HTML5 & websockets

Olivier Ertz - Jérome Freyre
2011 - heig-vd

Au menu

- Deux mots sur html5
- Rappels sur ajax
- Présentation de l'API websocket
- Démonstration
- Conclusion

HTML

De nouvelles balises

De nouveaux attributs

```
<!DOCTYPE html>

<html lang="fr">
<head>
     <meta charset="UTF-8">
     <title>here comes a title</title>
     <link rel="stylesheet" media="screen" href="ecran.css">
</head>
<body>
     <header>
          <h1>Wake up sheeple!</h1>
          <p>
               <a href="news.html">News</a> -
               <a href="blog.html">Blog</a> -
               <a href="forums.html">Forums</a>
          </p>
          <p>Last Modified: <time>2009-04-01</time></p>
          <nav>
               <h1>Navigation</h1>
               <ul>
                    <li><a href="articles.html">Index of all articles</a></li>
                    <li><a href="today.html">Things sheeple need to wake up for today</a></li>
                    <li><a href="successes.html">Sheeple we have managed to wake</a></li>
               </ul>
          </nav>
     </header>
     <div>
          <article>
               <header>
                    <h1>My Day at the Beach</h1>
               </header>
               <div>
                    <p>Today I went to the beach and had a lot of fun.</p>
                    ...more content...
               </div>
               <footer>
                    <p>Posted <time pubdate="" datetime="2009-10-10T14:36-08:00">Thursday</time>.</p>
               </footer>
          </article>
          ...more blog posts...
     </div>
     <footer>
          ...
     </footer>
</body>
</html>
```

mais aussi de nombreuses API

Stockage en local

Web Workers

Base de donnée local + pseudoSQL

Capture de données sur le device

HTML

Manipulation de fichiers

Mode déconnecté

Push events et notifications

Stockage de donnée indexée

Géolocalisation

Websockets

http://slides.html5rocks.com/

Rappel ajax

- Requêtes sur LE serveur
  - cross-domain possible via un proxy ou grâce à JSONP
- Uni-directionnelle; à la demande du client uniquement
- Chaque requête est « complète » en ce qui concerne les headers

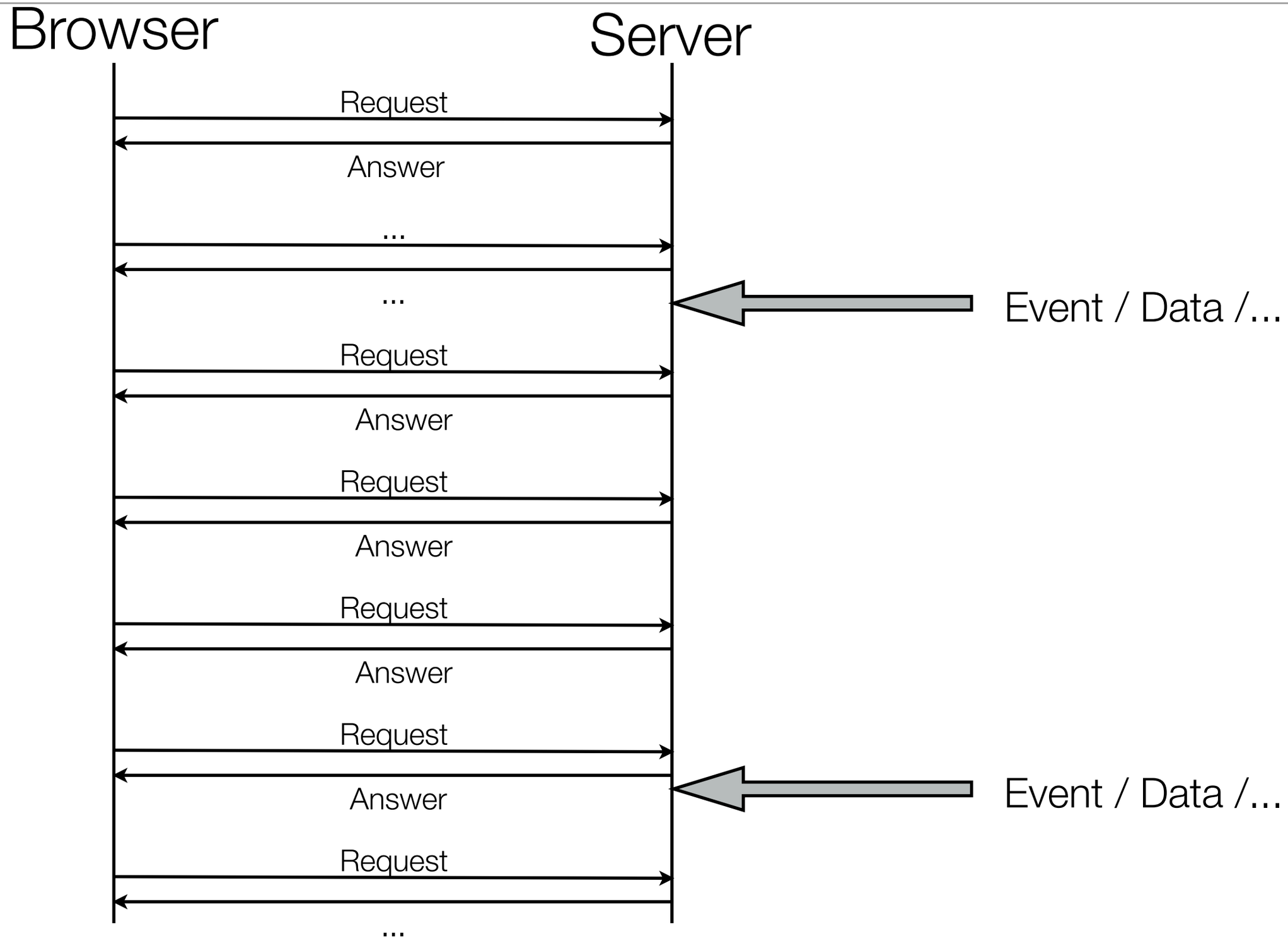
Rappel ajax (2)

- Très utile pour
    - Vérifier du contenu
    - Chargement différé de contenus
    - Améliorer l’interactivité avec les clients

malheureusement, quand on veut rendre
l'interface plus
« real time »

http://www.flickr.com/photos/its-only-lines/4588242859/

polling - Attente active



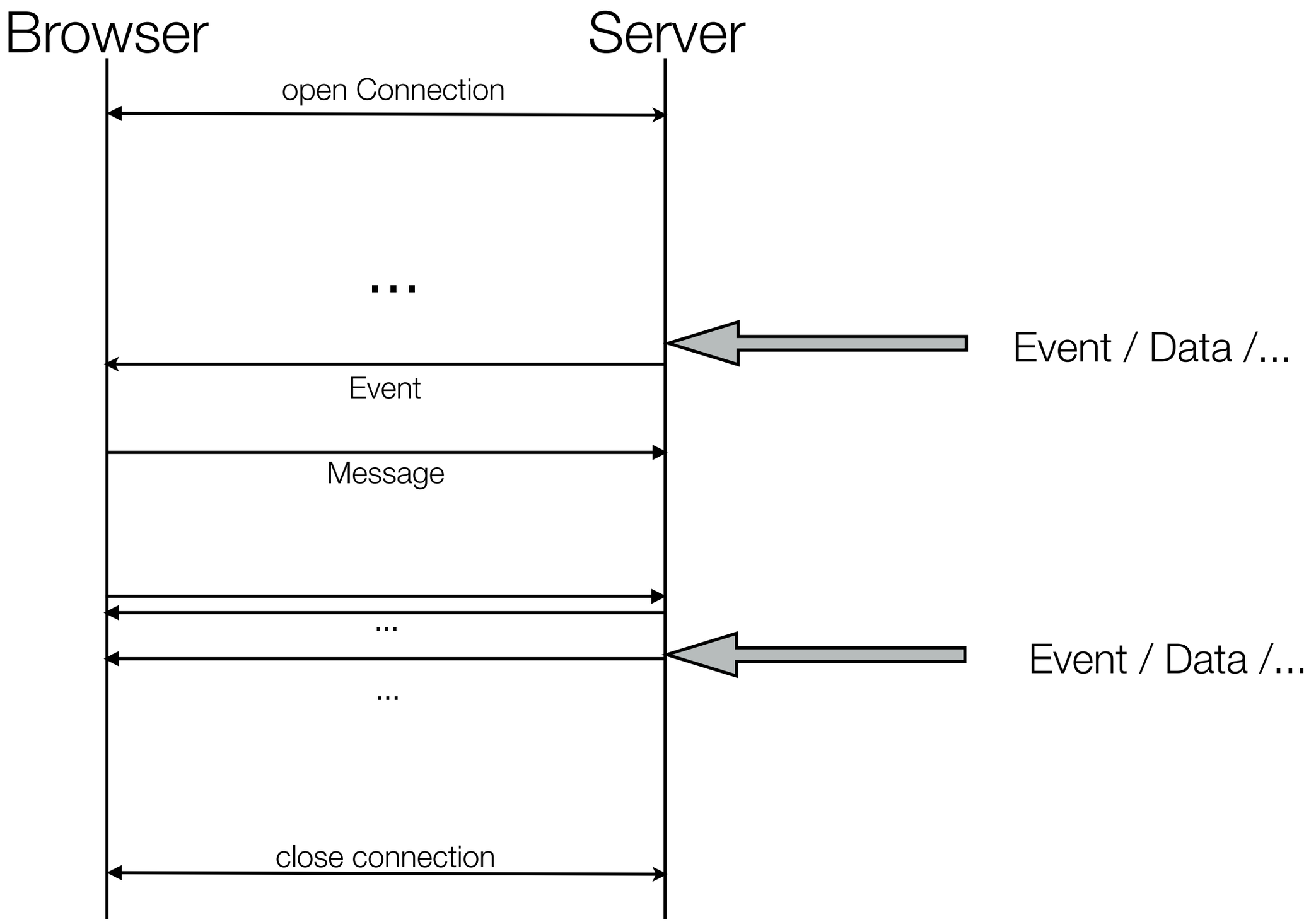
Les websockets

- Connexion persistante
- Communication lorsque nécessaire
- Communication bi-directionnelle
- Réduction du volume des headers (2 bytes!)

  `\0x00`Message`\0xff`
- Diminution de la latence car plus besoin d'établir de connexion
- Protocole spécifique (ws:// au lieu de http://)
- Un mode «secure» wss://

```
interface WebSocket {
        readonly attribute DOMString url;

        // ready state
        const unsigned short CONNECTING = 0;
        const unsigned short OPEN = 1;
        const unsigned short CLOSING = 2;
        const unsigned short CLOSED = 3;
        readonly attribute unsigned short readyState;
        readonly attribute unsigned long bufferedAmount;

        // networking
        attribute Function onopen;
        attribute Function onmessage;
        attribute Function onerror;
        attribute Function onclose;

        readonly attribute DOMString protocol;

        void send(in DOMString data);
        void close();
};
```

Websocket


Browser
Server
open Connection
...
Event / Data /...
Event
Message
...
Event / Data /...
...
close connection

Petite analyse

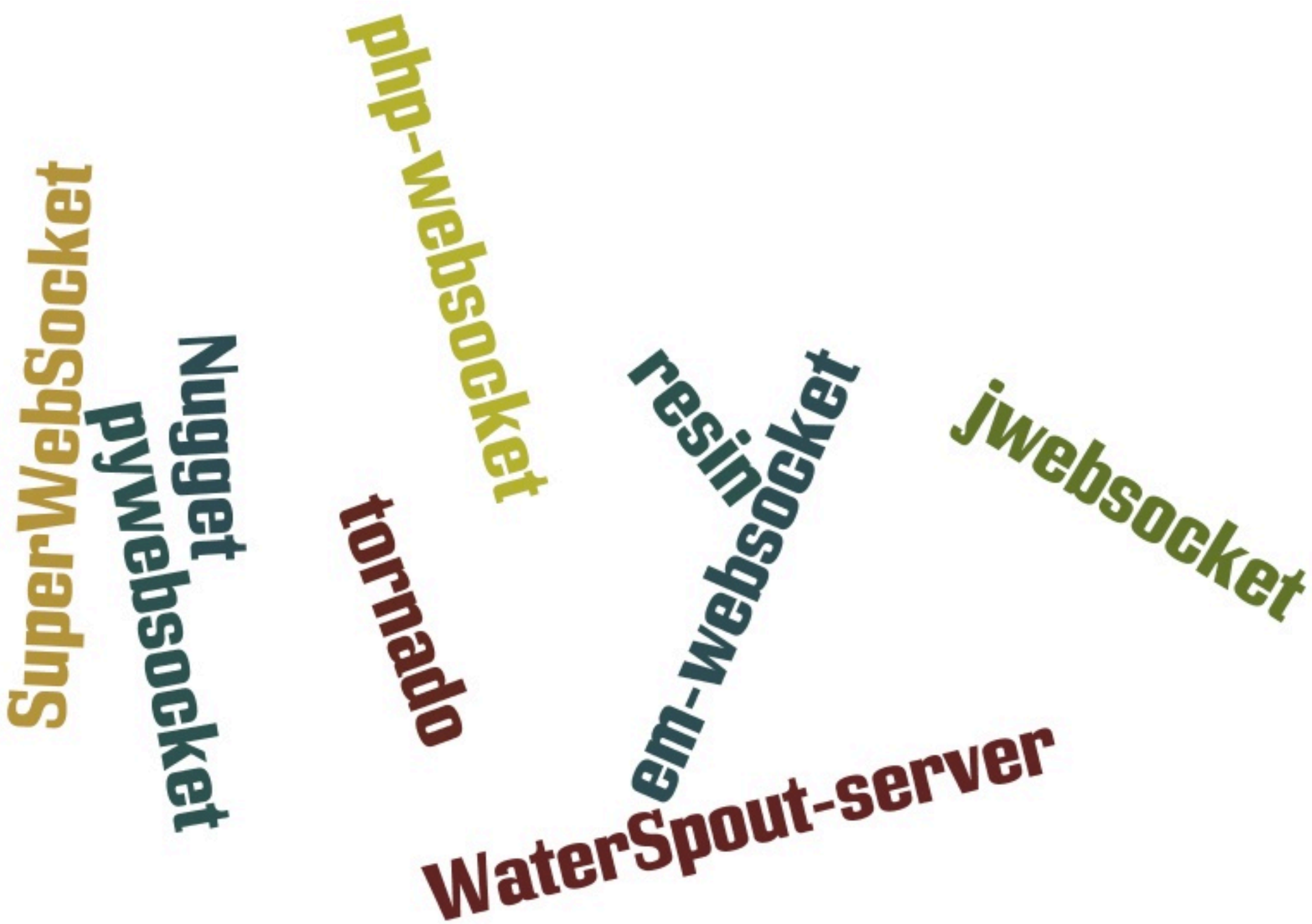
	Ajax	Websockets
Taille du header	~ 871 bytes	2 bytes
10’000 requêtes par seconde	8’710’000 bytes / sec	20’000 bytes / sec
	66Mbps	156 kbps

430

source: http://heig.ch/guba


Ca à l’air bien,
mais ça marche?
http://www.flickr.com/photos/15708236@N07/2754478731/

Côté serveur



Côté client

Mais

flash peut nous aider

pour une fois....

Un exemple simple

```
var ws = new WebSocket('ws://monServeur:monPort/monApplication');

ws.onopen = function (evt) {
    // Que faire lorsque la connexion est ouverte ?
}
ws.onclose = function(evt) {
    // Que faire lorsque la connexion est fermée ?
}
ws.onmessage = function(evt) {
    // Que faire lorsque je reçois un message ?
}
ws.onerror = function(evt) {
    // Que faire lorsque une erreur survient ?
}

ws.send('un message');
ws.send(JSON.stringify(unObjet));
ws.send(JSON.stringify({attr1:'value1', attr2: 'value2', ...}));
```

Un peu de pratique...

http://heig.ch/beretzu

Côté client

```
var wsUri = "ws://poulpe.heig-vd.ch:8080/echo";
var output;

$(document).ready(function() {
    output = $("div#log");
    initWebSocket();

        $('#send').click(function(){
            console.log('Message sent: ' + $('#message').val());
            writeToScreen("Websocket called");
            websocket.send($('#message').val());
        });
    });
```

```
function initWebSocket() {
    websocket = new WebSocket(wsUri);

    websocket.onopen = function (evt) {
        console.log('Websocket open.');
        $('#status').removeClass().addClass('online').html('online');
        writeToScreen('connected to '+ wsUri)
        $('#send, #message').attr('disabled', false);
    }
    websocket.onclose = function(evt) {
        console.log('Websocket closed.');
        $('#status').removeClass().addClass('offline').html('offline');
        $('#send, #message').attr('disabled', true);
        $('#connection').text('Connect');
        setTimeout(initWebSocket, 1000);
    }
    websocket.onmessage = function(evt) {
        console.log('Message received.')
        var data = JSON.parse(evt.data);
        writeToScreen('['+data.who+']: ' + data.msg);
        output.animate({scrollTop: output.attr('scrollHeight')})
    }
    websocket.onerror = function(evt) {
        console.error('Error')
        $('#status').removeClass().addClass('error').html('error');
    }
}

function writeToScreen(message) {
    var pre = document.createElement("pre");
    pre.style.wordWrap = "break-word";
    pre.innerHTML = message;
    output.append(pre);
}
```

Côté serveur (php) avec http://heig.ch/bodine

```
class EchoApplication extends Application
{
    private $clients = array();

    public function onConnect($client) {
        $this->clients[] = $client;
    }

    public function onDisconnect($client) {
        $key = array_search($client, $this->clients);
        if ($key) {
            unset($this->clients[$key]);
        }
    }

    public function onData($data, $client) {
        $obj = (object) null;
        $obj->who = $client->toString();
        $obj->msg = $data;
        foreach ($this->clients as $sendto) {
            $sendto->send(json_encode($obj));
        }
    }
}
```

Conclusion

- Les websockets peuvent être d'ores et déjà très utiles
- L'argument de gain (volume + latence) peut être très intéressant pour les gros fournisseurs
- La spécification n'est pas encore reconnue sur tous les navigateurs mais peut être contrecarrée grâce à flash
- Les websockets ne vont pas éradiquer les requêtes ajax!
- Il faut être en mesure de les faire cohabiter selon le contexte

Questions?