Programmation web

aghiles.benkaoudjt.up13@gmail.com

Plan

1. Introduction au web
2. PHP
3. Javascript

aghiles.benkaoudjt.up13@gmail.com

1. Introduction au web

aghiles.benkaoudjt.up13@gmail.com

1. Introduction au web

1.0 Avant tout...

Que se passe-t-il quand je fais une recherche Google?

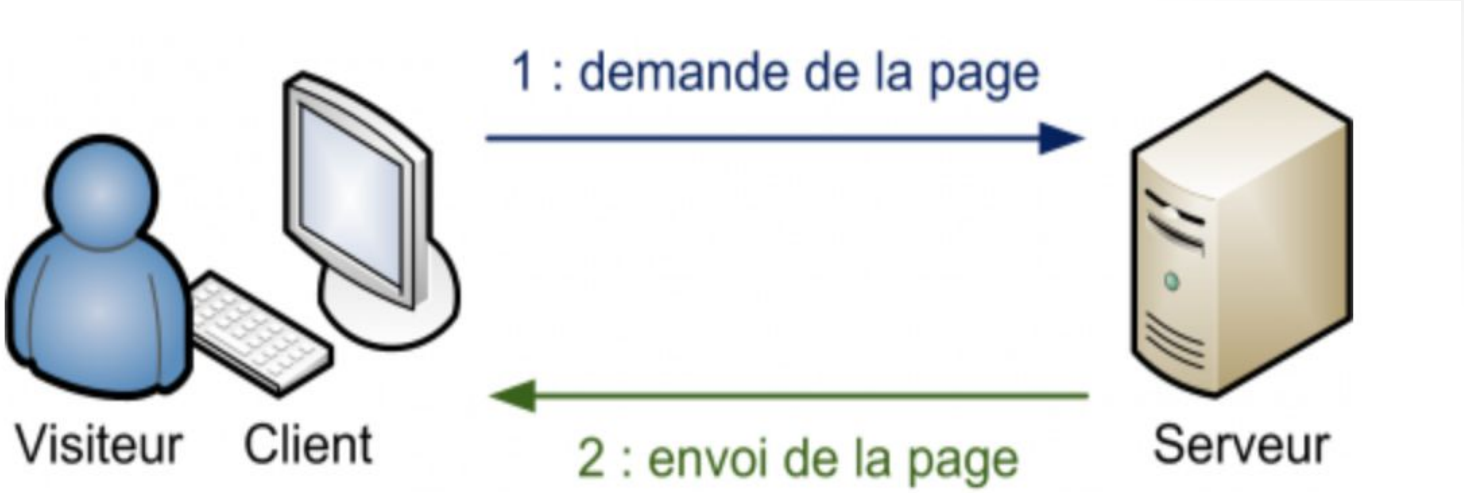
1. Introduction au web

1.1 Quelques définitions

- **client**: (ou frontend) appareils servant à la navigation (ordinateurs, téléphones, tablettes…)
- **serveur**: (ou backend) ordinateurs puissants qui stockent et délivrent des sites web / de la donnée aux internautes (ie aux clients)
- **sites statiques**: sites composés de fichiers HTML que les navigateurs web demandent à des serveurs
- **sites dynamiques**: la page web est générée à chaque fois que le client la demande

1. Introduction au web

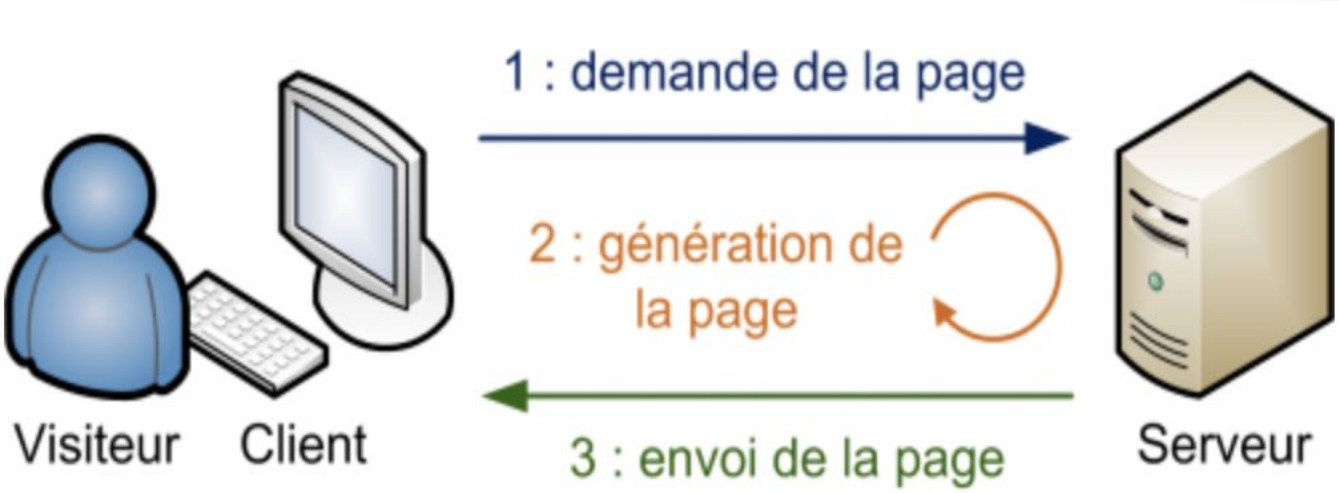
1.2 Sites statiques



aghiles.benkaoudjt.up13@gmail.com

1. Introduction au web

1.3 Sites dynamiques



1. Introduction au web

1.4 HTTP

- Hypertext Transfer Protocol
- protocole de **communication client / serveur**
- Protocole de la couche d'application (modèle OSI)
- **HTTPS**: variante HTTP sécurisée (utilisation des protocoles SSL ou TLS)

Les **navigateurs web** sont un exemple de **clients HTTP** (ie: ce sont de logiciels qui utilisent le protocole HTTP)

Autres exemples de protocoles de communication: SSH, FTP, etc...

1. Introduction au web

1.5 HTTP (status codes)

Success (2xx)

200	OK
201	Created
202	Accepted
204	No content
206	Partial content

Redirection (3xx)

302	Moved temporarily

Client issue (4xx)

400	Bad request
401	Unauthorized
403	Forbidden
404	Not found
405	Method not allowed
406	Not acceptable

Server issue (5xx)

500	Internal server error
501	Not implemented
502	Bad gateway
503	Service unavailable
504	Gateway timout

aghiles.benkaoudjt.up13@gmail.com

1. Introduction au web

1.6.1 Méthodes HTTP (ou verbes HTTP)

- **GET** : méthode la plus utilisée pour **récupérer une ressource**. elle n'est pas censée modifier la ressource (ie si on fait n fois le même appel, on aura n fois le même résultat)
- **POST** : utilisée pour **envoyer une entité** vers la ressource indiquée. Cela entraîne généralement un **changement d'état** ou des effets de bord sur le serveur. Ces requêtes possèdent un **corps (body)** contenant les données à envoyer
- **PUT** : **remplace** la représentation actuelle de la **ressource visée** par le contenu de la requête. Cette requête possède aussi un **corps (body)**

1. Introduction au web

1.6.2 Méthodes HTTP (ou verbes HTTP)

- **PATCH**: appliquer des **modifications partielles** à une ressource
- **DELETE**: **supprime** la ressource indiquée
- **HEAD** : demande les **en-têtes (headers)** qui seraient retournés si la ressource spécifiée était demandée avec une méthode HTTP GET
- **CONNECT**: établit un **tunnel** vers le serveur identifié par la ressource cible
- **OPTIONS**: permet d'obtenir les **options de communication** d'une ressource ou du serveur en général
- **TRACE**: réalise un message de **test aller/retour** en suivant le chemin de la ressource visée

2. PHP

aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.1 Introduction

- PHP Hypertext Processor
- Langage **transpilé** (!= compilé)
- Langage **typé dynamiquement** (ajout typage statique php 7+)
- Plutôt conçu pour le développement WEB
- S'exécute côté serveur

2. PHP

2.2.1 Installation

- Windows: wamp
  (à essayer aussi: uWamp, peut être plus simple d'utilisation)

- Linux: xampp

- Mac: mamp

NB: xampp peut être utilisé sur les 3 OS

2. PHP

2.2.2 Installation (Erreurs)

Il se peut que les erreurs ne soient pas visibles => changer config

- Ouvrir le fichier php.ini (pour voir où il se trouve `<?php phpinfo() ?>`)
- `error_reporting = E_ALL`
- `display_errors = On`
- Enregistrer le fichier
- relancer le serveur

2. PHP

2.3 Les variables

- Une variable est une petite information qui reste stockée en mémoire le temps de la génération de la page PHP. Elle a un nom et une valeur.
- Il existe plusieurs types: string, int, float, bool, etc...
- Exemples:

```
$userName = 'my name';
$user_name = 'my other name';
$user_name_2 = 'last one';
```


aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.4 Concaténation de string

On peut écrire des strings de différentes manières (en savoir plus):

```php
$name = 'martin';
echo 'Je suis $name';
// => affichage: Je suis $name

echo "Je suis $name";
// => affichage: Je suis martin

echo <<<HERODOC_ID
Je suis $name
HERODOC_ID;
// => affichage: Je suis martin
```

2. PHP

2.5.1 Les tableaux (ou *array*) numérotés

- permettent de stocker plusieurs valeurs dans une seule variable
- /!\ Les indices commencent à 0

```php
$names = array('Jean', 'Martin', 'Antoine');
$names2 = ['Jean', 'Martin', 'Antoine']; // php 7+
$names3 = []; // ou array()
$names3[] = 'Jean';
$names3[] = 'Martin';
$names3[] = 'Antoine';

echo $names3[2]; // affichage: Antoine
```

2. PHP

2.5.2 Les tableaux (ou *array*) associatifs

- même principe mais on donne un nom aux cases (équivalent d'un hashmap en Java)

```php
$user1 = array(
  'name' => 'Jean',
  'age' => 21
);
$user2 = [
  'name' => 'Jean',
  'age' => 21
];
$user2['sexe'] = 'M';

echo $user2['age']; // affichage: 21
```

2. PHP

2.6.1 Les conditions

```php
if ($age >= 18){
  echo 'tu es majeur';
} elseif ($age > 0 && $age < 18){
  echo 'tu es mineur';
} else{
  echo 'ah bon?';
}
```

2. PHP

2.6.2 Les conditions (syntaxe alternative)

```php
if ($age >= 18):
  echo 'tu es majeur';
elseif ($age > 0 && $age < 18):
  echo 'tu es mineur';
else:
  echo 'ah bon?';
endif;
```

=> Cette syntaxe peut être très utile en utilisation avec l'HTML

aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.6.3 Les conditions (== vs ===)

```php
$age = 21;
if ($age == '21') {
  echo 'on rentre ici';
}
if ($age === '21') {
  echo 'on ne rentre pas ici';
}
if ($age === 21) {
  echo 'on rentre ici';
}
```

2. PHP

2.7.1 Les boucles (while)

- Permet d'exécuter des instructions tant que la condition est vraie

```php
while ($condition === true)
{
    // instructions à exécuter dans la boucle
}
```

/!\ Il faut **TOUJOURS** s'assurer que la condition sera fausse au moins une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini

2. PHP

2.7.2 Les boucles (for)

- Permet d'exécuter des instructions un certain nombre de fois

```php
$names = ['Jean', 'Martin', 'Antoine'];
for ($i = 1; $i <= count($names); $i++)
{
  echo $names[$i]. ' ';
}
// affichage: Jean Martin Antoine
```

/!\ Il faut **TOUJOURS** s'assurer que la condition sera fausse au moins une fois. Si elle ne l'est jamais, alors la boucle s'exécutera à l'infini

2. PHP

2.7.3 Les boucles (foreach 1)

- Permet d'itérer facilement sur un tableau

```php
$names = ['Jean', 'Martin', 'Antoine'];
foreach ($names as $name)
{
  echo $name . ' ';
}
// affichage: Jean Martin Antoine
```


aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.7.4 Les boucles (foreach 2)

- On peut aussi accéder au nom de la clé en plus de la valeur

```php
$user = array(
  'name' => 'Jean',
  'age' => 21
);
foreach ($user as $key => $value){
  echo "$key: $value \n";
}
// affichage:
// name: Jean
// age: 21
```


aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.8 Les fonctions

- série d'instructions qui effectue des actions et qui retourne une valeur
- PHP propose des centaines et des centaines de fonctions prêtes à l'emploi pour tous types de tâches : envoyer un e-mail, récupérer l'heure, crypter des mots de passe, etc…
- Si une fonction existe déjà dans le langage, essayer de l'utiliser plutôt que de la réécrire
- Contrairement aux variables, les fonctions peuvent être écrites n'importe où dans le fichier (hoisting)

```php
function getSquare($number) {
  return $number * $number;
}
echo getSquare(3); // affichage: 9
```

TP 1

aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.8 Les inclusions

- Il est possible de faire appel (d'inclure) à d'autres fichiers dans un fichier

```php
require 'path_to/menu.php';
require_once 'path_to/menu.php';
include 'path_to/menu.php';
include_once 'path_to/menu.php';
```

NB: require est identique à include mis à part le fait que lorsqu'une erreur survient, il produit également une **erreur fatale**

2. PHP

2.9 HTML dans le php

- Il est possible d'écrire de l'html directement dans un fichier PHP

```
<?php
$page = 'Accueil';
?>
<h2>Je suis ici: <?php echo $page ?></h2>
<h3>Je suis ici: <?= $page ?></h3>
<!-- Résultat -->
<h2>Je suis ici: Accueil</h2>
<h3>Je suis ici: Accueil</h3>
```

- Tout ce qui est entre `<?php ?>` sera considéré comme du php, tout le reste comme du HTML

2. PHP

2.10 Les variables superglobales

- Ce sont des variables globales (on peut y accéder partout) automatiques
- Nous en verrons quelques unes dans la suite. pour l'instant, les voici

```
$superGlobales = [
  $GLOBALS,
  $_SERVER,
  $_GET,
  $_POST,
  $_FILES,
  $_COOKIE,
  $_SESSION,
  $_REQUEST,
  $_ENV
];
```

2. PHP

2.11 Transmettre des données avec l'url

- Une URL représente l'adresse d'une page web (http:// ou https://)
- Lorsqu'on fait un lien vers une page, il est possible d'ajouter des paramètres sous la forme bonjour.php?nom=Dupont&prenom=Jean qui seront transmis à la page.
- En accédant à la page, **$_GET** vaudra:

```
['nom' => 'Dupont', 'prenom' => 'Jean'];
```

- /!\ Il ne faut pas faire aveuglément confiance à ces informations, et TOUJOURS tester prudemment leur valeur avant de les utiliser

2. PHP

2.12.1 Les formulaires

- Moyen principal pour créer de l'interactivité avec l'utilisateur
- HTML => crée le formulaire | PHP => traite les informations du formulaire

```
<form action="cible.php" method="post">
  <input type="text" name="prenom" value="Jean"/>
</form>
```

- **action** : page appelée par le formulaire | **method** : méthode utilisée pour l'envoi (get | post)
- dans cet exemple, dans cible.php, `$_POST['prenom'] === 'Jean'`
- si on change la **method** en **get**, alors on aura `$_GET['prenom'] === 'Jean'`
- Éviter la faille **XSS** en échappant le HTML: appliquer la fonction **htmlspecialchars** sur tous les textes envoyés par les visiteurs qui seront affichés
- /!\ Il ne faut pas faire aveuglément confiance à ces informations, et **TOUJOURS** tester prudemment leur valeur avant de les utiliser

2. PHP

2.12.2 Les formulaires (fichiers)

- Il faut ajouter `enctype="multipart/form-data"` à la balise **<form>**
- cas particulier: le fichier ne se trouvera pas dans **$_GET** ou **$_POST** mais dans **$_FILES**

```
<form action="cible_envoi.php" method="post" enctype="multipart/form-data">
    Formulaire d'envoi de fichier :<br />
    <input type="file" name="monfichier" /><br />
    <input type="submit" value="Envoyer le fichier" />
</form>
```

- on aura $_FILES[“monfichier”] qui contiendra le fichier téléchargé

2. PHP

2.13 La session ($_SESSION)

- **$_SESSION** permet de stocker des informations qui seront automatiquement transmises de page en page
- Il faut toujours démarrer la session (avant tout code HTML et avant tout code PHP l'utilisant)

```php
session_start(); // démarre la session
$_SESSION['isConnected'] = true;
$_SESSION['user'] = ['name' => 'Jean', 'age' => 21]; // On peut aussi stocker des données complexes
session_destroy(); // détruit la session de l'utilisateur
```

2. PHP

2.14 Les cookies ($_COOKIE)

- Un cookie, c'est un petit fichier (contenant du texte) que l'on enregistre sur l'ordinateur du visiteur
- **$_COOKIE** permet de stocker des informations qui seront automatiquement transmises de page en page
- setcookie doit toujours être appelé avant tout code HTML

```php
$cookieName = 'pseudo';
$cookieValue = 'up13';
$expirationDate = time() + 365 * 24 * 3600; // exprimé en timestamp
$httpOnly = true;
setcookie($cookieName, $cookieValue, $expirationDate, null, null, false, $httpOnly);
// On a maintenant: $_COOKIE['pseudo'] === 'up13'
```

2. PHP

2.15 Les fichiers

- Il existe énormément de fonctions que l'on peut utiliser sur le système de fichiers(voir ici)
- On en retiendra principalement 2 :

```
file_get_contents($pathToFile);
file_put_contents($pathToFile, $data);
```


aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.16.1 Les classes

```php
class User {
  public $name;
  private $age;

  public function __construct($name, $age) {
    $this.$name = $name;
    $this.$age = $age;
  }

  public function sayHello() {
    echo "bonjour, je suis $this->name et j'ai $this->age ans";
  }
}

$user = new User('Jean', 21);
$user->sayHello(); // affichage: bonjour, je suis Jean et j'ai 21 ans
```

- pour en savoir plus, c'est ici

2. PHP

2.16.2 Les classes (statiques)

```php
class Form {
  private static $DEFAULT_CLASS = 'form-field';

  public static function input($name, $classes = []){
    $class = implode(' ', $classes) . ' ' . self::$DEFAULT_CLASS;
    return <<<HTML
      <input name="$name" class="$class"/>
HTML;
  }
}

Form::input('firstName', ['is-small', 'bg-gray']);
```

2. PHP

2.17 Les interfaces

```php
interface UserRepository
{
  public function checkUserExistence(string $username, string $password): bool;

  public function getUserByUsername(string $username);

  public function createUser($firstName, $lastName, $username, $password): void;
}
```

- Toujours préférer dépendre d'une interface plutôt que d'une classe (implémentation) (principe SOLID)

TP 2

aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.18.1 BDD - MySQL (connexion)

- PDO: permet d'accéder à n'importe quelle BDD (anciennement mysql_ ou mysqli_)
- Avant tout, activer `php_pdo_mysql` dans la configuration

```php
$bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8',
  'root',
  '',
  [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
);
// localhost: le nom d'hôte
// test: le nom de la base de donnée
// root: le login
// root: le mot de passe (/!\ sous WAMP password = '')
// Le dernier paramètre permet de mieux afficher les erreurs
```

2. PHP

2.18.2 BDD - MySQL (lecture)

```php
$reponse = $bdd->query('SELECT nom FROM user WHERE id=\'' . $_GET['userId'] . '\'');

$donnee = $reponse->fetch(); // pour récupérer la première ligne
$donnees = $reponse->fetchAll(); // pour récupérer toute la réponse
```

- $_GET['userId'] => /!\ faille XSS: on s'expose ici à une injection SQL

  => solution: les **requêtes préparées**

aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.18.3 BDD - MySQL (lecture - requêtes préparées)

```php
$req = $bdd->prepare('SELECT nom FROM user WHERE id = ?');
$req->execute([$_GET['userId']]);

// avec des marqueurs nominatifs
$req2 = $bdd->prepare('SELECT nom FROM user WHERE id = :id');
$req2->execute(['id' => $_GET['userId']]);
```

- On peut maintenant spécifier qu'on attend une variable
- Ensuite, on peut exécuter la requête en donnant les paramètres à la requête préparée

2. PHP

2.18.4 BDD - MySQL (écriture)

- On écrira en BDD grâce à **INSERT**, **UPDATE** ou **DELETE**
- Les requêtes préparées peuvent aussi servir pour l'écriture en BDD
  Donc de la même façon, on pourrait avoir:

```php
$req = $bdd->prepare('INSERT INTO user (nom) VALUES (:nom1), (:nom2)');
$req->execute([
  'nom1' => 'Jean',
  'nom2' => 'Nicolas'
]);
```

2. PHP

2.18.5 BDD - MySQL (un peu d'abstraction)

```php
class DatabaseClient { // To use in repositories
  private static $instance = null;
  private $database;

  private function __construct()
  {
    $this->database = new PDO("mysql:host=localhost;dbname=myDB;charset=utf8",
      'root',
      'root',
      [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION]
    );
  }

  public static function getDatabase() {
    if(is_null(self::$instance)) {
      self::$instance = new DatabaseClient();
    }
    return self::$instance->database;
  }
}
// /!\ Mauvaise pratique(difficile à tester) -> préférer déléguer l'unicité au code
```

2. PHP

2.18.6 BDD - MySQL (un peu d'abstraction bis - ORM)

```php
// Exemple avec Doctrine + Symfony
public function show($id)
{
    $product = $this->getDoctrine()
        ->getRepository(Product::class)
        ->find($id);

    if (!$product) {
        throw $this->createNotFoundException(
            'No product found for id '.$id
        );
    }

    return new Response('Check out this great product: '.$product->getName());
}
```


aghiles.benkaoudjt.up13@gmail.com

TP 3

aghiles.benkaoudjt.up13@gmail.com

2. PHP

2.19 pour aller un peu plus loin

1. Les Exceptions plutôt que les `if (is_null(...))`
2. Les espaces de nom (namespace)
3. L'autoloader (à combiner avec les namespaces)
4. Écrire un Router pour des Urls plus propres
5. cURL pour appeler des APIs
6. Les **Tests Automatisés (!!!!!!!!!!** exemple: Php unit)

Les points 1, 2 et 3 pourraient être mis en place facilement pour le projet
Le point 4, serait un joli bonus
Le point 6 serait un très gros bonus (pasa difficile à mettre en place)

3. Javascript

aghiles.benkaoudjt.up13@gmail.com

3. Javascript

3.1 Introduction

- Langage **transpilé** (!= compilé)
- Langage **typé dynamiquement**
- S'exécute sur le navigateur du client
- S'exécute aussi côté serveur (nodejs)
- Est à la base des frameworks component (React, Vuejs, Angular…)
- Sert à manipuler les éléments du DOM
- Sert à réagir aux événements de la page
- Ajax
- S'insère plutôt à la fin du body grâce à `<script>`

3. Javascript

3.2 Les variables

```
var age = 18; // number
var lastName = 'Dupont'; // string
var firstNames = ["Jean", "Martin"]; // array
var isMarried = false; // boolean
var wifeName; // undefined
wifeName = null; // undefined
var user = {
  age: age,
  nom: lastName,
  prenoms: firstNames
}; // object
```


aghiles.benkaoudjt.up13@gmail.com

3. Javascript

3.3 La concaténation

- /!\ Contrairement à PHP, la concaténation se fait avec <**+**>, et non pas avec <**.**>

```
var lastName = 'Dupont';
var firstName = 'Jean';
var fullName = lastName + ' ' + firstName;
```


aghiles.benkaoudjt.up13@gmail.com

3. Javascript

3.4 Les conditions / les boucles

- ...

aghiles.benkaoudjt.up13@gmail.com

3. Javascript

3.5 Les fonctions

```
function sayHello(name) {
  console.log('Hello ' + name);
}

var sayHelloBack = function() {
  console.log('Hello to you too!!');
}

sayHello('Jean');
sayHelloBack();
```

3. Javascript

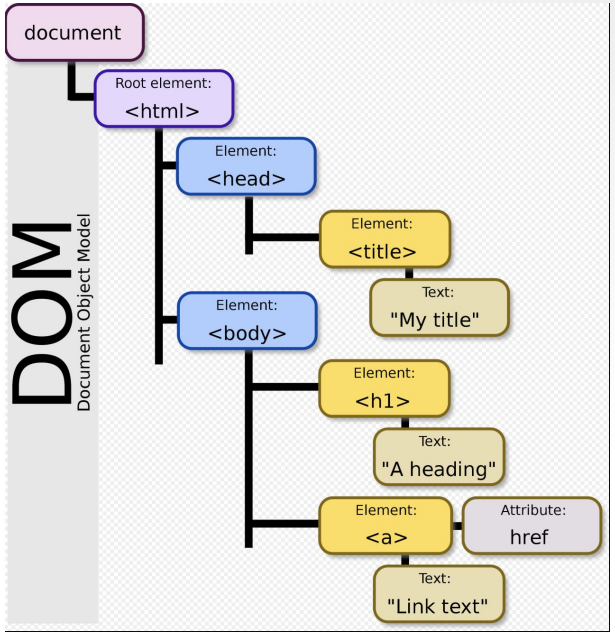
3.6 Le hoisting

```
name = 'Jean';
console.log(name); // affiche Jean
var name; // la déclaration est remontée tout en haut de a fonction

sayHello('Martin'); // Affiche Hello Martin

function sayHello(name) { // La déclaration est remontée tout en haut aussi
  console.log('Hello ' + name);
}
```

3. Javascript

3.7 Le DOM (Document Object Model)

```
<!DOCTYPE html>
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="">Link text</a>
  </body>
</html>
```



aghiles.benkaoudjt.up13@gmail.com

3. Javascript

3.8 Ajax

- Asynchronous Javascript And XML
- Permet de faire des appels au serveur de façon dynamique

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
     document.getElementById("demo").innerHTML = this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```


aghiles.benkaoudjt.up13@gmail.com

3. Javascript

3.9.1 jQuery

- Bibliothèque Javascript
- Simplifie l’écriture permettant l’accès au DOM
- Simplifie l’écriture pour la gestion des événements
- Simplifie les appels Ajax
- Grosse communauté

aghiles.benkaoudjt.up13@gmail.com

3. Javascript

3.9.2 jQuery (Ajax)

```
$.get( "ajax_info.txt", function( data ) {
  $("#demo").html(data);
});
```


aghiles.benkaoudjt.up13@gmail.com

TP 4

aghiles.benkaoudjt.up13@gmail.com