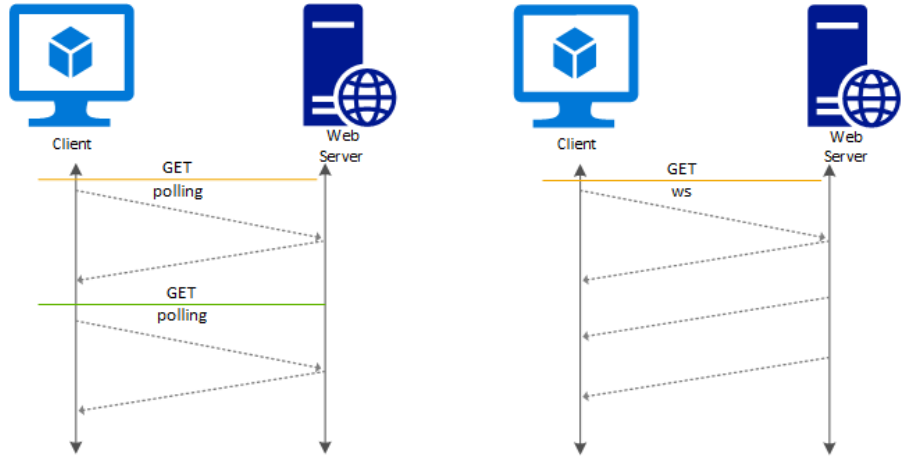
Les WebSockets et Angular

Animé par Mazen Gharbi

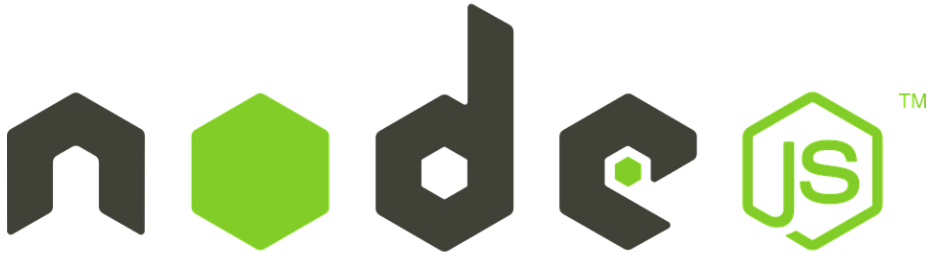
Introduction

- Les WebSockets sont apparus avec HTML5 ;
- Permettent de maintenir une connexion ouverte entre back et front ;
- Très simple à implémenter !



La partie NodeJS

- Côté serveur, il existe plusieurs librairies pour les WS
  - WS ;
  - Socket.io ;
- sont les deux plus populaires
- Pour la suite du cours, nous utiliserons WS



```
> npm install ws
```

Côté serveur

```
const wsServer = new WebSocket.Server({port: 8081});
wsServer.on('connection', ws => {
    ws.on('message', message => {
        const objMessage = JSON.parse(message);
        if (!objMessage.pseudo || !objMessage.content) {
            ws.send(JSON.stringify({type: 'error', message: 'Veuillez envoyer un pseudo et un message.'}));
            return;
        }

        ...

        wsServer.clients.forEach(function each(client) {
            if (client.readyState === WebSocket.OPEN) {
                client.send(JSON.stringify({type: 'newMessage', ...newMessage}));
            }
        });
    });
    ws.on('error', error => {
        console.error(error);
    });
    ws.on('close', ws => {
        console.log("Fin de communication");
    });
});
```

Côté Angular

- C’est là que ça devient intéressant !
- RxJS va encore une fois nous simplifier la vie

```
import {webSocket, WebSocketSubject} from 'rxjs/webSocket';
```

- Puis on enclenche la connexion

```
const wsConnection: WebSocketSubject = webSocket('ws://localhost:8081');
```

- Et c’est tout…
  - Oui, vraiment !

Ecouter et envoyer des messages

- Pour écouter de nouveaux messages, il suffit de s'abonner à l'observable :

```
connectionWebsocket.subscribe(
        dataFromServer => console.log("Nouveau message : " + dataFromServer),
        (err) => console.error(err),
        () => console.log("Connexion fermée")
);
```

- Et en envoyer :

```
connectionWebsocket.next({content: 'Salutations !', pseudo: 'Joe'});
```

- Le back et front communiquent avec des chaînes de caractères
  - A vous de parser la donnée avec JSON.parse ou JSON.stringify

Mettons cela en pratique

- Nous avons préparé un serveur pour vous permettre de connecter votre websocket

```
webSocket('ws://serveur-demo.macademia.fr:8081')
```

- L'objectif va être de créer un Chat
  - Vous, utilisateurs et développeurs, pourraient y envoyer et y visualiser tous les messages

- Un protocole de communication va être établi entre front et back

Protocole d'envoi

- Le serveur attend de votre part un objet formaté de cette manière :

```
interface DataFromClient {
  type: 'sendMessage' | 'getMessages';
}
```

- Pour sendMessage, il faudra ajouter 2 propriétés :

```
content: string;
pseudo: string;
```

Protocole de réception

▷ De la même manière, le serveur vous enverra régulièrement des informations :

```
interface Response {
  type: 'error' | 'newMessage' | 'allMessages';
  datas: any;
}
```

▷ Chaque message est constitué ainsi :

```
interface Message {
  pseudo: string;
  content: string;
  date: number;
}
```

A vos claviers

Questions?