

Introduction au Web

Mathieu ANCELIN

Mathieu ANCELIN

- Ingénieur d'étude @SERLI
- Java, Scala, Web & OSS
 - ReactiveCouchbase, Weld, etc
 - Poitou-Charentes JUG
- Membre de l'expert group CDI 1.1 (JSR-346)
- Membre de l'EEG OSGi
- @TrevorReznik



Dissertation

Dissertation

Le web, c'est bien !

Dissertation

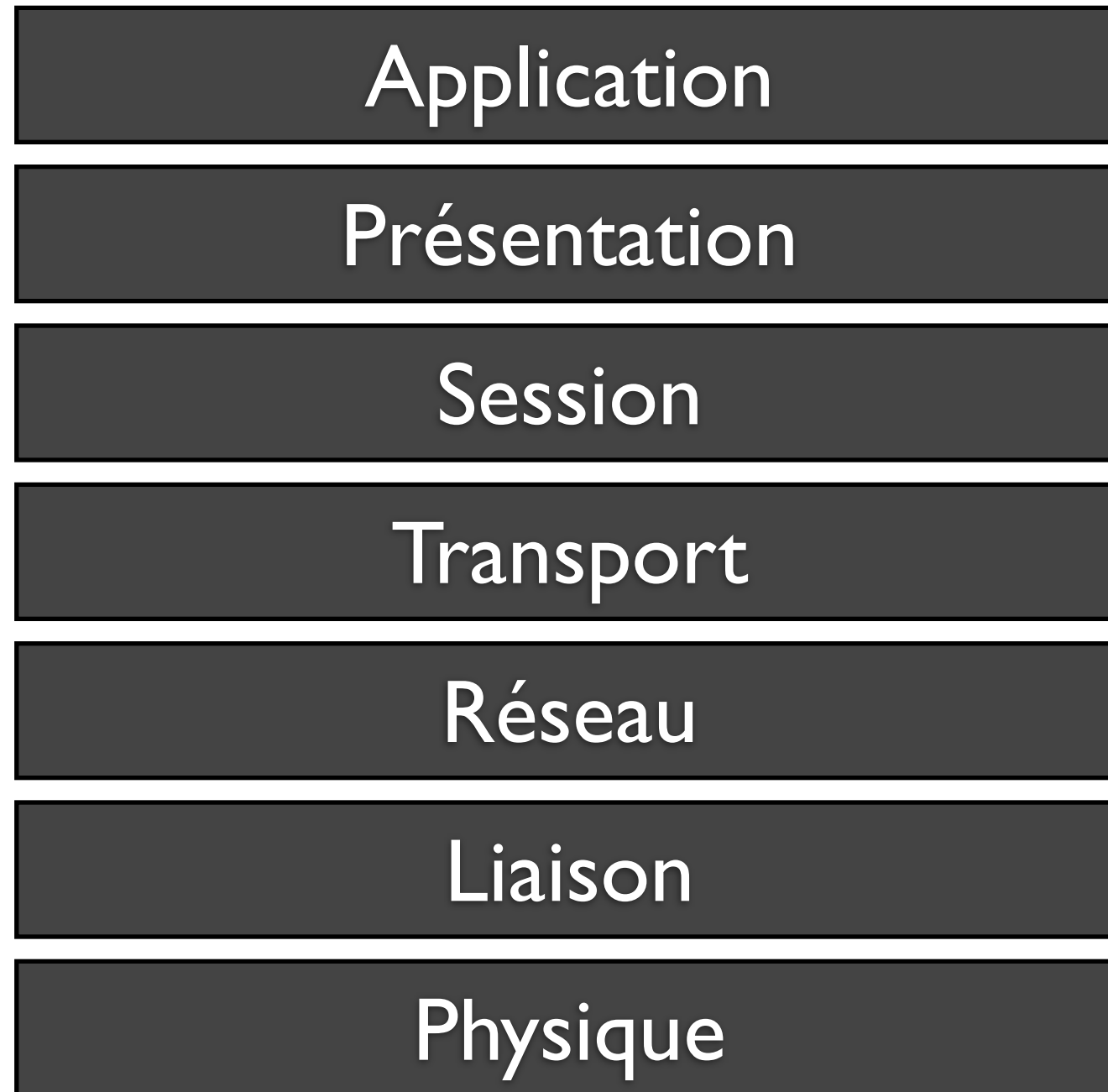
Le web, c'est bien !

vous avez 2 heures ...

HTTP

- Protocole de communication client/serveur développé pour le web
- Protocole de la couche application
 - est censé fonctionner sur n'importe quel type de connexion fiable
 - dans les faits, basé sur TCP

HTTP



HTTP



TCP



HTTP

- Par défaut utilise le port 80
 - port 443 pour le mode sécurisé (https)
- Au niveau du client, le plus connu est le navigateur
 - outils ligne de commande : curl, wget, etc ..
 - librairies
 - download managers, aspirateurs, etc ...

Un peu d'histoire

- HTTP a été inventé par Tim Berners-Lee au CERN
 - avec entre autre, le concept d'adresse web, HTML
 - création du web
- Apparemment inventé dans un bureau sur la partie française du CERN ;-)
- Depuis, standard IETF (RFC 1945, 2068, 2616)

HTTP

- Quatres méthodes disponibles en HTTP
 - GET
 - POST
 - PUT
 - DELETE

HTTP

- Autres méthodes moins fréquentes
 - HEAD
 - OPTIONS
 - CONNECT
 - TRACE

HTTP's anatomy

- Requête HTTP

Ligne de commande (Commande, URL, Version)

En-tête de requête

[Ligne vide]

Corps de requête

- Réponse HTTP

Ligne de statut (Version, Code-réponse)

En-tête de réponse

[Ligne vide]

Corps de réponse

HTTP's anatomy

```
GET /page.html HTTP/1.0
Host: example.com
Referer: http://example.com/
User-Agent: Firefox/Gecko 11.0
```

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Server: Apache/0.8.4
Content-Type: text/html
Content-Length: 59
```

```
<html><body><p>Hello World!</p></
body></html>
```

Paramètres HTTP

- Comment passer des paramètres à une ressource HTTP
- GET : passer les paramètres dans l'URL
 - <http://site.com/index?a=1&b=2&c=3>
- POST, PUT : passer les paramètres dans le corps de la requête

```
POST /index HTTP/1.0
```

```
Host: site.com
```

```
User-Agent: Firefox/Gecko 11.0
```

```
a=1&b=2&c=3
```

Paramètres HTTP

- Possibilité de passer autre chose dans le corps de la requête

```
POST /index HTTP/1.0
```

```
Host: site.com
```

```
User-Agent: Firefox/Gecko 11.0
```

```
<root><a>1</a><b>2</b><c>3</c></  
root>
```

```
POST /index HTTP/1.0
```

```
Host: site.com
```

```
User-Agent: Firefox/Gecko 11.0
```

```
{a: 1, b: 2, c: 3}
```

Paramètres HTTP

- Il est également possible de passer des paramètres qui font partie intégrante de l'URL
 - <http://myblog.com/posts/1>
 - <http://myblog.com/posts/1/comments>
 - <http://myblog.com/posts/1/comments/30>
 - <http://myblog.com/authors/mathieu>

Paramètres HTTP

- Il est également possible de passer des paramètres qui font partie intégrante de l'URL
 - <http://myblog.com/posts/1>
 - <http://myblog.com/posts/1/comments>
 - <http://myblog.com/posts/1/comments/30>
 - <http://myblog.com/authors/mathieu>

REST

- Style d'architecture apparu avec l'invention du web
- Fournit la définition de «ressources»
 - localisation de quelque chose n'importe où dans le monde (coord. GPS)
 - représentées par les URLs
 - une ressource possède une ou plusieurs représentations
 - html, xml, json, texte, image, etc ...

REST

- URL => nom
 - universel et unique
 - contrairement aux machines
- Méthodes HTTP => verbes
 - notion de polymorphisme des verbes
 - on applique le même verbe sur les différents noms
 - contrairement à d'autres protocoles où chaque ressource expose ses propres méthodes

RESTful

- Sur une ressource particulière
 - GET => récupérer une représentation de la ressource
 - POST => mise à jour d'une ressource (valeurs dans le corps de la requête)
 - PUT => création d'une ressource (valeurs dans le corps de la requête)
 - DELETE => suppression d'une ressource

RESTful

- Le sens des URLs
 - <http://www.myapp.com/mycompany/user/1>
 - <http://www.myapp.com/blog/post/123>
 - [http://www.myapp.com/blog/post/123/
comment/12](http://www.myapp.com/blog/post/123/comment/12)
- Bookmarkable, lisible, échangeable, etc ...

REST

<http://tomayko.com/writings/rest-to-my-wife>

Connecté vs. déconnecté

- HTTP est un protocole connecté
 - ouverture d'une socket vers le serveur
 - écriture de la requête dans la socket
 - lecture de la réponse depuis la socket
 - fermeture de la socket
- Le serveur traite une connexion, puis l'oublie
 - traite chaque requête unitairement
 - protocole HTTP

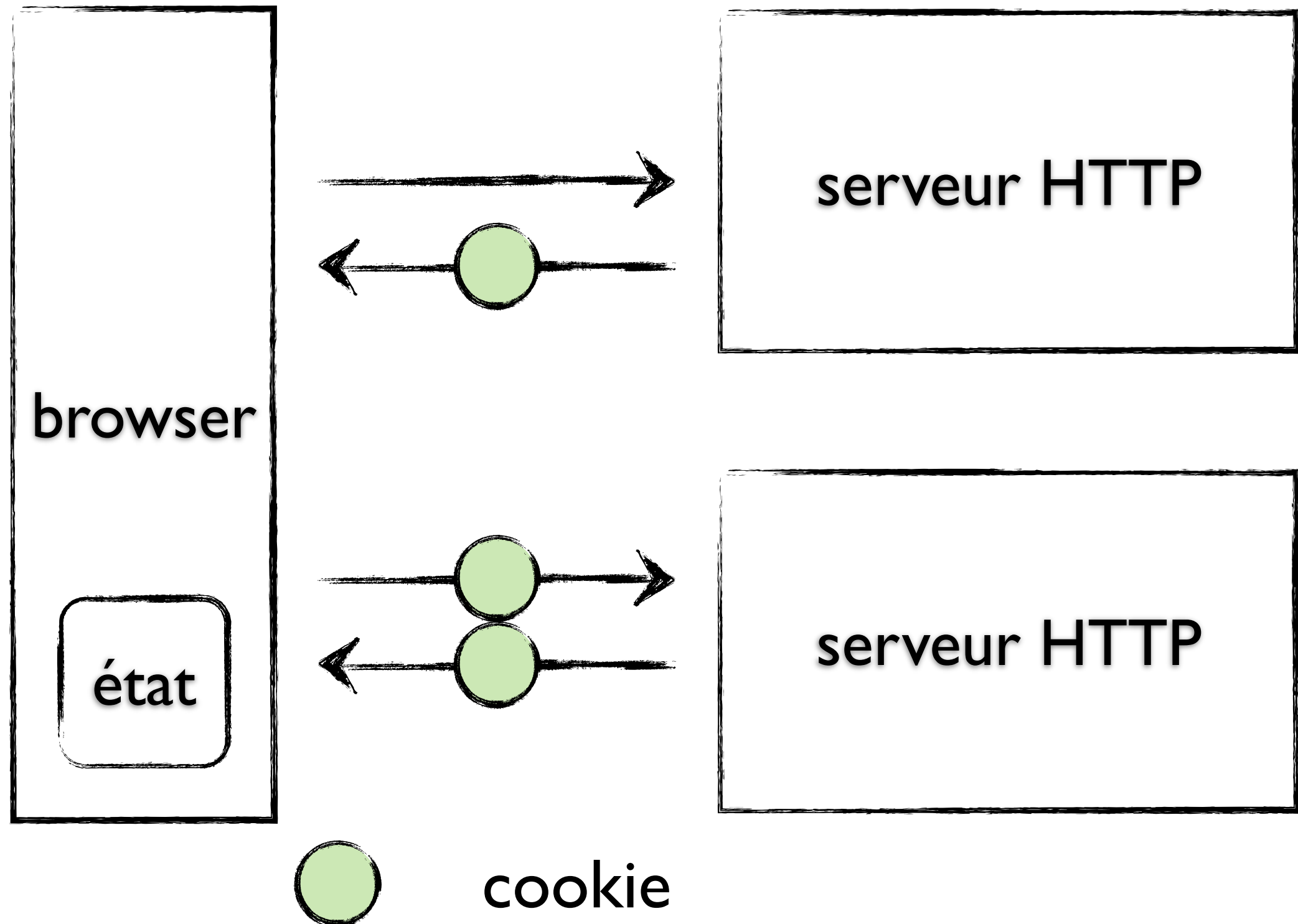
Stateless

- Suivant le style d'architecture REST, le serveur n'est pas censé conserver d'état de l'application en court
- Stateless
- Mais dans une application, il est nécessaire de conserver un état
 - contenu du panier
 - valeurs d'un formulaire
 - etc ...

Cookies



Cookies



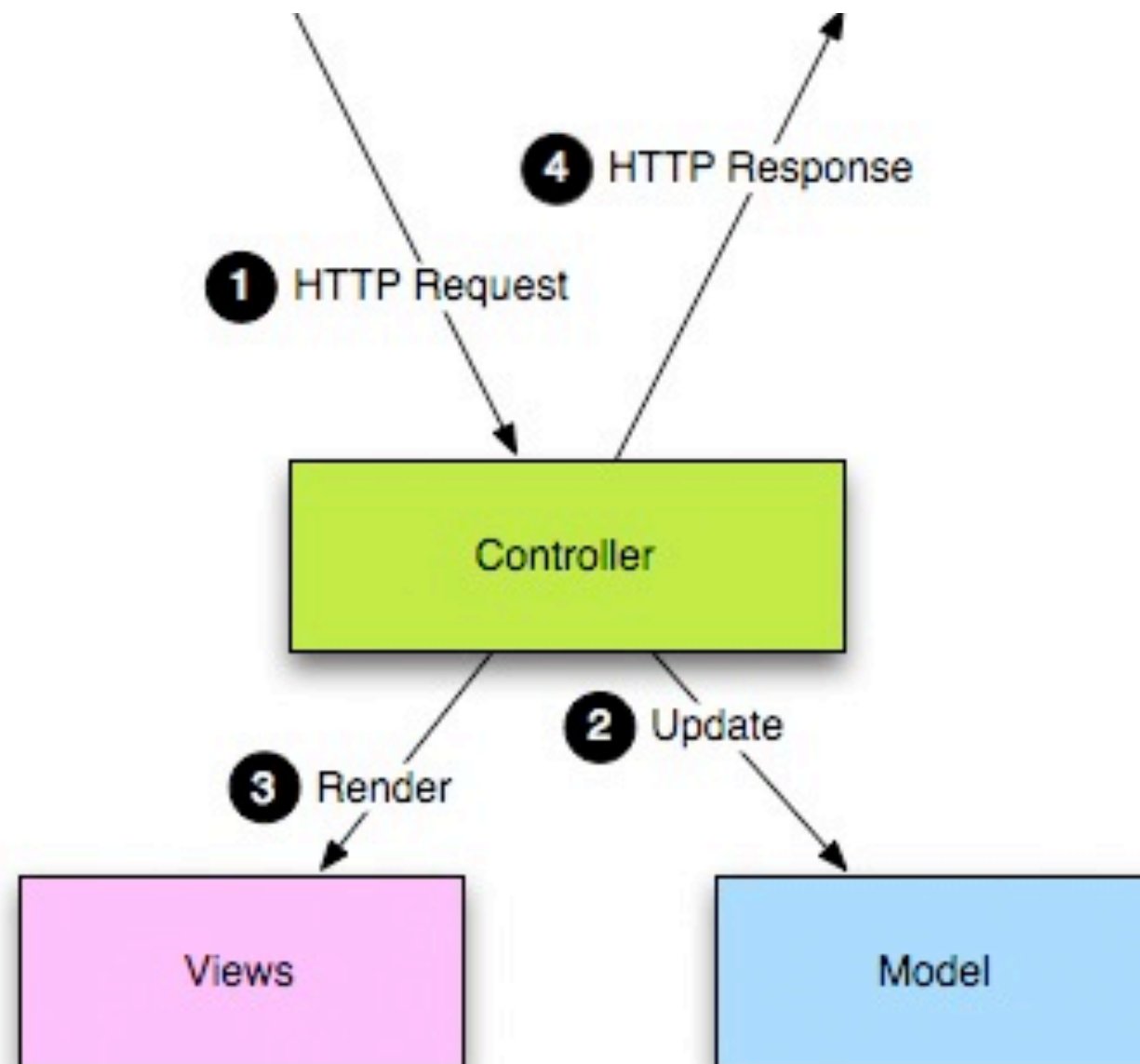
Cookies

- Suite de données nommée envoyée par le serveur puis renvoyée par le navigateur à chaque requête suivant certaines conditions
- Stocké au niveau du client
 - pas plus de 4096 octets
- Durée de validité
 - Session (onglet courant)
 - Heures, jours, mois, années ...

Cookies

- Très utile pour gérer l'état de l'application
- Un cookie contient peu d'informations
 - stocker des ids dans les cookies
 - récupérer les données au moment opportun
 - stocker directement les petites données

MVC



MVC

- Le routeur
 - Sorte de super contrôleur capable de faire correspondre un pattern d'URL et un type de méthode HTTP à une action

```
(GET, /index) => Application.index()  
(GET, /add/{id}) => Base.add(id)
```

MVC

- Le contrôleur
 - Unité de code (souvent une classe) comprenant une ou plusieurs actions

MVC

- Des actions
 - Contenues dans les contrôleurs, les actions correspondent à une URL et une méthode HTTP
 - Une action est déclenchée par une méthode HTTP, exécute du code métier et retournent une vue (où une représentation de la ressource)

MVC

- Des modèles
 - Objets de tout types, reliés à une base de données, un fichier, ou autre
 - Représentent les données à afficher dans une vue (ou une représentation)

MVC

- Des vues / représentations
 - Templates textuels
 - Binaires
 - Données structurées ou semi-structurée
 - XML
 - JSON
 - YAML
 - autre

HTML

- Langage de données conçu pour représenter des pages web
- Permet d'écrire de l'hypertexte (hyperliens)
- Permet d'inclure des ressources multimédia
- Souvent utilisé avec
 - des feuilles de style
 - un langage de programmation

HTML

- Langage à balise dérivé du SGML
- La version 4 contient 91 éléments
 - Structure du document, sémantique
 - Listes, tables, liens, inclusion multimedia
 - Formulaires, scripts, cadres, regroupements
 - Style, présentation du texte
 - etc ...

HTML

```
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML  
2.0//EN">
```

```
<html>
```

```
  <head>
```

```
    <title>Exemple de HTML</title>
```

```
  </head>
```

```
  <body>
```

```
    Ceci est une phrase avec un <a  
href="cible.html">hyperlien</a>.
```

```
    <p>
```

```
      Ceci est un paragraphe sans hyperlien.
```

```
    </p>
```

```
  </body>
```

```
</html>
```

HTML

- Base
 - `<div>`, `<p>`
 - ````
 - `<table>`
 - `<form>``<input>`
 - `<a>`
 - `<script>`, `<style>`

CSS

- Langage permettant de décrire la présentation de documents HTML et XML
- Séparer la structure de la présentation
 - permet de supporter plusieurs environnements depuis une même source
 - présentation unifiée
 - conception en parallèle

CSS

```
a {  
    color: #FF0000;  
}  
  
.btn {  
    background: #000000;  
    height: 30px;  
    color: #FFFFFF;  
}  
  
#bouton23 {  
    background: #123456;  
    height: 30px;  
    color: #543211;  
}
```


JavaScript

- Langage de programmation orienté script
- Créé pour surfer sur la vague du succès de Java et pour contrer les API propriétaires de Microsoft
- Langage orienté objets à prototype
- Langage avec une incursion dans la programmation fonctionnelle
- Indispensable de nos jours si on fait du web

Manipulation du DOM

- Javascript est capable de parcourir et manipuler l'arbre représentant la page HTML
- à ce niveau cet arbre est complètement dynamique
- Javascript est donc capable de modifier l'arbre DOM en temps réel
- Utilisation simplifiée via des frameworks

Manipulation du DOM

```
<div id="valeur">Hello World ! </div>  
<button id="goodbye" href="#">  
    Goodbye</button>
```

```
$( '#goodbye' ).click(function(e) {  
    $( '#valeur' )  
        .html( «Goodbye World !» );  
});
```

Templating

- Comme Javascript permet de manipuler le DOM, il est possible d'ajouter des éléments complexes dynamiquement à la page
- Au lieu de faire ça par concaténation de chaînes de caractères
 - bibliothèques de templating en JavaScript
- Mustache.js

JavaScript UI

- En plus de gérer le côté dynamique de la page, Javascript est capable de gérer des composants graphiques
 - en manipulant le DOM ;-)
- Il existe énormément de bibliothèques permettant de gérer des composants graphiques riches
 - JQuery UI, Dojo, Prototype
 - YUI, MooTools, AllowUI

JavaScript MVC

- MVC côté client
- Lorsque l'UI devient très complexe
 - plus simple à gérer
- Databinding bi-directionnel
 - très très pratique, pas besoin de tout faire à la main
- Ember.js, Batman.js, Knockout.js, Backbone.js, Angular.js, Cappuccino, etc ...

AJAX ?



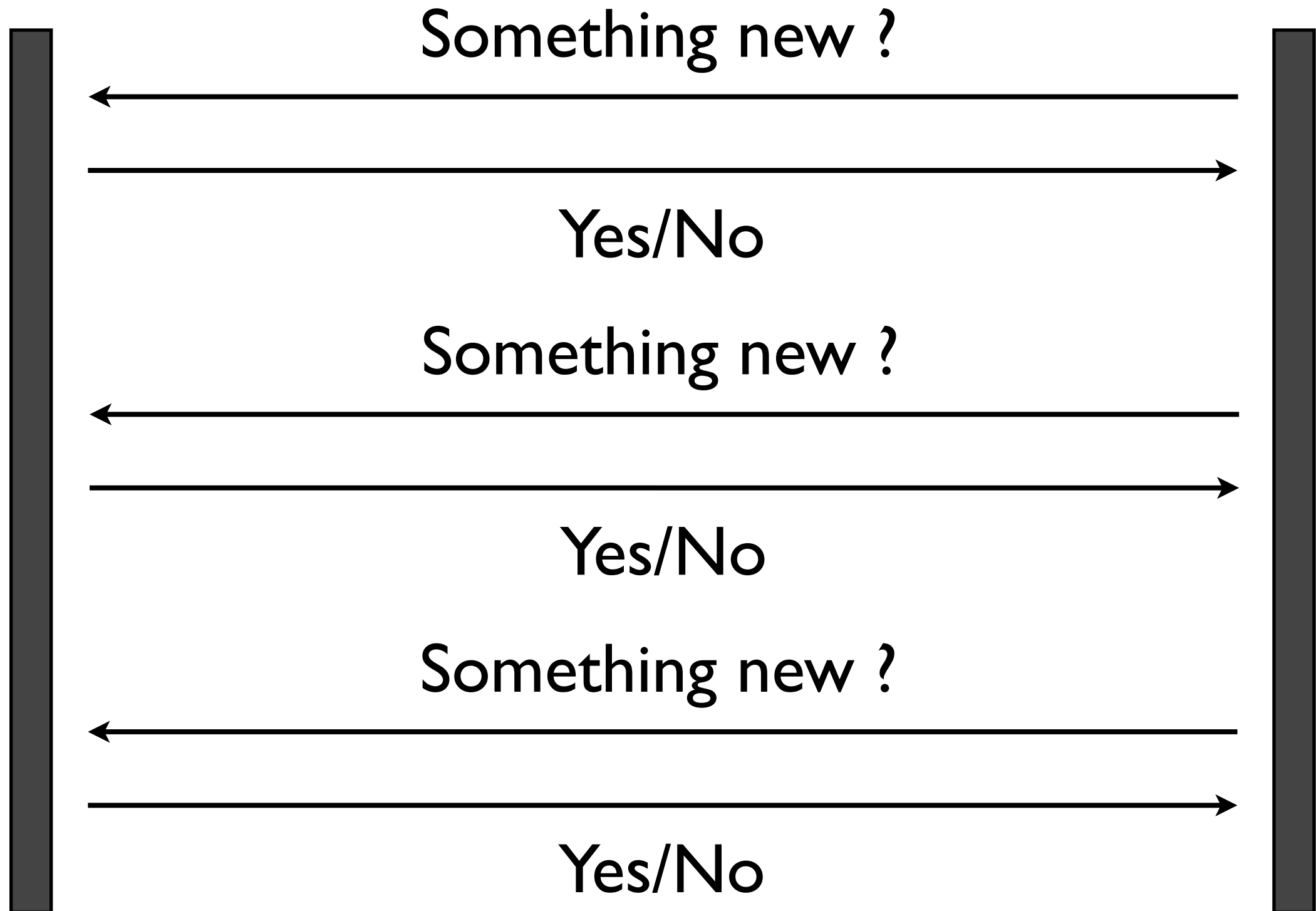
XMLHttpRequest

- La star qui se cache derrière Ajax
- API présente dans Javascript permettant de faire des requêtes HTTP de manière asynchrone
 - appel non bloquant dans Javascript
 - Callback via une fonction
- En le combinant avec la manipulation du DOM
 - Comportement dynamique proche des application de bureau

Polling

Serveur

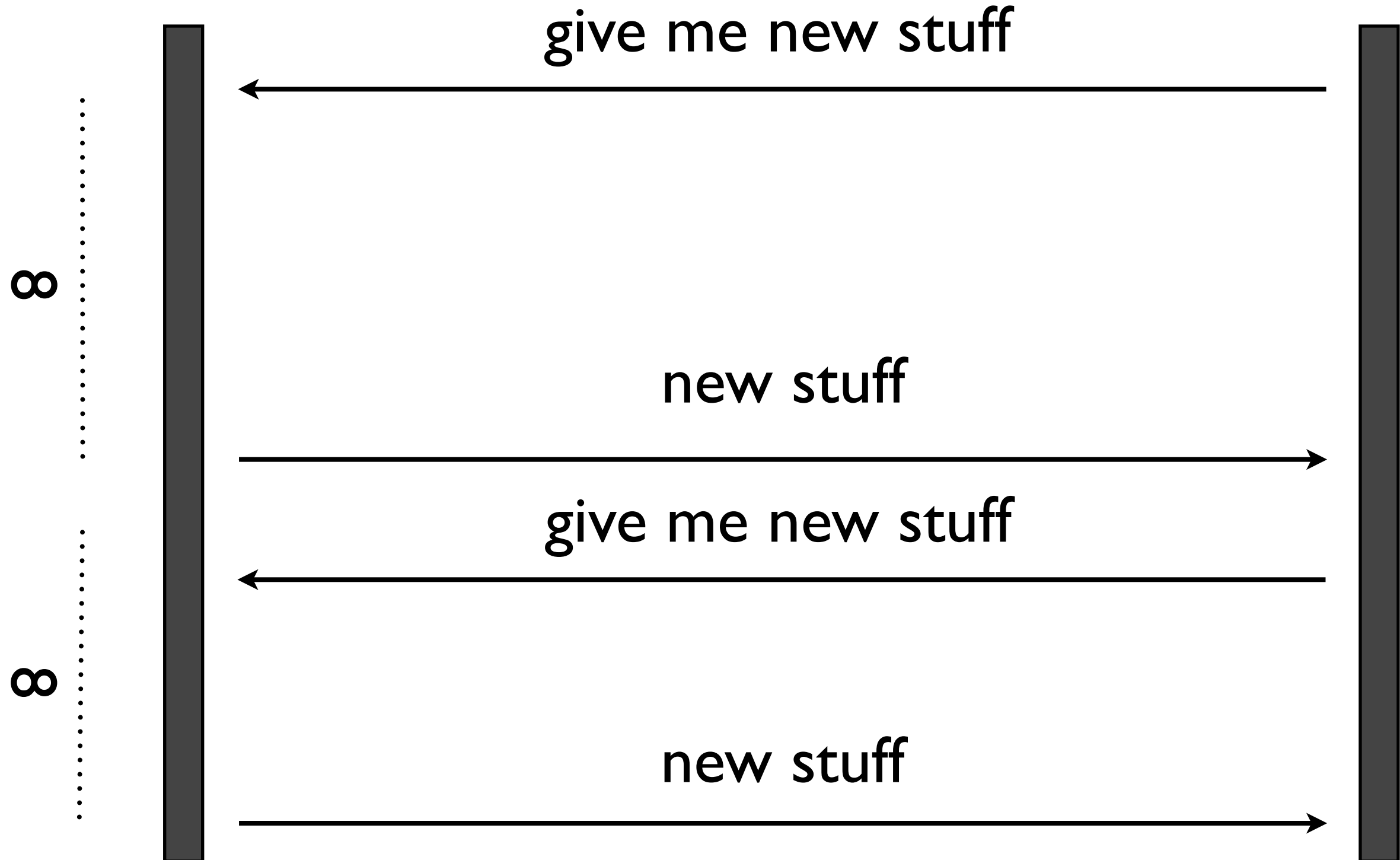
Client



Long polling

Serveur

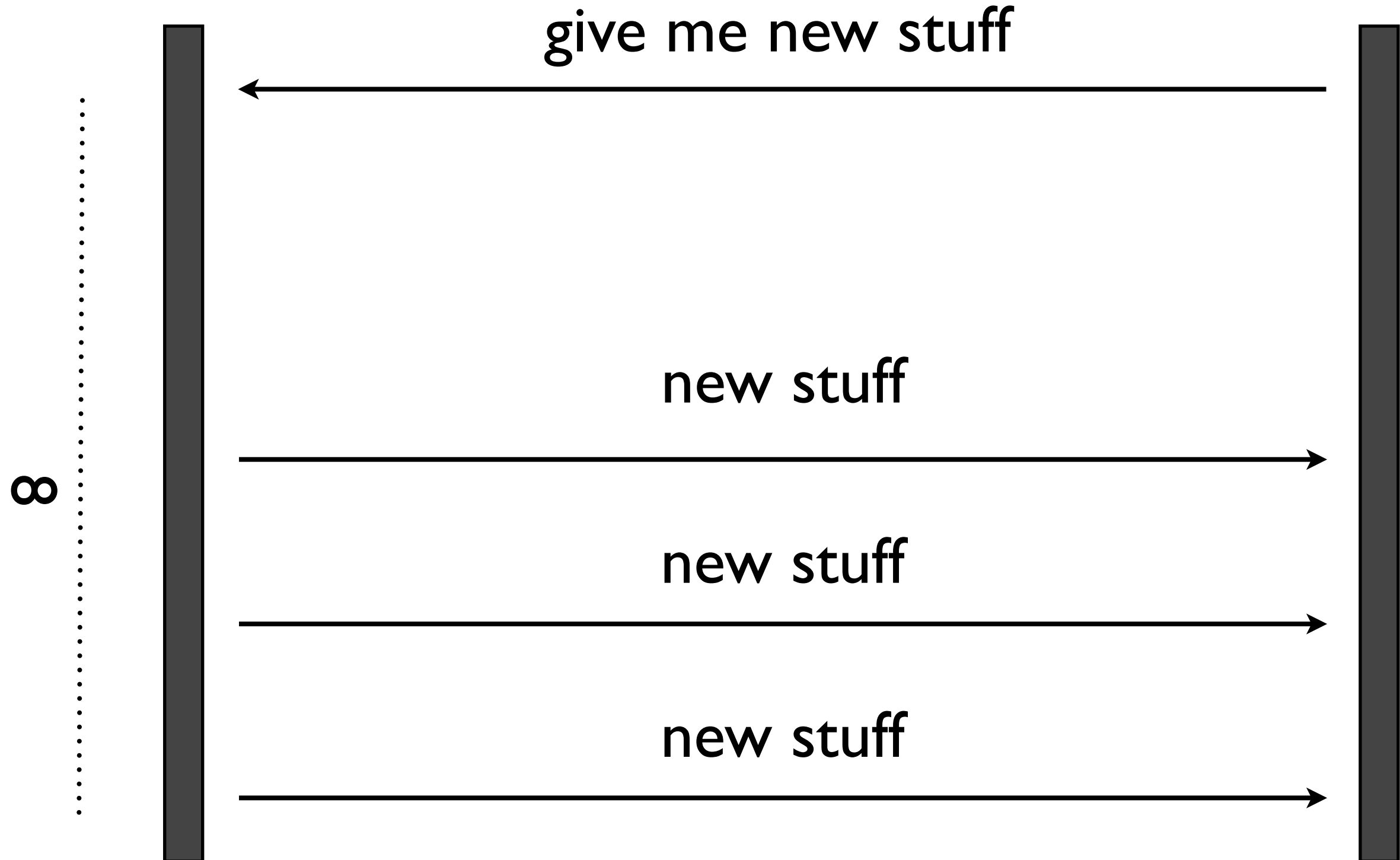
Client



Http Streaming

Serveur

Client



Client riche ?

- Combo HTML/CSS/Javascript considéré comme client riche en face de
 - Flash / Flex
 - JavaFX
 - Silverlight

Le futur

- Les standards du web évoluent
 - certes, lentement
- HTML5
 - nouvelles balises, nouvelles API
 - stockage, temps réel, 2D, 3D
- CSS 3
 - nouvelles capacités
 - transistions

HTML5

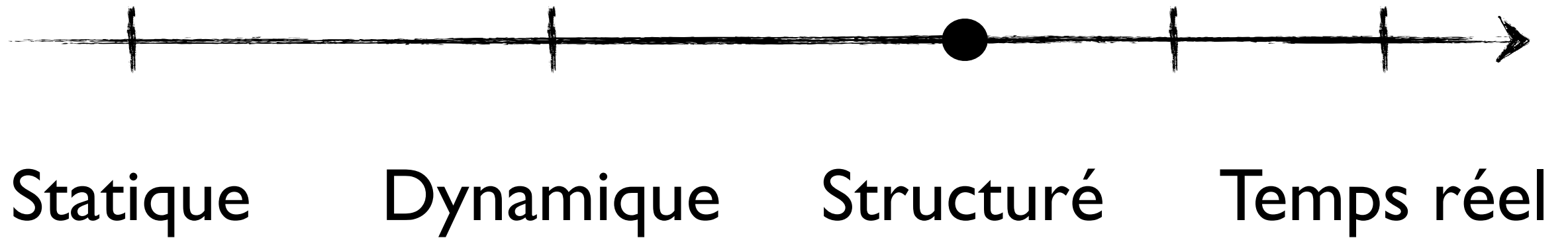
HTML5 = HTML + CSS + JS
+ SVG + WebGL

slides additionnelles

Storage

- Session storage
- Local storage
- Indexed storage

Nouvelle ère



HTTP connecté ?

- Depuis HTTP 1.1 une connexion HTTP peut avoir l'option 'Keep-Alive'
- plusieurs requêtes/réponses pour une même connexion
- streaming
- HTML 5 va profiter au maximum de cette capacité

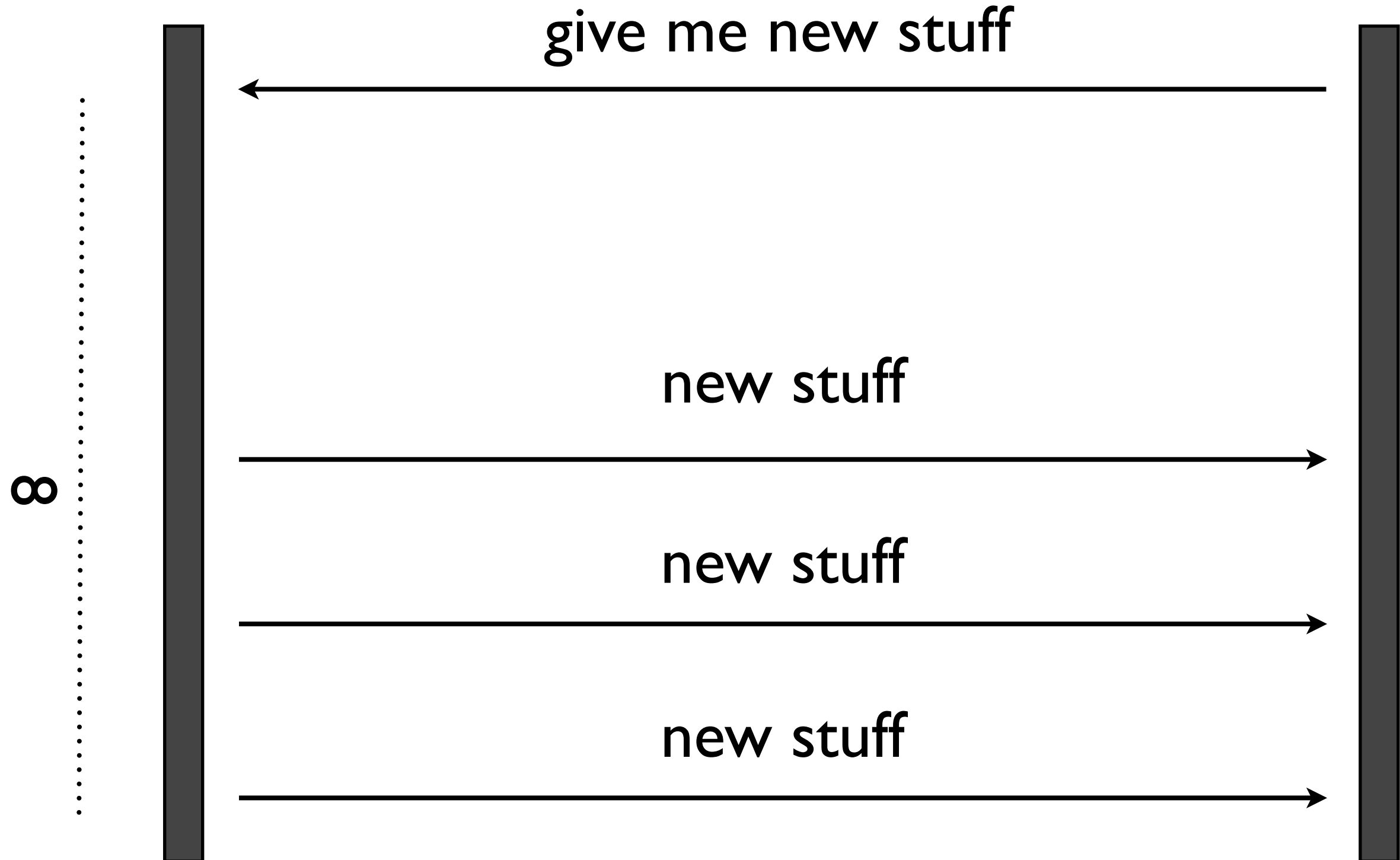
Server Sent Events

- Standardisation du concept de push serveur mais en mieux
- Le navigateur ouvre une connexion persistante et enregistre des callback Javascript
- Le serveur envoie des informations quand bon lui semble ce qui actionne les callbacks sur le navigateur

Server Sent Events

Serveur

Client



Web Sockets

- Ouvre une socket bi-directionnelle entre le client et le serveur
- API Javascript réactive à base de callback
- Privilégier l'envoi d'informations structurées pour ne pas repartir sur les problèmes de formats

Web Sockets

Serveur

Client

connexion



Web Mobile

- L'intérêt de concevoir une application web est son côté universel
- De nos jours tous les terminaux mobiles possèdent de très bon navigateurs

Quand ?

- Chaque constructeur d'OS propose son SDK
- Pour faire une application native, il faut souvent redévelopper x fois une application
- Si celle-ci ne demande pas une intégration système poussée
- l'application web est la meilleure solution

Frameworks JS

- Il existe beaucoup de frameworks Javascript ciblés pour le développement d'applications web pour terminaux mobiles
 - JQuery mobile
 - EmbedJS
 - Mobilize.js
 - Sencha

Interopérabilité système

- Certains navigateurs de terminaux mobiles proposent des API d'interopérabilité
- Possibilité de manipuler le carnet d'adresses
- Possibilité de lancer des appels
- Possibilité de jouer avec la géolocalisation
- etc ...

Conclusion

- Respectez l'architecture REST
- Evitez les frameworks se servant de HTTP comme d'un protocole couteau-suisse
- Servez vous du web
 - universel
 - très capable