MobApp – HTML, CSS, Javascript

Laurent Daverio, Olivier Hermant

Centre de recherche en informatique
MINES Paris, Université PSL

Plan

1. XML
2. HTML
3. CSS
4. JS

1. XML

XML - eXtensible Markup Language

Classification des espèces :

```
<règne>
    <animal>
        ...
        <mammifère>
            ...
            <homoSapiens bras="2" jambes="2" ailes="0">
                <formation>
                    <ingénieur ecole="mines">
                        ...
                    </ingénieur>
                    <faux-ingenieur ecole="X" />
                    <commercial />
                </formation>
                ...
            </homoSapiens>
        </mammifère>
    </animal>
    <végétal>
        ...
```

XML – composition chimique

- forme arborescente
- tag/balise : `<animal>`
- balises ouvrantes (`<animal>`) **doivent** se fermer (`</animal>`)
  - cas particulier : balises autofermantes “stériles” (`<hec />`)
- les balises peuvent avoir
  - des attributs :
    `<homoSapiens bras="2" jambes="2" ailes="0">`
  - des descendants (si non autofermantes)

Organiser les informations.

2. HTML

HTML : HyperText Markup Language

- inventé en Europe (CERN)
- l'ancêtre de XML : mêmes principes
- **pages web** : avec des liens hypertexte
- DÉMO: l'inspecteur de code de Firefox

Une page HTML assez minimale

```
<!doctype html>
<html lang="fr">
<head>
    <meta charset="utf-8" />
    <title>Page Vide</title>
</head>
<body>
    <!-- Ceci est un commentaire dans une page vide -->
</body>
</html>
```

- séparation entre en-tête <head></head> (non affichée) et corps <body></body>
- [page_vide.html]
- que l'on peut ouvrir dans un navigateur

Quelques balises

- organisation du contenu en *divisions*

```
<div>contenu</div>
```

- et en *paragraphes*

```
<p></p>
```

- *passer à la ligne* avec

```
<br />
```

- les *liens*

```
<a href="URL">contenu</a>
```

- [page_div.html]

D'autres balises

- liste non ordonnée (`<ul>...</ul>`) ou ordonnée (`<ol>...</ol>`)
  - chaque item de la liste (`<li>...</li>`)
- un tableau à 5 lignes et 2 colonnes:

```
<table>
<tr> <td> (1,1) </td> <td> (1,2) </td> </tr>
<tr> <td> (2,1) </td> <td> (2,2) </td> </tr>
<tr> <td> (3,1) </td> <td> (3,2) </td> </tr>
<tr> <td> (4,1) </td> <td> (4,2) </td> </tr>
<tr> <td> (5,1) </td> <td> (5,2) </td> </tr>
</table>
```

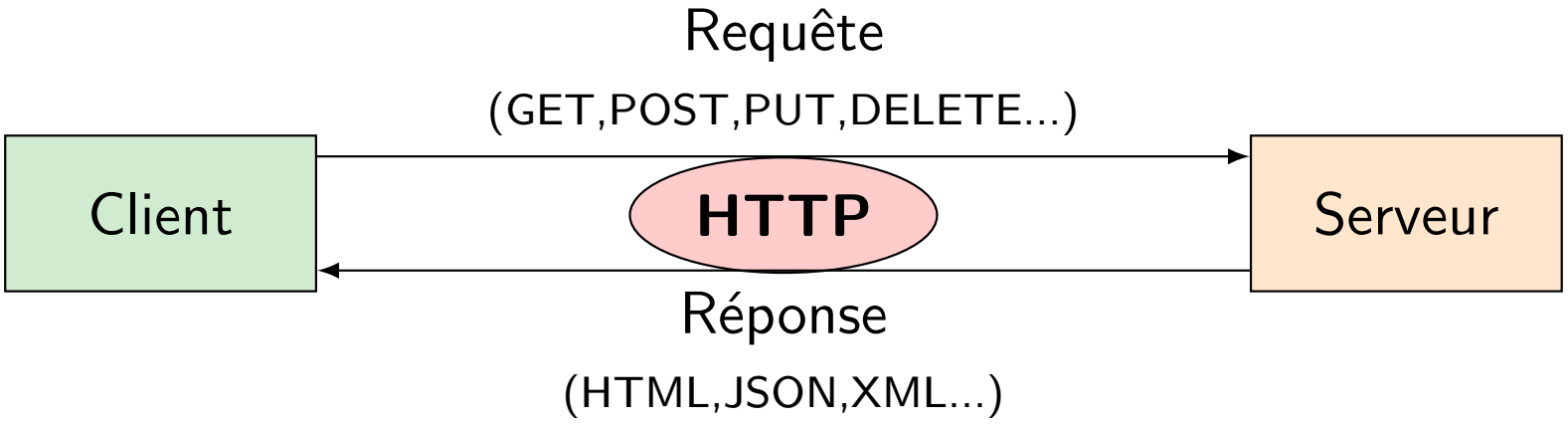
- [page_liste.html]
- les balises ne sont pas affichées. Caractères spéciaux :

  `<` $\rightsquigarrow$ `<`

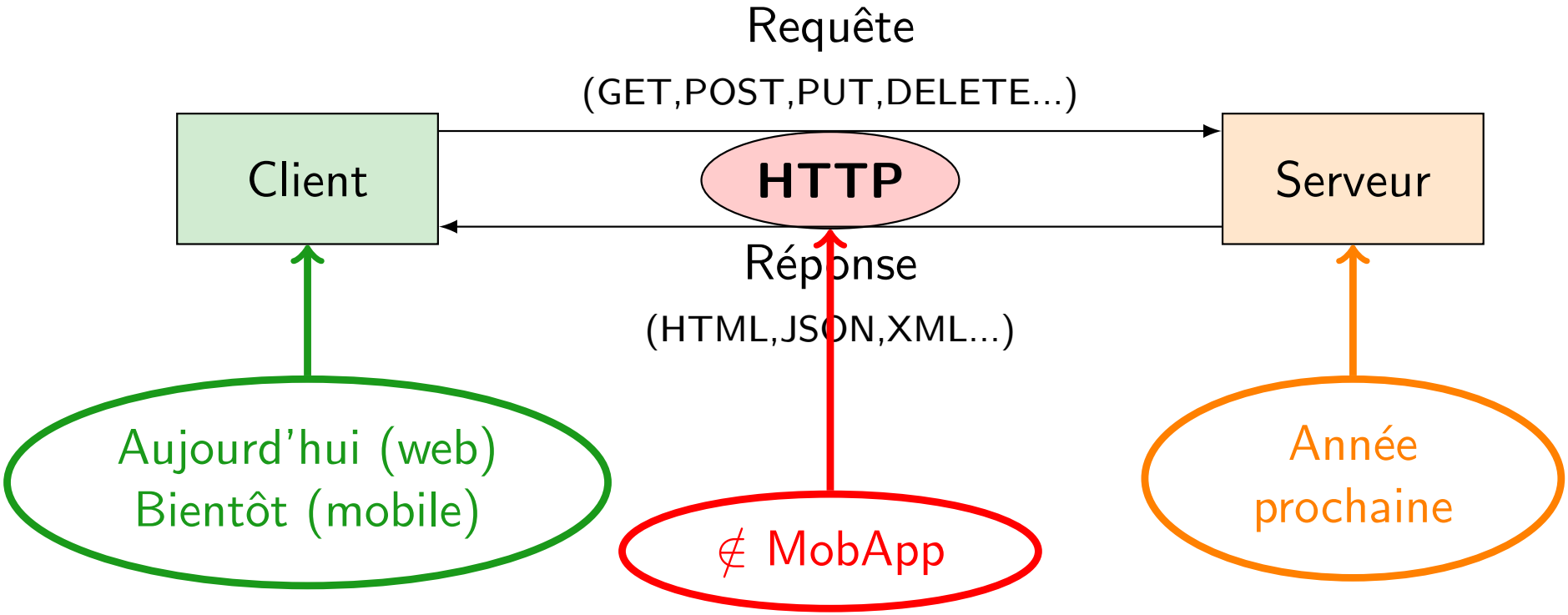
  `>` $\rightsquigarrow$ `>`

  $\vdots$ $\rightsquigarrow$ $\vdots$

Le modèle client–serveur



Le modèle client–serveur



Les formulaires

- comment envoyer des requêtes au serveur ?
- la balise <`form`> et ses attributs
  - `method` : restreint à GET ou POST
  - `action` : l'URL où sera envoyée la requête
- la balise <`input`> et ses attributs
  - `type`
  - `name` : le nom du paramètre de la requête
  - `value` : sa valeur
- la balise <`button`> et ses attributs
  - `type` (submit pour envoyer le formulaire)
- bien sûr, il faut un serveur en face
- [page_formulaire.html]

3. CSS

CSS – Cascading Style Sheet

- l'attribut `style`
  - trop de propriétés pour les mentionner
  - schéma :

```
propriété :  valeur;
```

- [page_style.html]
- encombrant : factoriser dans le <head>, dans une balise de <style> où l'on référence
  - les balises ayant des identifiants (attribut `id`)
  - un certain type de balise
  - une classe de balise (attribut `class`)
- et bien d'autre choses...
- [page_css.html]

Fichiers séparés

Encore trop lourd

- mettre le CSS dans un fichier séparé

```
<link rel="stylesheet" href="style1.css">
```

- [page_css_separee.html]

Fichiers séparés

Encore trop lourd

- mettre le CSS dans un fichier séparé

```
<link rel="stylesheet" href="style1.css">
```

- [page_css_separee.html]
- on peut mettre du style partout
  - fichier séparé, balise `<style>`, attribut des balises
  - gestion des priorités : oui. (*Cascading*)

4. JS

Un bouton sans formulaire

- mais qui réagit
- [page_bouton.html]

Un bouton sans formulaire

- mais qui réagit
- [page_bouton.html]

JAVASCRIPT

- langage de programmation à part entière
- rien à voir avec Java
- exécuté (interprété) localement, **par le navigateur**

Un bouton sans formulaire

- mais qui réagit
- [page_bouton.html]

JAVASCRIPT

- langage de programmation à part entière
- rien à voir avec Java
- exécuté (interprété) localement, **par le navigateur**
- accès à tous les éléments du document, hiérarchiquement
  - DOM: Document Object Model
  - `let aut = document.querySelector('#auteur')`
    - sélectionne la première balise `<... id="auteur" ...>`
    - `#` pour l'attribut `id`, autres manières possibles (liens avec CSS)
    - représentation sous la forme d'un objet
  - lire/modifier les attributs, les nœuds fils... Exemple:

```
aut.innerHTML='Olivier Hermant'
aut.style.backgroundColor = 'red'
aut.style.fontSize = '300\%'
```

  - [page_auteur.html]

Javascript rapide

- nous n'allons pas apprendre JS en 1h

Javascript rapide

- nous n'allons pas apprendre JS en 1h
- mais en 5 minutes :
  - déclarer une variable : `let`, sans le type
  - `;` obligatoire si on ne change pas de ligne
  - `'une chaîne'` et `"une chaîne"` sont valides
  - séparer le JS du HTML : balise `<script>`
- [page_auteur_2.html]

Javascript rapide

- nous n'allons pas apprendre JS en 1h
- mais en 5 minutes :
  - déclarer une variable : `let`, sans le type
  - `;` obligatoire si on ne change pas de ligne
  - `'une chaîne'` et `"une chaîne"` sont valides
  - séparer le JS du HTML : balise `<script>`
- [page_auteur_2.html]
- on peut aussi faire des fichiers séparés, à inclure
- [page_auteur_3.html]

JS et le modèle client–serveur

- AJAX : Asynchronous JavaScript and XML
- en JS :
  - envoi de la requête [à moitié à la main]
  - traitement de la réponse (une fois reçue) [automatisé]
  - mise à jour de la page [à la main]
- utilisation de librairies : aujourd'hui, aucune
  - par ordre chronologique : `jQuery`, `Angular`, `Node`
  - évolution vers dse frameworks complets (`Vue`, `React`)

Attribution dynamique de propriétés/code

- Le sélecteur des objets du DOM est `document.querySelector()`
- l'objet du DOM sélectionné a de nombreuses fonctions très utiles
- exemple : [page_statique.html]
  - usage double lecture/écriture

Attribution dynamique de propriétés/code

- Le sélecteur des objets du DOM est `document.querySelector()`
- l'objet du DOM sélectionné a de nombreuses fonctions très utiles
- exemple : [page_statique.html]
  - usage double lecture/écriture
- **ne plus coder en dur** l'appel à `changer()` dans l'attribut `onClick`:
  - [page_dyn.html]

Attribution dynamique de propriétés/code

- Le sélecteur des objets du DOM est `document.querySelector()`
- l'objet du DOM sélectionné a de nombreuses fonctions très utiles
- exemple : [page_statique.html]
  - usage double lecture/écriture
- **ne plus coder en dur** l'appel à `changer()` dans l'attribut `onClick`:
  - [page_dyn.html]
- `addBehavior` affecte une **fonction** à la propriété `onclick`
- Possible uniquement quand le document est **déjà** chargé
  - retarder l'appel à `addBehavior`
  - enregistrer `addBehavior` comme callback

Faire des requêtes

- avec `fetch` : assez complexe
- on vous a simplifié la tâche avec `simplerequest.js`
  - utiliser la fonction `req`
- format de retour : JSON (standard)
- enregistrer une fonction de callback
  - en charge de la mise à jour de la page
  - en fonction de la réponse reçue
  - ... quand elle sera reçue
- [page_requete.html]

Outils de base pour développer avec HTML, CSS, JS

- un bon éditeur (e.g. vscode)
- inspecteur de code & outils développeur des navigateurs