M3104

PHP
Développement Web côté serveur

david.durand@u-picardie.fr

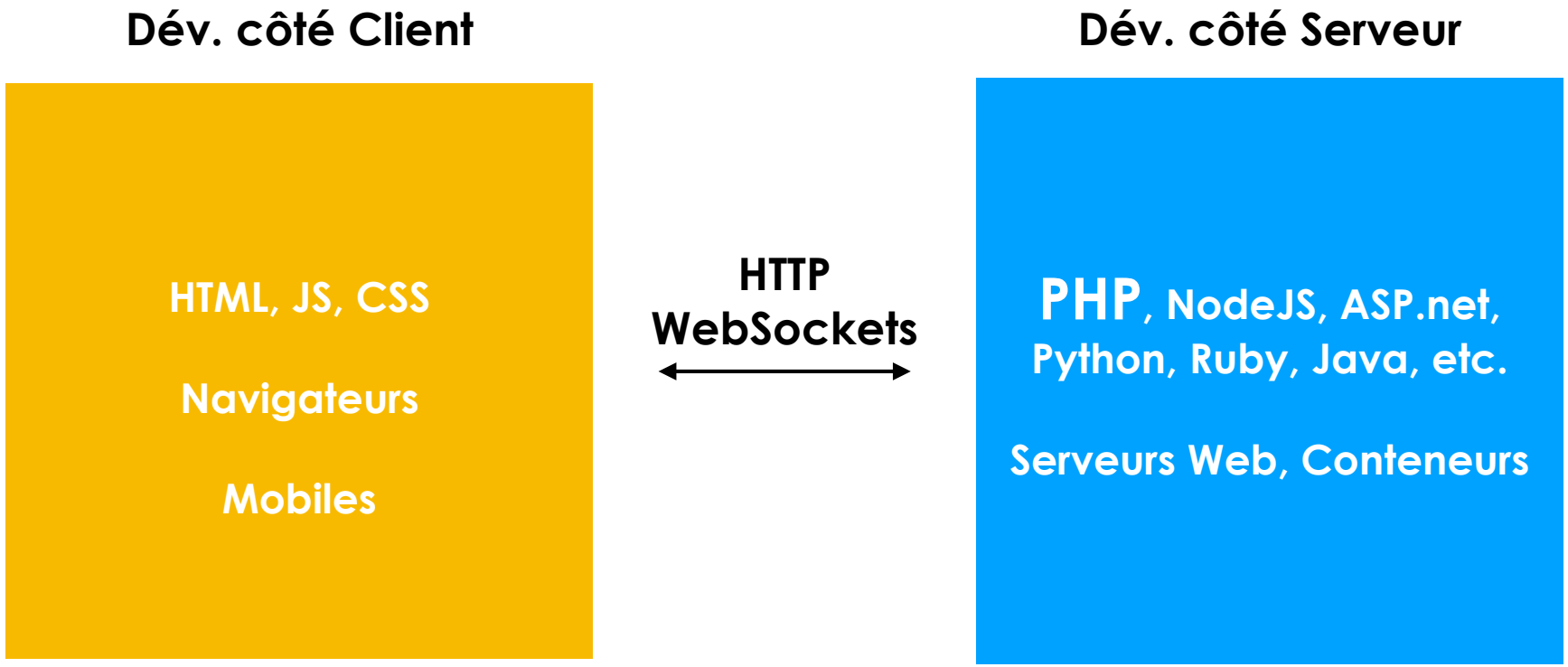
Notions abordées

- Web, HTML, environnement de développement
- PHP, bases du langage
- Accès aux BD MySQL
- Formulaires
- Sessions, Cookies & Cie
- Expressions Régulières

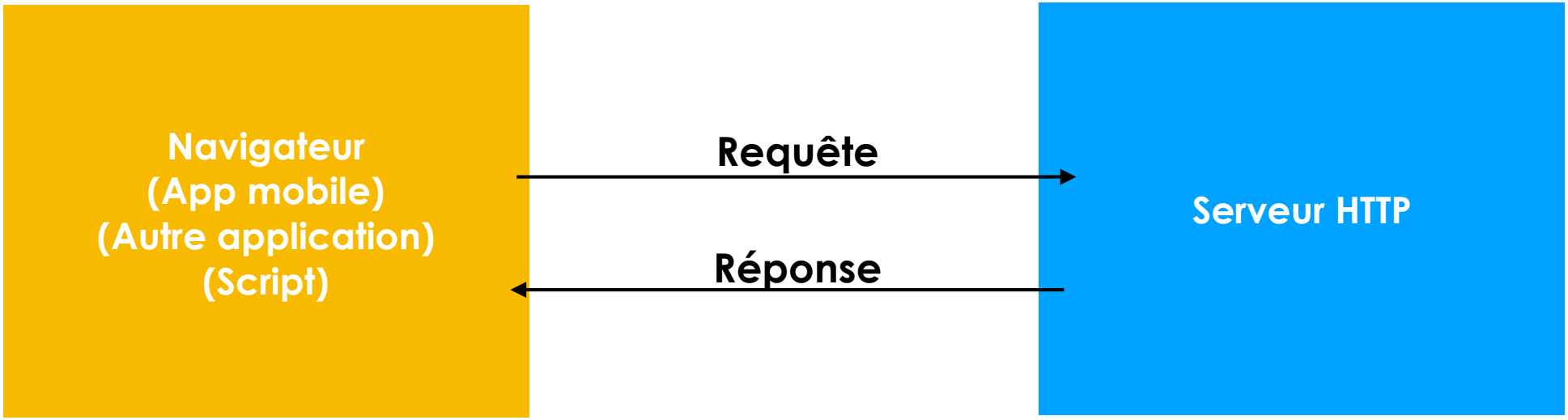
Côté client

- Javascript
- CSS
- AJAX

Web, HTML & Cie


Dév. côté Client
HTML, JS, CSS
Navigateurs
Mobiles
HTTP
WebSockets
Dév. côté Serveur
PHP, NodeJS, ASP.net,
Python, Ruby, Java, etc.
Serveurs Web, Conteneurs

Protocole HTTP



GET https://duckduckgo.com/?q=php&t=ffab&ia=web

Host: duckduckgo.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.13; rv:61.0) Gecko/20100101 Firefox/61.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate, br
Cookie: p=-2; 5=1
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/2.0 200 OK

server: nginx
date: Tue, 04 Sep 2018 08:31:59 GMT
content-type: text/html; charset=UTF-8
server-timing: total;dur=19;desc="Backend Total"
x-duckduckgo-results: 1
strict-transport-security: max-age=31536000
x-xss-protection: 1;mode=block
x-content-type-options: nosniff
referrer-policy: origin
expires: Tue, 04 Sep 2018 08:32:00 GMT
cache-control: max-age=1
x-duckduckgo-locale: fr_FR
content-encoding: br

Environnement de dév.

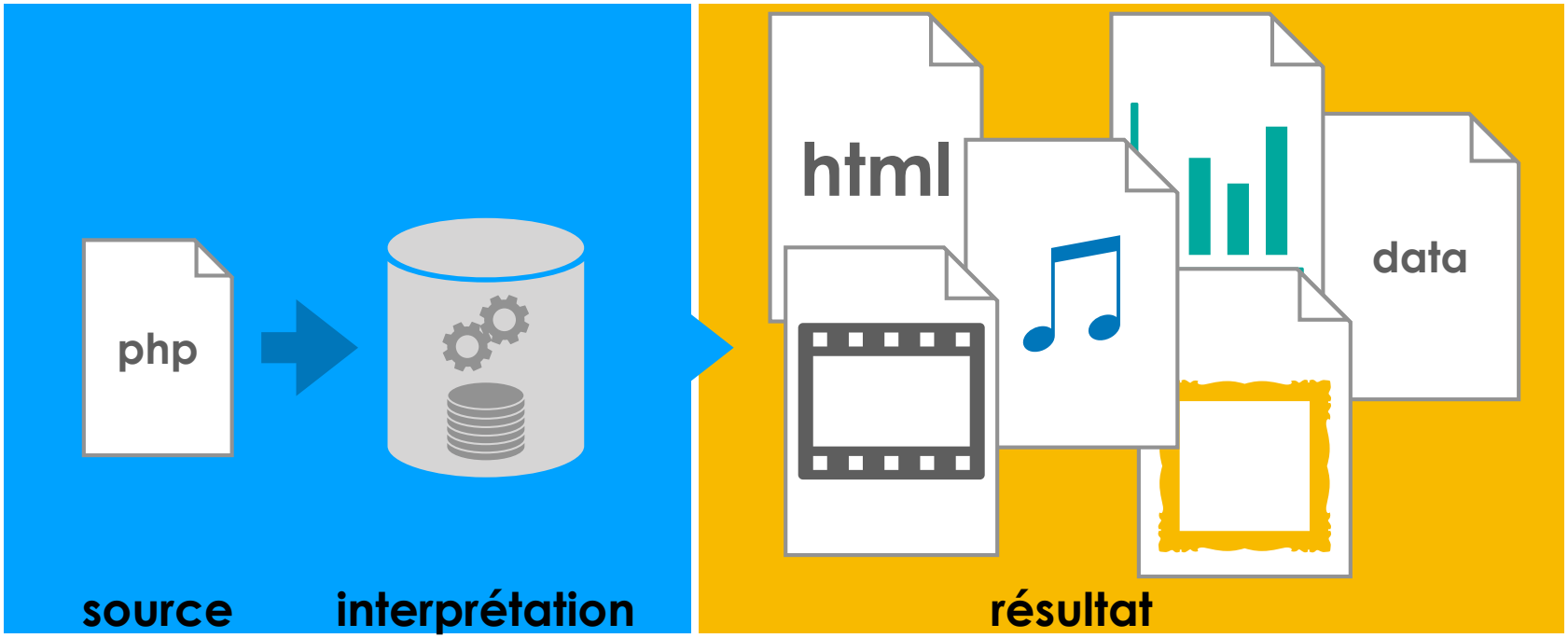
- Un serveur Web : Apache, nginx
- Un module pour interpréter le code : PHP
- Un serveur de Base de Données : Mysql, MariaDB, …

Environnement de dév.

- **PC** : WampServer, EasyPHP, uWAmp, Laragon
- **Linux** : LAMP, ou installation manuelle
- **Mac** : MAMP, ou installation manuelle
- **Editeur** : Atom, Visual Code Studio, Sublime, PHPStorm (PAS notepad ni word!)
- **Navigateurs** : FF, Opera, Safari, Chrome, Edge…
- **Outils** en ligne de commande : wget, cURL
- Extensions/apps : RESTED, POSTMAN

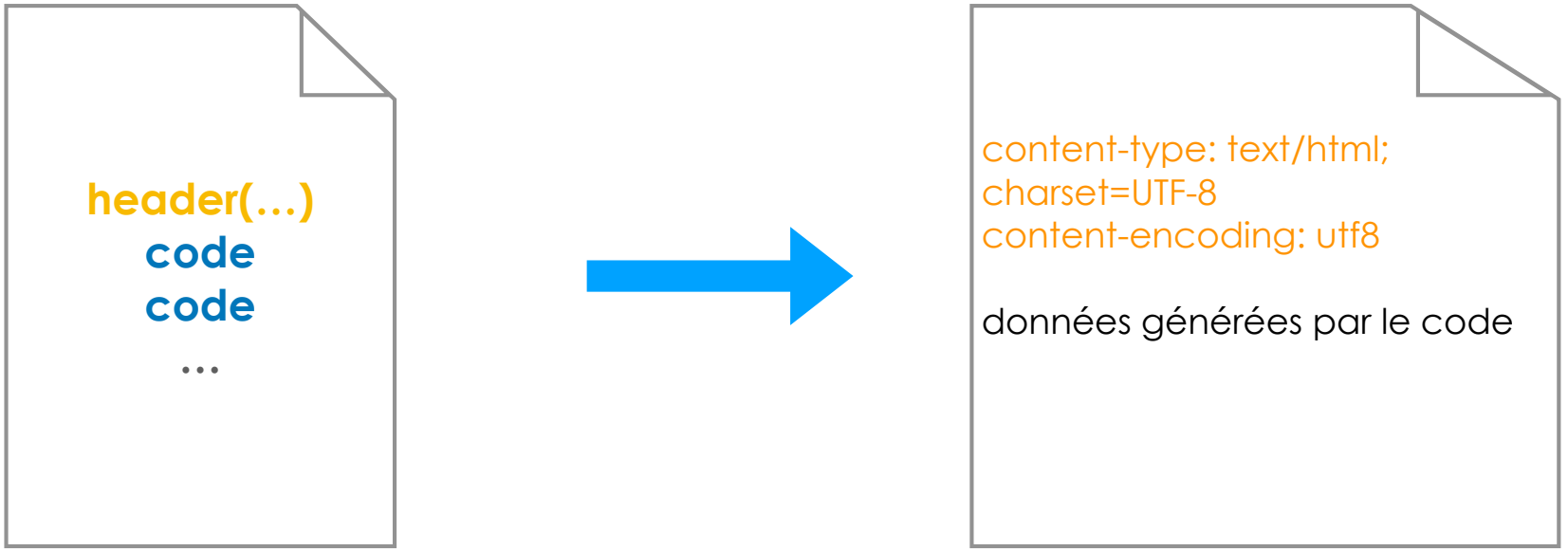
PHP

- Langage de programmation utilisé **côté serveur**
- Syntaxe inspirée de celle du langage C
- Génère *la plupart du temps* du contenu HTML



PHP

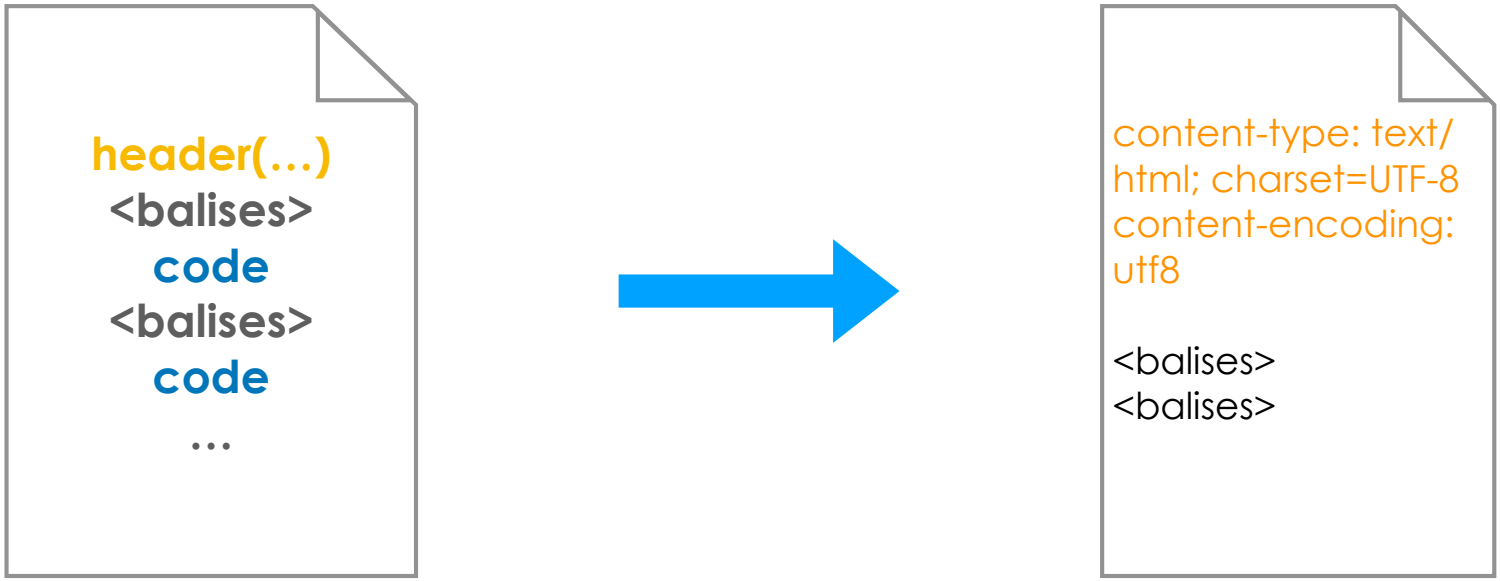
- **Mais aussi** peut simplement effectuer des traitements (scripts, bots), générer des images, du texte brut, de contenu structuré (json, xml, …)



PHP

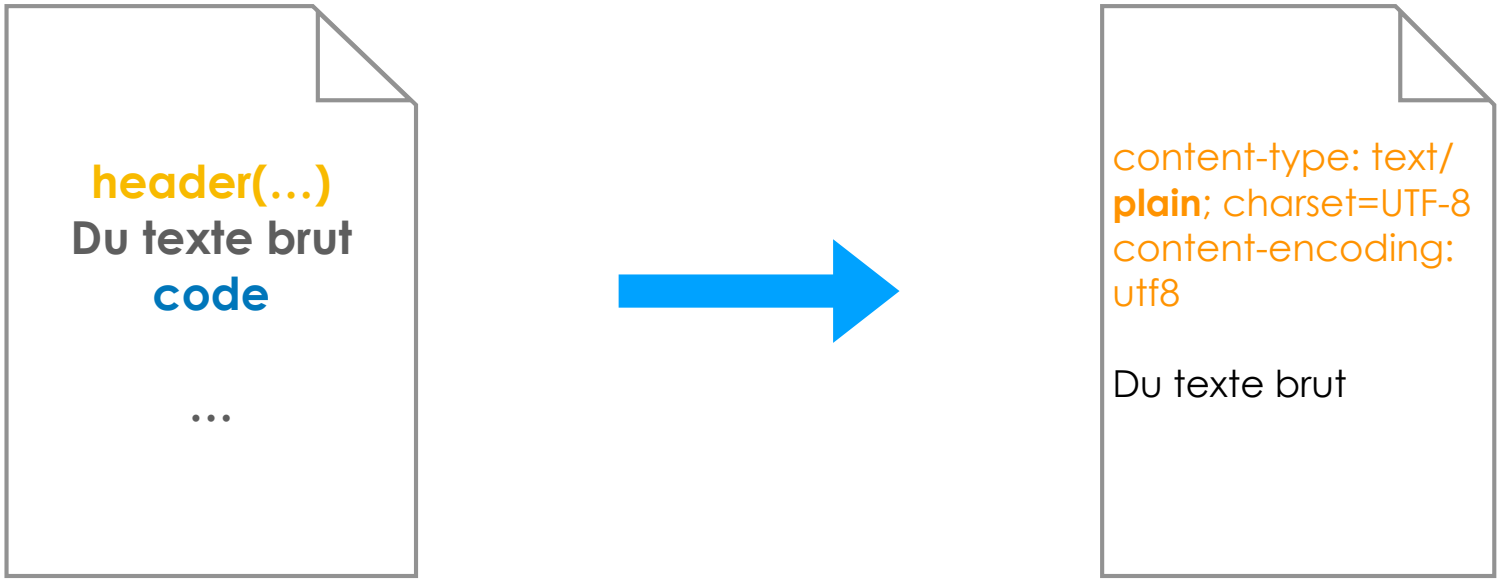
- Dans tous les cas, génère une **en-tête** et éventuellement un **contenu**

- Le code **n'apparait pas** dans le navigateur



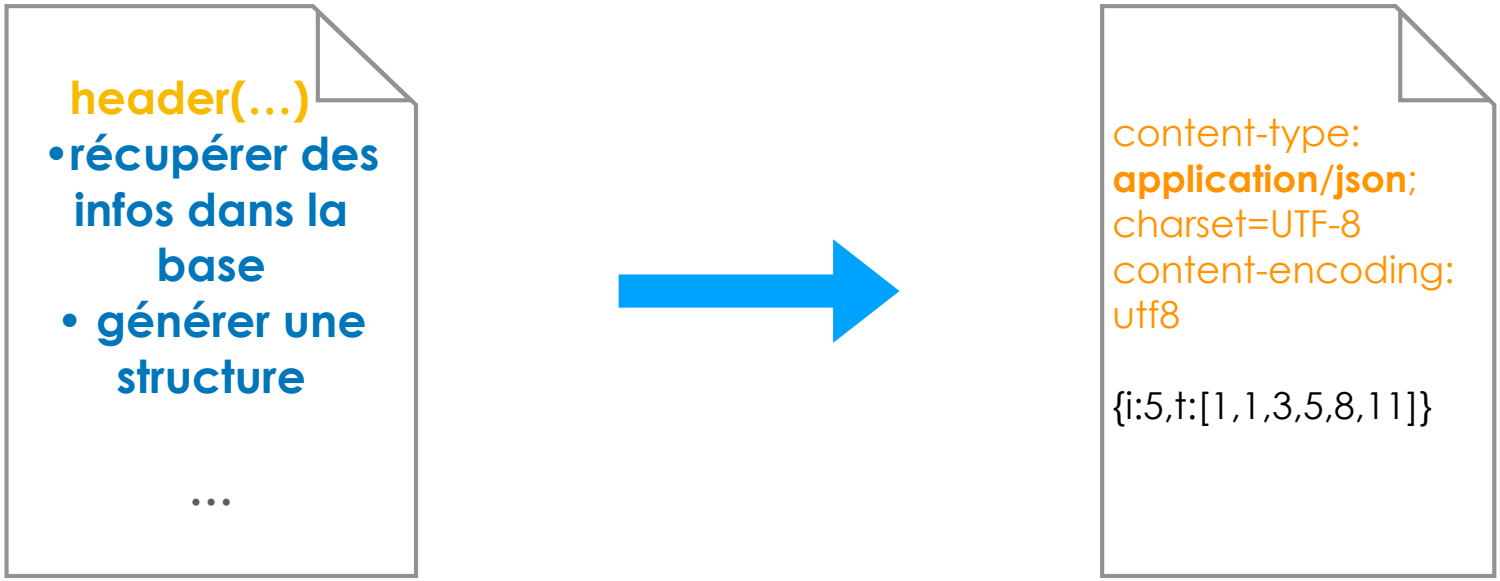
PHP

- exemple : texte brut



PHP

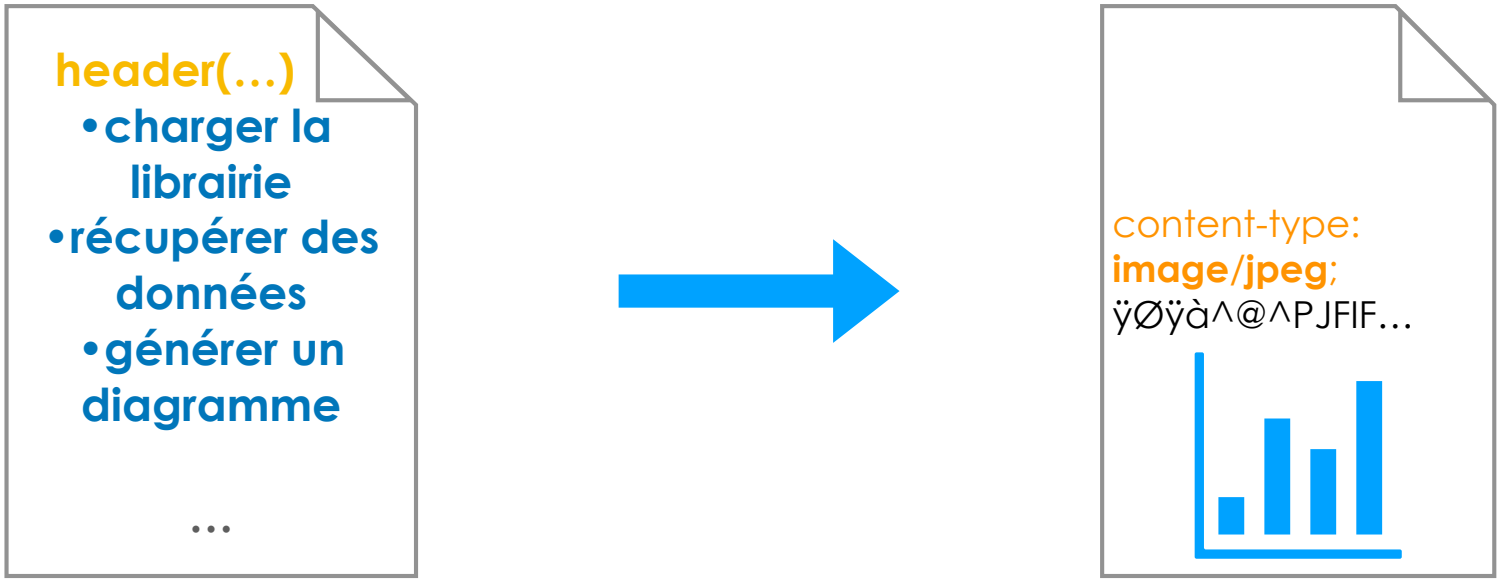
- exemple : données structurée en json



source **résultat**

PHP

- exemple : image



source **résultat**

Bases du langage

- Les scripts PHP sont des fichiers textes portant l'extension « .php »
- Le contenu peut-être
  - uniquement du code php (traitement, envoi d'image, connexion à un SGBD, …)
  - un mix de code HTML dans lequel on insère du code PHP à l'aide de marqueurs spécifiques.

Code

Source (poste de développement)

```
<html>
  <head>
    <title>Exemple</title>
  </head>
  <body>
  <?php
  echo "Une chaine de caractère générée en php";
  ?>
  </body>
</html>
```

Code

Résultat (côté navigateur)

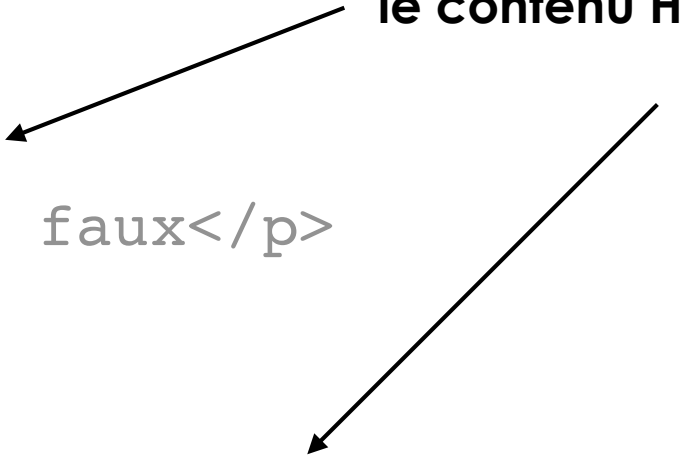
```
<html>
  <head>
    <title>Exemple</title>
  </head>
  <body>
  Une chaine de caractère générée en php
  </body>
</html>
```

Code

```
<html>
  <head>
    <title>Exemple</title>
  </head>
  <body>
  <?php
   if(true){
  ?>
    <p>C'est pas faux</p>
  <?php
   }else{
  ?>
    <p>C'est pas vrai</p>
  <?php } ?>
  </body>
</html>
```

Source (poste de développement)

le contenu HTML est généré en fonction du code PHP



Code

Résultat (côté navigateur)

```
<html>
  <head>
    <title>Exemple</title>
  </head>
  <body>
  <p>C'est pas faux</p>
  </body>
</html>
```

Code

- Bonne pratique : ne pas mélanger le code, mais utiliser un moteur de templates pour éviter de mélanger les langages.

```
<?php
//  chercher les données dans la base
//  charger et initialiser le gabarit
//  afficher le gabarit
?php>
```

Variables

- On déclare une variable à l'aide du caractère **$ (dollar)**
- A l'inverse du C, il n'y a pas besoin de préciser de type, l'interpréteur le détecte lorsque l'on affecte un contenu à la variable.
- On peut également déclarer des constantes à l'aide de la fonction **define**.

Variables

```
$bool = TRUE; // un booléen
$str = 'foo'; // une chaîne de caractères
$int = 12; // un entier
$decimal = 1.234 // un nombre décimal;
define ("PI", 3.14159265358979323846264);
```

Chaînes

- Une chaîne de caractères peut être déclarée avec des guillemets simples ou doubles.

- A l'intérieur des **guillemets simples**, les variables et caractères spéciaux (\t,\n,\,$...) ne sont **pas** interprétés.

- A l'intérieur des **guillemets doubles**, les variables et caractères spéciaux (\t,\n,\,$...) **sont** interprétés.

Chaînes

```
$nb=5;
echo "Il reste $nb articles dans votre panier";
//ou
echo "Il reste ${nb} articles dans votre panier";
//affiche Il reste 5 articles dans votre panier
echo 'Il reste $nb articles dans votre panier';
//affiche Il reste $nb articles dans votre panier
```

Chaînes

- L'opérateur de concaténation est le « . » (point)

```
echo "Le contenu de la variable est <b>" .  $variable
         . "</b>";

print ("Le contenu de la variable est <b>". $variable
        . "</b>");
```

Affichage/Sortie

- Tout affichage génère une sortie dans le flux envoyé au client (ex: le navigateur).

- Les commandes de base qui peuvent être utilisées sont : **echo**, **print**, **print_r**, **printf**

```
echo "Le contenu de la variable est <b> $variable </b>";
print ("Le contenu de la variable est <b> $variable </b>");
```

Tableaux

- le tableau est une liste d'éléments (quelconques)
- accès aux données :
  - en utilisant une position (**indice**)
  - en utilisant une **clé associée** à la case
  - ou les deux (l'indice devient une clé)

Tableaux indexés

- La position de chaque élément est représentée par un entier.
- Par défaut, le premier indice commence à 0.
- Exemple : déclaration d'un tableau de quatre éléments, en utilisant array( )

```
$fruits = array('Pomme','Banane','Pêche','Mangue');
// => l'index de Banane est 1

//création implicite :
$fruits[0] = 'Pomme';
$fruits[1] = 'Banane';
$fruits[2] = 'Pêche';
$fruits[3] = 'Mangue';
```

Tableaux indexés

- Autre syntaxe, sans indices. Ici, l'indice est déduit automatiquement en fonction de la taille du tableau.

- Chaque nouvel élément est inséré en fin de liste.

```
$fruits[] = 'Pomme';
$fruits[] = 'Banane';
$fruits[] = 'Pêche';
$fruits[] = 'Mangue';
```

Tableaux indexés

- Forcer le premier indice à une autre valeur que 0. On précise l'index (ici 1) avec la notation **=>**

```
$fruits = array(
        1=>'Pomme',
        'Banane',
        'Pêche',
        'Mangue'
        );
```

Tableaux associatifs

- Ces tableaux sont également appelés **tables de hachage**
- la position des éléments n'a pas d'importance : on utilise un ensemble de paires de clés/valeurs.
- Exemple : déclaration d'une table DNS, en utilisant array( ) et la notation clé => valeur

```
//deux paires de clés/valeurs
$dns = array('www.iut-amiens.fr'=>'194.57.109.129',
'localhost'=>'127.0.0.1');

//autre syntaxe
$dns['www.iut-amiens.fr'] = '194.57.109.129' ;
$dns['localhost'] = '127.0.0.1' ;
```

Accès aux valeurs

$tableau["case1"]

- $tableau["tab"][1]
- $tableau["lang"]["fr"]

$tableau["lang"]

```
$tableau = array(

  "case1" => 50,
  "tab" => array(
            1,
            1,
            2,
            3,
            5
            ),
  "lang" => array(
            "fr" => "Français",
            "en" => "Anglais",
            "de" => "Allemand"

  )
)
```

Parcours de tableaux

- La syntaxe **foreach** permet d'itérer chaque élément du tableau de manière simple, qu'il soit associatif ou indexé.

```
//Affiche chaque élément du tableau
foreach ($fruits as $element){
   echo $element;
}

//Affiche chaque élément du tableau, avec sa clé ou son indice
foreach ($fruits as $cle => $element){
   echo "$cle : $element";
}
```

Parcours de tableaux

- Pour afficher le contenu brut d'un tableau, utiliser la commande **print_r**

- Utile pour le debug, comprendre la structure d'un tableau

Fonctions des tableaux

- **count**($tab) : nombre d'éléments du tableau
- **asort**($tab) : trie les éléments du tableau
- **ksort**($tab) : trie le tableau selon les clés
- **array_flip**($tab) : transforme les clés en valeurs et les valeurs en clés
- **array_keys**($tab) : renvoie les clés
- **array_values**($tab) : renvoie les éléments
- **array_key_exists** ($cle, $tab) : vérifie si la clé est présente dans le tableau
- **in_array**($valeur,$tab) : vérifie si la valeur est présente dans le tableau
- etc… voir la doc php.net