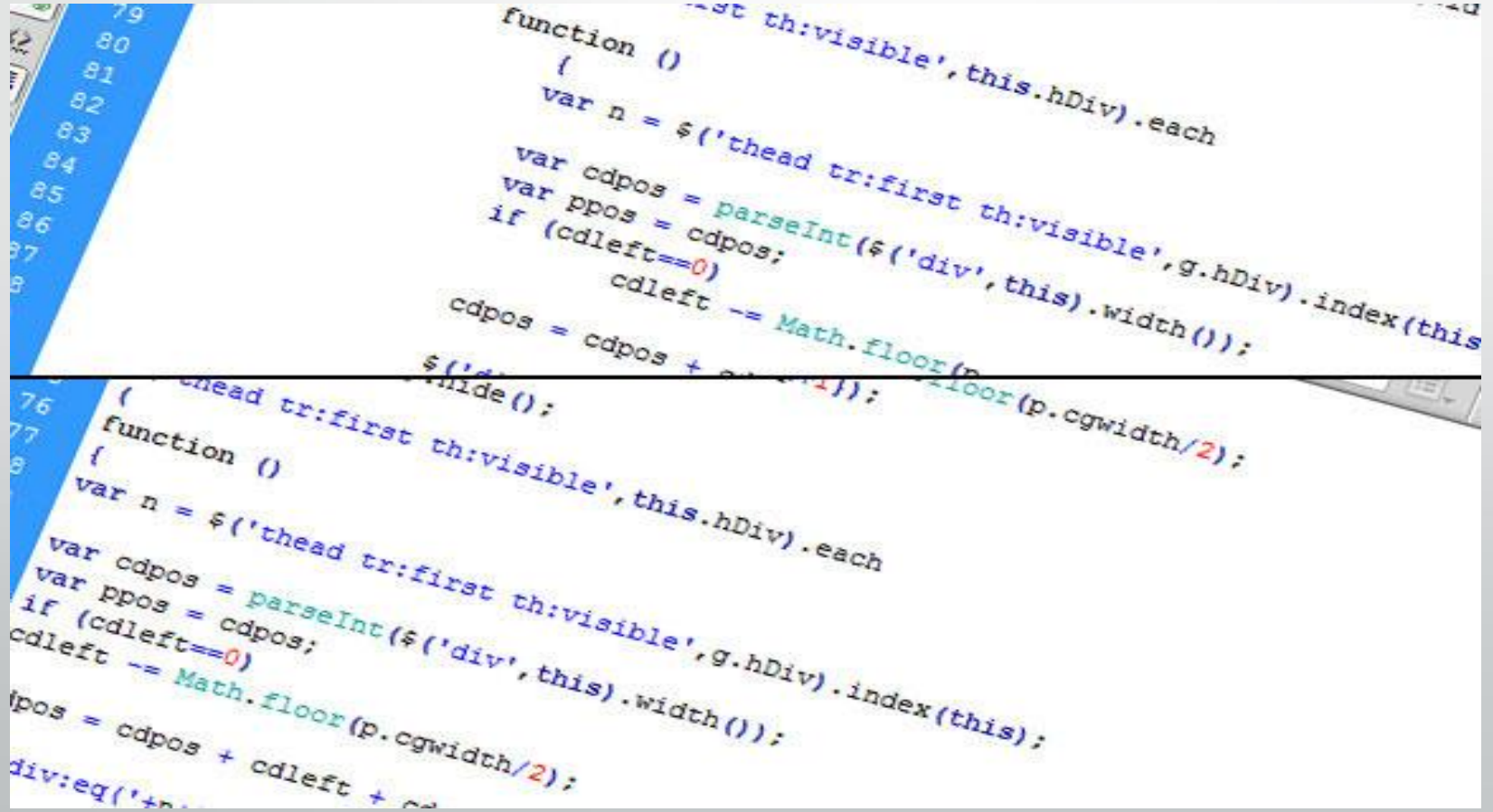
نوشتن انتهای توابع یا کلاس ها و ...


",4); return false;
return false;" style="font-size:10; font-
style="margin-left:5px;"/>');
$("#clase").fadeIn();
$("#num_filtro").html(5);
}
}
}
}//End of validation statements
updateGraph(data);//Ending procedure of graph update
}
function drillDown(info){ //This method obtains the information for the FLASH
var dat = info.split(",");
var registro = dat[0];
dat[1]-1;
== "0")&&(checks[2]

۴- فاصله‌گذاری مناسب

پشتیبانی از اصل بالا، برای رعایت ساختار مناسب، نیاز به مجزا کردن شروع و پایان هر المان دارد. اگر همه‌ی خطهای کد پشت هم در سمت چپ صفحه‌ی نمایش باشند، تشخیص اینکه یک المان در کد، دقیقاً کجا تمام می‌شود کار واقعاً دشواری خواهد بود.

همچنین، تلاش برای طراحی یک ساختار کامل را غیر ممکن می‌کند.



۵- نمایش واضحی از ساختار صفحه

```
<div id="main-container">
    <div id="header">
        <div id="logo">...</div>
        <div id="main-menu">...</div>
    </div>
    <div id="content">
        <div id="left-column">...</div>

        <div id="center-column">...</div>
        <div id="right-column">...</div>
    </div>
    <div id="footer">
        <div id="footer-menu">...</div>
        <div id="disclaimer">...</div>
    </div>
</div>
```

```
<html>
<body>

<div style="float:left ; padding-right:100px">
<p>paragraph number 1</p>
</div>

<div style="float:left">
<p>paragraph number 2</p>
</div>

</body>
</html>
```

۶- برای اطلاعاتی که مربوط به یک چیز است از یک متغیر استفاده کنید.(از آرایه استفاده کنید)

```php
//Bad:
$userInfo = '';
$userData = '';
$userRecord = '';
$userProfile = '';


//Good:
$user = [
    'info' => '',
    'data' => '',
    'record' => '',
    'profile' => ''
];
```

۷ـ آرگومان های تابع خود را به حداقل برسانید .

ورودی توابع ، اگر ورودی توابع بیش از ۲ عدد باشد بهتر است از قابلیت شی گرایی استفاده کنیم.

کد کثیف:

```
function createNewUser($id,$firstName,$lastName,$email,$password,$phone);
```

کد تمیز (با استفاده از قابلیت شی گرایی):

```
function createNewUser(User $user)
{
   //do something ...
}

$user = new User();

$user->setId(1);
$user->setFirstName("Reza");
$user->setLastName("Eskandari");
$user->setEmail("user@mail");
$user->setPhone("123456");

createNewUser($user);
?>
```

یک توصیه عمومی وجود داره که میگوید اگر آرگومان های یک تابع دو یا کمتر از دو باشند ان تابع در حالت ایده آل خودش به سر میبرد.
محدود کردن تعداد پارامترهای یک تابع فوق العاده مهم است زیرا باعث می شه به کار بردن تابع و فهم ان ساده تر شود.

```php
//Bad:
function createOrder(string $title, int $price, Carbon $createAt, bool $cancellable): void
{
    // ...
}


//Good:
class OrderItem
{
    public $title;
    public $price;
    public $createAt = false;
    public $cancellable = false;
}

$order = new OrderItem();
$order->title = 'title';
$order->price = 1000;
$order->createAt = Carbon::now();
$order->cancellable = true;

function createOrder(OrderItem $items): void
{
    // ...
}
```

۸- توابع فقط باید یک کار و وظیف مشخص را انجام دهند. این مهم ترین قاعده در مهندسی نرم افزار است. وقتی توابع بیش از یک کار انجام دهند، ساختن، تست کردن و استفاده از انها سخت تر میشود.

```php
//Bad:
function emailClients(array $clientIds): void
{
    foreach ($clientIds as $id) {
        $client = Client::find($id);
        if ($client->isActive()) {
            email($id);
        }
    }
}

//Good:
function emailClients(array $clientIds): void
{
    $activeClients = activeClients($clientIds);
    array_walk($activeClients, 'email');
}

function activeClients(array $clientIds): array
{
    return array_filter($clientIds, 'isClientActive');
}

function isClientActive(int $id): bool
{
    $clientRecord = Client::find($id);

    return $clientRecord->isActive();
}
```

این مثال خوبی بود تا روشن کند مفهوم clean code لزوما کوتاه کردن کدنویسی نیست بلکه هدف خوانا بودن کد هست

۹- به پراپرتی های یک کلاس پیشوند اضافه نکنید.

کد کثیف:

```php
<?php

class Animal
{
    private $animalName;

    private $animalType;

    private $animalFamily;
}


?>
```

کد تمیز :

```php
<?php

class Animal
{
    private $name;

    private $type;

    private $family;
}


?>
```

۱۰- اجتناب از نوشتن شرط های کثیف و اضافه.

کد کثیف :

```php
<?php
if (condition == true) {
    continue;
} else {
    die("Error") or exit();
}
?>
```

کد تمیز و کمتر:

```php
<?php

if (!condition){
    die("invalid") or exit();
}

?>
```

۱۱- پرهیز از ترکیب دو زبان برنامه‌سازی به طور یکسره :

بهترین نمونه از مسئله‌ی ترکیب نابجای زبان‌های برنامهسازی، می‌تواند CSSهای درونخطی و یاJavaScript های پراکنده با رویه های کوچک در کد HTML باشد. عدم توجه به این مسئله، سبب ایجاد حجم عظیمی از تگ‌های المان‌ها به همراهCSS های درون آنها می‌شود. خیلی از مشکلات در جریان ساختار کار به سبب همین توابع درون‌برنامه‌ای ایجاد می‌شوند؛ که البته، بسیار گیج کننده هم خواهد بود .

نمونه کد تمیز تر:

```
<html>
<body>
<table>
    <tr>
        <td><?php echo "hello world !!!"; ?></td>
    </tr>
</body>
</html>
```

کد کثیف

```
<?php

echo "<table>";
echo “<tr>”;
echo “<td>”;
echo "hello world !!!";
echo “</td>”;
echo </tr>”;
echo “</table>”;
?>
```

۱۲-از flag ها به عنوان پارامتر توابع استفاده نکنید

پرچم (flag) ها در واقع به زبان بی زبانی می گویند این تابع بیش از یک کار انجام می دهد. در صورتی که به عنوان مهمترین اصل این آموزش گفتیم توابع بهتر است فقط یک کار انجام دهند.در چنین حالتی بهترین کار این است تابع خود را به توابع بیشتری تقسیم کنیم.

```php
//Bad:
function createFile(string $name, bool $temp = false): void
{
    if ($temp) {
        touch('./temp/'.$name);
    } else {
        touch($name);
    }
}

//Good:
function createFile(string $name): void
{
    touch($name);
}

function createTempFile(string $name): void
{
    touch('./temp/'.$name);
}
```

۱۳- به جای مقادیر از ثابت ها(با رعایت اصل یک و نام های قابل جستجو) استفاده کنید. یا حتی روش بهتر اینه که از کلاس استفاده کنید.

```php
// •Bad
if ($user->access & 4) {
    // ...
}


//Good:
class User
{
    const ACCESS_READ = 1;
    const ACCESS_CREATE = 2;
    const ACCESS_UPDATE = 4;
    const ACCESS_DELETE = 8;
}

if ($user->access & User::ACCESS_UPDATE) {
    // do edit ...
}
```

۱۴- از به کار بردن جمله های شرطی تو در تو اجتناب کنید و خلاقانه تر کد بنویسید.

```php
//Bad:
function isShopOpen($day): bool
{
    if ($day) {
        if (is_string($day)) {
            $day = strtolower($day);
            if ($day === 'friday') {
                return true;
            } elseif ($day === 'saturday') {
                return true;
            } elseif ($day === 'sunday') {
                return true;
            } else {
                return false;
            }
        } else {
            return false;
        }
    } else {
        return false;
    }
}
```

```php
//Good:
function isShopOpen(string $day): bool
{
    if (empty($day))
        return false;


    $openingDays = [
        'friday', 'saturday', 'sunday'
    ];

    return in_array(strtolower($day), $openingDays, true);
}
```

۱۵ - اگر چند جمله ی IF بعد از یکدیگر می آیند به جای if های تو در تو از AND استفاده کنید:
*نکته ای که باید بدانید این است که زمانی که در IF از OR استفاده می کنید. اگر در هر جمله ای به TRUE برسد جملات دیگه را چک نمی کنده و در AND هم بر عکس این ماجرا وجود داره.

```php
//Bad:
function fibonacci(int $n)
{
    if ($n < 50) {
        if ($n !== 0) {
            if ($n !== 1) {
                return fibonacci($n - 1) + fibonacci($n - 2);
            } else {
                return 1;
            }
        } else {
            return 0;
        }
    } else {
        return 'Not supported';
    }
}


//Good:
function fibonacci(int $n): int
{
    if ($n === 0 || $n === 1)
        return $n;


    if ($n > 50)
        throw new \Exception('Not supported');


    return fibonacci($n - 1) + fibonacci($n - 2);
}
```