Semana da Eng. - Facens 2014

Desenvolvendo jogos multiplayer utilizando Html5 + WebSockets.

Eng. Carlos Augusto

Sobre o Palestrante

- Carlos Augusto (@gutopro)
  - -Engenheiro da Computação
  - -Scrum Master pela Scrum.org
  - -Especialista na plataforma .NET
- Consultor
  - -Na plataforma .NET e em arquitetura.

Inicio da vida gamer.

Desenvolvimento de jogos.

Inicio do desenv. de jogos

XNA Framework

Layers

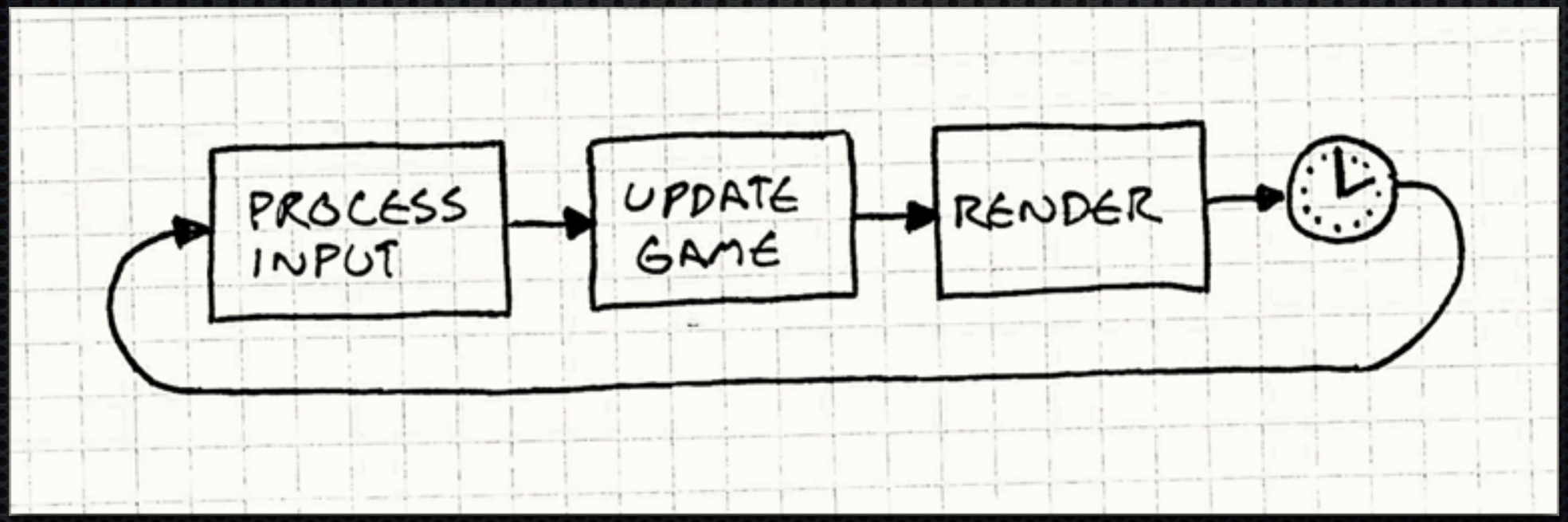
Games	Starter Kits	Code	Content	Components
Extended Framework	Application Model	Content Pipeline		
Core Framework	Graphics	Audio	Input	Math	Storage
Platform	Direct3D	XACT	XINPUT	XCONTENT

Legend: XNA Provides | You Provide | Community

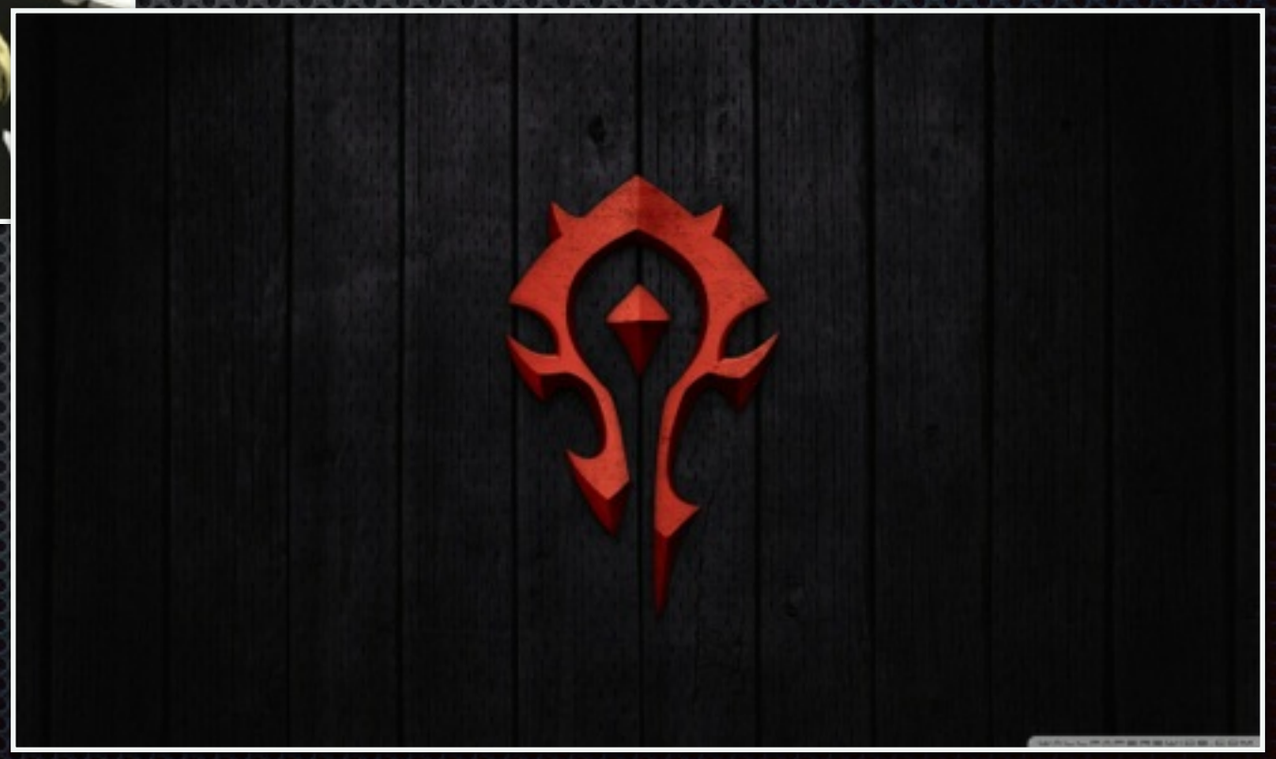
xna

Game Developer Conference 2007

Estrutura simples dos jogos

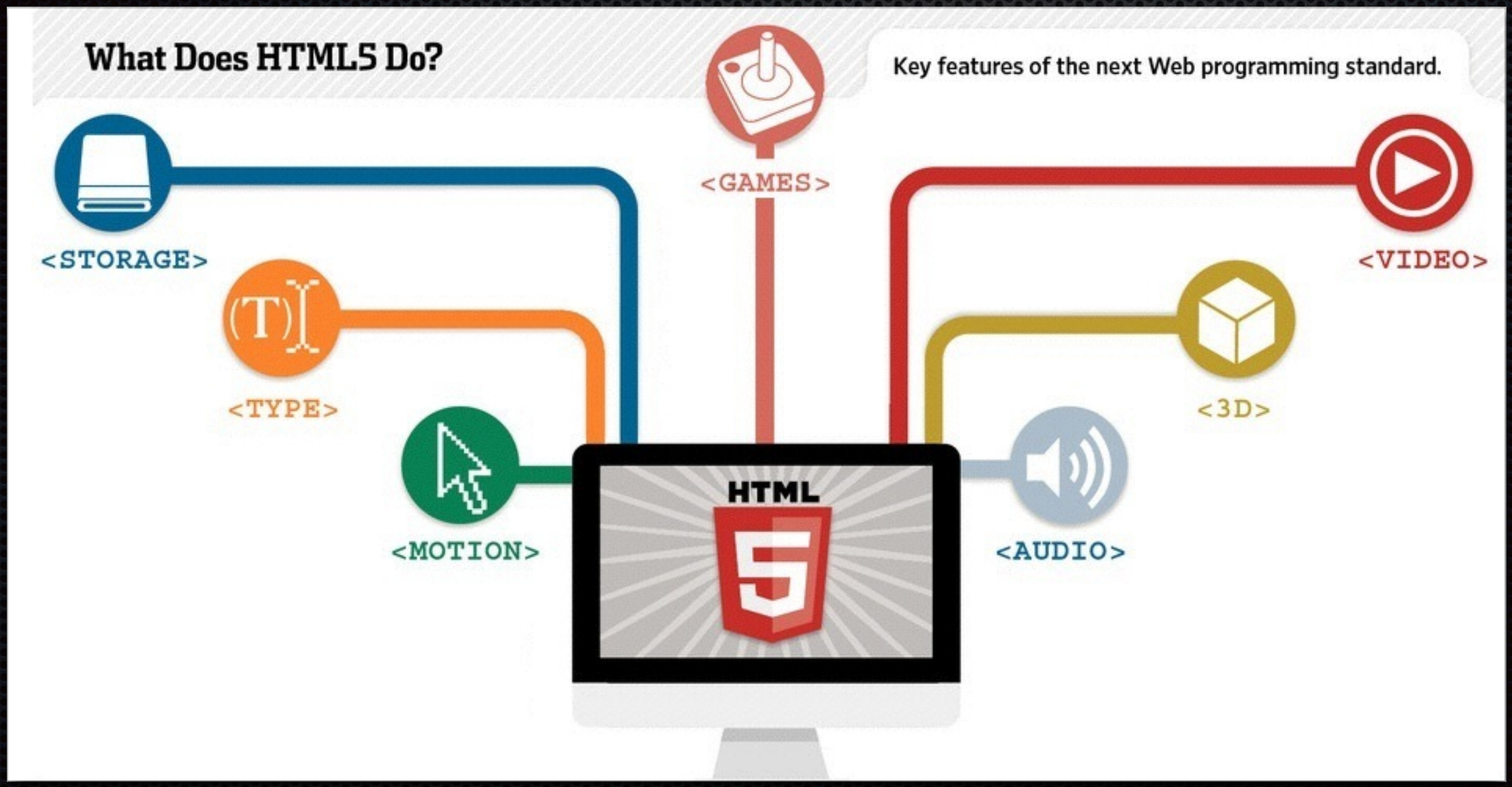


Jogos multiplayer !



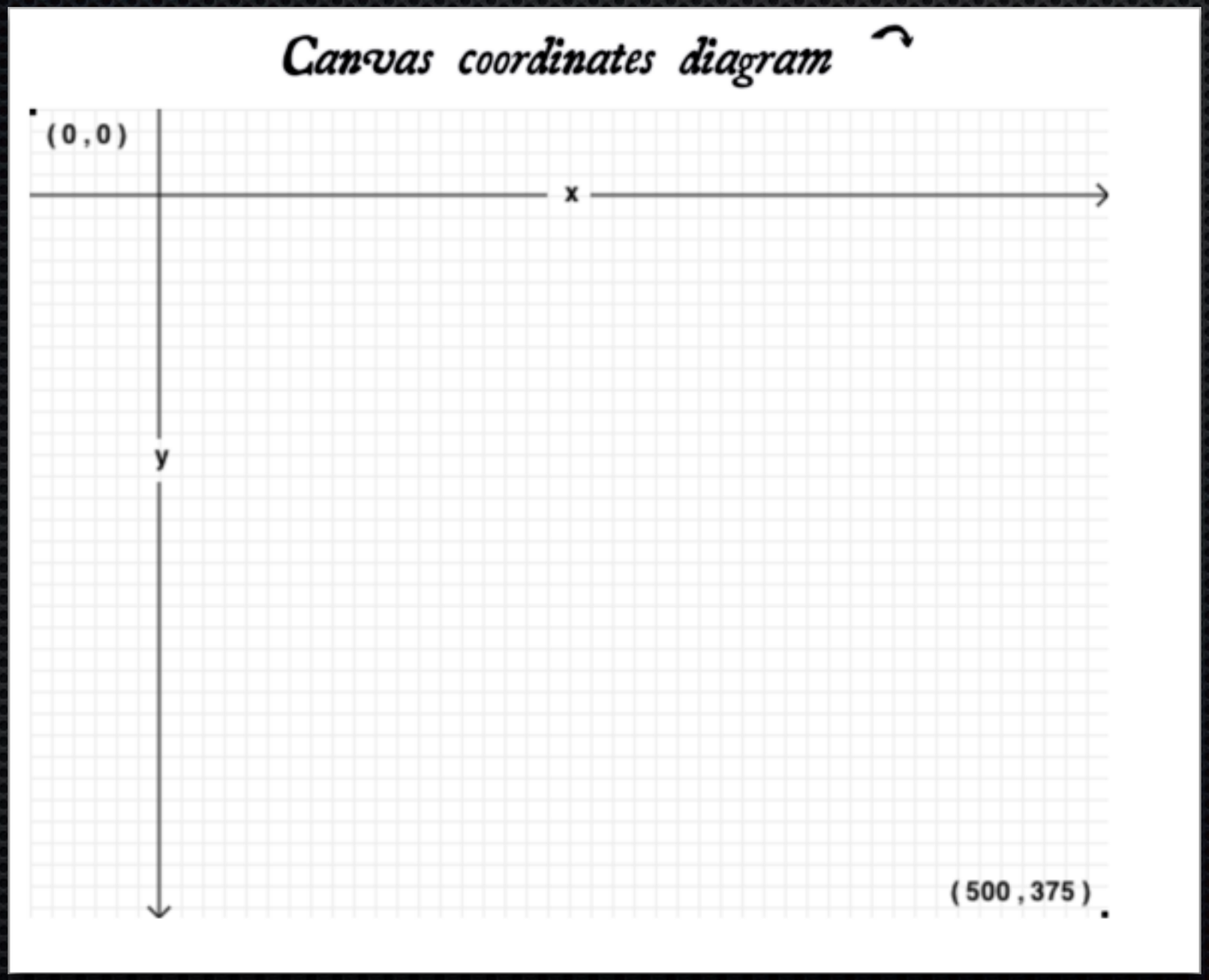
As tecnologias

HTML5


What Does HTML5 Do?
Key features of the next Web programming standard.
<GAMES>
<STORAGE>
<VIDEO>
<TYPE>
<3D>
<MOTION>
<AUDIO>
HTML
5

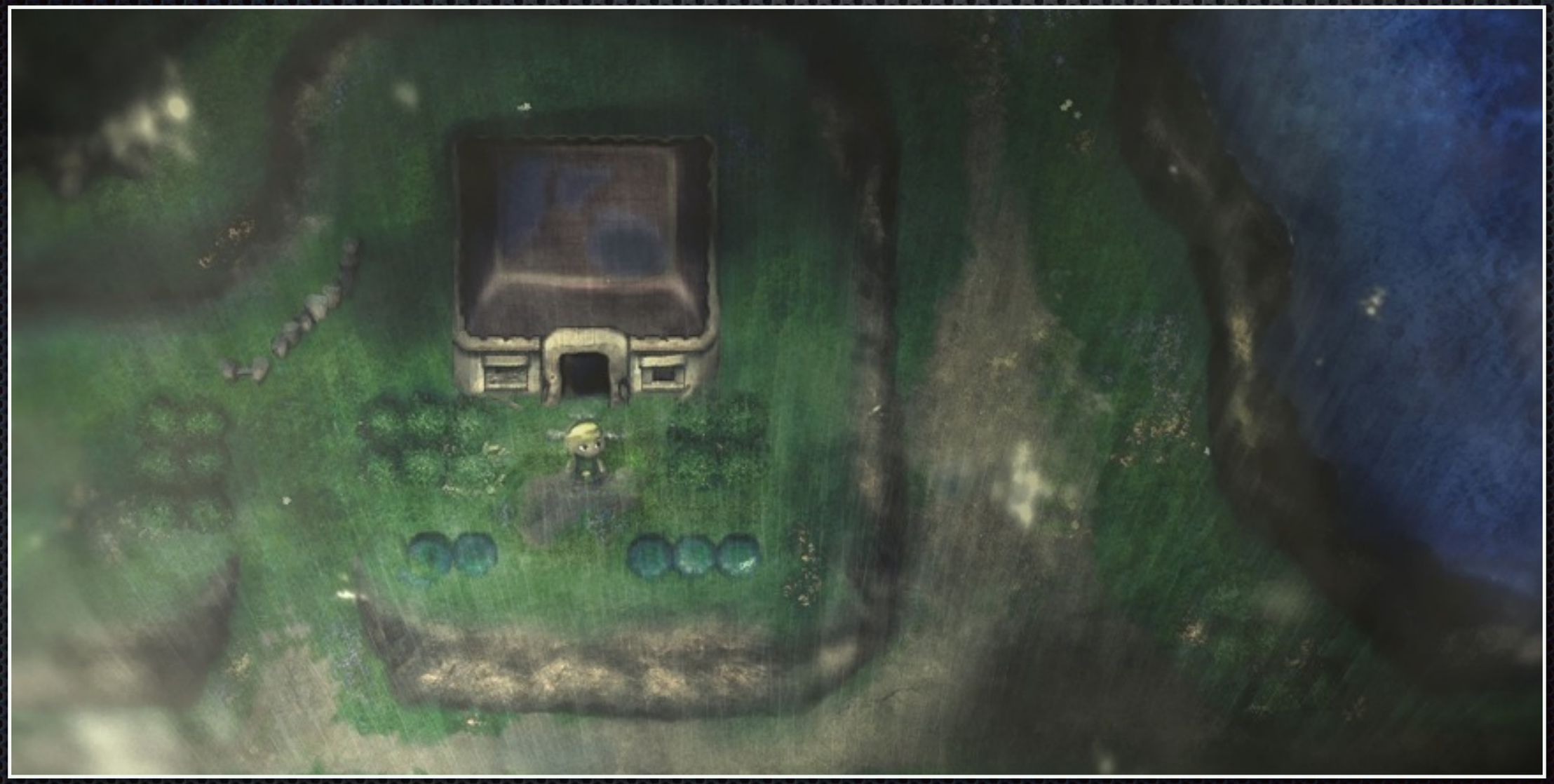
HTML5 - Canvas

Canvas - Coordenadas



Demo

HTML5 - Audio



main.html

```html
<html>
  <head> </head> <body> </body>
    <script type="text/javascript">
        //nome do som
        var soundName = "zelda.mp3";

        //criando o "canal" de audio
        var audioEle=document.createElement('audio');
        audioEle.setAttribute('preload','auto');
        audioEle.autobuffer=true;

        //criando a fonte (o arquivo do som)
        var source1=document.createElement('source');
        source1.type='audio/mpeg';
        source1.src=soundName;
        audioEle.appendChild(source1);

        //finalmente
        audioEle.play();

        //adicionado botes playStop
        var btnStop = document.createElement('input');
        btnStop.type='button';
        btnStop.value ='stop';
        btnStop.onclick=function(){
            audioEle.pause();
            audioEle.currentTime=0;
        }
        document.body.appendChild(btnStop);
        //btnplay
        var btnPlay = document.createElement('input');
        btnPlay.type='button';
        btnPlay.value ='play';
        btnPlay.onclick=function(){
             audioEle.play();
        }
        document.body.appendChild(btnPlay);
    </script>
```

tst

- loadImg
- loadSound
  - main.html
  - zelda.mp3

Demo

HTML5 - Video

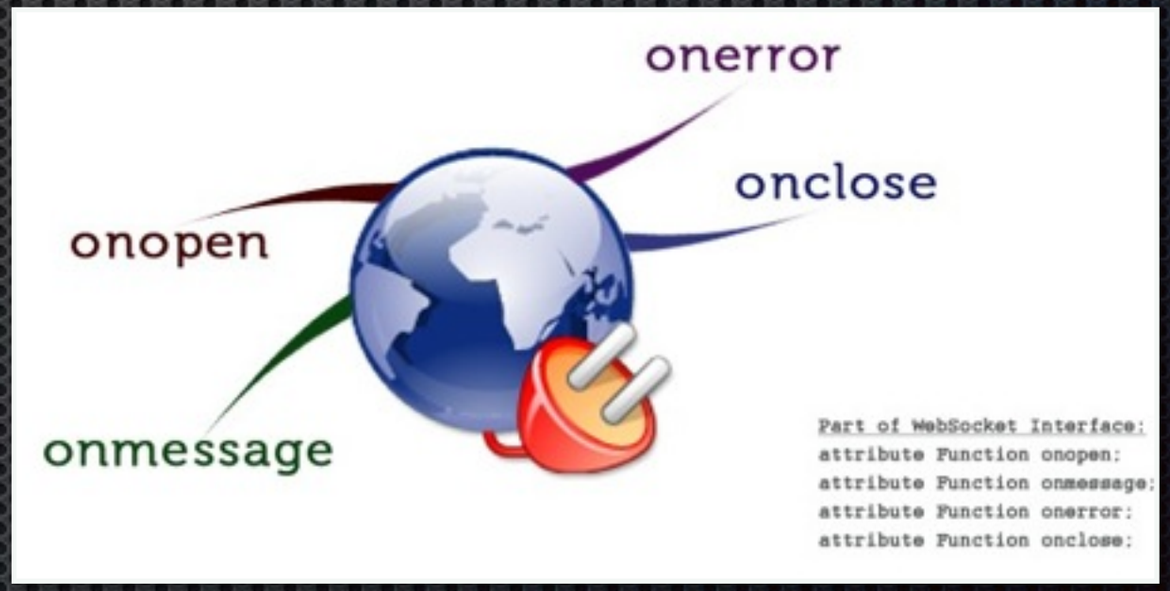
Demo

Html5 - Storage

- Características
  - Armazenagem de dados de 5 MB (recomendado).
  - Tipo de armazenagem - Key-Value pair.
- Tipos
  - SessionStorage
    - Apenas para uma sessão (janela).
    - Se o aba for fechado os dados se perderam.
  - LocalStorage
    - Compartilhado pelas sessões.
    - Permanece disponível até que seja removido pelo usuário/site/programador.

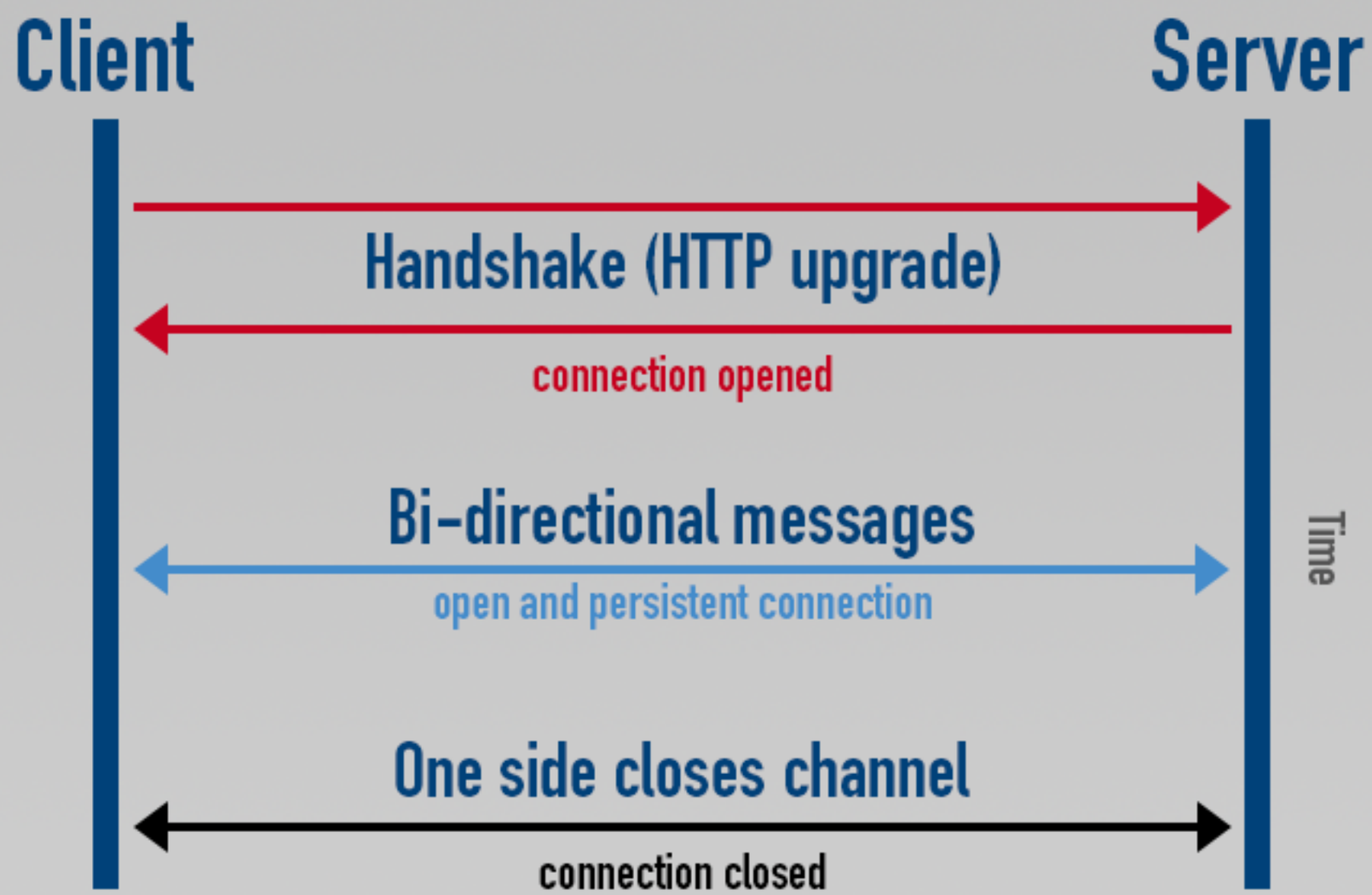
Html5 - Websocket

- É um protocolo de comunicação.
- Modelo “socket”.
- Esquema de url novo “ws” ou “wss”.
- Várias linguajes suportadas : Objective-C, .NET, Ruby, Java, NodeJS e Actionscript.
- Comunicações com o servidor e o cliente.



WEBSOCKETS

A VISUAL REPRESENTATION



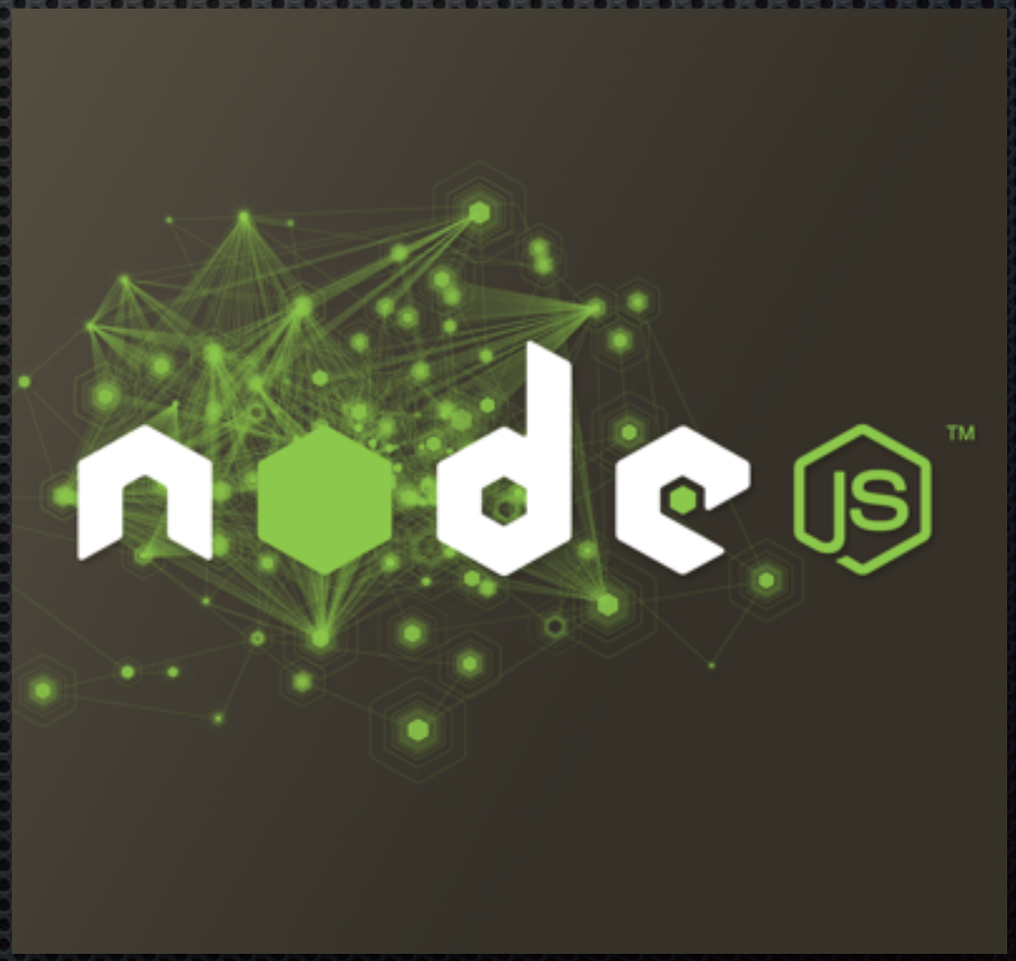
PubNub

Javascript

- Vantagens
  - Executado no cliente.
  - Rápido.
  - Cross-Plataform.
  - Cross-Browser.
  - Aceito por todas as grandes indústrias (Apple, Microsoft, Google, Facebook e etc…)
- Desvantagens
  - Lento se mal programado.
  - Não compilável. (Coffee script)
  - Falta de ferramentas (IDE, Debugger, etc…).
  - Dependente excessivo do Browser.

Node JS

- Javascript no server !!.
- MUITO rápido !. (V8 Javascript Engine).
- Orientado a evento.
- Documentação excelente.
- Comunica com o banco de dados.
- Compartilha código entre o servidor e o cliente.
- Facilita a comunicação via Websocket.
- Possui diversos “plugins”.



NPM-Node Package Manager

- Gerenciador de pacotes do Node (similar apt-get).
- Instalação de pacotes via linha de comando.
- Pacotes com resolução de sub-dependências automático.

Node JS + NPM = S2

Socket.IO

- Simplifica o Websocket.
- Extremamente simples.
- Cross-browser.
- Cross-Plataform.
- Executa em “tempo-real”.

Express JS

- Servidor web pequeno e flexível para o NodeJS.
- Integração com o Socket.IO.
- Executa sobre o NodeJS.
- Cross-Browser.
- Cross-Plataform.

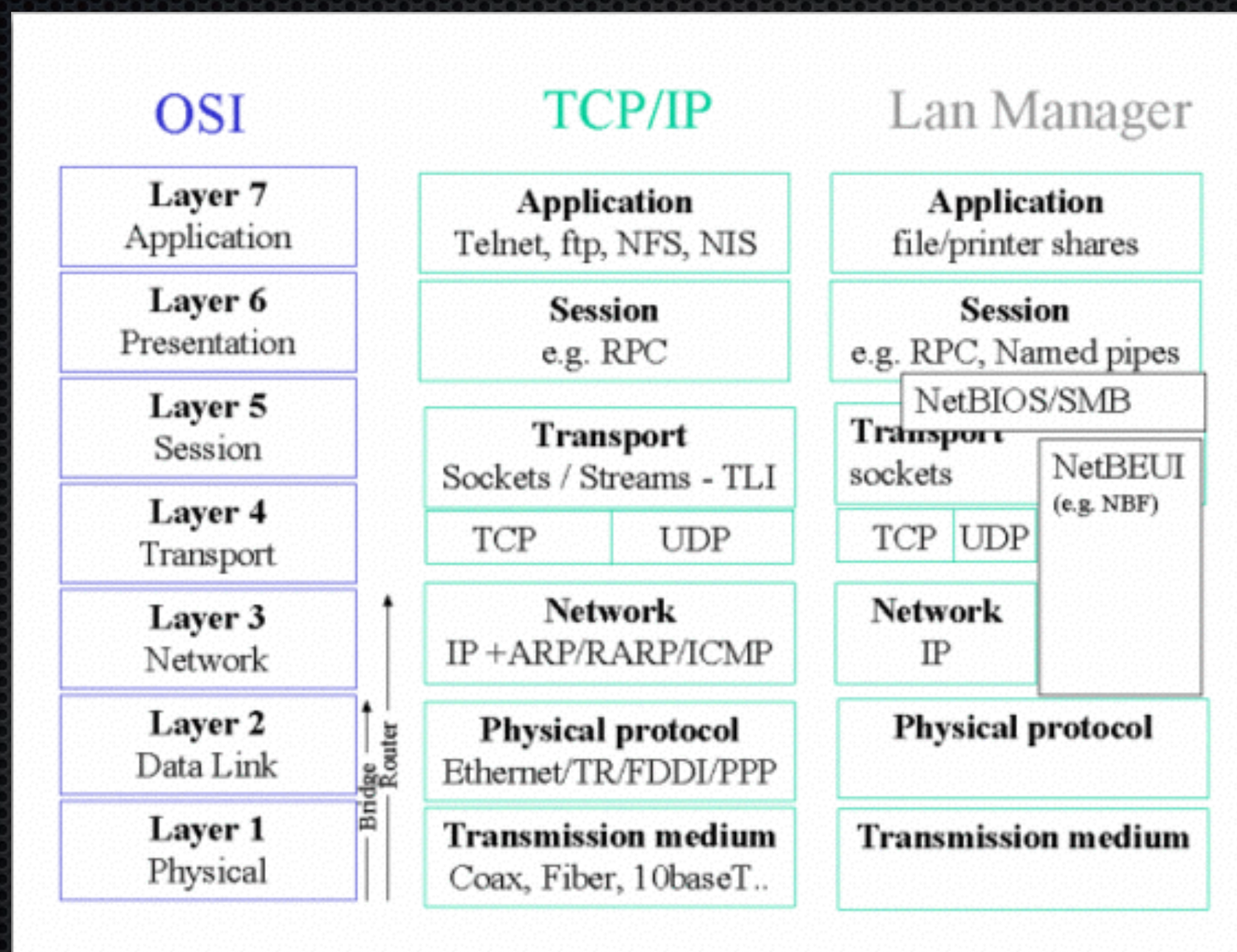
express

High performance, high class web development for **Node.js**

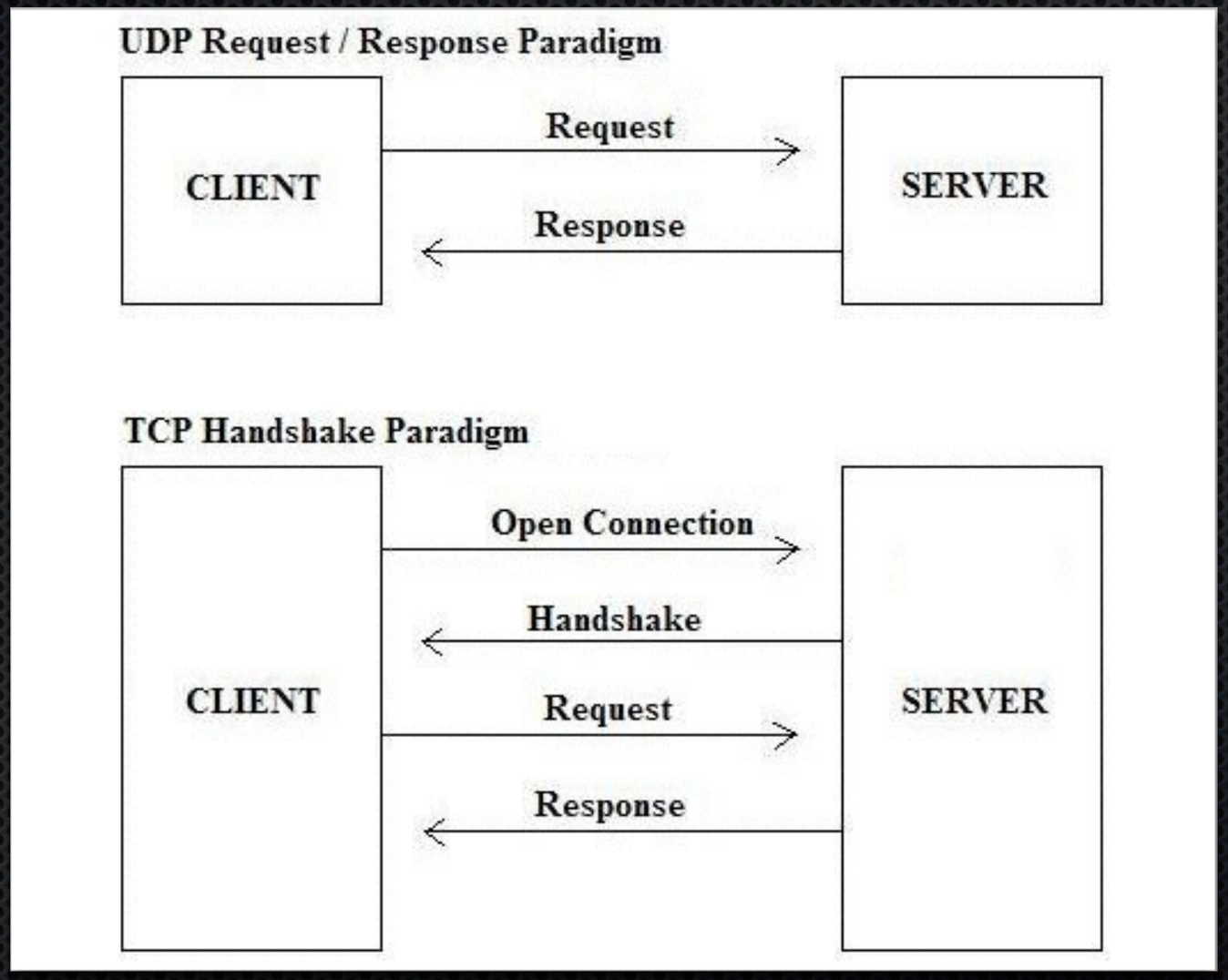
Vamos para o multiplayer...

Lag

Rede - Camadas

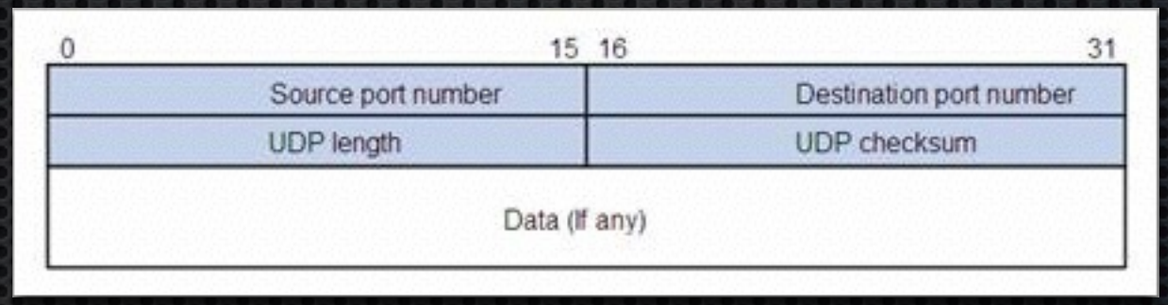


Rede - TCP e UDP

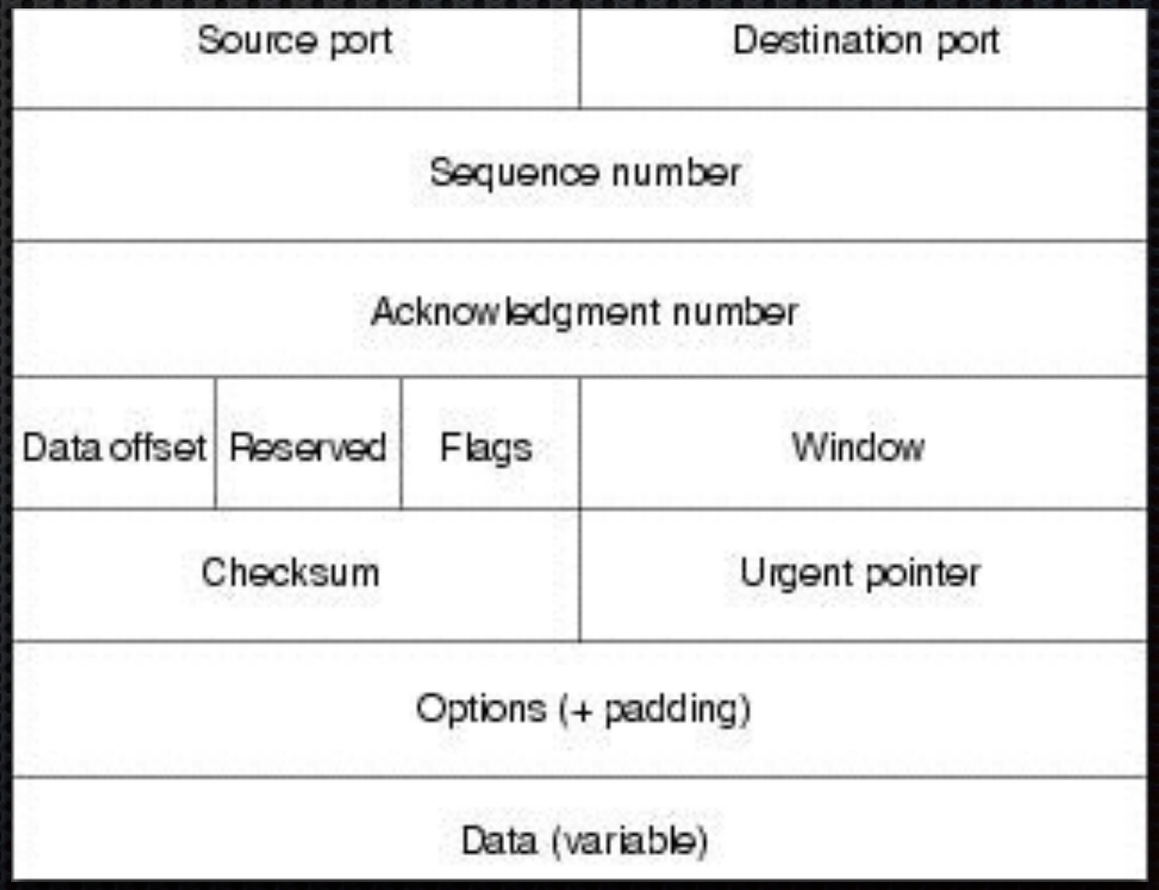


Rede - Formato de msg.

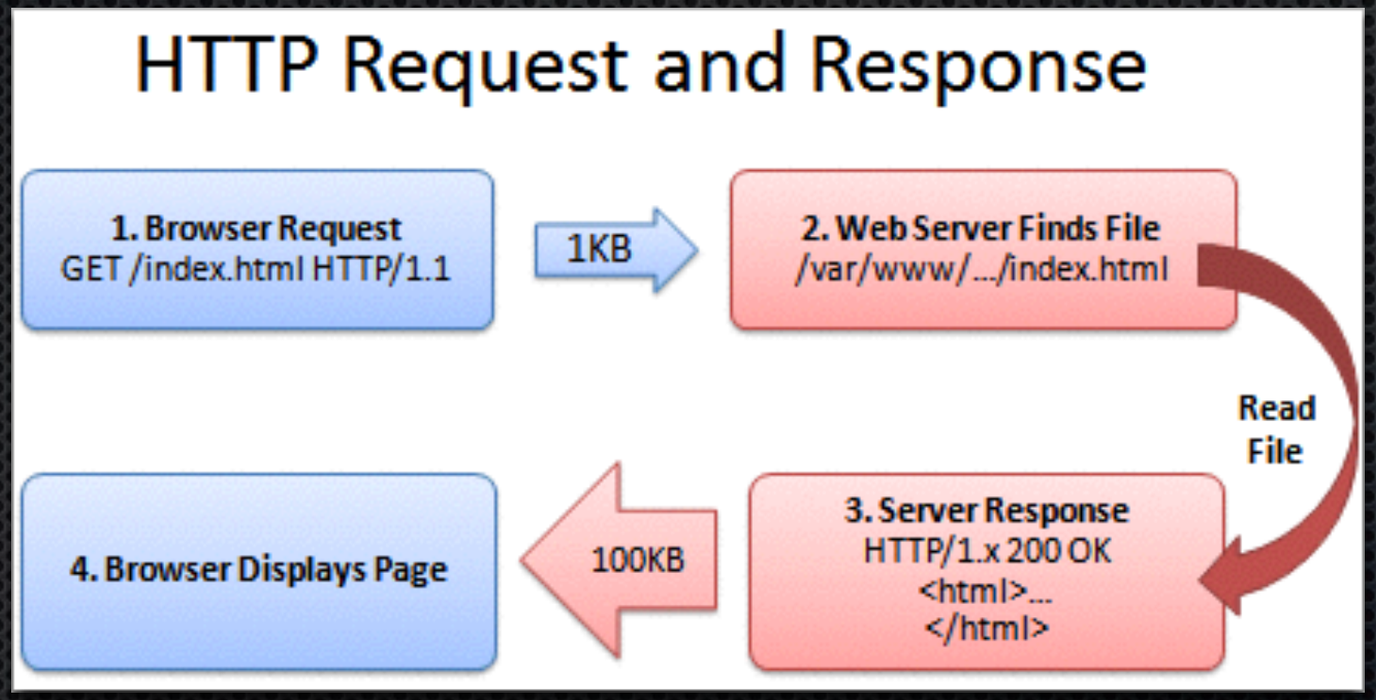
UDP



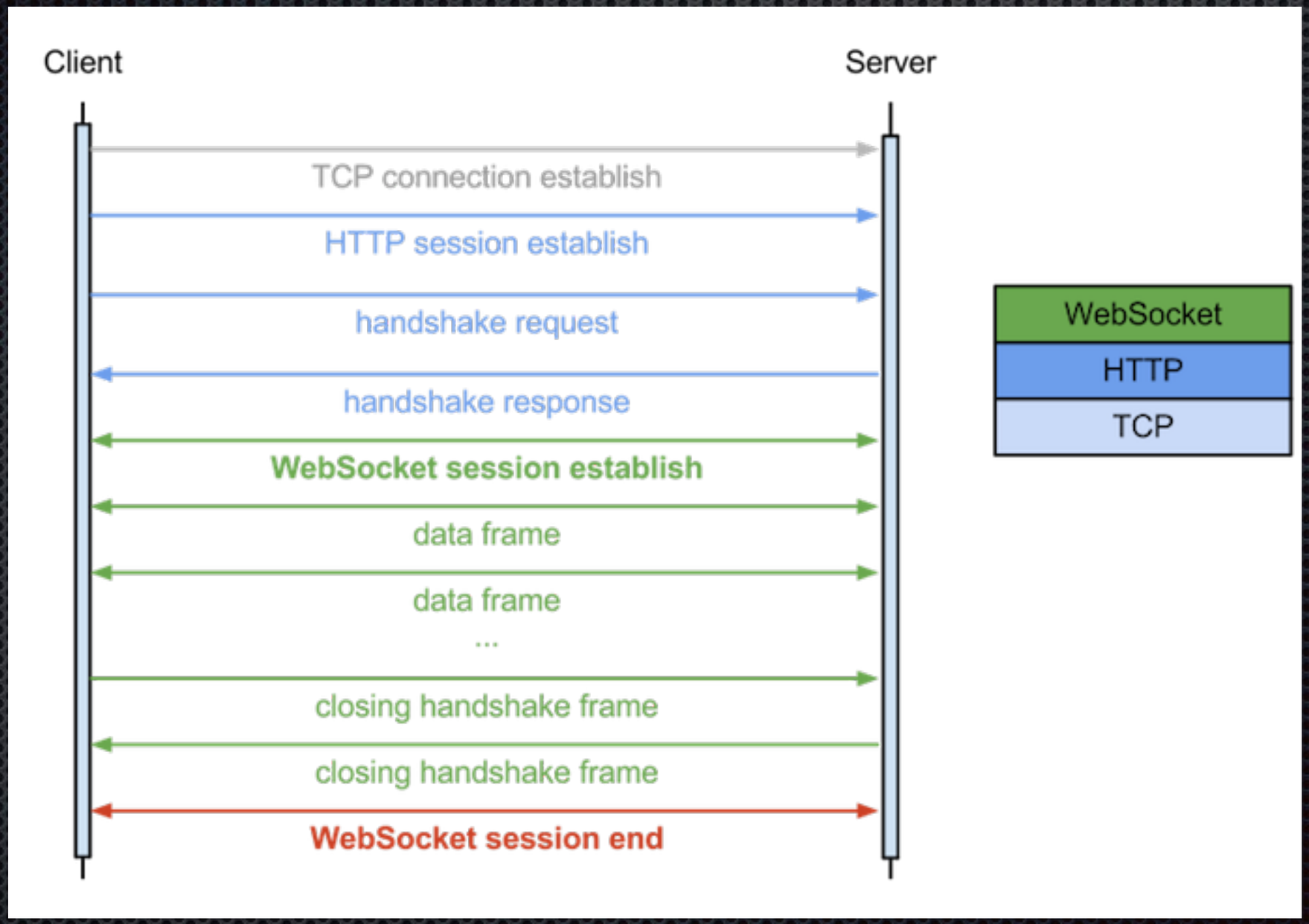
TCP



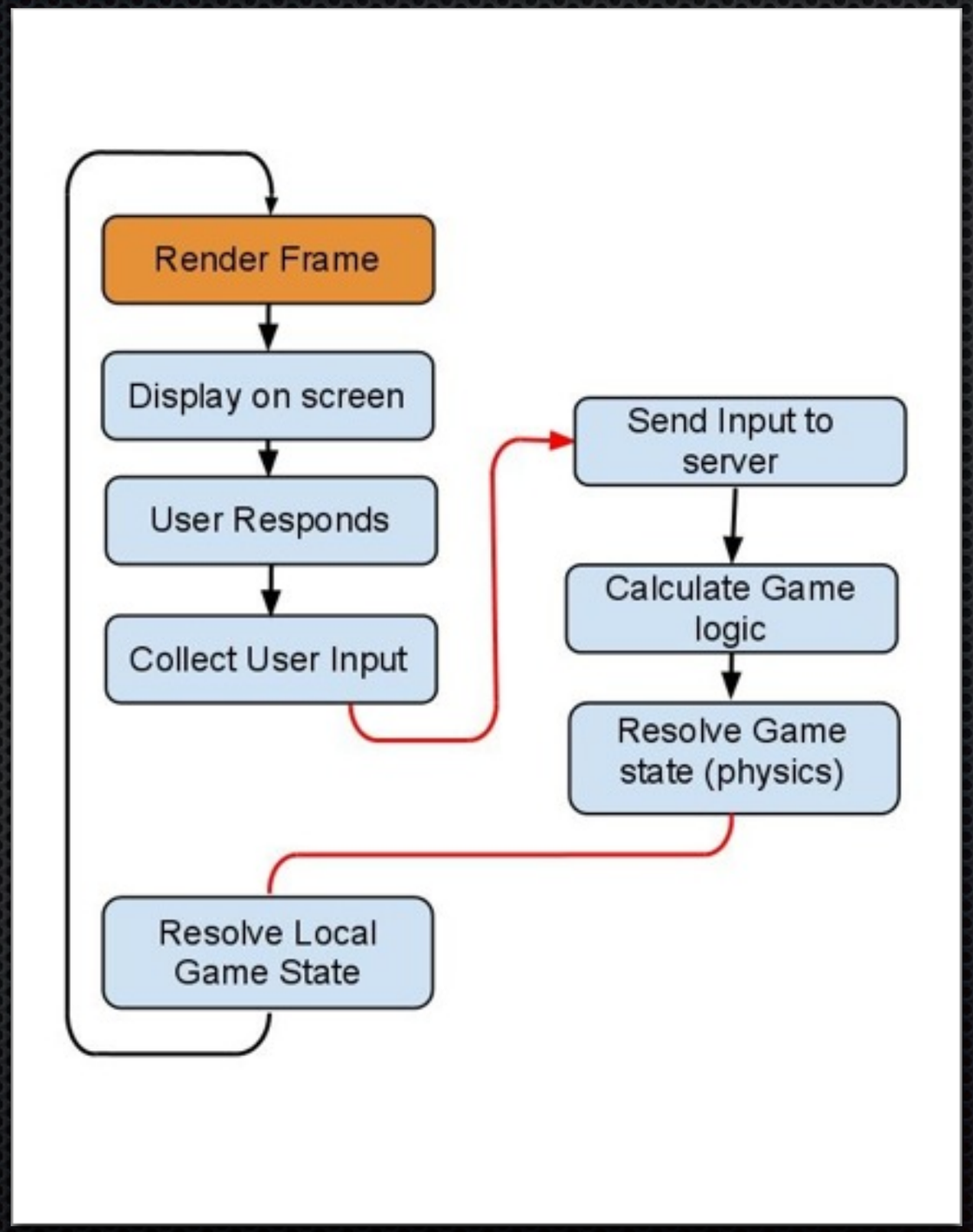
Web - Modelo atual.



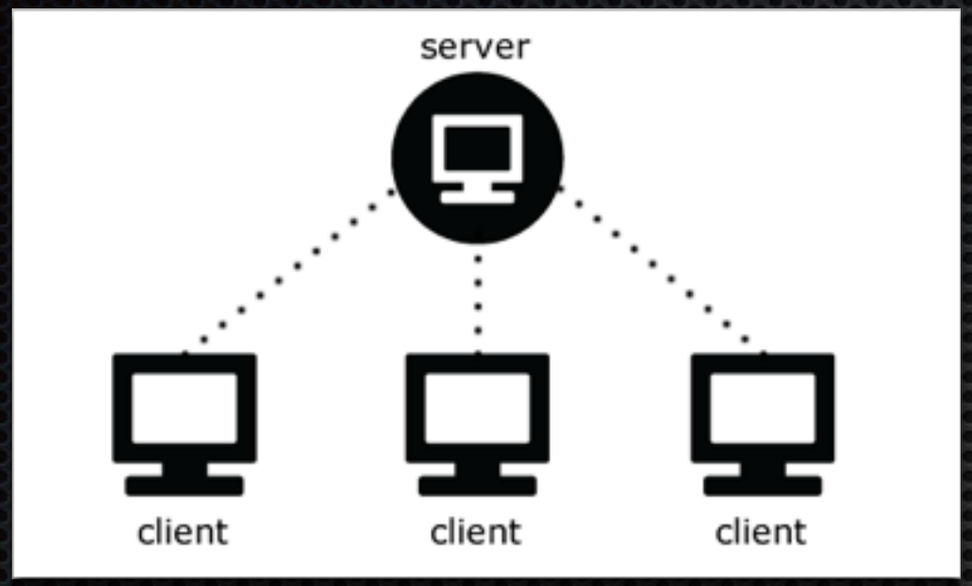
Mas qual a diferença ?

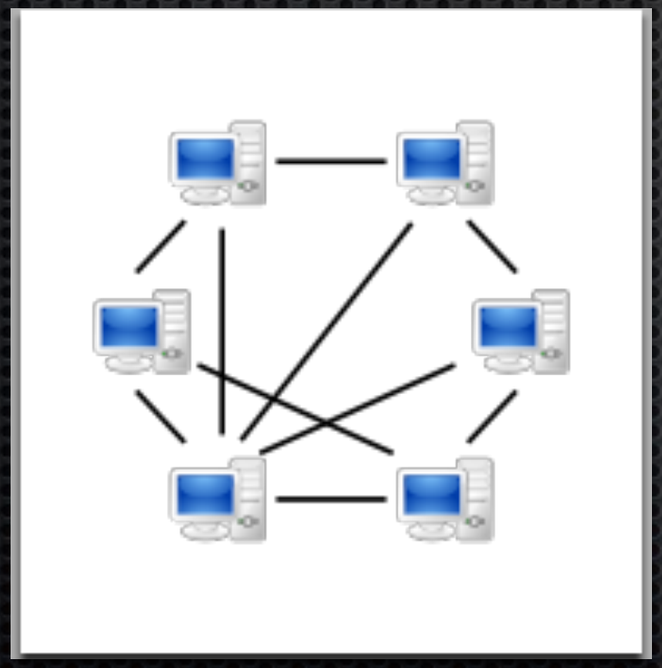


Estrutura simples multiplayer



Tipos de arquitetura

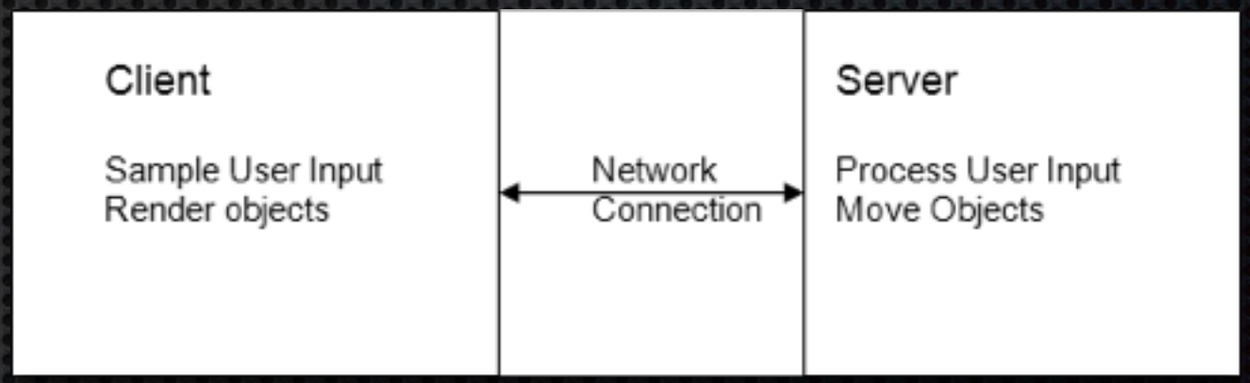




Websockets utiliza somente o TCP e o modelo cliente-server !

Separação de conceitos

- Cliente
  - Renderizar imagens.
  - Receber comandos do player.
  - Receber e processar comandos do Server.
  - Conectar-se ao server.
  - Enviar comandos ao server.
- Server
  - Processar comandos.
  - Definir lógica do jogo.
  - Atualizar todos que estão conectados.



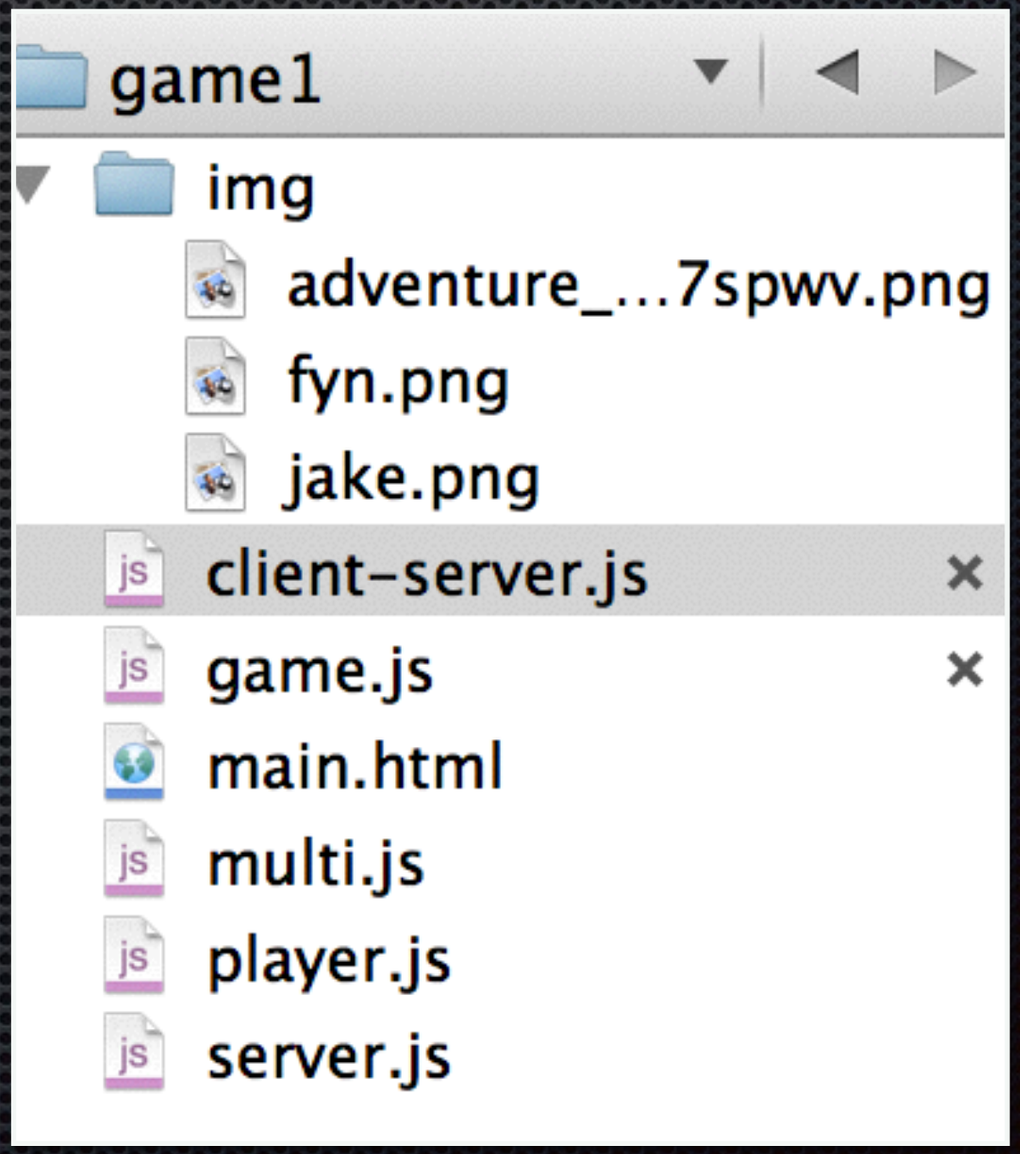
Mensagem

- Definir formato da mensagem a ser trafegada.
- Trafegar somente o necessário
- Exemplo: posição do player (inteiro x, inteiro y).

O jogo

Estrutura do jogo

- Client
  - Client-server - Servidor web do Client (jogador) (NodeJS+Express).
  - game1.js - (lógica do jogo).
  - main.html - (Html do cliente).
  - multi.js - script responsável pelo multiplayer.
  - player.js - “classe” do jogador.
- Server
  - server.js - Servidor do Websocket. (NodeJS + Socket.IO).



Estrutura do jogo - Cliente

- Engine
  - Criar o canvas..
  - Adicionar imagens.
- O jogo
  - Inicio
  - Atualizar
    - Atualizar Rede
  - Desenhar
    - Desenhar

Estrutura do jogo Cliente(Net)

- Socket.On (connect) - Invocada ao conectar.
- Socket.On(‘<NomeDaFunção>’) função customizada recebida pelo server.

```
28 ▾ socket.on('connect', function () {
29
30 ▾     socket.on('getUpdate', function(objData){
31           objMultiRet = objData;
32 ▲     });
33
34 ▾     socket.on('newP', function(data){
35           NewPlayerConnected(data);
36 ▲     });
37
38 ▾     socket.on('fimjogo', function(){
39          fimjogo=true;
40 ▲     });
41
42 ▾     socket.on('getPlayers', function (data) {
43           // my msg
44           NetPlayers = data;
45           //start the game after anyone connect
46           ready();
47 ▲      });
48 ▲ });
```

Estrutura do jogo - Server

- Socket.IO - Webksocket
- Socket.emit - Envia msg. para o cliente.
- Socket.On(‘connection’) - Evento ao conectar.
- Broadcast - Envia para todos os clientes exceto o cliente mesmo.
- Socket.On(“<InsiraONome”>) - Função customizada disponível ao cliente.

```
var io = require('socket.io').listen(8052);
io.sockets.on('connection', function(socket) {
    qtdPlayer++;

    socket.emit('getPlayers', objArray);
    socket.broadcast.emit('newP', sendNewP());

    socket.on('updateNet', function(data) {
        //console.log('updateNet called');
        if (data.num == 1) {
            objArray[0].x = data.x;
            objArray[0].y = data.y;
        } else if (data.num == 2) {
            objArray[1].x = data.x;
            objArray[1].y = data.y;
        }
        var bolColisao = colisaoPlayer();
        if (bolColisao) {
            socket.broadcast.emit('fimjogo');
        }
        socket.broadcast.emit('getUpdate', objArray);
    });

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

Demo

Obrigado!

- http://www.toptal.com/nodejs/why-the-hell-would-i-use-node-js
- http://www.websocket.org/
- http://diveintohtml5.info/canvas.html
- http://dev.w3.org/html5/webstorage/
- http://pusher.com/websockets
- http://html5gameengine.com
- http://html5quintus.com/#demo