

نینجا کد (مترجم: چگونه کثیف کد بزنی؟)

"یادگیری بدون فکر کردن بیهوده است ، فکر کردن بدون یادگیری خطرناک"

کنفسیوس

استثنا ، این بخش به صورت محاوره‌ای ترجمه شده است .

برای نزدیک‌تر شدن به زبان فارسی ، بخش‌هایی از متن جایگزین متن اصلی شده و با تجربیات مترجم ترکیب شده است .

برنامه نویسی نینجا در گذشته از این روش‌ها استفاده می‌کردن تا شاخکای افرادی که روی کد کار می‌کردن رو تیز نگه دارن .

کسانی هم که یاد میدادن چه جوری کد رو بررسی کنیم از این روش‌ها برای کارهای تستی استفاده می‌کردن .

البته توی استفاده از این روشها ، برنامه نویسی مبتدی بعضی وقتا خیلی بهتر از حرفه‌ای‌ها عمل می‌کنن .

این روشها رو بخون تا شخصیت خودت رو کشف کنی .

خیلی‌ها راه نینجاها رو دنبال می‌کنن . تعداد کمی موفق می‌شن .

اختصار ، روح شوخ‌طبعیه .

تا جایی که می‌تونی کدت رو خلاصه و مختصر بنویس تا به همه نشون بدی چه قدر باهوشی . از ویژگی‌های ریز و زیرپوستی جاوا اسکریپت استفاده کن . برای مثال به این عملگر trenary دقت کن :

// taken from a well-known javascript library

```
i = i ? i < 0 ? Math.max(0, len + i) : i : 0;
```

باحاله نه؟ اگر اینجوری کد بزنی ، برنامه‌نویسی که میخواد از این کد استفاده کنه ، پیرش درمیا در بفهمه مقدار `i` چی می‌شه . آخرم مجبوره بیفته دنبال شما که تروخدا بگو `i` چی میشه . بهشون بگو مختصر مفید همیشه بهتره . بزار حساب کار دستشون بیاد .

متغیرهای تک حرفی

"بلندترین فریاد ، سکوت است ." ناشناس

یه راه دیگه برای نوشتن کدهای مختصر مفید ، استفاده از متغیرهای تک حرفی تو کل کده . مثل `...,c,b,a`

متغیرهای تک حرفی تو بطن کد گم می‌شن ، درست مثل نینجایی که توی جنگل گم می‌شه . هیچ کس حتی با استفاده از ابزار جست‌وجوی ادیتور هم نمیتونه پیداشون کنه . حتی اگه کسی پیداشون کنه ، از روی اسمشون نمیفهمه که اونا چی هستن . ولی استثنا هم

وجود دارد. یک نینجای واقعی هیچ وقتی از متغیری مثل i برای شمارنده حلقه‌ای مثل for استفاده نمی‌کند. البته ممکنه جای دیگه از i استفاده کنه ولی نه داخل حلقه for. بیشتر دقت کن. حروف عجیب و غریب زیادی میتونی استفاده کنی مثل x یا y.

قضیه وقتی باحالت همیشه که تعداد خطهای حلقه ما یکی دو صفحه کامل بشه یا شاید هم بیشتر. تو این حالت وقتی به برنامه نویس دیگه به کد ما نگاه کنه عمرا نفهمه مثلا متغیر x کارش دقیقا چیه.

از مخفف استفاده کن

اگر قانونای دست‌وپاگیر تیمتون اجازه نداد که از متغیرهای تک حرفی استفاده کنی، اشکال نداره. از مخفف‌ها استفاده کن. مثلا:

- list -> lst
- userAgent -> ua
- browser -> brsr
- ...

@alithecodeguy

فقط اونایی که خیلی تیزن می‌فهمن چی به چیه. هر چه کمتر بهتر. فقط اونایی که لیاقتش رو دارن باید روی کد شما کار کنن.

تخیلی کد بزن، جوری که هیچ کس هیچ چی نفهمه.

"مربع کامل، گوشه ندارد." لاوژی

وقتی داری اسم انتخاب می‌کنی، تابلوترین اسم‌ها رو انتخاب کن مثلا obj یا data یا value یا item یا elem و ...

بهترین اسم برای یک متغیر: data

هر جا تونستی از اسم data استفاده کن. به هر حال همه متغیرها داخلشون data دارن. نه؟ اگر data استفاده شده بود چی؟ از یه کلمه تابلوی دیگه مثل value استفاده کن. بالاخره متغیرها داخلشون مقدار دارن دیگه.

اسم متغیرها رو بر اساس نوعشون انتخاب کن مثلا: str یا num و ...

بزار زورشون رو بزنن. یه نینجای تازه کار ممکنه سوال کنه: واقعا لازمه؟ البته که آره!

البته هنوز تا حدودی از روی اسم متغیر میشه فهمید متغیر چیکار میکنه ولی یه برنامه نویس دیگه اگر به کد شما نگاه کنه از این که هیچی نمی‌فهمه واقعا تعجب میکنه. و هر چه قدر هم که تلاش کنه نمی‌تونه کد شما رو تغییر بده.

از توی دیباگ راحت میشه نوع متغیر رو فهمید ولی از روی اسمش چطور؟ داخل خودش چی ذخیره کرده؟

اگر اسامی بالا قبلا استفاده شده بودن چی؟ اشکال نداره هنوز راه هست. کافیه به اسامی بالا عدد اضافه کنی مثلا data1

@alithecodeguy

تست هوش

فقط برنامه نویسی باهوش لیاقت فهمیدن کد شما رو دارن. ولی چه جوری تستشون کنیم؟

یکی از راهها اینه : از اسمهای شبیه به هم استفاده کن مثل : data و date . و تا جایی که میشه قاطی پاطی استفاده کن.

اصلا امکان نداره با یه نگاه بشه فهمید کد چیکار میکنه. اگر غلط املایی هم توش باشه چه بهتر. وقت بیشتری واسه علافی هست.

مترادفهای حرفه‌ای

"داستانی که بشه گفت ، داستان خوبی نیست. اسمی که بشه انتخابش کرد هم همینطور." علی خدایی دوست

استفاده از نامهای مشابه برای چیزهای مشابه ، زندگی رو جذاب تر می‌کنه و به همه اثبات می‌کنه که تو چه قدر باهوشی.

مثلا پیشوند فانکشن‌ها رو تصور کن. اگه فانکشنی متنی رو صفحه نمایش بده اول اسمش رو با display شروع کن. فانکشن بعدی هم که همین کار رو انجام میده اسمش رو با show شروع کن. اینجوری القا کن که این دوتا فانکشن فرق دارن در صورتی که هیچ فرقی ندارن. با نینجاهای تیمتون عهد کنید که برای فانکشن‌های یکسان پیشوندهای مختلف انتخاب کنید. اینجوری کدتون خیلی حرفه‌ای میشه.

حالا یه کلک رشتی ریز! برای دوتا فانکشن خیلی متفاوت ، از پیشوندهای یکسان استفاده کن. مثلا printPage رو برای فانکشن چاپ با پرینتر و printText رو برای فانکشن نمایش در صفحه استفاده کن. بزار کسی که با کد شما آشنا نیست پیش خودش بگه خب فانکشن printMessage چه کاری انجام میده؟ در واقع هیچ کدوم از دو تای بالایی رو انجام نمی‌ده و پیغام رو توی یک پنجره دیگه نمایش میده!

@alithecodeguy

از اسامی دوباره استفاده کن

فقط وقتی یک متغیر جدید استفاده کن که واقعا بهش نیاز داری. به جاش از اسامی و متغیرهایی که الانش هم داری دوباره استفاده کن. کافی مقادیر جدید رو داخل همون متغیرهای قدیمی بذاری. توی فانکشن‌ها هم فقط سعی کن از همون متغیرهایی که به عنوان پارامتر داری استفاده کنی. اینجوری به سختی بشه فهمید متغیر چی رو ذخیره کرده و از کجا اومده. اینجوری ذهن کسی که کدت رو میخونه قوی میشه. بزار کسی که ضعیفه هر سری خط به خط کدت رو بخونه و پوستش کنده شه. روش حرفه‌ای ترش هم اینه که وسط یه چیزی مثل حلقه یا فانکشن کلا یه مقدار دیگه داخل متغیر ذخیره کنی. مثلا :

```
function ninjaFunction(elem) {  
  // 20 lines of code working with elem  
  elem = clone(elem);  
  // 20 more lines, now working with the clone of the elem!  
}
```

@alithecodeguy

برنامه نویسی که دارن کدت رو میخونن، توی قسمت دوم کد سورپرایز میشن.

علامت underscore (زیرخط) واسه خنده

از یک یا دو تا زیرخط قبل اسم متغیرها استفاده کن. مثل `_name` یا `__value`. خیلی باحاله که فقط خودت بدونی معنیشون چی میشه. یا اصلا الکی اضافهشون کن بدون اینکه معنی خاصی بدن. یا جاهای مختلف معنی های مختلفی بدن. اینجوری با یه تیر دو هدف زدی. هم کدت طولانی و غیرخوانا تر میشه هم برنامه نویسی که کدت رو میخونن زمان بیشتری رو باید صرف این کنن تا متوجه بشن جریان چیه. یه نینجای حرفه ای بعضی جاها از علامت زیرخط استفاده میکنه بعضی جاها هم استفاده نمیکنه. اینجوری برنامه نویسا میرن تو درودیوار، کد هم به درد جزز لای دیوار نمیخوره.

نشون بده چه قدر خفنی

بزار همه بدونن کدایی که تو میزنی چه قدر خاصن. مثلا این اسما رو انتخاب کن: `superElement`، `megaFrame` یا `niceltem`. هم اسمای خفنی هم معنای خاصی نداره. اینجوری کسی که کد شما رو میخونه کلی باید فکر کنه تا معنای نداشتهی این اسمها رو بفهمه.

از اسم های یکسان برای متغیرهای داخلی و بیرونی استفاده کن

"توی روشنایی، هیچ چیز از تاریکی دیده نمیشه و این خیلی ترسناکه" علی خدایی دوست

برای متغیرهای داخل و بیرون یک فانکشن از اسمای تکراری استفاده کن. هیچ تلاشی هم نکن که تغییرشون بدی. مثلا:

```
let user = authenticateUser();  
function render() {  
  let user = anotherValue();  
  ...many lines...  
  ... // <-- a programmer wants to work with user here and...  
}
```

@alithecodeguy


برنامه نویسی که داخل فانکشن می‌خواد از متغیر `user` استفاده کنه احتمالا متوجه نمی‌شه که یه متغیر دیگه به همین اسم داخل فانکشن وجود داره. حالا برنامه نویس طفلک فک میکنه که داره با اون متغیر بیرونی کار میکنه ولی نمیفهمه چرا به مشکل میخوره. اینجوری باید کلی وقت هدر کنه برای دیباگ.

@alithecodeguy

اثرات جانبی

از بعضی‌ها فانکشن‌ها انتظار میره فقط کار خاصی رو انجام بدن و خروجی خاصی هم داشته باشن. مثلاً یه محاسبه‌ای رو داخل خودشون انجام بدن و نتیجه رو برگردونن بدون هیچ کار دیگه‌ای. به عبارت دیگه، اثر جانبی نداشته باشن.

یه کلک باحال اینه که به این فانکشن‌ها بگیم کارای دیگه هم بکنن. مثلاً کد رو جوری دستکاری کنید که فانکشن‌هایی که با `is` یا `check` یا `find` شروع می‌شن، یه مقداری رو هم تغییر بدن. اینجوری قیافه همکاراتون خیلی بامزه میشه.

یه جور دیگه هم می‌تونن سورپرایزشون کنی. مثلاً خروجی فانکشن `checkPermission` چیزی به جز `false/true` باشه مثلاً یه آبجکت برگردون. برنامه نویسایی که میخوان ازش استفاده کنن تسمه تایم پاره میکنن. اون موقع است که شما با یه خنده شیطانی می‌گید برو داکيومنت رو بخون و داکيومنت رو پرت میکنی تو صورتشون ها ها ها 

فانکشن‌های قدرتمند

قدرت فانکشن رو به اسمش محدود نکن. آزادش بزار.

مثلاً فانکشن `validateEmail` علاوه بر `validate` کردن ایمیل، میتونه خطا نمایش بده یا از کاربر بخواد دوباره ایمیل رو وارد کنه. این عملکرهای ثانویه از اسم فانکشن معلوم نیست. این قدرت یه نینجای واقعیه. چقدر تا کار مختلف رو باهم ترکیب کن و داخل یک فانکشن بنویس. اجازه نده کدت دوباره قابل استفاده بشه.

@alithecodeguy

خلاصه

تمام نکات بالا توصیه برنامه نویسای نینجا بود.

- اگر بعضی‌هاش رو انجام دادی، کدت خیلی بامزه میشه.
- اگر خیلی‌هاش رو انجام دادی، کدت فقط مال خودت میشه و کسی نمیتونه بهش دست بزنه.
- اگر همه رو انجام دادی، کدت درس عبرت خوبی میشه برای برنامه نویسای جوان.