„**Socket.IO** aims to make **realtime apps** possible **in every browser** and mobile device, blurring the differences between the different transport mechanisms. **It's care-free realtime 100% in JavaScript.**“

Scratch Free
¡No Raya!
AJAX
With BLEACH
Powder Cleanser
LIMPIADOR EN POLVO CON BLANQUEADOR
NET WT 14 OZ (396g)
Easy Rinse Formula
Fórmula de Enjuague Fácil

Einschränkungen bei Ajax Requests?

Einschränkungen bei Ajax Requests?

Einseitige Kommunikation

Die Kommunikation geht immer vom Client (Browser) aus

Einschränkungen bei Ajax Requests?

Normaler HTTP Request

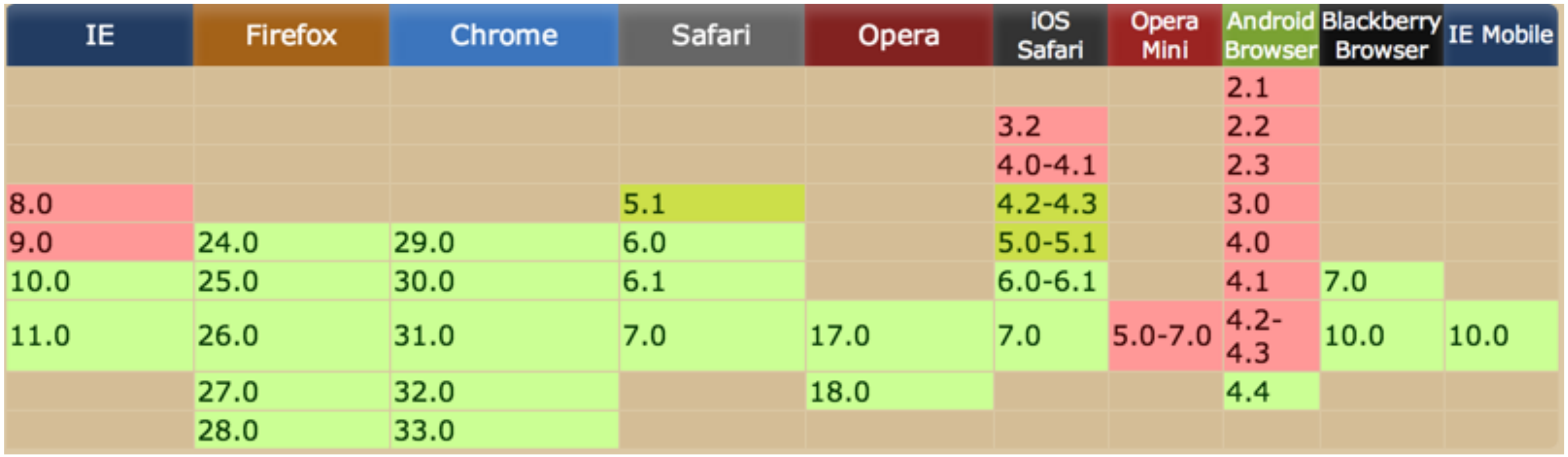
Ein Ajax Request ist ein normaler HTTP Request der vom Browser asynchron ausgeführt wird.

Lösung

WebSockets

Can I use WebSockets?

Can I use WebSockets?



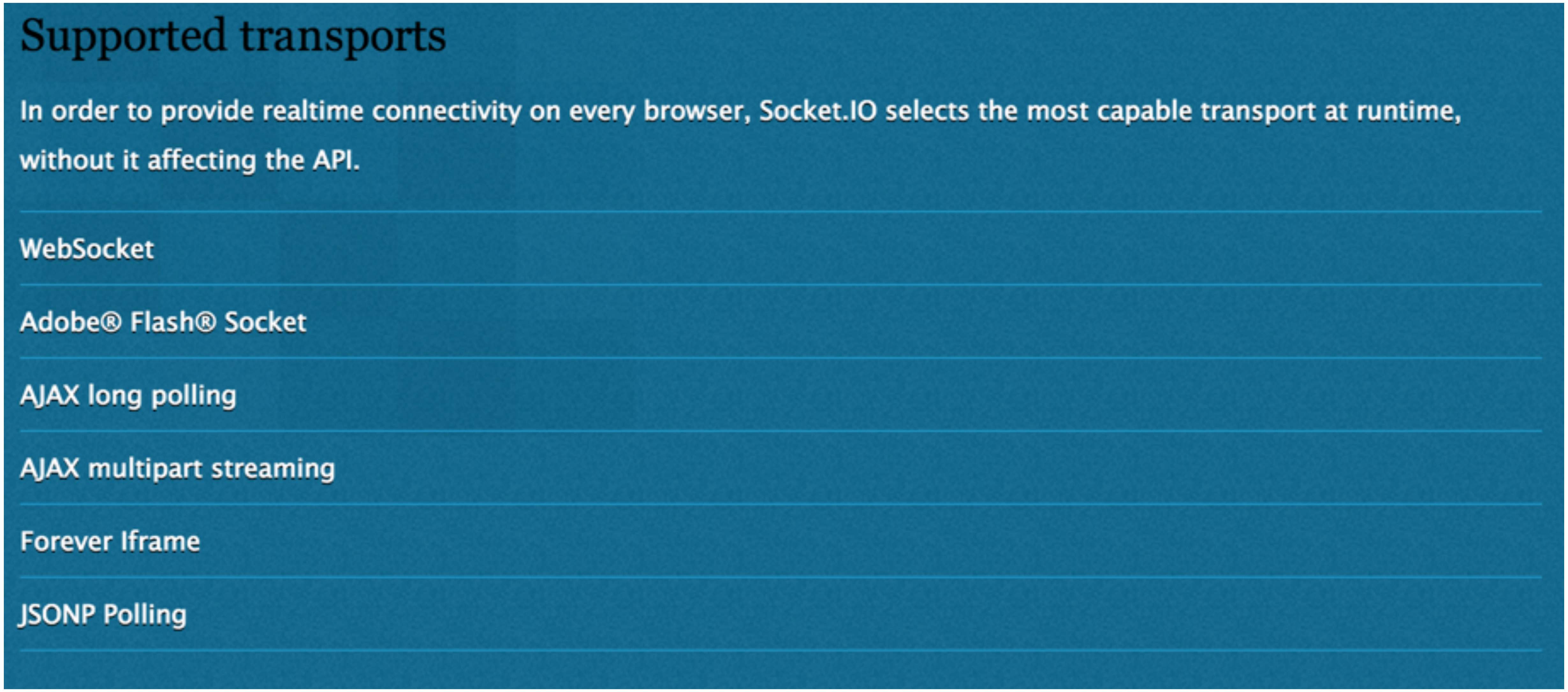
IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
							2.1		
					3.2		2.2		
					4.0-4.1		2.3		
8.0			5.1		4.2-4.3		3.0		
9.0	24.0	29.0	6.0		5.0-5.1		4.0		
10.0	25.0	30.0	6.1		6.0-6.1		4.1	7.0	
11.0	26.0	31.0	7.0	17.0	7.0	5.0-7.0	4.2-4.3	10.0	10.0
	27.0	32.0		18.0			4.4		
	28.0	33.0							

Ja und **Nein**

Say hello to

Socket.IO

Cross Browser Fallbacks



Supported transports

In order to provide realtime connectivity on every browser, Socket.IO selects the most capable transport at runtime, without it affecting the API.

- WebSocket
- Adobe® Flash® Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

Internet Explorer 5.5+, Safari 3+, Google Chrome 4+, Firefox 3+, Opera 10.61+

Features

- Sending and receiving events
- Storing data associated to a client
- Restricting yourself to a namespace
- Sending volatile messages
- Sending and getting data (acknowledgements)
- Broadcasting messages

Sending and receiving events

```
var io = require('socket.io').listen(80);
io.sockets.on('connection', function (socket)
  io.sockets.emit('this', {
    will: 'be received by everyone'
  });

  socket.on('private message', function (from, msg) {
    console.log(from, msg);
  });
});
```

Storing data associated to a client

```
var io = require('socket.io').listen(80);
io.sockets.on('connection', function (socket) {
  socket.on('set nickname', function (name) {
    socket.set('nickname', name, function () {
      socket.emit('ready');
    });
  });

  socket.on('msg', function () {
    socket.get('nickname', function (err, name) {
      console.log('Chat message by ', name);
    });
  });
});
```

Restricting yourself to a namespace

```
// server
var io = require('socket.io').listen(80);
var news = io.of('/news').on('connection', function (socket) {
  socket.emit('item', { news: 'item' });
});

// client
<script>
  var news = io.connect('http://localhost/news');
  news.on('news', function () {
    news.emit('woot');
  });
</script>
```

Sending volatile messages

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  var tweets = setInterval(function () {
    getBieberTweet(function (tweet) {
      socket.volatile.emit('bieber tweet', tweet);
    });
  }, 100);

  socket.on('disconnect', function () {
    clearInterval(tweets);
  });
});
```

Sending and getting data (acknowledgements)

```
// server
var io = require('socket.io').listen(80);
io.sockets.on('connection', function (socket) {
  socket.on('ferret', function (name, fn) {
    fn('woot');
  });
});

// client
<script>
  var socket = io.connect('http://localhost');
  socket.on('connect', function () {
    socket.emit('ferret', 'tobi', function (data) {
      console.log(data); // data will be 'woot'
    });
  });
</script>
```

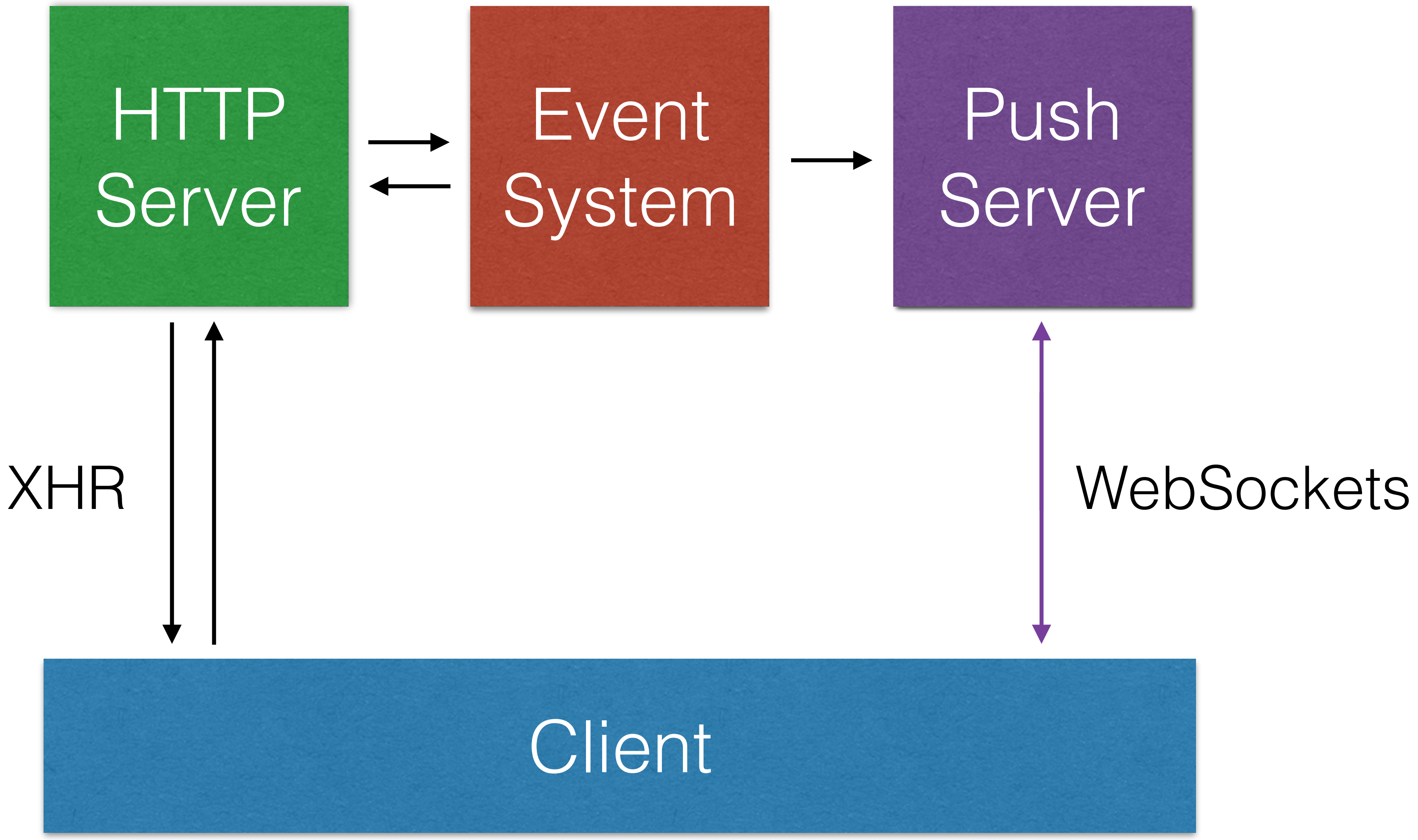
Broadcasting messages

```
var io = require('socket.io').listen(80);

io.sockets.on('connection', function (socket) {
  socket.broadcast.emit('user connected');
});
```

Anwendungsbeispiel

Travian 5 Architektur



Aufgabe: Chat