Syntaxe des namespaces

Déclaration d'un `namespace`

Pour déclarer un `namespace` il suffit de faire :

```
<?php
  declare(encoding='UTF-8');
  namespace Projet
```

Il faut que ce soit la première instruction (à l'exception du mot clé `declare` pour spécifier éventuellement l'encodage) du fichier. Il ne faut aucun code `HTML` ou `PHP` avant.

Les `namespaces` s'appliquent aux classes, aux interfaces, aux traits, aux fonctions et aux constantes.

Utiliser les `namespaces` dans le contexte global

Par défaut, du code déclaré à l'extérieur d'un `namespace` spécifié est dans **l'espace de noms global**.

Créons un fichier `projet.php` dans lequel nous mettons :

```
<?php

  namespace Projet;

  const A = 42;

  function compter(string $chaine)
  {
      echo strlen($chaine);
  }

  class User
  {
      function __construct(public string $name)
      {
```

```
    }
}
```

Puis utilisons celui-ci dans notre fichier `index.php` :

```
<?php

  require_once 'projet.php';

  $user = new Projet\User('Paul');
  Projet\compter($user->name); // 4
  echo '<br>';
  echo Projet\A; // 42
```

Remarquez que nous devons préfixer chaque fonction, classe ou constante avec le nom du `namespace`. Dans le cas contraire, l'interpréteur regardera dans l'espace de noms global et elles ne seront pas trouvées.

Il y a heureusement une syntaxe plus simple qui permet d'utiliser des éléments d'un autre `namespace`, l'utilisation de `use` :

```
<?php

  use Projet\User;
  use const Projet\A;
  use function Projet\compter;

  require_once 'projet.php';

  $user = new User('Paul');
  compter($user->name); // 4
  echo '<br>';
  echo A; // 42
```

Utiliser des éléments de l'espace global dans un `namespace`

Nous avons vu comment utiliser des éléments d'un `namespace` dans l'espace global, mais nous n'avons pas encore vu comment faire l'inverse.

Nous pouvons soit préfixer par un antislash `\` les éléments que nous souhaitons récupérer de l'espace global :

```
<?php

  namespace Projet;

  $date = new \DateTime();
  echo $date->format(\DateTime::ATOM); // 2021-05-06T12:54:37+00:00
```

Soit utiliser `use` :

```
<?php

  namespace Projet;

  use DateTime;

  $date = new DateTime();
  echo $date->format(DateTime::ATOM); // 2021-05-06T12:54:37+00:00
```

Lorsque nous utilisons `use` suivi d'un identifiant sans antislash, cela signifie que nous le récupérons de l'espace de noms global.

Utiliser dans le même fichier des éléments ayant le même nom

Si vous utilisez par exemple dans le même fichier deux fonctions ayant le même nom : par exemple l'une provenant de l'espace global et l'autre d'un `namespace`, vous pouvez utiliser un alias avec `as`.

Ainsi, si nous avons un fichier `projet.php` :

```
<?php

  namespace Projet;
```

```
function direBonjour()
{
    echo 'Coucou !';
}
```

Nous pouvons utiliser un alias dans `index.php` :

```
<?php

use function Projet\direBonjour as direCoucoud;

require_once 'projet.php';

function direBonjour()
{
    echo 'Bonjour !';
}

direCoucoud();
```