

DON'T USE FLAGS AS FUNCTION PARAMETERS



```
function createFile(name: string, temp: boolean)
{
  if (temp) {
    fs.create(`./temp/${name}`);
  } else {
    fs.create(name);
  }
}
```



```
function createTempFile(name: string) {
  createFile(`./temp/${name}`);
}

function createFile(name: string) {
  fs.create(name);
}
```



USE MEANINGFUL VARIABLE NAMES



```
function between<T>(s9: T, n1: T, m1: T): boolean {  
    return n1 <= s9 && s9 <= m1;  
}
```



```
function between<T>(value: T, car: T, engine: T): boolean {  
    return car <= value && value <= engine;  
}
```



Sina Riyahi

FUNCTION ARGUMENTS (2 OR FEWER IDEALLY)



```
function createMenu(title: string, body: string, buttonText: string, cancellable: boolean) {  
  // ...  
}
```

```
createMenu('Foo', 'Bar', 'Baz', true);
```




```
function createMenu(options: { title: string, body: string, buttonText: string, cancellable: boolean }) {  
  // ...  
}
```

```
createMenu({  
  title: 'Foo',  
  body: 'Bar',  
  buttonText: 'Baz',  
  cancellable: true  
});
```




Sina Riyahi

USE PRONOUNCEABLE VARIABLE NAMES



```
type Stu = {  
  lan: string;  
  bir: Date;  
  We: number;  
}
```



```
type Student = {  
  language: string;  
  birth: Date;  
  weight: number;  
}
```



USE THE SAME VOCABULARY FOR THE SAME TYPE OF VARIABLE



```
function getStudentInfo(): Student;  
function getStudentDetails(): Student;  
function getStudentData(): Student;
```



```
function getStudent(): Student;
```



Sina Riyahi

USE EXPLANATORY VARIABLES



```
declare const students: Map<string, Student>;  
  
for (const keyValue of students) {  
  // iterate through students map  
}
```



```
declare const students: Map<string, Student>;  
  
for (const [id, student] of students) {  
  // iterate through students map  
}
```



Sina Riyahi

FUNCTIONS SHOULD DO ONE THING



```
function emailActiveClients(clients: Client[]) {  
  clients.forEach((client) => {  
    const clientRecord = database.lookup(client);  
    if (clientRecord.isActive()) {  
      email(client);  
    }  
  });  
}
```



```
function emailActiveClients(clients: Client[]) {  
  clients.filter(isActiveClient).forEach(email);  
}  
  
function isActiveClient(client: Client) {  
  const clientRecord = database.lookup(client);  
  return clientRecord.isActive();  
}
```

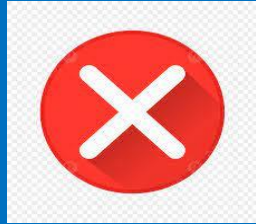


Sina Riyahi

DON'T ADD UNNEEDED CONTEXT



```
type Car = {  
  carMake: string;  
  carModel: string;  
  carColor: string;  
}  
  
function print(car: Car): void {  
  console.log(`${car.carMake} ${car.carModel} (${car.carColor})`);  
}
```



```
type Car = {  
  make: string;  
  model: string;  
  color: string;  
}  
  
function print(car: Car): void {  
  console.log(`${car.make} ${car.model} (${car.color})`);  
}
```



Sina Riyahi

USE DEFAULT ARGUMENTS INSTEAD OF SHORT CIRCUITING OR CONDITIONALS



```
function loadPages(count?: number) {  
  const loadCount = count !== undefined ? count : 10;  
  // ...  
}
```



```
function loadPages(count: number = 10) {  
  // ...  
}
```



Sina Riyahi

WANT TO LEARN MORE ?

Sina Riyahi

FOLLOW



FIND THIS POST HELPFUL