1. Générer du HTML dans une variable pour un email (ou log)
2. Faire un post-traitement (exemple : minifier, encoder, etc.)

1️⃣ Stocker le HTML pour l’envoyer par email

```
use App\UI\Template;

// Initialisation
$tpl = new Template(ROOT_PATH . '/templates/email');
$tpl->setFile('mail', 'welcome.html');

// Variables dynamiques pour le mail
$tpl->setVar([
    'USERNAME' => $user['name'],
    'ACTIVATION_LINK' => $activationLink
]);

// Stocker le rendu dans une variable (aucun affichage direct)
$tpl->parse('body', 'mail');
$htmlContent = $tpl->get('body');

// Utilisation : envoyer un mail HTML
mail(
    $user['email'],
    "Bienvenue sur Git.Docs",
    $htmlContent,
    "Content-Type: text/html; charset=UTF-8"
);
```

2️⃣ Post-traiter le rendu avant affichage

Par exemple, on veut compresser ou transformer le HTML avant de l’envoyer au navigateur :

```
$tpl->setFile('main', 'page.html');
// ... setVar etc ...
$tpl->parse('cache', 'main');

$html = $tpl->get('cache');

// Post-traitement : minifier le HTML (exemple simple)
$html = preg_replace('/\s+/', ' ', $html); // simplification

echo $html;
```

3️⃣ Envoyer un PDF, un export, une API, etc.

```
$tpl->setFile('main', 'export.html');
$tpl->setVar(...);
// Générer dans une variable
$tpl->parse('export', 'main');
$html = $tpl->get('export');

// Ici : convertir en PDF, exporter, envoyer, etc.
```

En résumé :

- parse → stockage dans une variable interne
- get → récupérer la chaîne
- pparse → affichage direct

On peut faire autant de parse que l'on veux pour préparer différents blocs, puis décider de ce que l'on affiche ou stocke à la fin.