WebSocket

Aula 1

Tiago Lage Payne de Pádua

❑ Introdução

- **WebSocket** é um protocolo, descrito na especificação RFC 6455, fornece uma maneira de trocar dados entre o navegador e o servidor através de uma conexão persistente;

- Depois que uma conexão de WebSocket é estabelecida, o cliente e o servidor podem enviar os dados uns aos outros;

- O WebSocket é especialmente excelente para serviços que exigem troca contínua de dados, por exemplo, jogos online, sistemas de negociação em tempo real e assim por diante;

❑ RFC 6455 - https://tools.ietf.org/html/rfc6455

"O protocolo WebSocket permite a comunicação bidirecional entre um cliente executando código não confiável em um ambiente controlado para um host remoto que tenha optado por comunicações desse código. O modelo de segurança usado é baseado em origem comumente usado pelos navegadores da web;

O protocolo consiste em um hand-shake de abertura seguido pelo envio de frames de mensagens, operando sobre TCP. O objetivo de esta tecnologia é fornecer um mecanismo para aplicativos baseados em navegador que precisam de comunicação bidirecional com servidores e que não depende da abertura de várias conexões HTTP (por exemplo, XMLHttpRequest ou <iframe> e long polling)."

❑ Exemplo

- Para abrir uma conexão precisamos criar um WebSocket usando o protocolo especial ws na URL:

```
let socket = new WebSocket("ws://exemplo.com");
```

- Há também um `wss://` que é um protocolo criptografado. É como o HTTPS para WebSockets;

❑ Eventos

- Depois que o socket é criado podemos ouvir os eventos nele. Existem um total de 4 eventos:

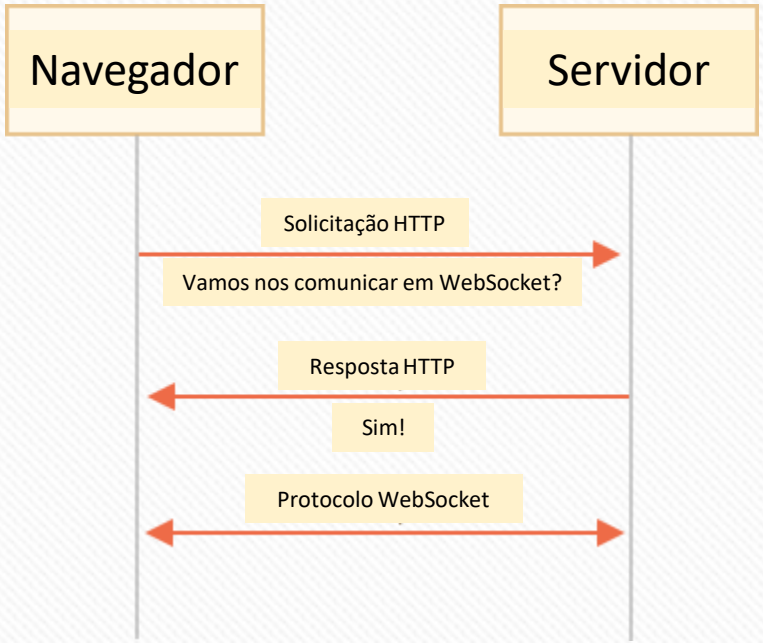
  - `open` - conexão estabelecida;
  - `message` - dados recebidos;
  - `error` - erro de WebSocket;
  - `close` - conexão fechada;

- Para enviar dados usamos `socket.send(data);`

```
let socket = new WebSocket("wss://exemplo.com/websocket");
socket.onopen = function(e) {
  alert("[open] Conexão estabelecida, enviando dados ao servidor");
  socket.send("Meu nome é Tiago");
};
socket.onmessage = function(event) {
  alert(`[message] Dados recebidos do servidor: ${event.data}`);
};
socket.onclose = function(event) {
  if (event.wasClean) {
    alert(`[close] Conexão fechada corretamente, código=${event.code} razão=${event.reason}`);
  } else {
    // Ex.: Processo do servidor encerrou ou rede caiu
    // event.code é geralmente 1006 neste caso
    alert('[close] Conexão encerrada com erro');
  }
};
socket.onerror = function(error) {
  alert(`[error] ${error.message}`);
};
```

❑ Abrindo um WebSocket

- Quando new WebSocket é criado, inicia um handshake HTTP (HTTPS para wss://);
- O navegador pergunta ao servidor: "Você suporta WebSocket?" E se o servidor disser "sim", a conversa continua no protocolo WebSocket, que não é HTTP;



❑ Abrindo um WebSocket

Veja um exemplo de solicitação de navegador para:

```
new WebSocket("wss://exemplo.com.br/chat"):

GET /chat
Host: exemplo.com.br
Origin: https://exemplo.com.br
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Key: Iv8io/9s+lYFgZWcXczP8Q==
Sec-WebSocket-Version: 13
```

❑ Abrindo um WebSocket

- `Origin`– a origem da página do cliente. O WebSocket é de cross-origin por natureza. Não há cabeçalhos especiais ou outras limitações. Os servidores antigos são incapazes de lidar com o WebSocket de qualquer maneira, portanto, não há problemas de compatibilidade. Mas o cabeçalho `Origin` é importante, pois permite que o servidor decida se deve ou não falar WebSocket com este site;

- `Connection: Upgrade` - sinaliza que o cliente gostaria de alterar o protocolo;

❑ Abrindo um WebSocket

- `Upgrade: websocket` - o protocolo solicitado é "websocket";
- `Sec-WebSocket-Key` - uma chave gerada aleatoriamente pelo navegador para segurança;
- `Sec-WebSocket-Version` - Versão do protocolo WebSocket, 13 é a atual;

❑ Abrindo um WebSocket

- Se o servidor concordar em mudar para o WebSocket, ele deve enviar o código de resposta 101:

```
101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: hsBlbuDTkk24srzEOTBUlZAlC2g=
```

- Aqui `Sec-WebSocket-Accept` é o `Sec-WebSocket-Key` recodificado usando um algoritmo especial. O navegador usa para garantir que a resposta corresponda à solicitação;

- Posteriormente, os dados são transferidos usando o protocolo WebSocket;

❑ Dados WebSocket

- A comunicação WebSocket consiste em “frames” que podem ser enviados de ambos os lados:

  - Frames de texto - contêm dados de texto que as partes enviam entre si;
  - Frames de dados binários - contém dados binários que as partes enviam entre si.
  - Frames de ping/pong - são usados para verificar a conexão, enviados pelo servidor, o navegador responde a eles automaticamente;
  - Frame de conexão fechada - e alguns outros frames de serviço;

❑ Dados WebSocket

- No navegador, nos preocupamos apenas com frames de texto ou binários;
- WebSocket `.send()` pode enviar texto ou dados binários;
- Os dados textuais sempre vêm como string. Para receber dados binários, podemos escolher entre os formatos Blob e ArrayBuffer;

```
socket.bufferType = "arraybuffer";
socket.onmessage = (event) => {
  // event.data é uma string ou arraybuffer (se binário)
};
```

❑ Encerramento da Conexão

- Normalmente, quando uma parte deseja fechar a conexão (o navegador e o servidor têm direitos iguais), eles enviam um "frame de fechamento de conexão" com um código numérico e um motivo como texto;

- O método é:

```
socket.close([código], [razão]);
```

❑ Encerramento da Conexão

- Em seguida, a outra parte no gerenciador do evento `close` pode obter o código e o motivo, por exemplo:

```
// um lado:
socket.close(1000, "Tarefa concluída");

// Outro lado:
socket.onclose = event => {
  // event.code === 1000
  // event.reason === "Tarefa concluída"
  // event.wasClean === true (fechamento limpo)
};
```

❑ Encerramento da Conexão

O `código de fechamento` não é um número qualquer, mas um código de fechamento especial do WebSocket:

- `1000` - o padrão, fechamento normal;
- `1006` - não se pode definir esse código manualmente, indica que a conexão foi interrompida (sem frame de fechamento);

Existem outros códigos como:

- `1001` - o servidor está sendo desligado ou um navegador saiu da página;
- `1009` - a mensagem é muito grande para processar;
- `1011` - erro inesperado no servidor;

...e assim por diante.

❑ Estado da Conexão

Para obter o estado da conexão, há também há a propriedade `socket.readyState` com valores:

0 - CONNECTING: a conexão ainda não foi estabelecida;

1 - OPEN: comunicando;

2 - CLOSING: a conexão está sendo fechada;

3 - CLOSED: a conexão está fechada;

Ano: 2019 Banca: UFC Órgão: UFC Prova: UFC - 2019 - UFC - Técnico de Tecnologia da Informação - Desenvolvimento de Sistemas

Sobre WebSockets, assinale a alternativa correta:

a) WebSockets utiliza AJAX para possibilitar a comunicação RESTful.
b) WebSockets e Sockets de rede são sinônimos, representando o mesmo protocolo e especificação.
c) De acordo com a RFC 6455 que descreve o protocolo WebSocket, não é possível utilizar WebSockets em ambientes que possuam um servidor proxy.
d) O handshake WebSocket usa o cabeçalho HTTP Upgrade para mudar do protocolo HTTP para o protocolo WebSocket, com uma requisição de Upgrade.
e) Por padrão, o protocolo WebSocket usa a porta 8080 para conexões WebSocket comuns e a porta 443 para conexões WebSocket encapsuladas por TLS (Transport Layer Security).

❑ Abrindo um WebSocket

Veja um exemplo de solicitação de navegador para:

```
new WebSocket("wss://exemplo.com.br/chat"):

GET /chat
Host: exemplo.com.br
Origin: https://exemplo.com.br
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Key: Iv8io/9s+lYFgZWcXczP8Q==
Sec-WebSocket-Version: 13
```

Ano: 2019 Banca: UFC Órgão: UFC Prova: UFC - 2019 - UFC - Técnico de Tecnologia da Informação - Desenvolvimento de Sistemas

Sobre WebSockets, assinale a alternativa correta:

a) WebSockets utiliza AJAX para possibilitar a comunicação RESTful.
b) WebSockets e Sockets de rede são sinônimos, representando o mesmo protocolo e especificação.
c) De acordo com a RFC 6455 que descreve o protocolo WebSocket, não é possível utilizar WebSockets em ambientes que possuam um servidor proxy.
d) O handshake WebSocket usa o cabeçalho HTTP Upgrade para mudar do protocolo HTTP para o protocolo WebSocket, com uma requisição de Upgrade.
e) Por padrão, o protocolo WebSocket usa a porta 8080 para conexões WebSocket comuns e a porta 443 para conexões WebSocket encapsuladas por TLS (Transport Layer Security).