XMPP Web-Programmierung

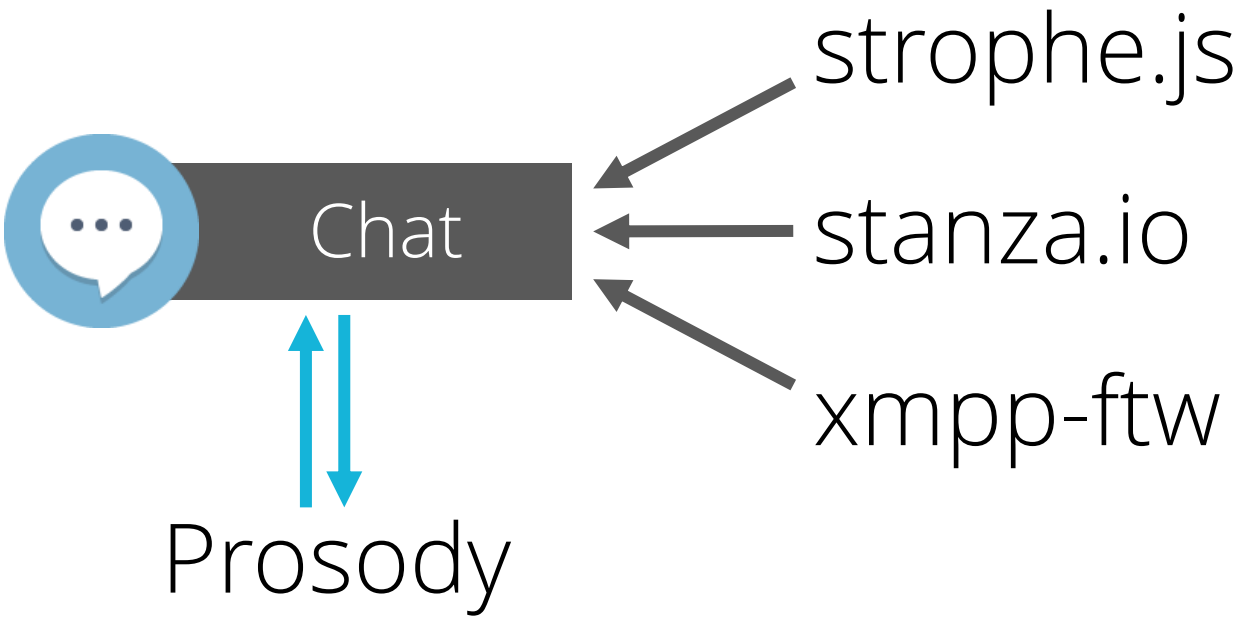
Philipp Heisig
Robert Starke
Kai Pfrötschner (ausgeschieden)

Übung Multimediakommunikation Sommersemester 2015

1	2	3
Aufgaben-stellung	Organisation	Demo Video

4	5	6
Vergleich	Auswertung	Fragen

1 Aufgabenstellung



Benutzbarkeit der API

Code-Qualität

Wiederver-wendbarkeit

Dokumentation

2 Organisation

Kaltstart mit Prosody

Zeitaufwendige Einarbeitung

Austritt von Kai (XMPP-FTW)

Vergleiche aller **3** Frameworks

Implementierung von **2** Frameworks

3

Demo Video

4 Vergleich

Strophe.js

```
function ( jid, password ) {
    client = new Strophe.Connection( 'websocket-url' );
    client.connect( jid, password, callback );
},
```

```
callback: function ( status ) {
    if ( status === Strophe.Status.CONNECTED ) {
        client.roster.requestRoster( rosterCallback );
        client.addHandler( onMessageHandler, null, 'message, );
        client.addHandler(
            onChatStateHandler,
            Strophe.NS.CHATSTATES,
            'message,
        );
        client.send( $pres() );
    }
}
```

Strophe.js

Schwer lesbar

Payload in XML

Funktionalitäten in Plugins gekapselt

Strophe.js

Schwer lesbar

Payload in XML

Funktionalitäten in Plugins gekapselt

Generischer Aufbau führt zu spezifischem Code

Abhängigkeiten durch Verkettung der Handler und Callbacks

Plugins schaffen zusätzliche Abhängigkeiten

Strophe.js

Schwer lesbar

Payload in XML

Funktionalitäten in Plugins gekapselt

Generischer Aufbau führt zu spezifischem Code

Abhängigkeiten durch Verkettung der Handler und Callbacks

Plugins schaffen zusätzliche Abhängigkeiten

Frameworkeigene Dokumentation dürtig

Viele Beispiele und Hilfestellungen in Foren und Google Groups

Stanza.io

```
XMPP.createClient({
    jid: jid,
    password: password,
    wsURL: ,websocket-url'
});
initListeners();
client.connect();
```

```
 client.on( 'session:started', function () {
    client.getRoster( function ( err, response ) {
        client.sendPresence();
        var roster = response.roster;
    });
});
```

Stanza.io

Gut lesbar

Payload in JSON

Funktionalitäten vorimplementiert und verwendungsbereit

Stanza.io

Gut lesbar

Payload in JSON

Funktionalitäten vorimplementiert

Implizite Vorgaben und klare Trennung der Funktionen

Spezifisches Event Handling

Node, Browserify und Grunt Workflow

Stanza.io

Gut lesbar

Payload in JSON

Funktionalitäten vorimplementiert und verwendungsbereit

Implizite Vorgaben und klare Trennung der Funktionen

Spezifisches Event Handling

Node, Browserify und Grunt Workflow

Frameworkeigene Dokumentation zählt Funktionen nur auf

Wenig gutes Material für den Einstieg

xmpp-ftw

Gut lesbar

Payload in JSON

Konsistenz durch gleichbleibende Erweiterung von Socket-Events

xmpp-ftw

Gut lesbar

Payload in JSON

Konsistenz durch gleichbleibende Erweiterung von Socket-Events

Erweitert Websocket-Verbindung durch XMPP-Events

Völlig unabhängig vom Websocket-Framework

Node, Browserify und Grunt Workflow

xmpp-ftw

Gut lesbar

Payload in JSON

Konsistenz durch gleichbleibende Erweiterung von Socket-Events

Erweitert Websocket-Verbindung durch XMPP-Events

Völlig unabhängig vom Websocket-Framework

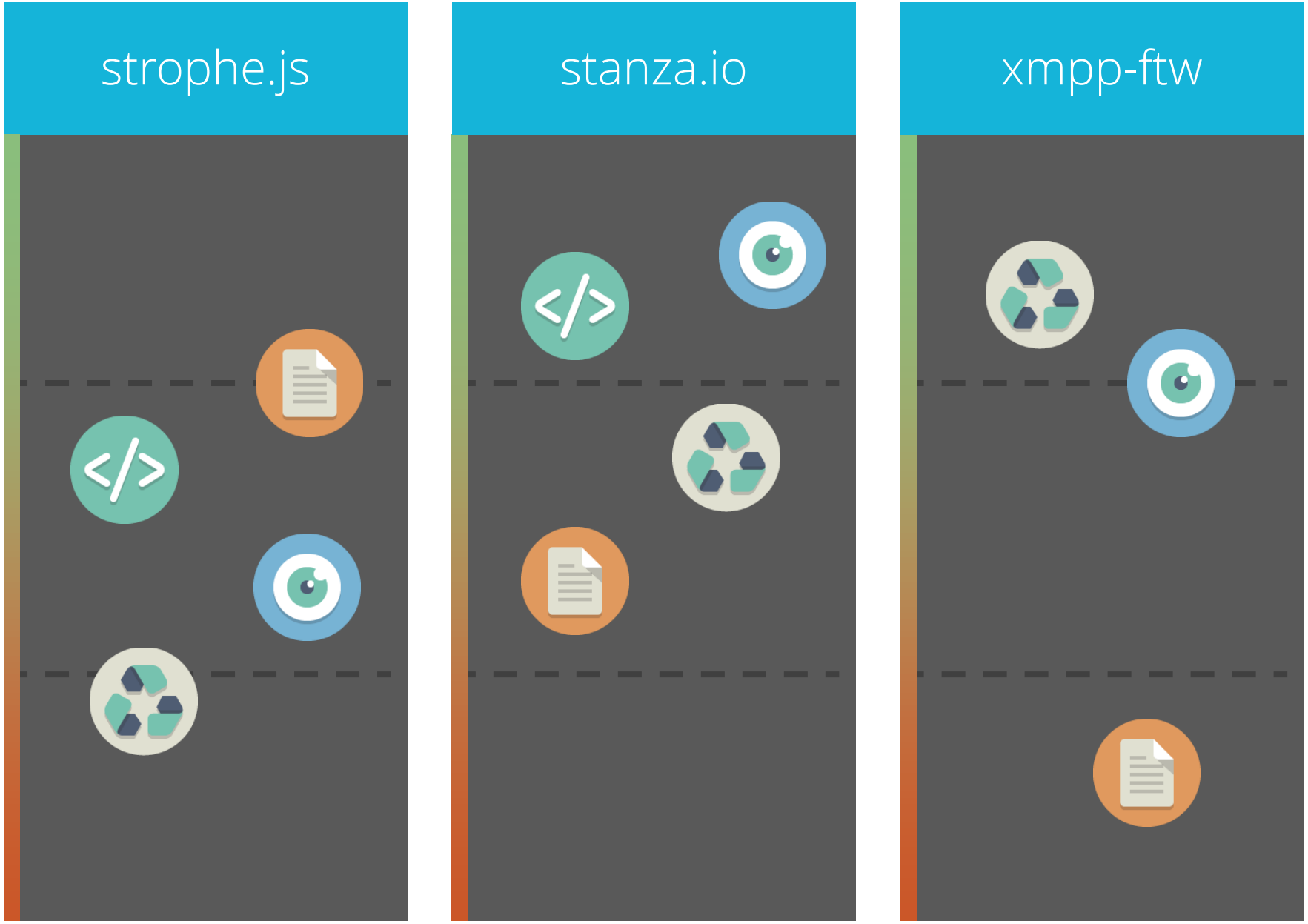
Node, Browserify und Grunt Workflow

Frameworkeigene Dokumentation unübersichtlich

Kein Material für den Einstieg

Web-Demo ist ein Witz

5 Auswertung


strophe.js
stanza.io
xmpp-ftw

6

Fragen

Philipp Heisig
Robert Starke
Kai Pfrötschner (ausgeschieden)

Übung Multimediakommunikation Sommersemester 2015