

کامنت‌ها

همانطور که از فصل‌های قبل می‌دانیم ، کامنت‌ها را می‌توان به وسیله‌ی // به صورت یک خطی و به وسیله‌ی /*...*/ به صورت چندخطی نوشت . ما معمولا از کامنت‌ها برای تشریح چگونگی و چرایی عملکرد یک پروژه استفاده می‌کنیم . در نگاه اول ممکن است کامنت‌ها ساده به نظر برسند ولی برنامه نویسان مبتدی معمولا از آنها اشتباه استفاده می‌کنند .

@alithecodeguy

کامنت‌های بد

افراد مبتدی از کامنت ها برای شرح نحوه چگونگی عملکرد کد استفاده می‌کنند . مثال :

```
// This code will do this thing (...) and that thing (...)
```

```
// ...and who knows what else...
```

```
very; complex; code;
```

ولی در یک کد خوب ، مقدار چنین توضیحاتی باید به حداقل برسد . کد باید حتی بدون کامنت نیز به راحتی قابل فهم باشد .

یک قانون کلی در مورد آنها وجود دارد : اگر کد به قدری غیر واضح است که نیاز به کامنت دارد ، پس احتمالا باید دوباره نوشته شود .

دستورالعمل : فانکشن کردن عملکردهای تکراری

گاهی اوقات بهتر است بخشی از کد با یک فانکشن جایگزین شود ، مثال :

```
function showPrimes(n) {  
  nextPrime:  
  for (let i = 2; i < n; i++) {  
    // check if i is a prime number  
    for (let j = 2; j < i; j++) {  
      if (i % j == 0) continue nextPrime;  
    }  
    alert(i);  
  }  
}
```

@alithecodeguy

بهتر است کد فوق به کمک فانکشن مجزای isPrime به شکل زیر نوشته شود:

```
function showPrimes(n) {  
  for (let i = 2; i < n; i++) {  
    if (!isPrime(i)) continue;  
    alert(i);  
  }  
}
```

```
function isPrime(n) {  
  for (let i = 2; i < n; i++) {  
    if (n % i == 0) return false;  
  }  
  return true;  
}
```

@alithecodeguy

حال کد فوق به راحتی قابل فهم است. خود فانکشن به نوعی کامنت محسوب می شود. به این کدها خود-مشرح می گویند.

دستورالعمل : ساخت فانکشن های متعدد

اگر کدی طولانی مطابق مثال زیر داشته باشیم :

```
// here we add whiskey  
for(let i = 0; i < 10; i++) {  
  let drop = getWhiskey();  
  smell(drop);  
  add(drop, glass);  
}  
  
// here we add juice  
for(let t = 0; t < 3; t++) {  
  let tomato = getTomato();  
  examine(tomato);  
  let juice = press(tomato);  
  add(juice, glass);  
}
```

@alithecodeguy

بهتر است آن را به چند فانکشن مختلف تقسیم کنیم:

```
addWhiskey(glass);
addJuice(glass);
function addWhiskey(container) {
  for(let i = 0; i < 10; i++) {
    let drop = getWhiskey();
    //...
  }
}
function addJuice(container) {
  for(let t = 0; t < 3; t++) {
    let tomato = getTomato();
    //...
  }
}
```

@alithecodeguy

بار دیگر ، فانکشن‌ها خودشان گویای عملکرد خودشان هستند . نیازی به کامنت نیست . همچنین این نحوه نوشتن برای ساختار کد نیز مناسبتر است . واضح است که عملکرد هر فانکشن چیست ، چه چیزی دریافت می کند و چه چیزی برمی گرداند . در واقعیت نمی توانیم به طور کلی از کامنت های مشروح صرف نظر کنیم . الگوریتم های پیچیده ای وجود دارند که برای هدف خاصی تغییر کرده اند . ولی به طور کلی باید تا جایی که امکان دارد کدها را به شکل ساده و خود-مشروح بنویسیم .

کامنت های خوب

اگر کامنت های مشروح بد هستند ، کامنت های خوب کدامند؟

@alithecodeguy

۱- کامنت هایی که معماری را توضیح می دهند :

کامنت هایی که یک دید کلی از اجزا ، نحوه تعامل آنها ، جریان کد در موقعیت های مختلف و به طور کلی یک دید کلی از کد به ما می دهند . زبان مخصوصی به نام UML وجود دارد که بوسیله آن می توان نمودارهای حرفه ای از معماری کد و نحوه کار آن ایجاد کرد .

۲- کامنت هایی که پارامترهای مختلف و کاربرد آنها را شرح می دهد :

گرامر مخصوصی به نام JSDoc برای داکيومنت کردن یک فانکشن وجود دارد : کاربرد ، پارامتر ، مقدار بازگشتی

مثال :

```
/**
 * Returns x raised to the n-th power.
 *
 * @param {number} x The number to raise.
 * @param {number} n The power, must be a natural number.
 * @return {number} x raised to the n-th power.
 */
function pow(x, n) {
  ...
}
```

چنین کامنت‌هایی به ما کمک می‌کنند تا هدف کد را بدون اینکه آن را مطالعه کنیم درک کنیم و از آن به درستی استفاده نماییم. همچنین ادیتورهای مانند WebStorm اینگونه کامنت‌ها را فهمیده و بوسیله آنها می‌توانند autocomplete و codechecking خودکار در اختیار برنامه نویس قرار دهند.

همچنین ابزارهایی مانند JSDOC3 وجود دارند که می‌توانند از روی این کامنت‌ها داکيومنت‌هایی با فرمت html تولید کنند. برای اطلاعات بیشتر به سایت زیر مراجعه نمایید:

<http://usejsdoc.org/>

چرا فلان کار به این شکل انجام شده است؟

برای اینکه بفهمیم کد چگونه کار می‌کند، مهم‌تر از چیزی که نوشته شده، چیزی است که نوشته نشده است. چرا فلان کار به این شکل حل شده است؟ خود کد جواب سوال ما را نمی‌دهد. اگر راه‌حل‌های مختلفی برای حل مساله وجود دارد، چرا این روش انتخاب شده است؟ مخصوصاً اینکه این روش خیلی روش واضحی نیست.

بدون اینگونه کامنت‌ها ممکن است خود را در یکی از شرایط زیر بیابید:

۱- شما یا همکارانتان کدی که قبلاً نوشته شده است را باز کنید و احساس کنند که کد بهینه نوشته نشده.

۲- پیش خود فکر کنید: قبلاً چه قدر احمق بودم و الآن چه قدر باهوشم. و سعی کنید کد را به روش هوشمندانه‌تر و بهینه‌تری بنویسید.

۳- هنگام بازنویسی به سختی خاطرتان هست که قبلاً چه کار کرده‌اید. اگر هم بخواهید دوباره کد را مطالعه کنید وقت زیادی صرف شده است.

کامنت‌هایی که راه حل‌ها را شرح می‌دهند بسیار مهم است. اینگونه کامنت‌ها کمک می‌کنند که توسعه کد ادامه پیدا کند.

اگر کد ویژگی های ظریف و خاصی دارد قطعا ارزش دارد که در مورد آن کامنت گذاری شود.

@alithecodeguy

خلاصه

یکی از نشانه های یک برنامه نویس خوب ، بودن یا نبودن کامنت هاست .

کامنت خوب به ما کمک می کند که کد را به شکل صحیحی نگهداری کنیم ، بعد از مدتی دوباره به آن رجوع کنیم و از آن به شکل بهتری استفاده کنیم .

این موارد را کامنت کنید :

- ساختار کلی . نگاه بالا به پایین به کد .
 - کاربرد توابع .
 - راه حل های مهم . مخصوصا آنها که خیلی واضح نیستند .
- از این موارد خودداری کنید :
- کامنت اینکه کد چگونه کار می کند و چه کاری انجام می دهد .
 - کامنت گذاری در مواقعی که کد آنقدر سخت است که نمی شود آن را ساده تر و خود-مشرح نوشت .

@alithecodeguy

همچنین کامنتها توسط خود ادیتورها و ابزارهای مختلف نیز استفاده می شوند .