

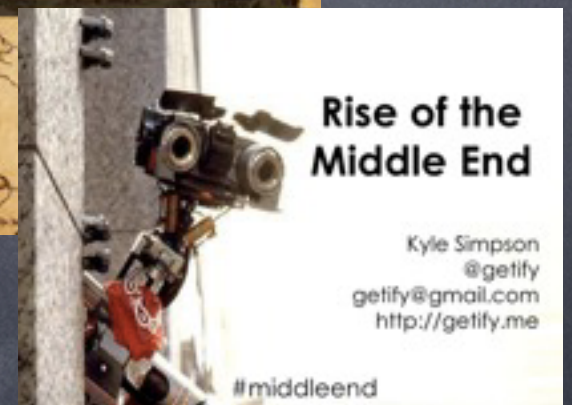
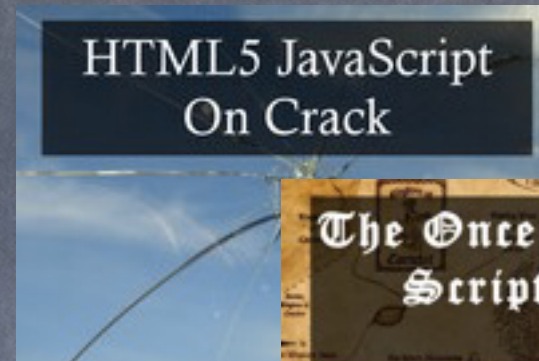
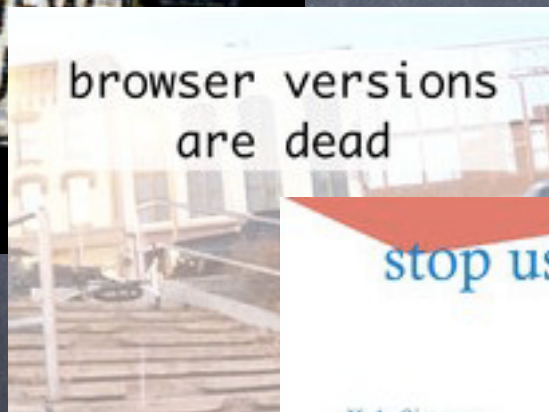
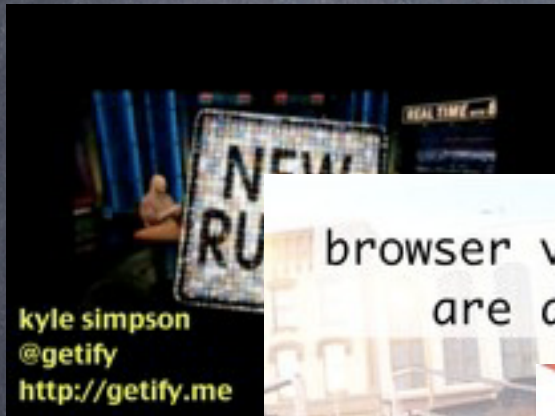
Kyle Simpson
@getify
<http://getify.me>

- LABjs
- grips
- asynquence

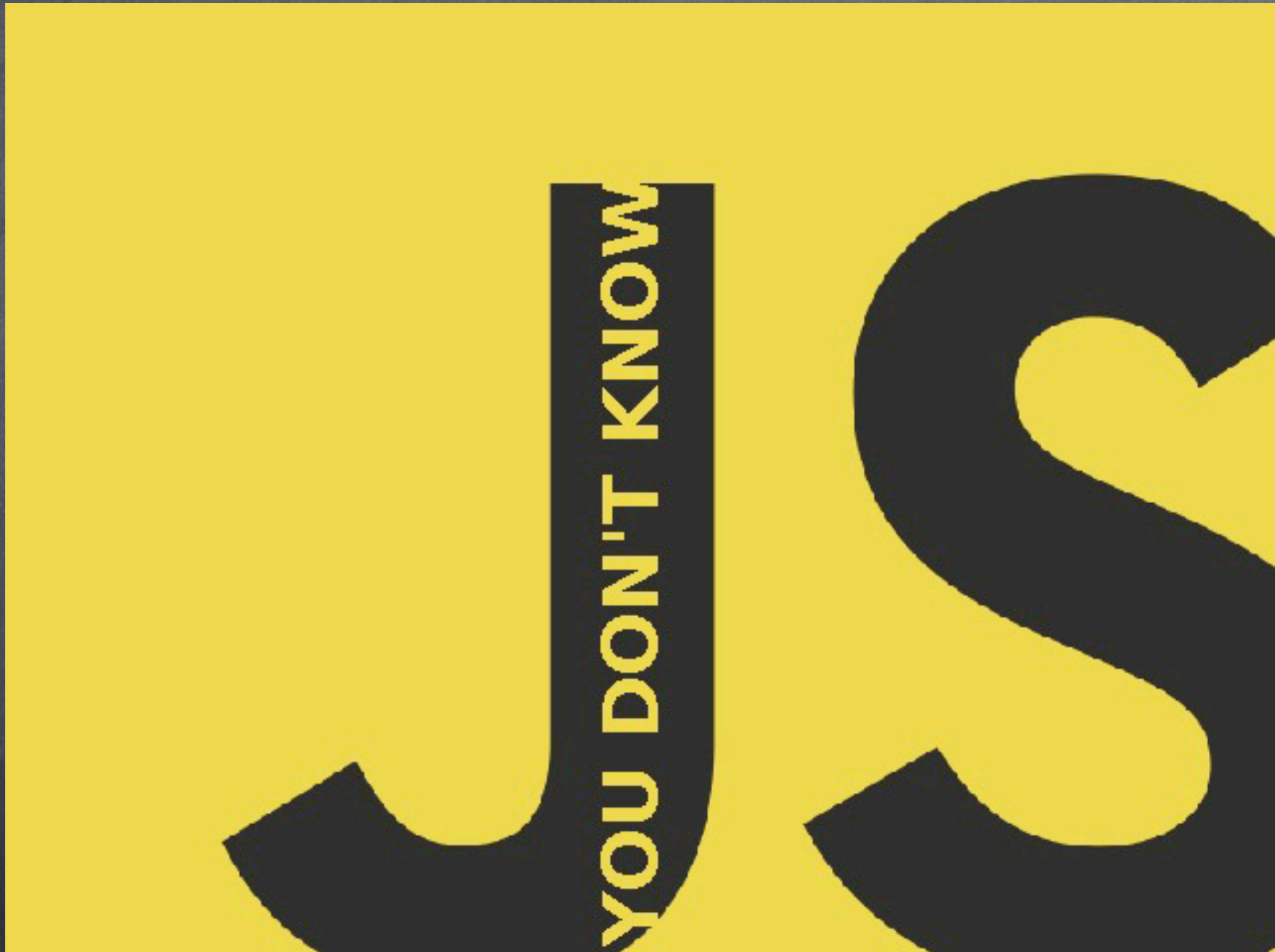
Kyle Simpson

@getify

<http://getify.me>



<http://speakerdeck.com/getify>



<http://YouDontKnowJS.com>

Into node.js

HTML



WebRTC

setup

<http://nodejs.org/download/>

you need node.js

Agenda

- HTML5
- node.js
- Web Sockets (socket.io)
- WebRTC

HTML5

facades

<https://github.com/getify/h5ive>

storage


```
1 var keep = h5.storage();
2 var session = h5.storage({ expires: "session" });
3 var temp = h5.storage({ expires: 5*60*1000 }); // 5 minutes
4
5 keep
6 .save({
7     prefs: { /* ... */ }
8 });
9
10 session
11 .save({
12     session_id: 123456,
13     foo: "bar baz"
14 })
15 .discard(["foo"]);
16
17 temp.save({
18     active_login: session.get("session_id") // only keep this ar
19 });
```


<canvas>


```
1  var cnv = h5.canvas({
2      width: 500,
3      height: 500,
4      matchDimensions: true // make the CSS dimensions match the attribute dimen
5  });
6
7  document.body.appendChild(cnv.element());
8
9  cnv
10 .clear()
11 .setStyles({
12     alpha: 0.8,
13     stroke: {
14         width: 3,
15         color: "#fc6"
16     },
17     fill: {
18         color: "#009"
19     }
20 })
21 .startPath(75,100)
22 .defineSegments([
23     { lineTo: [200,200] },
24     { lineTo: [100,50] }
25 ])
26 .endPath({
27     close: true,
28     stroke: true,
29     fill: true
```


getUserMedia,
<video>


```
1 h5
2 .userMedia({
3     video: true
4 })
5 .stream(function(src){
6     var video = document.getElementById("mycam");
7     video.src = src;
8     video.play();
9 })
10 .failed(function(){
11     alert("Access to the media failed.");
12 });
```


requestAnimationFrame


```
1  var aFrame = h5.animationFrame,
2      body = document.body,
3      text, id1, id2, id3
4  ;
5
6  id1 = aFrame.queue(function(){
7      // canceled, won't ever get called
8      text = document.createTextNode("##");
9      body.appendChild(text);
10 });
11 id2 = aFrame.queueAfter(function(){
12     // canceled, won't ever get called
13     text = document.createTextNode("!!");
14     body.appendChild(text);
15 });
16 id3 = aFrame.queueAfter(function(){
17     // canceled, won't ever get called
18     text = document.createTextNode("$$");
19     body.appendChild(text);
20 });
21
22 aFrame.queueAfter(function(){
23     text = document.createTextNode("third.");
24     body.appendChild(text);
25 });
26 aFrame.queue(function(){
27     aFrame.cancel(id2); // still time to cancel something before the
```


WebSockets


```
55     socket.on("game_session_invalid", function(data) {
56         self.postMessage({error: "game_session_invalid"});
57     });
58
59     socket.on("reset_player", function() {
60         self.postMessage({reset_player: true});
61     });
62 }
63
64 if (!game_session_started) {
65     game_session_started = true;
66     socket.emit("establish_game_session", {game_session_id: game
67 }
68 }
```

socket.io

node.js

“middle end”

Hello World


```
1 function handleHTTP(req,res) {
2     if (req.method == "GET") {
3         if (req.url == "/") {
4             res.writeHead(200,{ "Content-type": "text/plain" });
5             res.end("Hello World: " + Math.random());
6         }
7         else {
8             res.writeHead(403);
9             res.end();
10        }
11    }
12    else {
13        res.writeHead(403);
14        res.end();
15    }
16 }
17
18 var http = require("http"),
19     httpserv = http.createServer(handleHTTP),
20     port = 8006,
21     host = "127.0.0.1";
22
23 httpserv.listen(port, host);
```


async flow

<https://github.com/getify/asyncquence>


```
1 ASQ()  
2 .gate(  
3     function(done){ setTimeout(done,100); },  
4     function(done){ setTimeout(done,200); },  
5     function(done){ setTimeout(done,300); }  
6 )  
7 .then(function(){  
8     alert("All tasks are complete, and that only took ~300ms, not 600ms!");  
9 });
```

```
1 ASQ()  
2 .then(function(done){ setTimeout(done,100); })  
3 .then(function(done){ setTimeout(done,200); })  
4 .then(function(done){ setTimeout(done,300); })  
5 .then(function(){  
6     alert("All tasks are complete, and that took ~600ms!");  
7 });
```

```
1 ASQ()  
2 .then(function(done){ setTimeout(done,100); })  
3 .gate(  
4     function(done){ setTimeout(done,200); },  
5     function(done){ setTimeout(done,300); }  
6 )  
7 .then(function(){  
8     alert("All tasks are complete, and that took ~400ms!");  
9 });
```


WebSockets

socket.io

<http://socket.io/>

Hello World


```
46     io = require("socket.io").listen(httpserv)
47 ;
48
49 // configure socket.io
50 io.configure(function(){
51     io.enable("browser client minification"); // send minified
52     io.enable("browser client etag"); // apply etag caching
53     io.set("log level", 1); // reduce logging
54     io.set("transports", [
55         "websocket",
56         "xhr-polling",
57         "jsonp-polling"
58     ]);
59 });
60
61 httpserv.listen(port, host);
62
63 io.on("connection", connection);
```

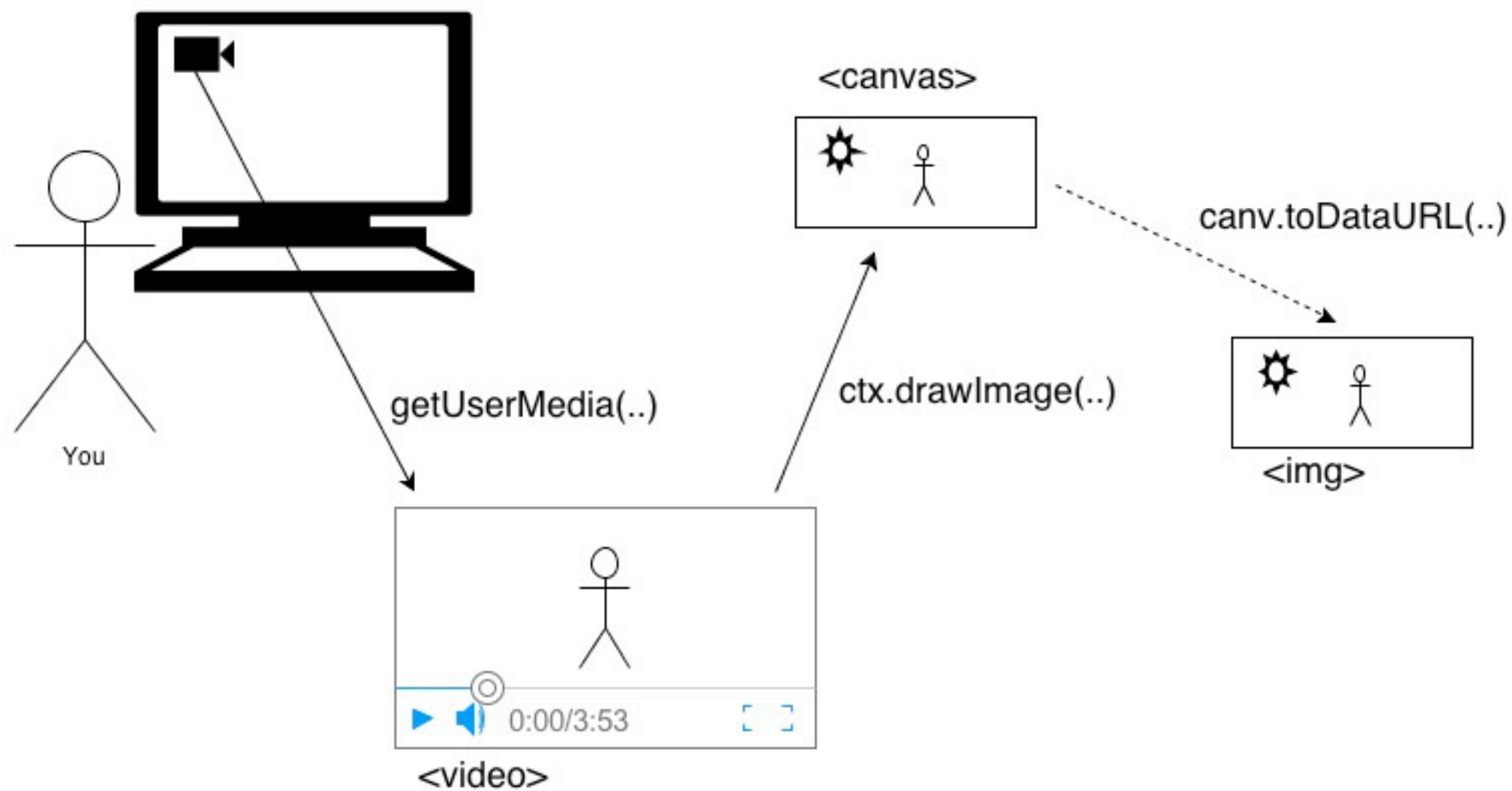


```
21 function connection(socket) {  
22  
23     function disconnect() {  
24         console.log("disconnected");  
25     }  
26  
27     socket.on("disconnect",disconnect);  
28  
29     var intv = setInterval(function(){  
30         socket.emit("hello",Math.random());  
31     },1000);  
32 }
```


WebRTC

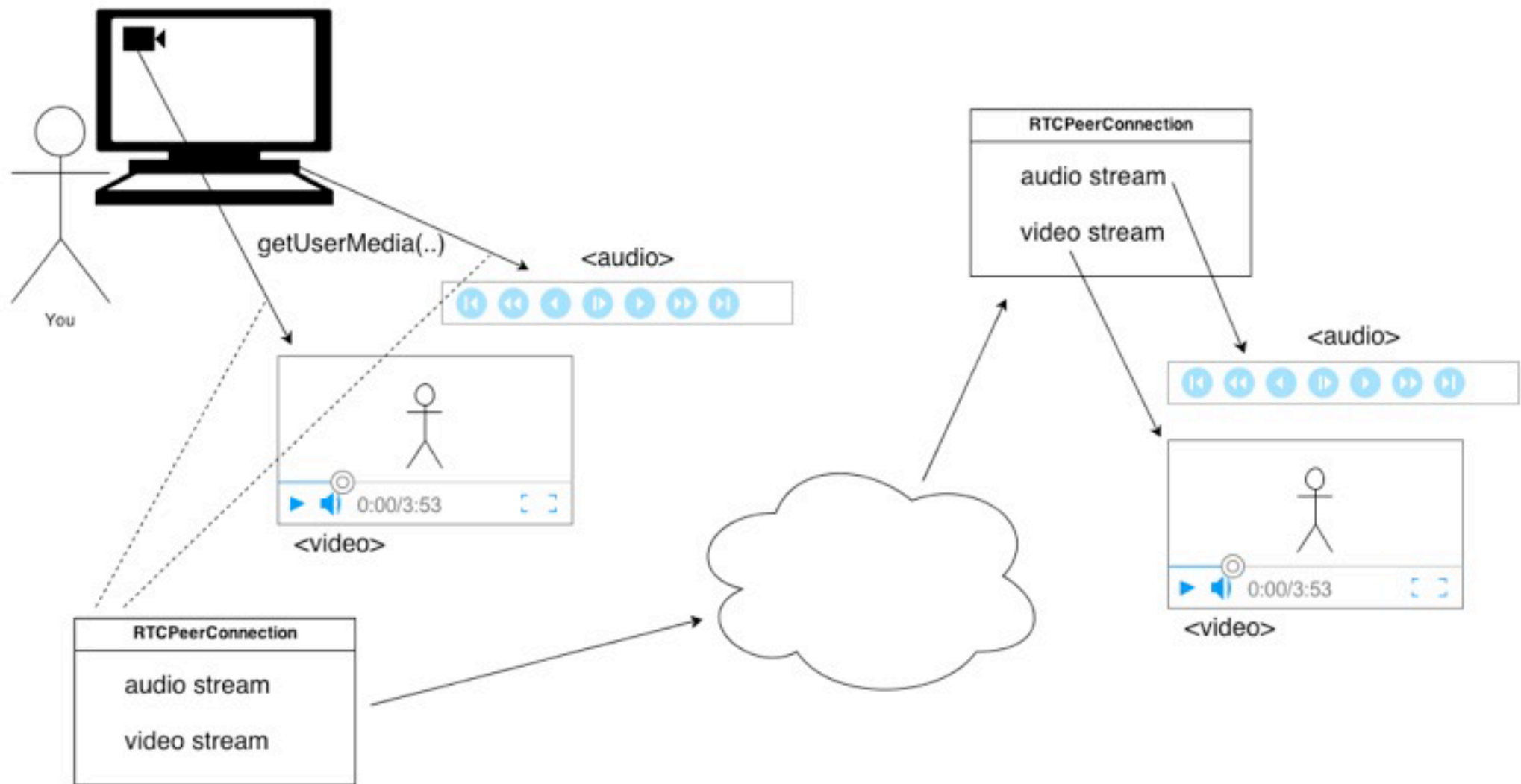
<http://www.webrtc.org/>

webcam capture

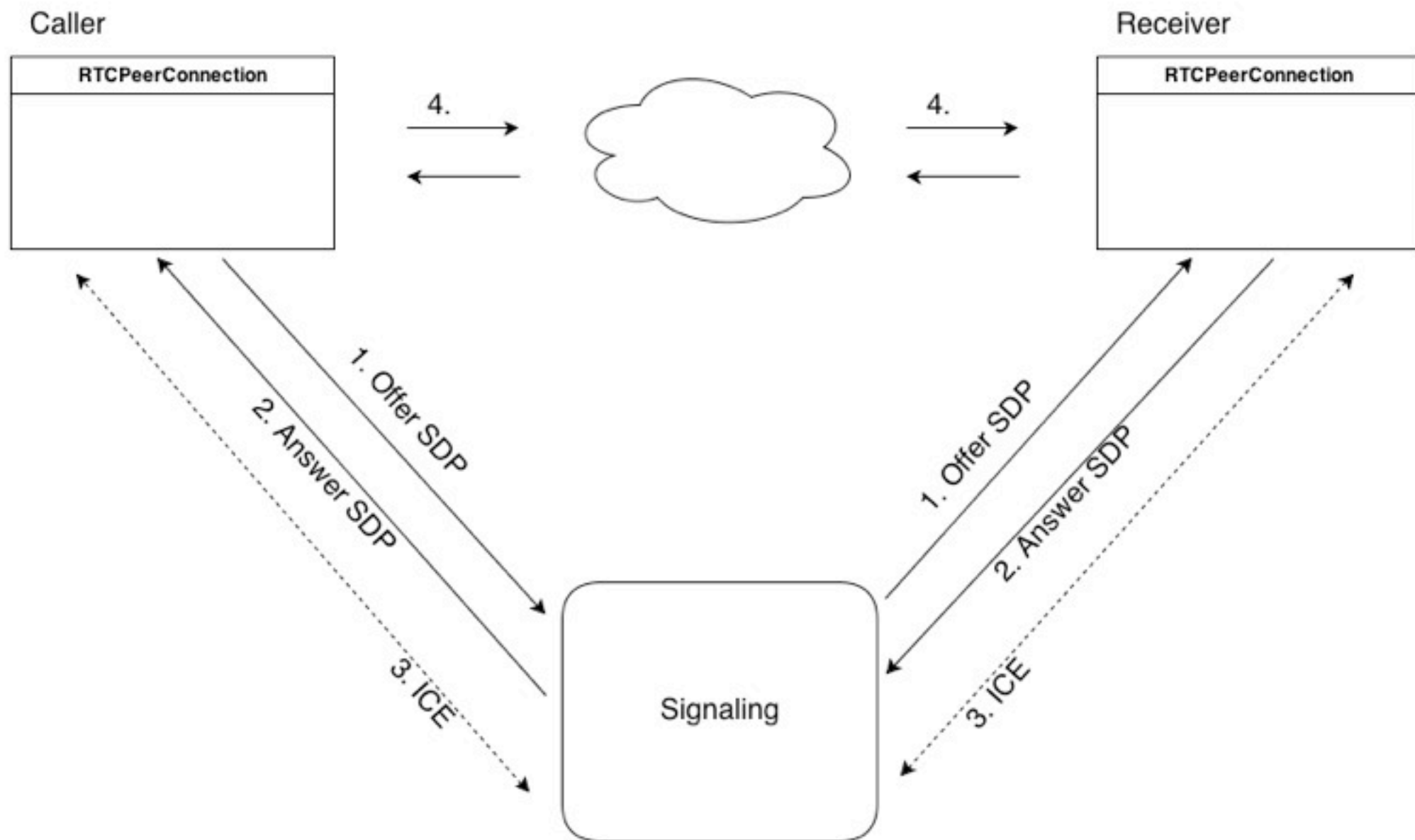


RTCPeerConnection

Video, Audio

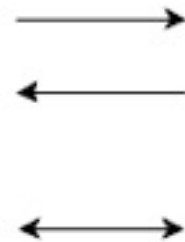
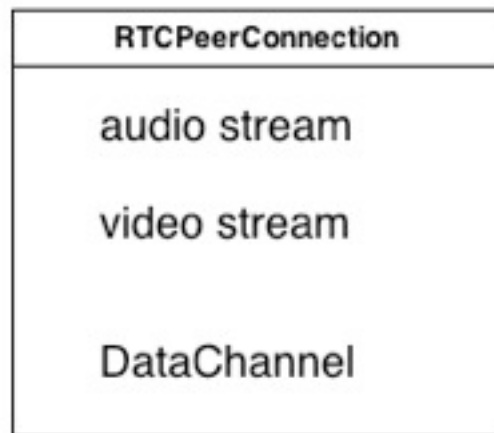


Signaling

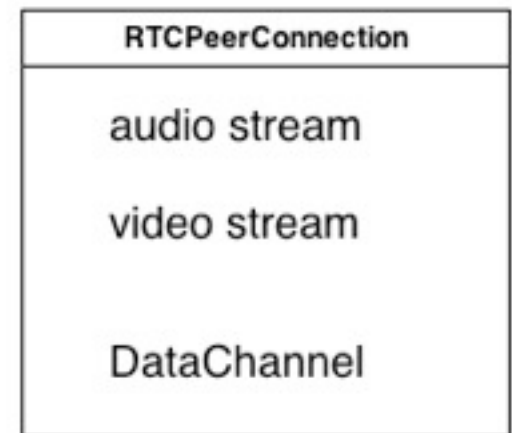


DataChannel

Caller



Receiver




```
20 // rtc stuff
21 function createPeerConnection(config,optional) {
22     if (global.RTCPeerConnection) return new RTCPeerConnection(config,opt
23     else if (global.webkitRTCPeerConnection) return new webkitRTCPeerConn
24     else if (global.mozRTCPeerConnection) return new mozRTCPeerConnection
25     throw new Error("RTC Peer Connection not available");
26 }
27
28 function createIceCandidate(candidate) {
29     if (global.RTCIceCandidate) return new RTCIceCandidate(candidate);
30     else if (global.webkitRTCIceCandidate) return new webkitRTCIceCandida
31     else if (global.mozRTCIceCandidate) return new mozRTCIceCandidate(can
32     throw new Error("RTC Ice Candidate not available");
33 }
34
35 function createSessionDescription(desc) {
36     if (global.RTCSessionDescription) return new RTCSessionDescription(de
37     else if (global.webkitRTCSessionDescription) return new webkitRTCSess
38     else if (global.mozRTCSessionDescription) return new mozRTCSessionDes
39     throw new Error("RTC Session Description not available");
40 }
41
42 function signal(message) {
43     if (socket) {
44         socket.emit("message",message);
45     }
46 }
```


Take a deep breath

Kyle Simpson
@getify
<http://getify.me>

Thanks!

Questions?