Nommage des éléments

Le nommage en programmation est la pratique de donner des **noms explicites** et **lisibles** aux éléments du code (variables, fonctions, classes, constantes, etc.) Des noms bien choisis rendent le code plus lisible, compréhensible et maintenable, ce qui est crucial pour le travail collaboratif et la révision de code. Le choix des conventions de nommage dépend des langages et des préférences du projet, mais certaines conventions sont largement adoptées.

Le choix du nom

Le choix du nom doit refléter le **rôle** et **l'intention** de l'élément. Par exemple, une variable qui stocke un âge pourrait s'appeler **age** ou **userAge**, alors qu'une fonction qui calcule le prix total pourrait s'appeler **calculateTotalPrice**. Un bon nom doit être **:**

Descriptif : il doit indiquer la fonction ou l'objet qu'il représente.
Concis : un nom court et significatif est préférable.
Standardisé : suivre les conventions du langage (ex. get, set pour des accesseurs en Java) facilite la lecture.

La notation

La notation est le style de formatage du nom en lui-même, particulièrement pour les noms composés. Elle est cruciale pour différencier les **types d'éléments** et **respecter les conventions du langage**. Ci-dessous voici les principales notations.

Upper Case

Description : Toutes les lettres sont en **majuscule**, et les mots sont souvent séparés par des underscores .

Utilisation : Elle est principalement utilisée pour nommer les **constantes**, surtout dans les langages comme le C ou le Java.

But : Le format **upper** case signale visuellement que la valeur de l'élément ne doit pas être modifiée. En programmant, cela aide le développeur à distinguer facilement les constantes des variables modifiables

```
"String NOM_UTILISATEUR": "THEO HASCOET","int NOMBRE_UTILISATEUR": 5,
```

.

Snake Case

Description : Les mots sont séparés par des underscores **(_)**, et tous les caractères sont généralement en **minuscules**.

Utilisation : Très courant pour nommer les variables et fonctions en Python et dans certains autres langages de script.

But : Améliore la **lisibilité** en ajoutant des séparateurs (**underscores**) entre les mots, tout en évitant la confusion avec les **majuscules/minuscules**. C'est également une convention standard dans Python pour les variables et les fonctions

```
int nombre_utilisateur = 5; String nom_utilisateur = "Theo Hascoet";
```

.

Camel Case

Description : Le premier mot est en minuscule et chaque mot suivant commence par une majuscule, sans aucun séparateur.

Utilisation : Couramment utilisé pour nommer les variables et fonctions en JavaScript, Java, et Swift.

But : Ce style conserve une bonne **lisibilité** tout en évitant les séparateurs comme _ . Il est fréquemment choisi dans les langages qui privilégient des noms de variables courts mais distinctifs, car la majuscule initiale de chaque mot facilite la lecture des mots composés.

Pascal Case

Description : Chaque mot commence par une **majuscule**, et il n'y a **aucun séparateur**.

Utilisation : Principalement utilisé pour nommer les **classes** et **structures** dans des langages comme le C#, Java et le .NET.

But : Le Pascal Case permet de distinguer rapidement les **classes et structures des autres éléments**, car la majuscule initiale de chaque mot indique visuellement une entité importante. Cela **facilite la compréhension** de la hiérarchie dans le code, notamment dans les environnements orientés objet.

From:
https://wiki.siochaptalqper.fr/ - **Wiki SIO Chaptal**

Permanent link:
https://wiki.siochaptalqper.fr/doku.php?id=bloc1:prog:nommage&rev=1732043752

Last update: **2024/11/19 20:15**