WebSockets

Matthieu Nicolas
Licence Pro CIASIE

Plan

- Pourquoi WebSockets?
- Utilisation

Pourquoi WebSockets?

WebSockets

Pourquoi WebSockets?

- Les requêtes HTTP permettent aux clients de contacter le serveur…
- .. mais c’est à l’initiative des clients

- Le serveur ne peut pas utiliser les requêtes HTTP pour contacter les clients de par lui-même

Mais existe des cas d’utilisations

- Envoi de notifications
- Envoi de nouveau contenu
  - Messages
  - Mise à jour d’un document collaboratif
  - Mise à jour de l’état d’un jeu en ligne

Solutions existantes - 1

- Polling
  - Client fait une requête HTTP à intervals réguliers
  - Le serveur envoie les nouveaux éléments, s’il y en a
- Limites
  - DDOS le serveur si interval trop court
  - Délai utilisateur si interval trop long

Solutions existantes - 2

- Long Poll
  - Client fait une requête HTTP
    - Le serveur ne répond pas tout de suite
- Maintient en vie la connexion jusqu'à la limite du `timeout`
  - Si `event`, le serveur répond et le client refait une nouvelle requête

Limites

- Solutions ad-hoc
- Lourd
  - Multiplie les requêtes HTTP
  - À chaque fois, refait l'`handshake` HTTP(S)
    - 1 à 3 RTTs

WebSockets

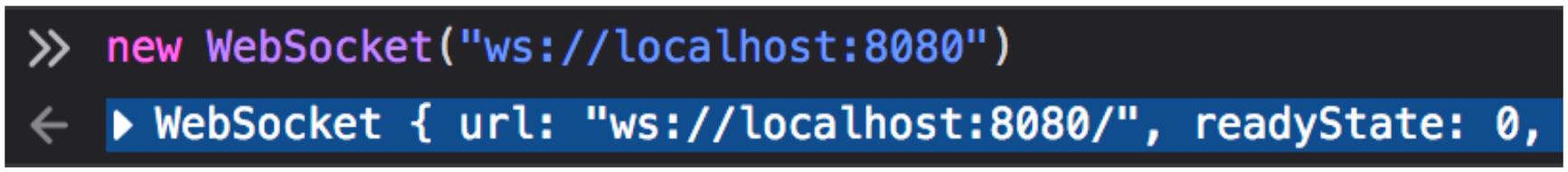
- Permet d'établir un lien de communication bi-directionnel entre navigateurs et serveurs
- Disponible dans les navigateurs récents
- Utilise une requête HTTP(S) qui est promue en WebSocket

Utilisation

WebSockets

Construction

- `new WebSocket(url)`
- Protocole
  - `ws://` pour connexion HTTP
  - `wss://` pour HTTPS



Events

- Met à disposition plusieurs types d'`events` pour réagir à l'activité de la WebSocket
  - `open`: lorsque la connexion est créée
  - `error`: lorsqu'une erreur survient (perte co)
  - `message`: lorsqu'on reçoit un message

Exemple d’events

- Rappel: utilise `addEventListener(event, callback)` pour réagir à un `event`

```
» const ws = new WebSocket("ws://localhost:8080")
  ws.addEventListener("open", function () {
    console.log("Yeah!")
  })
← undefined

  Yeah!
```

Envoi de message

- Dispose d'une méthode `send(data)`
  - `data` pouvant être du texte, du binaire…

```
» const ws = new WebSocket("ws://localhost:8080")
  ws.addEventListener("open", function () {
    ws.send(JSON.stringify({msg: "Hello"}))
  })
← undefined
```

TD

WebSockets

TD - 1

- Mettre en place un chat
  - Un salon de discussion unique
  - Les clients rejoignent le salon en accédant à la page
    - Peuvent envoyer des messages
    - Reçoivent les messages des autres clients

TD - 2

- Les messages ont le format suivant
  - `author`: une chaîne indiquant l'auteur du message
  - `content`: une chaîne correspondant au contenu du message
  - `date`: la date de génération du message

TD - 3

- Met à disposition un serveur HTTP et WebSockets
  - HTTP: Port 5000
  - WS: Port 8080

- Lien dispo sur Arche