
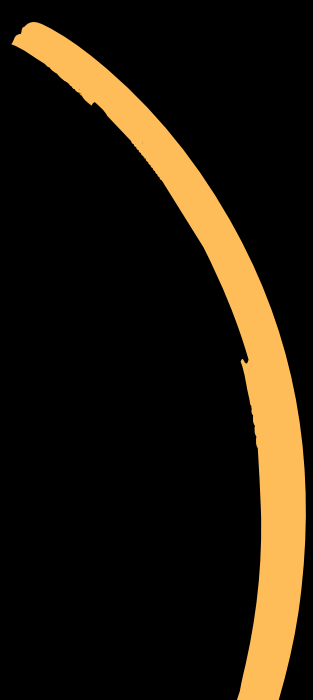



# 10 DICAS DE CLEAN CODE

```
export function getOrderMessage(status: string): string {  
  if (status === 'pending') {  
    return 'Order is pending.'  
  } else if (status === 'shipped') {  
    return 'Order has been shipped.'  
  } else if (status === 'delivered') {  
    return 'Order has been delivered.'  
  }  
}
```



```
const orderMessages = {  
  [OrderStatus.Pending]: 'Order is pending.',  
  [OrderStatus.Shipped]: 'Order has been shipped.',  
  [OrderStatus.Delivered]: 'Order has been delivered.',  
} as const
```



```
function getOrderMessage(status: OrderStatus): OrderMessage {  
  return orderMessages[status]  
}
```

# 1 - ELIMINE NÚMEROS E PALAVRAS MAGICAS



```
function fullDayElapsed(seconds){  
  return seconds >= 86400;  
}
```



```
function fullDayElapsed(seconds){  
  const SECONDS_IN_A_DAY = 86_400;  
  return seconds >= SECONDS_IN_A_DAY;  
}
```



## 2 - SUBSTITUA PARÂMETROS BOOLEANOS POR ENUMS



```
function sendNotification(isUrgent) {  
  console.log(isUrgent ?  
    "Sending URGENT notification!"  
    : "Sending regular notification.");  
}  
sendNotification(true);
```



```
enum NotificationPriority {  
  URGENT = "Urgent",  
  REGULAR = "Regular",  
}  
  
function sendNotification(priority) {  
  console.log(`Sending ${priority} notification.`);  
}  
  
sendNotification(NotificationPriority.URGENT);
```



# 3 - PARA FUNÇÕES COM MÚLTIPLOS PARÂMETROS USE OBJETOS



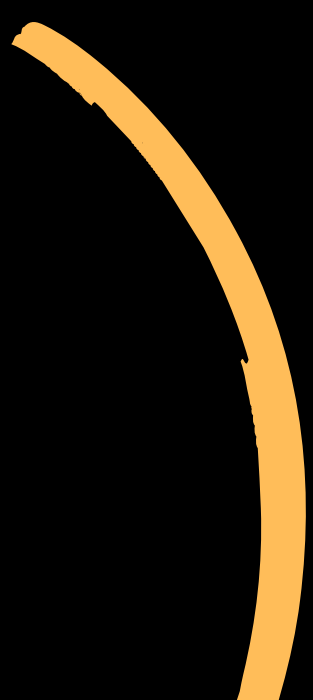
```
function createUser(  
  name: string,  
  age: number,  
  email: string,  
  isAdmin: boolean  
){}  
createUser("Alice", 30, "alice@example.com", true)
```



```
function createUser(params: CreateUserParams){  
  const { name, age, email, role } = params  
}  
createUser({  
  name: "Alice",  
  age: 30,  
  email: "alice@example.com",  
  role: UserRole.ADMIN  
})
```

# 4 - USE OBJETOS PARA REMOVER IFS

```
export function getOrderMessage(status: string): string {  
  if (status === 'pending') {  
    return 'Order is pending.'  
  } else if (status === 'shipped') {  
    return 'Order has been shipped.'  
  } else if (status === 'delivered') {  
    return 'Order has been delivered.'  
  }  
}
```



```
const orderMessages = {  
  [OrderStatus.Pending]: 'Order is pending.',  
  [OrderStatus.Shipped]: 'Order has been shipped.',  
  [OrderStatus.Delivered]: 'Order has been delivered.',  
} as const
```

```
function getOrderMessage(status: OrderStatus): OrderMessage {  
  return orderMessages[status]  
}
```

# 5 - USE EARLY RETURN PARA REDUZIR A INDENTAÇÃO DOS IFS



```
function processUser(user) {  
  if (user.isActive) {  
    if (user.age > 18) {  
      console.log("Processing user...");  
    } else {  
      console.log("User is underage.");  
    }  
  } else {  
    console.log("User is inactive.");  
  }  
}
```



```
function processUser(user) {  
  if (!user.isActive) return console.log("User is inactive.");  
  if (user.age <= 18) return console.log("User is underage.");  
  console.log("Processing user...");  
}
```






# 6 - USE **CONST** PARA TORNAR SEUS **IF** MAIS **LEGÍVEIS**



```
function canAccessDashboard(user){  
  return user.isActive &&  
    user.age >= 18 &&  
    user.subscriptionStatus === "active";  
}
```



```
function canAccessDashboard(user) {  
  const isUserActive = user.isActive;  
  const isUserOldEnough = user.age >= 18;  
  const hasActiveSubscription =  
    user.subscriptionStatus === "active";  
  
  return isUserActive &&  
    isUserOldEnough &&  
    hasActiveSubscription;  
}
```

# 7 - ESCREVA CÓDIGO AUTOEXPLICATIVO E ELIMINE COMENTÁRIOS



```
// Função que checa se um número é válido para fazer algum cálculo
function checkIfValidNumber(num: number): boolean {
  if (num <= 0) return false
  if (num % 1 !== 0) return false;
  if (num > 1000) return false;
  return true;
}
```



```
function isValidForCalculation(number: number){
  const isPositive = number > 0;
  const isInteger = number % 1 === 0;
  const isNotTooLarge = number <= 1000;
  return isPositive && isInteger && isNotTooLarge;
}
```



## 8 - UTILIZE **HAS** E **IS** PARA TORNAR SEUS **BOOLEANOS** MAIS **DESCRITIVOS**



```
const permission = user => user.role === 'admin';  
const age = age => age >= 18;  
const active = user => user.subscriptionStatus === 'active';
```



```
const hasPermission = user => user.role === 'admin';  
const isAdult = age => age >= 18;  
const hasActiveSubscription =  
user => user.subscriptionStatus === 'active';
```

# 9 - CRIE UM PADRÃO DE NOMENCLATURA



modifyUser()  
editUser()  
refreshUser()  
changeUser()  
adjustUser()  
alterUser()  
reviseUser()

■ - ação  
■ - escopo

updateUser()  
createUser()  
deleteUser()  
getUser()



updateUser()

# 10 - UTILIZE MÉTODOS DO JAVASCRIPT PARA MELHORAR A LEGIBILIDADE



```
function getActiveUsers(users){
  const activeUsers = [];
  for (let i = 0; i < users.length; i++) {
    if (users[i].isActive) {
      activeUsers.push(users[i].name);
    }
  }
  return activeUsers;
}
```

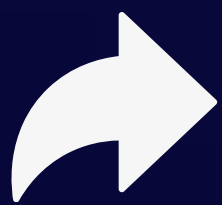


```
function getActiveUsers(users){
  return users
    .filter(user => user.isActive)
    .map(user => user.name);
}
```

# Gostou?



**Curta**



**Compartilhe**



**Salve**



**Ative as notificações**