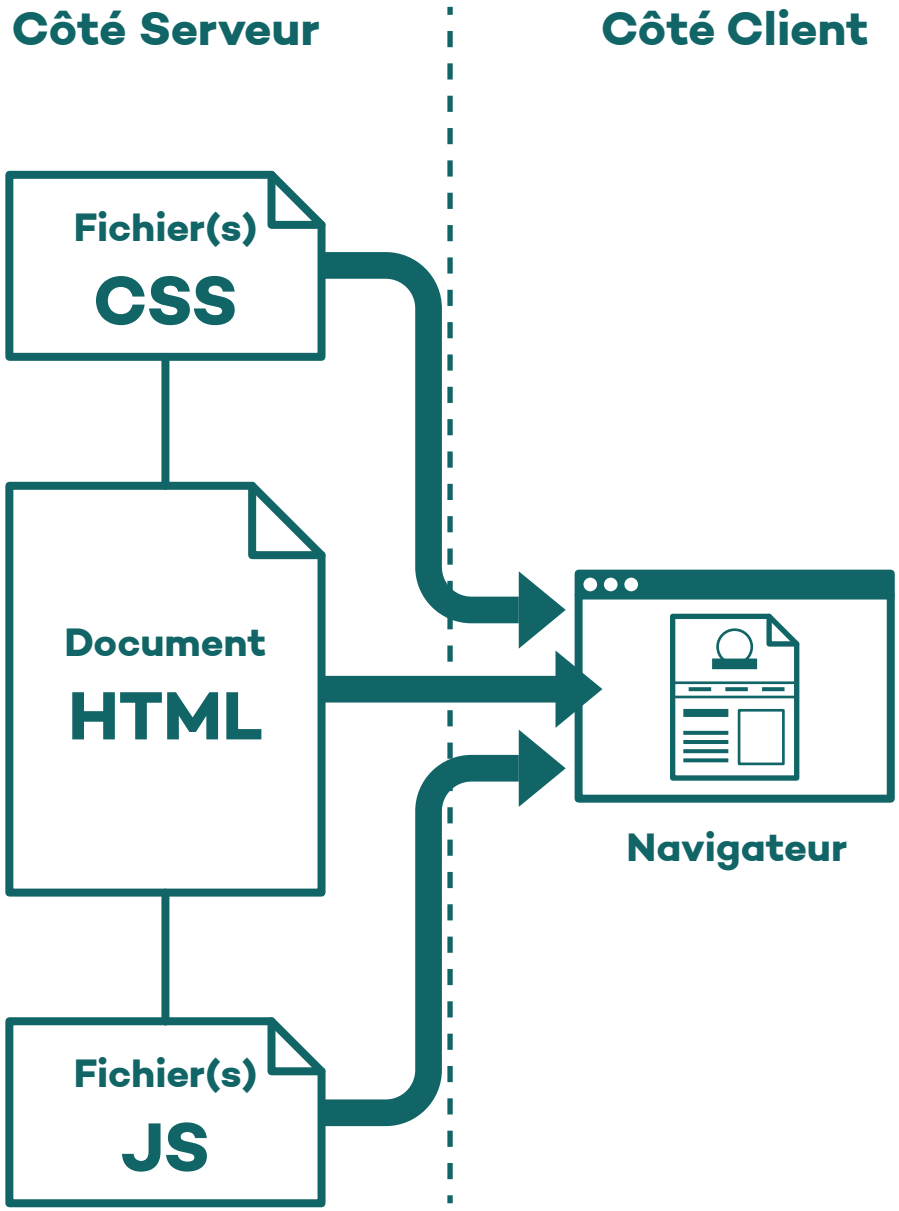
SCRIPTS **PHP**

et principes de base d'un

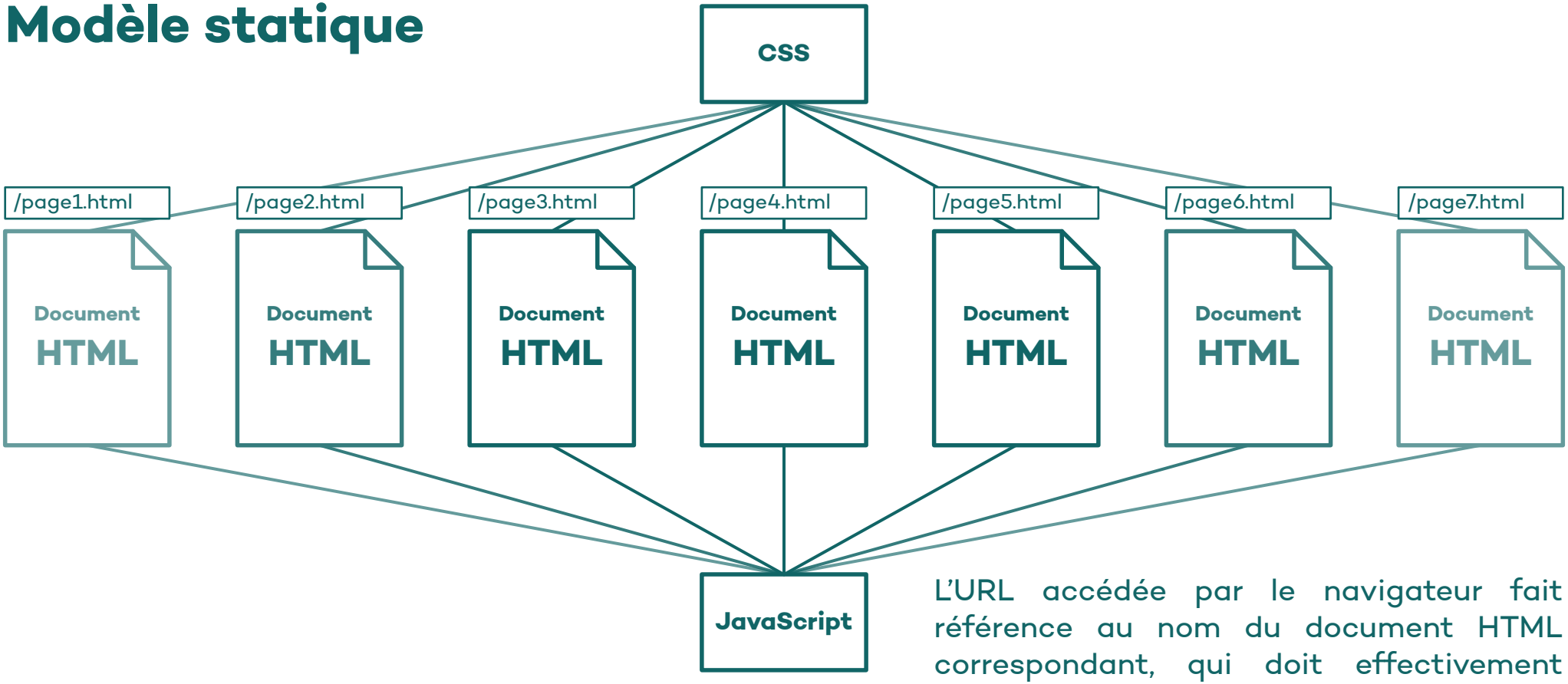
site web dynamique



Pour l'affichage d'un site statique, le serveur livre au client (le navigateur) les fichiers HTML, CSS et JavaScript, tels qu'ils ont été hébergés.

Selon ce modèle, chaque page du site doit faire l'objet de son propre fichier HTML. La structure et les contenus doivent y figurer «en dûr».

Modèle statique



L'URL accédée par le navigateur fait référence au nom du document HTML correspondant, qui doit effectivement exister sur le serveur, et présenter l'intégralité (structure et contenus) de la page que l'on veut afficher.

En imaginant par exemple un blog fonctionnant selon ce principe :

• Il faudrait mettre en ligne pour chaque article sa propre page HTML, en y faisant directement figurer les bons contenus, et en reproduisant sur chacune le code HTML correspondant à la structure-type d'une page «article».

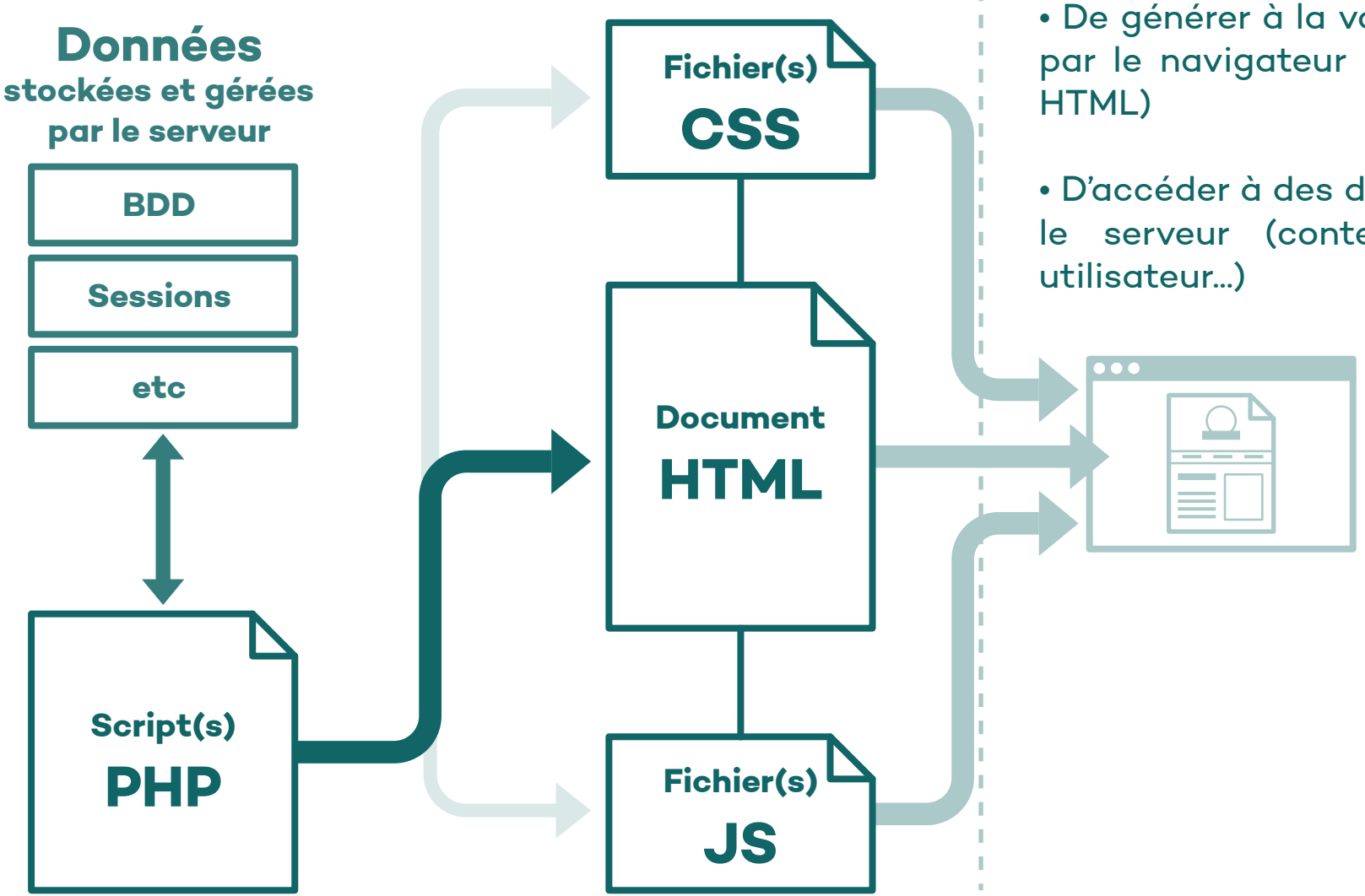
• Pour modifier le contenu d'un article il faudrait éditer directement son fichier, sans oublier les pages présentant éventuellement un aperçu de cet article.

• Les éléments de navigation tributaires de l'article où l'on se trouve (liens vers article suivant ou précédent, vers articles parus le même mois ou dans la même catégorie, etc) doivent être ajustés manuellement dans chaque fichier HTML.

• Une modification dans la structure HTML globale du site (menu, autres éléments d'interface) doit être répétée manuellement autant de fois qu'il existe d'articles et de pages.

• Le volume de données à placer en ligne, puis à potentiellement transmettre au client lors d'une navigation sur le site, pourra vite devenir (inutilement) colossal.

Génération dynamique des pages



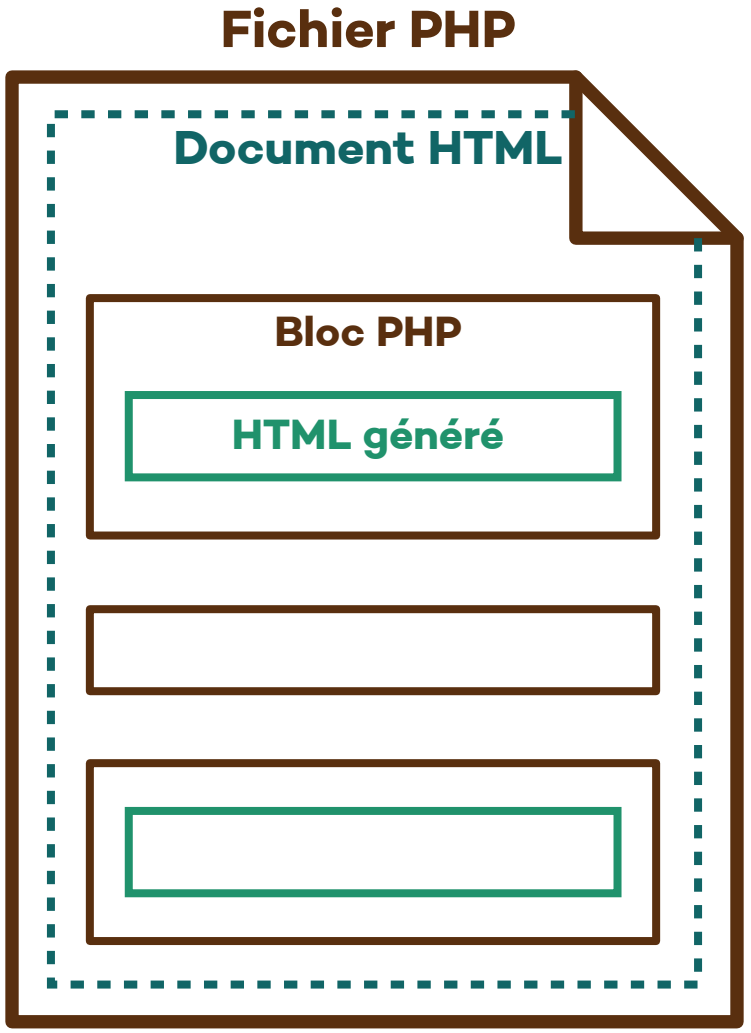
Dans le cas d'un site dynamique, on utilisera des scripts PHP. Ceux-ci permettront, en premier lieu :

- De générer à la volée le code requis par le navigateur (principalement le HTML)

- D'accéder à des données gérées par le serveur (contenus, préférences utilisateur…)

Pour permettre à PHP de modifier une page, il est important que le fichier soit au format .php et placé sur un serveur capable de l'interpréter (comme c'est le cas pour la plupart des serveurs web).

Dans ce fichier, tout ce qui est hors d'un bloc <?php ?> sera interprété comme du texte / HTML.



Un exemple de HTML généré par PHP :

Ci-dessous, deux manières d'insérer un titre dynamique dans un élément . Le titre affiché dépendra d'une variable $page, qui aura été définie plus tôt, dans un autre bloc PHP du document. L'instruction echo permettra ensuite d'écrire directement sur le document.

```
<div>

<?php

if ($page == "home") {
   $mon_titre = "Accueil";
} else {
   $mon_titre = "Une page";
}

echo "<h1>".$mon_titre."</h1>";

?>

</div>
```

```
<?php

if ($page == "home") {
   $mon_titre = "Accueil";
} else {
   $mon_titre = "Une page";
}

?>

<div>

<h1><?php echo $mon_titre; ?></h1>

</div>
```

Cet exemple nous permet par ailleurs de relever quelques points de syntaxe en PHP :

• Quelques similitudes avec celle de JavaScript. La syntaxe des tests conditionnels par exemple, incluant l'obligation du double =. En PHP les points-virgules sont obligatoires après chaque instruction (là où en JavaScript un retour à la ligne est techniquement équivalent).

• Pour nommer les variables, la liberté est la même. En revanche ici le nom d'une variable commencera toujours par «$». Notons également l'absence de mot-clé «var» à la déclaration.

• Pour concaténer (joindre à la suite) des chaînes de caractères, on utilisera des points, le + étant réservé aux opérations numériques.

NOTE : L'instruction suivante

```
echo "<h1>".$mon_titre."</h1>";
```

peut également se rencontrer sous la forme :

```
echo "<h1>$mon_titre</h1>";
```

En effet dans certains cas on peut se dispenser des points lors de l'inclusion d'une variable dans une chaîne. Il est nécessaire pour cela que la délimitation entre le nom de la variable et la chaîne qui la suit ne laisse aucun doute (cette chaîne doit donc commencer par un caractère ne pouvant pas faire partie d'un nom de variable).

Pour délimiter une chaîne, les guillemets simples et doubles sont admis. L'échappement de caractères fonctionne de la même manière qu'en Javascript, à l'aide de l'anti-slash :

```
echo 'des "guillemets"';
echo "des \"guillemets\"";
echo "l'apostrophe";
echo 'l\'apostrophe';
```

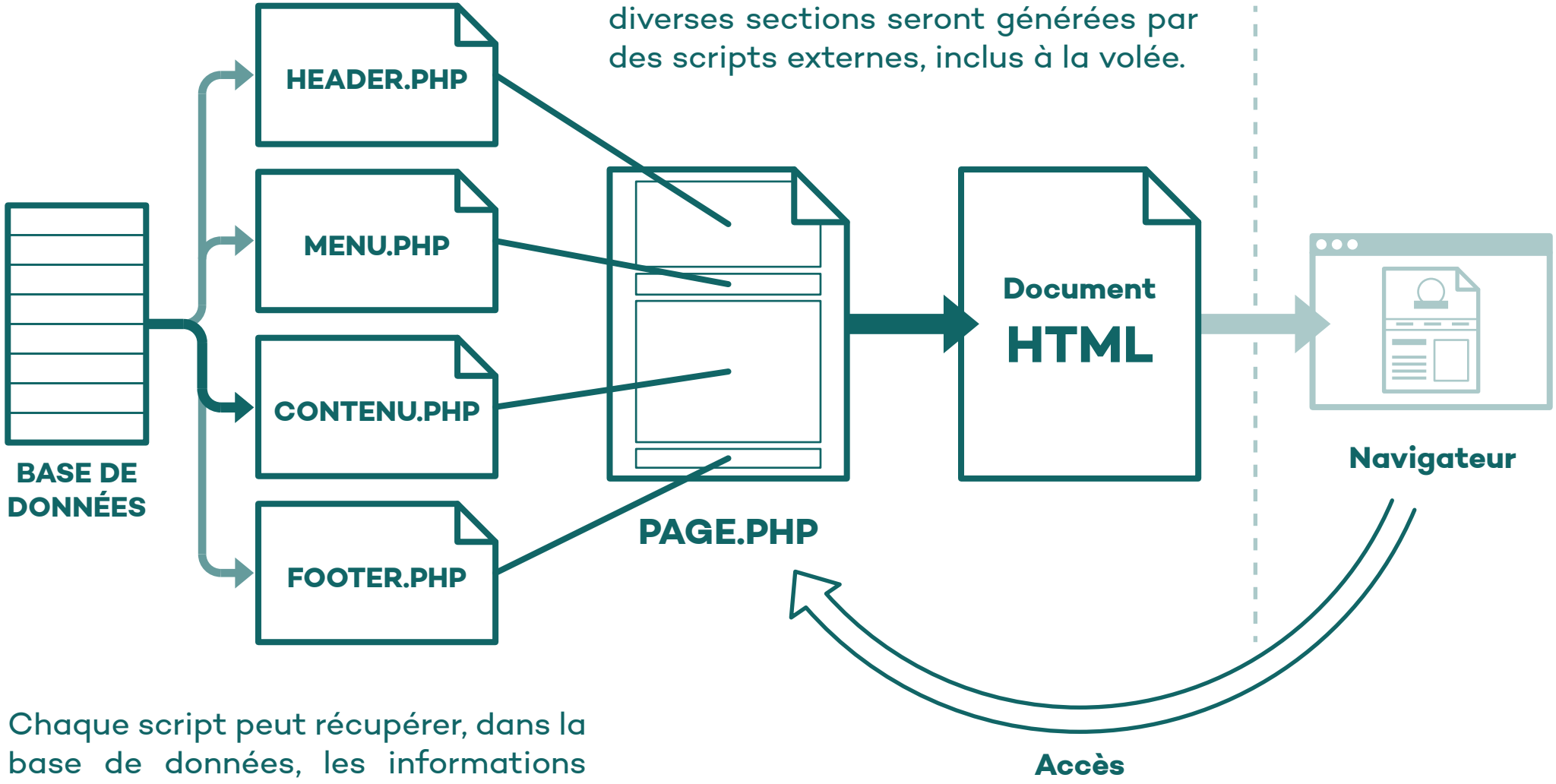
Comme déjà évoqué, dans un fichier .php le texte placé hors des blocs PHP peut être directement interprété comme du code HTML. De plus, il est possible d'ouvrir un test conditionnel (ou toute autre structure nécessitant des accolades) dans un bloc PHP et de le fermer dans un autre.

Il en résulte la possibilité de syntaxe présentée ci-contre, dans laquelle on est dispensé de l'instruction echo (et de guillemets en tant que délimitation des chaînes à afficher). Ici PHP ne gère que le test en lui-même, et c'est le HTML qui devient conditionnel.

Cette syntaxe peut s'avérer pratique lorsque le HTML à afficher est volumineux ou pénible à formater comme chaîne.

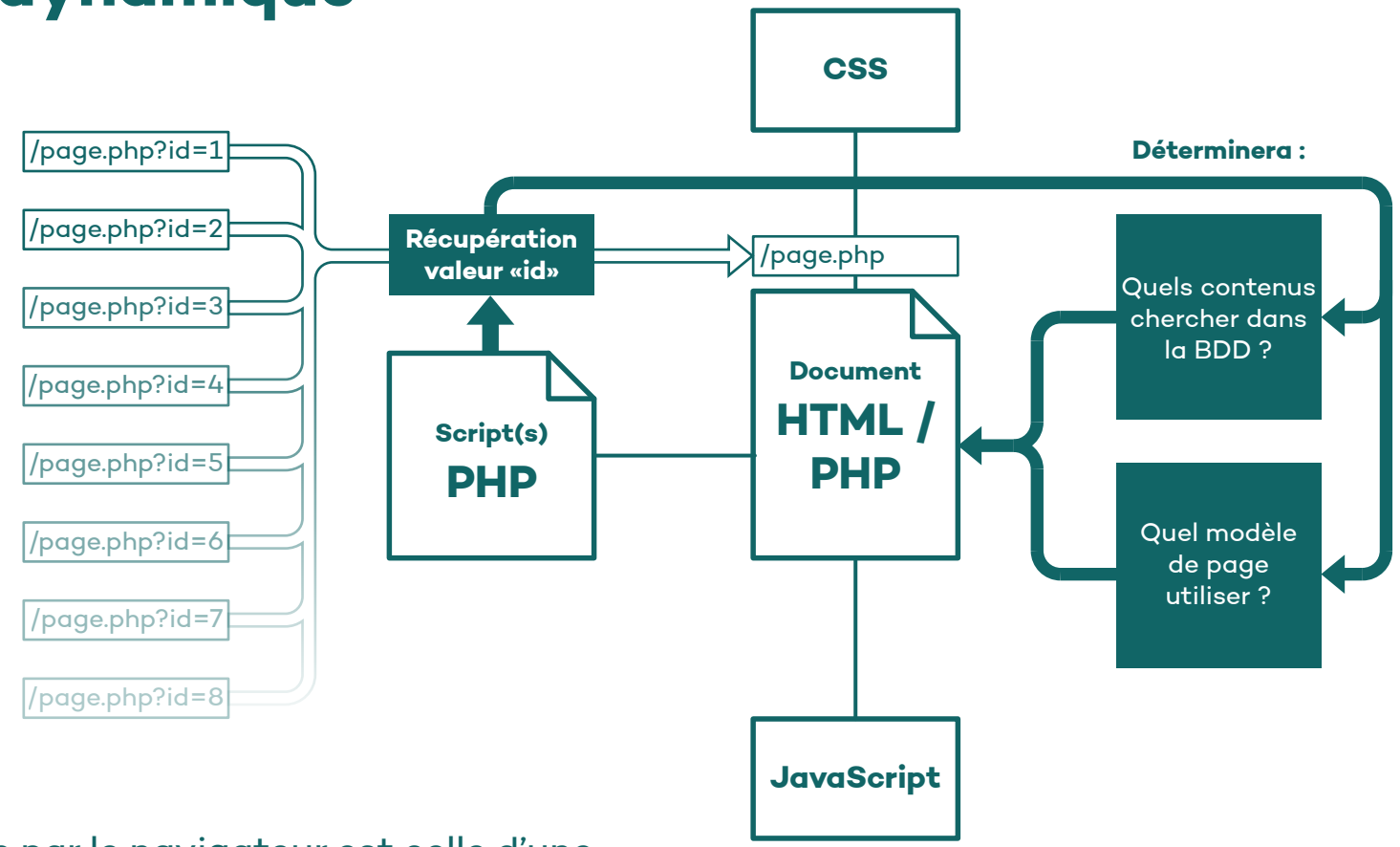
```
<div>

  <?php if ($page == "home") { ?>

   <div id = "maDiv" class = "accueil">
     <h1>Accueil</h1>
   </div>

  <?php } else { ?>

   <div id = "maDiv" class = "page">
     <h1>Une page</h1>
   </div>

  <?php } ?>

</div>
```

PHP permet de construire des modèles de pages intelligents. On utilise un script principal qui sera accédé en tant que page, et dont les diverses sections seront générées par des scripts externes, inclus à la volée.



Chaque script peut récupérer, dans la base de données, les informations dont il a besoin et générera sa partie du code HTML en conséquence.

Modèle dynamique



L'URL accédée par le navigateur est celle d'une page «template». Celle-ci pourra afficher différents contenus et structures, en se basant sur des informations supplémentaires passées le plus souvent comme paramètres à la fin de l'URL.

Inclure des scripts externes

Si un tronçon de PHP ou de HTML doit être reproduit sur plusieurs pages, il devient intéressant de le placer dans un fichier .php séparé, puis d'y faire appel dès que nécessaire à l'aide de la fonction include() de PHP :

```
<?php
include("chemin/vers/fichier.php");
?>
```

Comme le nom de la fonction l'indique, l'extrait placé dans le fichier externe sera littéralement inclus dans le fichier courant lors de l'exécution du PHP, à l'endroit où include() a été appelé.

Passer des variables dans une URL

Imaginons un fichier unique, que l'on voudrait pouvoir utiliser à la fois comme page d'accueil et comme page interne, son contenu dépendant du cas. On peut, lors de l'accès à cette page, lui transmettre sous forme de paramètres les informations déterminant quel cas appliquer.

Les paramètres fonctionnent comme des variables : ils ont un nom et une valeur. On les ajoute après l'URL du fichier, après un «?».

Dans notre cas, si le fichier s'appelle page.php et que l'on veut lui passer un paramètre appelé «p» contenant la valeur «home», on y accèder via :

page.php?p=home

Dans le cas où plusieurs paramètres sont passés, ils doivent être séparés par des «&» :

page.php?p=blog&article=43

À l'initialisation de la page, ces informations seront placées dans le tableau superglobal $_GET, où PHP peut les récupérer.

```
<?php

$page = $_GET["p"];

if ($page == "home") {
   include("script-home.php");
} else {
   include("script-default.php");
}

?>
```

• Dans les valeurs passées en paramètres, certains caractères sont interdits, comme l'espace qui ne peut pas faire partie d'une URL, ou «?» et «&» qui seront interprétés comme délimiteurs des paramètres. Ces caractères peuvent être remplacés par leurs codes ASCII précédés d'un «%». Par exemple pour passer la chaîne :

«ces caractères sont-ils autorisés ?»

Il faudra utiliser :

«ces%20caract%C3%A8res%20sont-ils%20autoris%C3%A9s+%3F»

On constate que les caractères accentués doivent également être encodés.

• Il n'est pas possible de passer des paramètres lors d'un appel via la fonction include(), celle-ci se contentant d'inclure du contenu sans modifier le tableau $_GET. Le code PHP inclus peut néanmoins utiliser les variables mises en place plus tôt sur la page courante.

```php
<?php // sur la page courante
  $page = "home";
  include("script.php?p2=home");
?>

<?php // contenu de script.php

  $page2 = $_GET["p2"];
  echo $page2; // n'affichera rien

  echo $page;
  /* affichera "home"
  car la variable a été définie
  avant l'appel à include() */

?>
```

Les principaux intérêts du modèle dynamique :

• **Gestion modulaire de la structure HTML :** chaque portion qui sera éventuellement utilisée n'existe qu'une seule fois et peut s'adapter au contexte de la page. Ce qui pourra réduire considérablement le nombre de fichiers nécessaires sur le serveur et le volume de chacun.

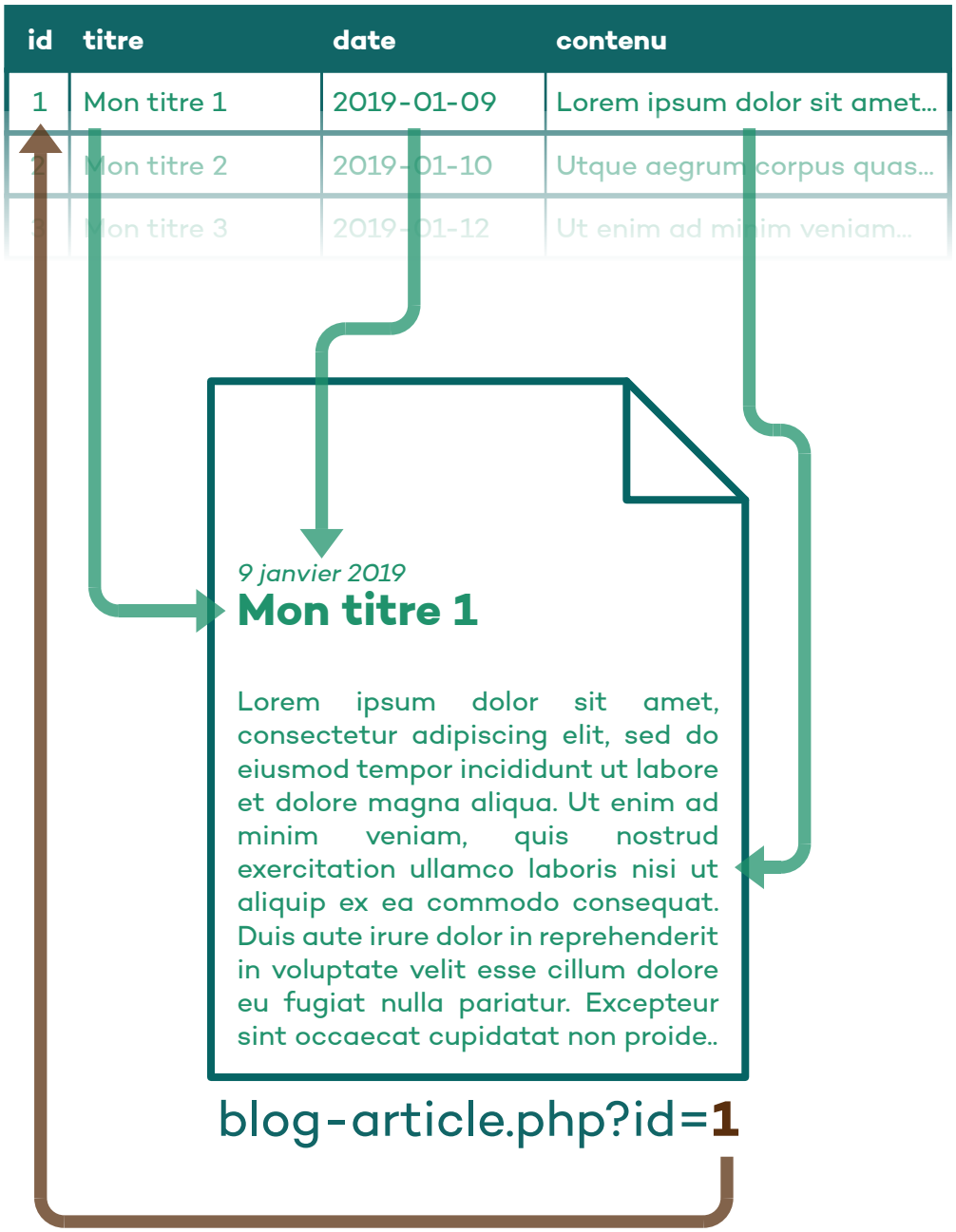
Éviter les redondances rend également la maintenance bien plus facile : les modifications structurelles faites sur un «module» (une section du HTML) s'appliqueront partout où cette section est appelée, sans avoir à être répétées sur les différentes pages concernées.

• **Utilisation des bases de données :** les contenus peuvent être modifiés sans intervenir sur le site lui-même et sans connaissances en HTML, à condition qu'une interface d'administration ait été mise en place.

Conjointement avec l'architecture HTML modulaire, cette manière de gérer les contenus permet aussi que la structure du site s'adapte intelligemment à eux, et rend possible des fonctionnalités de navigation plus pointues. Si par exemple tous les articles d'un blog sont listés et stockés dans une base de données, il sera possible d'utiliser cette même liste pour générer un menu ou une sidebar menant à une sélection d'articles, ou encore générer un lien menant toujours à l'article suivant.

Récupérer des informations dans une base de données

La plupart des serveurs web sont équipés d'un système de gestion de bases de données comme MySQL. Celui-ci permet de créer et gérer des tables, c'est à dire des listes de données à deux dimensions, idéales pour accueillir les contenus d'un site. Toujours dans l'exemple d'un blog, on peut imaginer une table pour les articles, dans laquelle chacun constituera une ligne, et chaque colonne sera destinée à contenir une information distincte concernant cet article (titre, date, corps du contenu, catégorie, etc).



PHP met à disposition diverses fonctions permettant d'interagir avec les bases de données mySQL, et ainsi y récupérer (ou encore modifier / ajouter) des informations.

On commencera toujours par établir une connexion au système mySQL, au moyen de la fonction mysqli_connect() et des identifiants nécessaires.

```
<?php

$connexion = mysqli_connect(
  "localhost", // hôte
  "root", // utilisateur
  "root", // mot de passe
  "ma_bdd" // nom de la base
);

?>
```

Il faut ensuite créer une requête SQL à envoyer à la base de données. La requête ci-dessous part du principe que la table concernée s'appelle blog_articles et qu'elle possède une colonne appelée id, contenant pour chaque ligne un identifiant numérique.

```
<?php

$req = "SELECT * FROM blog_articles
WHERE id = 1";

?>
```

«SELECT * » indique que l'on souhaite récupérer toutes les colonnes, «FROM blog_articles» indique dans quelle table chercher, et «WHERE id = 1» filtre les résultats pour n'obtenir que les lignes dont la colonne «id» contient la valeur 1.

Lorsque la connexion a été établie, un objet mysqli a été créé et stocké dans une variable $connexion. Ce type d'objet possède une méthode query() que l'on peut utiliser pour envoyer notre requête :

```
<?php

$res = $connexion->query($req);

?>
```

La variable $res contient maintenant les informations souhaitées. On peut les en extraire à l'aide de la fonction mysqli_fetch_array(), permettant de traiter chaque ligne extraite comme un tableau listant ses colonnes.

```
<?php

while (
  $col = mysqli_fetch_array($res)
) {

  $titre = $col["titre"];
  $cont = $col["contenu"];
  $date = $col["date"];

}

?>
```

Ici la boucle while fonctionne comme en JavaScript. Mais l'utilisation d'un simple égal indique bien une assignation et non une comparaison (nécessitant le double égal). Tant qu'il y a une nouvelle ligne de tableau à stocker dans $col, la boucle est répétée.

En PHP les éléments d'un tableau peuvent posséder une clé (un nom) en plus de leur indice (numérique) ; c'est le cas pour celui retourné par mysqli_fetch_array(). Dans la boucle on peut donc récupérer la valeur de chaque colonne de la ligne traitée, en utilisant comme clé le nom que celle-ci porte dans la table MySQL.

Une fois les différentes informations stockées dans des variables, celles-ci peuvent être insérées dans la page via l'instruction echo.

NOTE : Si plusieurs lignes sont traitées, l'affichage doit se faire dans la boucle while, car chacune de ses itérations remplace les valeurs stockées dans nos variables $titre, $cont et $date.

Dans notre exemple, on part du principe que la valeur de la colonne id est unique pour chaque ligne ; il n'y en aura donc qu'une seule contenant la valeur 1 et la boucle while ne s'exécutera qu'une fois.

Après avoir récupéré les informations, il est de bon usage de refermer la connexion (de l'objet mysqli créé au départ).

```
<?php

mysqli_close($connexion);

?>
```