Technologie des Applications Web

Chapitre 3: Les Applications Internet Riches (RIA)

Master 1 RSD 2020-2021

M. Laïdi FOUGHALI

Ressources utilisées :

https://www.nmedia.ca/articles/les-applications-internet-riches-rich-internet-application

https://www.w3schools.com/js/js_ajax_intro.asp

La raison d'être des RIA

- Les « applications Web » sont les logiciels accessibles par un navigateur Internet crées, en particulier, pour apporter des solutions aux désagréments des applications de bureau.

- Plus récemment, les applications Internet riches (*RIA: Rich Internet Application*) sont apparues pour combiner certains des avantages des « applications de bureau » et des « applications Web » en offrant un juste équilibre entre puissance, ergonomie et accessibilité.

- Les applications RIA se démarquent des applications Web par l'interactivité qu'elles offrent : support multimédia, rétroactions, accès aux périphériques du poste, interactivité avec d'autres internautes en temps réel, etc.

- Facebook est un excellent exemple d'une RIA: la plateforme supporte la vidéo, le son, l'image, possède un système de chat instantané et j'en passe… beaucoup!

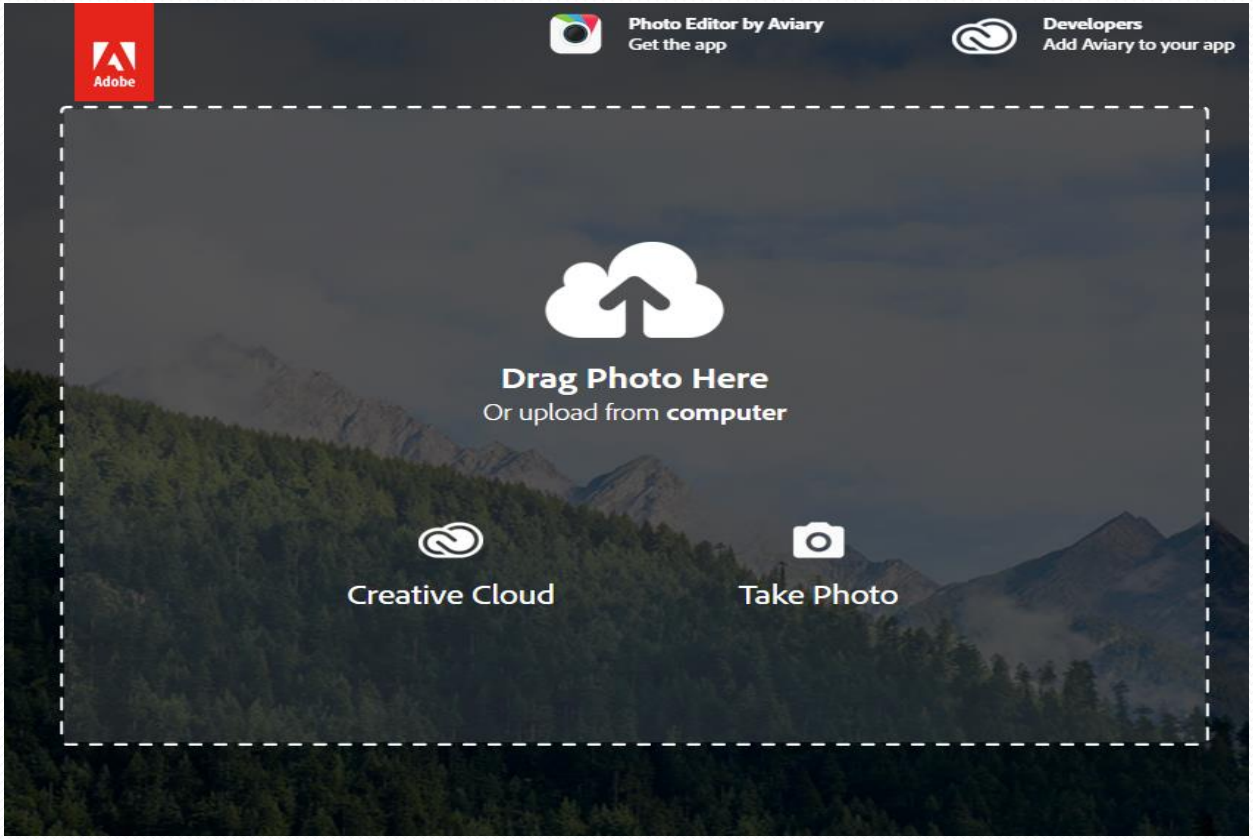
- En revanche, eBay, populaire site d'enchères en ligne, est une application Web dite pauvre, car elle offre peu d'interactions entre l'utilisateur et la plateforme elle-même.

Comparaison des avantages des types de logiciels applicatifs

	Ergonomie	Vitesse	Maintenance	Puissance	Collaboratif	Multimédia
Application de bureau	●●●	●●●●	●	●●●●	●	●●●
Application Web	●	●●	●●●	●●●	●●●	●
Application Internet riche	●●●	●●●	●●●	●	●●●●	●●●

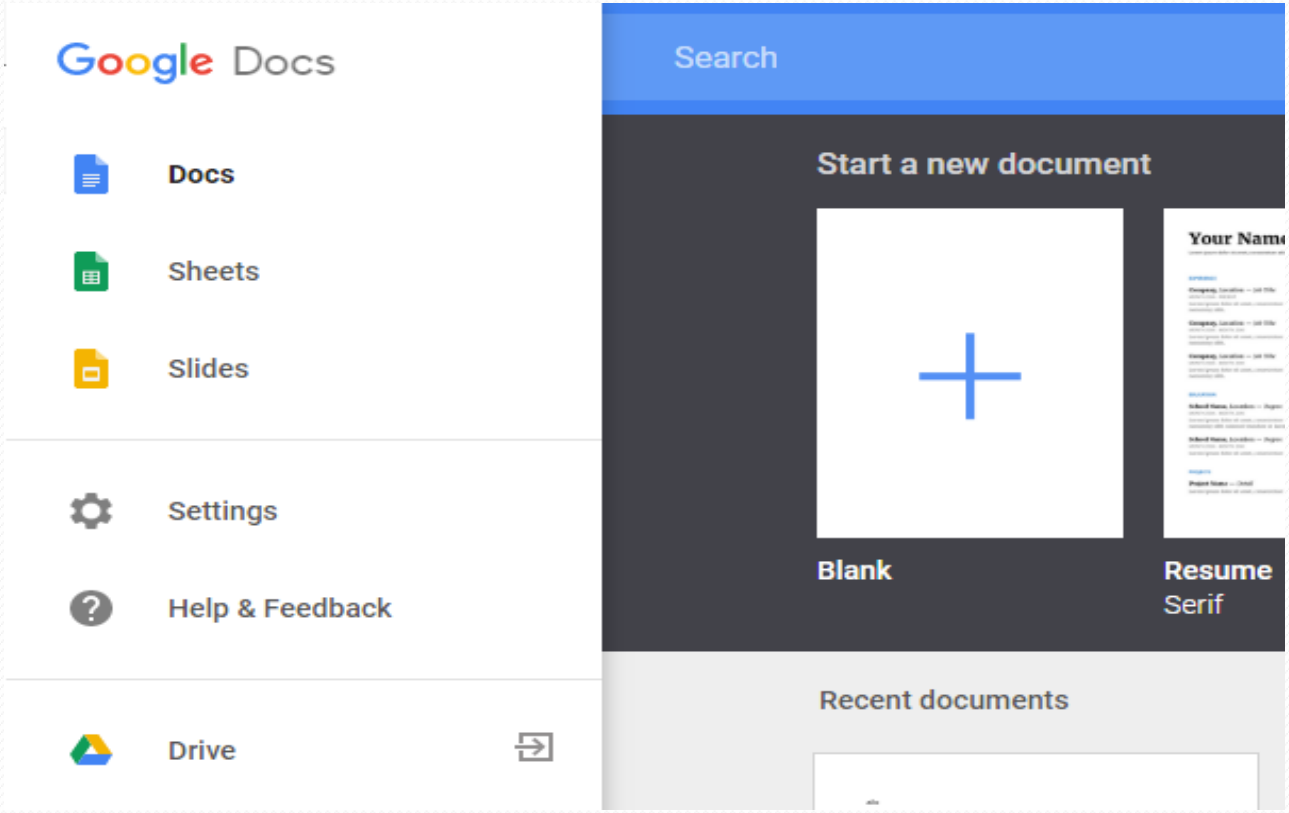
- Les RIA bien qu'elles soient réputées pour être moins puissantes que les applications de bureau ou les applications Web et qu'elles soient dépendantes du lien Internet, elles restent variées et très impressionnantes.
- Les RIA peuvent remplacer des logiciels complets tels que : le traitement de texte, la retouche d'images, le montage vidéo, la comptabilité et la gestion de clientèle, etc.
- Toutefois, elles sont particulièrement populaires pour le développement d'outils collaboratifs en ligne, d'outils de personnalisation à la volée, de *dashboard* et bien sûr de jeux!

Exemple 1: Logiciel – Aviary



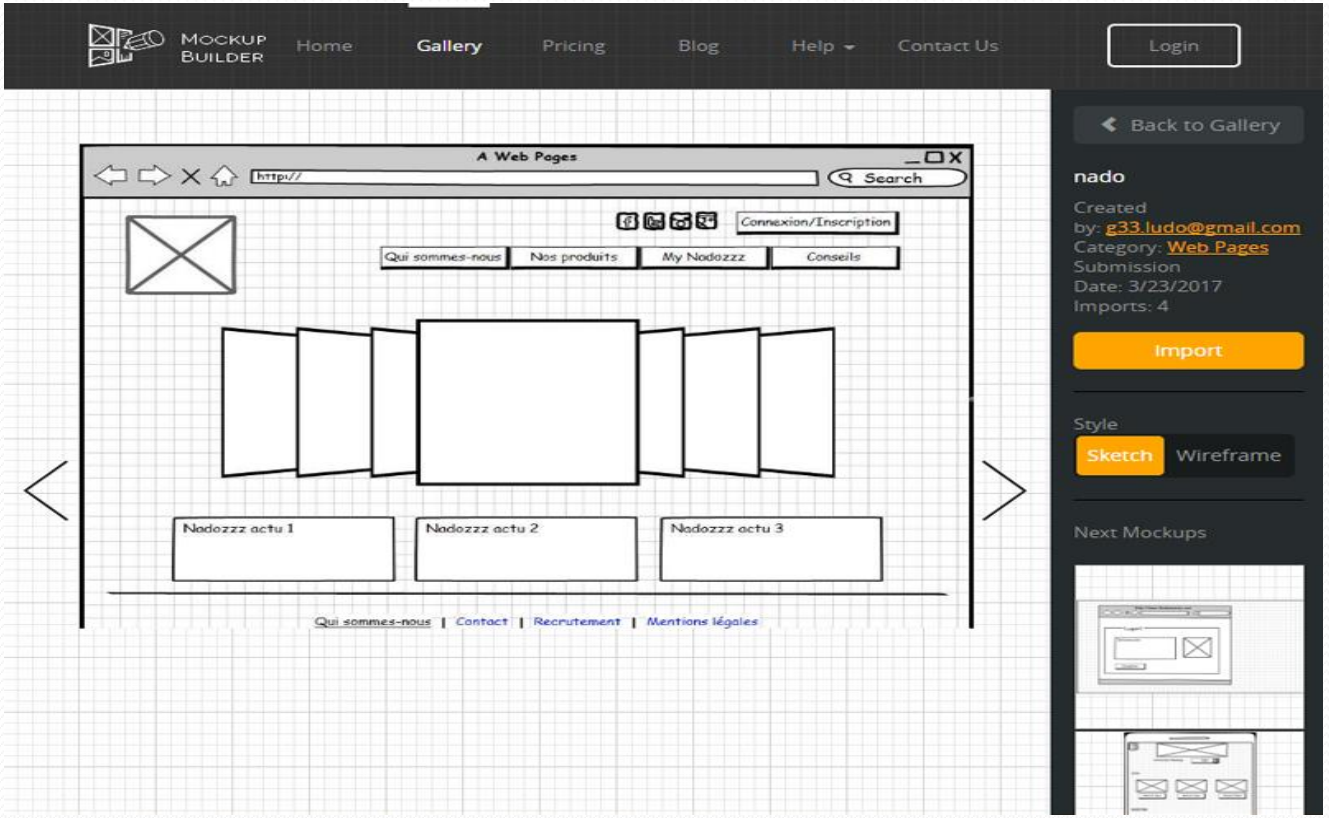
Traitement d'images et de sons
Flash

Exemple 2: Outils collaboratifs – Google Docs



Traitement de texte et chiffrier en temps réel
JavaScript

Exemple 3: Mockup Builder



Outils de création de diagrammes en fil de fer
Silverlight

Exemple 4: Jeux en ligne - Bearville

Plateforme ludo-éducative permettant l'interaction entre les joueurs à travers des mini jeux Flash

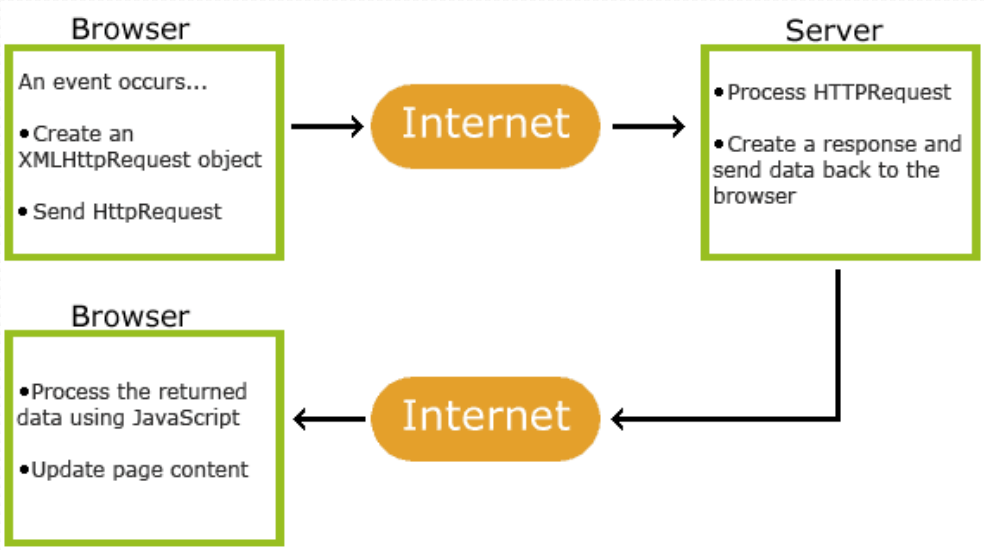
Les technologies RIA

Technologie	Niveau de pénétration
Adobe Flash	Offre un support de périphériques et une gestion de la 3D exceptionnelle. Nécessite le plug-in. L'indexation dans les moteurs de recherche est rarement optimale. Ne supporte pas iOS.
Microsoft Silverlight	Supporte plusieurs langages de programmation. Permet de bien scinder les aspects visuels de la programmation: une meilleure collaboration entre designers et programmeurs. Nécessite le plug-in. Les performances dépendent du poste de travail. Même problème d'indexation Flash. Ne supporte pas iOS.
HTML JavaScript AJAX	Les applications ont pris un essor considérable grâce à l'apparition de librairies puissantes et faciles d'intégration. Technologie libre. Permettent de développer des applications légères et flexibles. Les applications pourront interagir avec d'autres systèmes (*Web Services*, base de données) en utilisant un autre langage généralement plus puissant : PHP, Ruby, Perl, etc.
Applets Java	Désuétude. Revirement vers JavaFX. Il n'y a pas trop d'exemples.

AJAX

- AJAX est une technique utilisant différentes technologies ajoutées aux navigateurs entre 1995 et 2005, et dont la particularité est de permettre d'effectuer des requêtes aux serveur Web et, en conséquence, de modifier partiellement la page Web.
- AJAX dénote *Asynchronous JavaScript And XML* : JavaScript et XML asynchrones.
- AJAX combine JavaScript et DOM, XMLHttpRequest ou Fetch API qui servent au dialogue asynchrone avec le serveur Web ; ainsi qu'un format de données (XML ou JSON), afin d'améliorer maniabilité et confort d'utilisation des application RIA.
- XML, cité dans l'acronyme, était historiquement le moyen privilégié pour structurer les informations transmises entre Serveur Web et navigateur, de nos jours le JSON tend à le remplacer pour cet usage.
- AJAX fonctionne sur tous les navigateurs Web courants.

Comment fonctionne AJAX ?



1. Un événement se produit dans une page Web (la page est chargée, un bouton est cliqué)
2. Un objet XMLHttpRequest est créé par JavaScript
3. L'objet XMLHttpRequest envoie une requête à un serveur Web
4. Le serveur traite la demande
5. Le serveur renvoie une réponse à la page Web
6. La réponse est lue par JavaScript
7. Une action appropriée (comme la mise à jour de la page) est effectuée par JavaScript

The XMLHttpRequest Object

- L' XMLHttpRequestobjet peut être utilisé pour échanger des données avec un serveur Web dans les coulisses. Cela signifie qu'il est possible de mettre à jour des parties d'une page Web, sans recharger la page entière.

- Tous les navigateurs modernes (Chrome, Firefox, IE7 +, Edge, Safari, Opera) ont un XMLHttpRequestobjet intégré .

- Syntaxe de création d'un XMLHttpRequestobjet:

  variable = new XMLHttpRequest();

- **Accès à travers les domaines:** Pour des raisons de sécurité, les navigateurs n'autorisent pas l'accès entre les domaines: la page Web et le fichier XML doivent se trouver sur le même serveur.

- **Fetch API:** Les navigateurs modernes peuvent utiliser Fetch API au lieu de l'objet XMLHttpRequest. L'interface de l'API Fetch permet au navigateur Web d'envoyer des requêtes HTTP aux serveurs Web. Si vous utilisez l'objet XMLHttpRequest, Fetch peut faire de même d'une manière plus simple.

Méthodes d'objet XMLHttpRequest

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method, url, async, user, psw*)	Specifies the request: *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server; Used for GET requests
send(*string*)	Sends the request to the server; Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

Propriétés de l'objet XMLHttpRequest

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

AJAX - Envoyer une demande à un serveur

Method	Description
open(*method, url, async*)	Specifies the type of request: *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

GET ou POST ?

- GET plus simple et plus rapide.
- POST peut envoyer une grande quantité de données au serveur.
- POST est plus robuste et sécurisé que GET.

AJAX – Exemples d'envoi

- **GET**

```
xhttp.open("GET", "demo_get.php", true);
xhttp.send();
```

- **POST**

```
xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
xhttp.send();
```

- **Ajout d'un en-tête HTTP avec setRequestHeader():**

```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

Method	Description
setRequestHeader(*header, value*)	Adds HTTP headers to the request: *header*: specifies the header name *value*: specifies the header value

AJAX – Exemples d'envoi (bis)

- **L'url point un fichier ou un script sur le serveur:**

```
xhttp.open("GET", "ajax_test.php", true);
```

- La nature de de la réponse doit spécifiée : synchrone ou asynchrone. En envoyant de manière asynchrone, le JavaScript n'a pas à attendre la réponse du serveur.

- **onreadystatechange** permetde définir une fonction à exécuter lorsque la requête reçoit une réponse:

```
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();
```

AJAX - Réponse du serveur

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 403: "Forbidden" 404: "Page not found“ For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")
onreadystatechange event is triggered four times, one time for each change in the readyState.	

AJAX - Using a Callback Function

- If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task.

- The function call should contain the URL and what function to call when the response is ready.

```
loadDoc("url-1", myFunction1);

loadDoc("url-2", myFunction2);

function loadDoc(url, cFunction) {
  var xhttp;
  xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      cFunction(this);
    }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}

function myFunction1(xhttp) {
  // action goes here
}
function myFunction2(xhttp) {
  // action goes here
}
```

AJAX - Server Response

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

AJAX – Example of server Response

- **The responseText**

```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

- **The responseXML**

```
<script>
var xhttp, xmlDoc, txt, x, i;
xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    xmlDoc = this.responseXML;
    txt = "";
    x = xmlDoc.getElementsByTagName("ARTIST");
    for (i = 0; i < x.length; i++) {
      txt = txt + x[i].childNodes[0].nodeValue + "<br>";
    }
    document.getElementById("demo").innerHTML = txt;
  }
};
xhttp.open("GET", "cd_catalog.xml", true);
xhttp.send();
</script>
```

AJAX – Example of server Response (bis)

```
<!DOCTYPE html>
<html>
<body>
<h2>The XMLHttpRequest Object</h2>
<p>The getResponseHeader() function is used to return specific header
information from a resource, like length, server-type, content-type,
last-modified, etc:</p>

<p>Last modified: <span id="demo"></span></p>

<script>
var xhttp=new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getResponseHeader("Last-Modified");
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
</script>
</body>
</html>
```

Vos Questions!

…

Your Questions!