CSE104 Web Programming

Server Programming

General Introduction

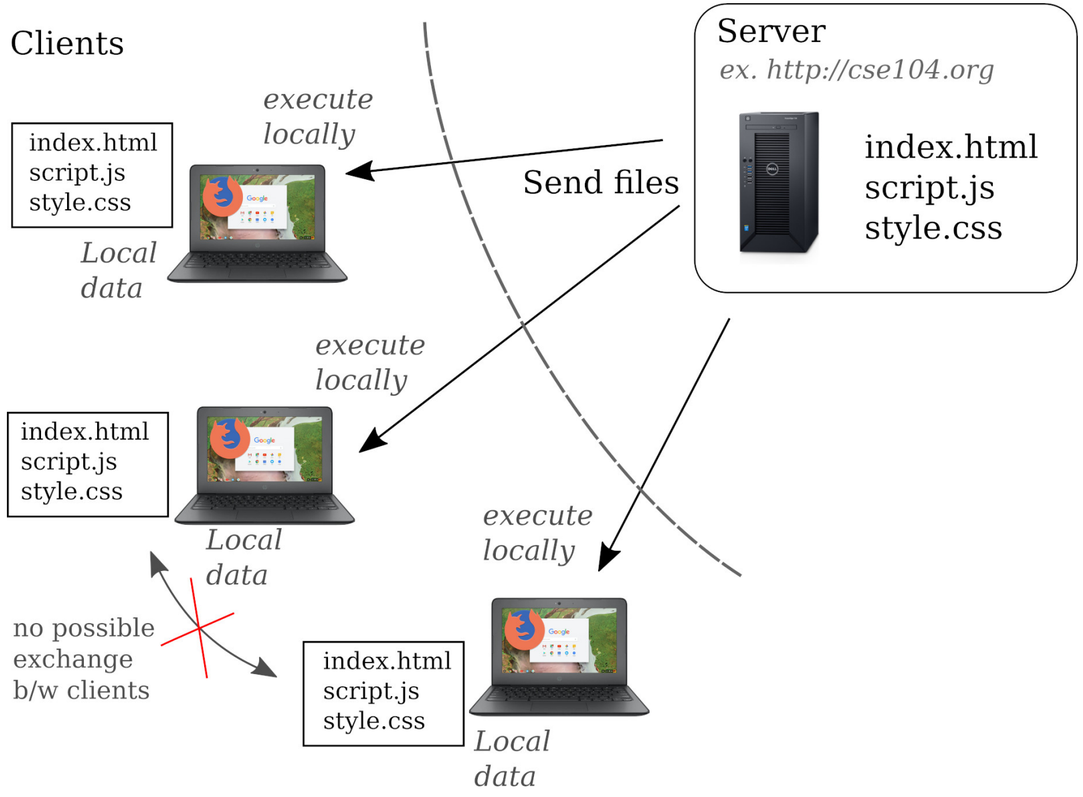
Why Server Programming

So far, we used HTML, CSS, JavaScript to create webpages that are interpreted on the **client side** (the browser).

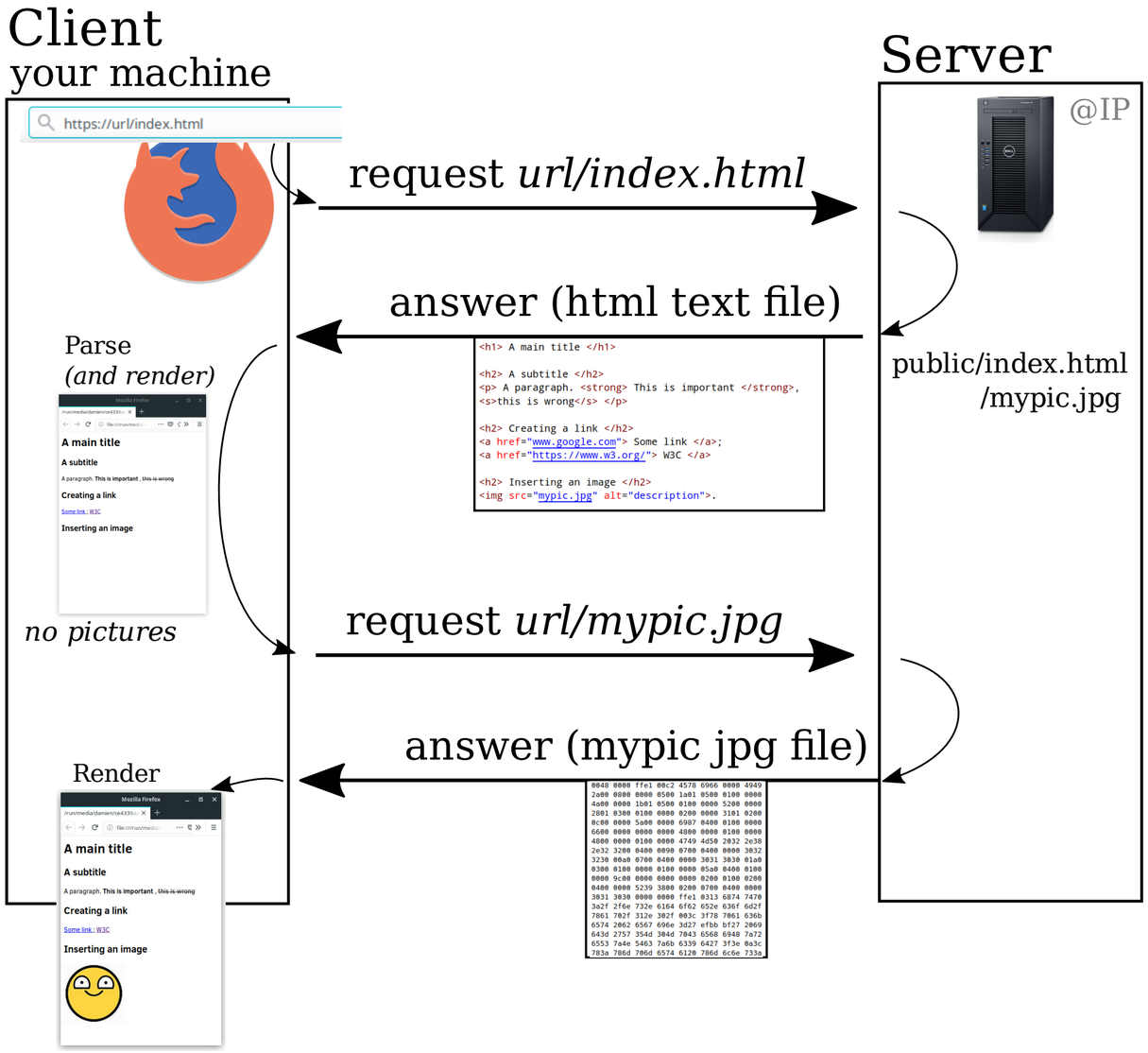
The server only distribute pre-made static files to the client.

Client side limitations:

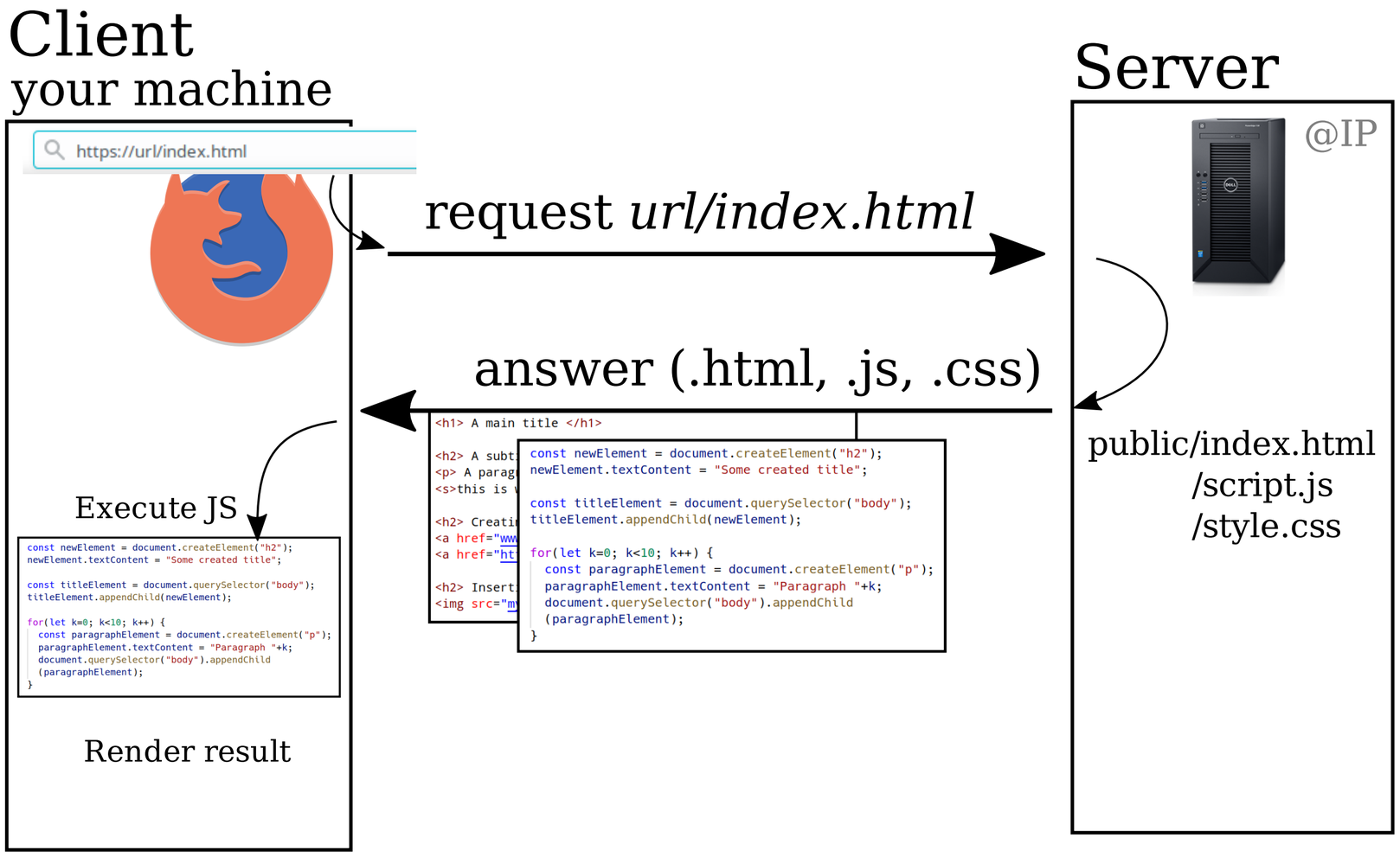
- *Cannot share information between users*
  - *No network games, shared high-scores, chats, etc.*
- *Limited access to the surrounding system (for security)*
  - *Cannot write into files, read automatically content of a directory, etc*
- *No memory after restarting the webpage (or limited: local-storage, cookies)*
  - *No data base*



Reminder on Client Side



Reminder on Client Side (with JS)



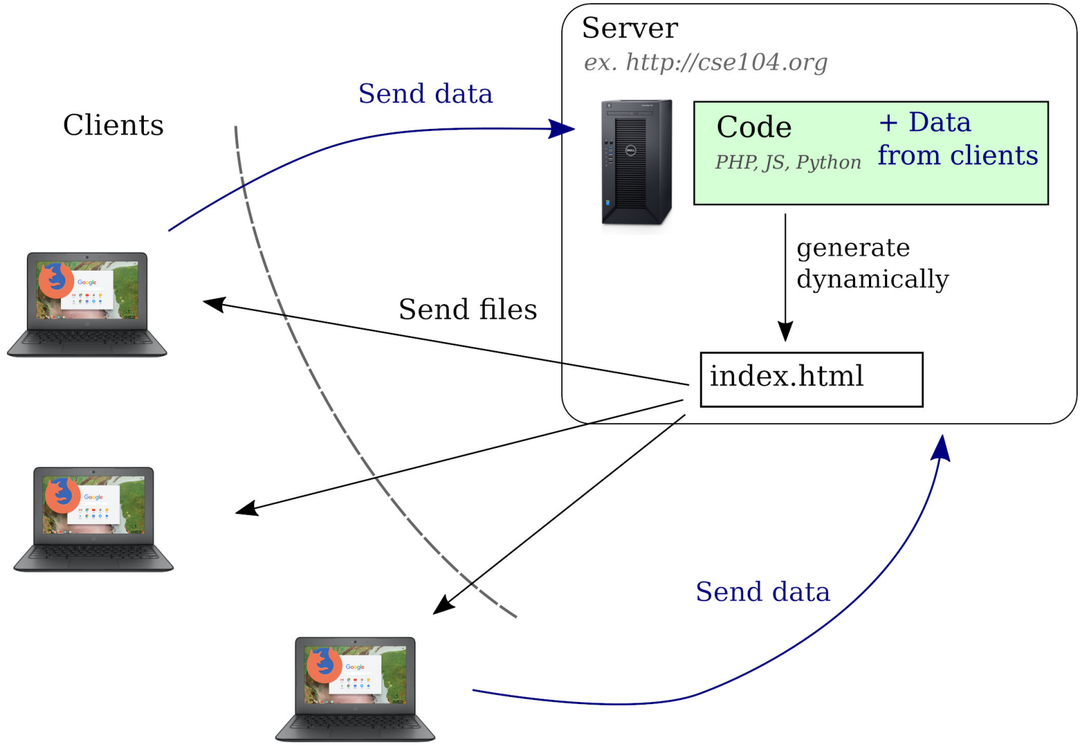
- Server only distribute files (same file for every client)
- JS code fully executed on the client computer

Idea of server programming

The server can receives data from the client.
A code run on the server. This code generates an HTML file.
This html file is sent to the client.

Server programming:

- *Information from a client can be used by the server*
  - *Allows sharing information b/w clients*
- *A code is run on the server. The code is not visible not accessible from the client (for security).*
- *Server code can write files on the server, and access its filesystem*
  - *Storage of data-base, automatisation for displaying images, etc*

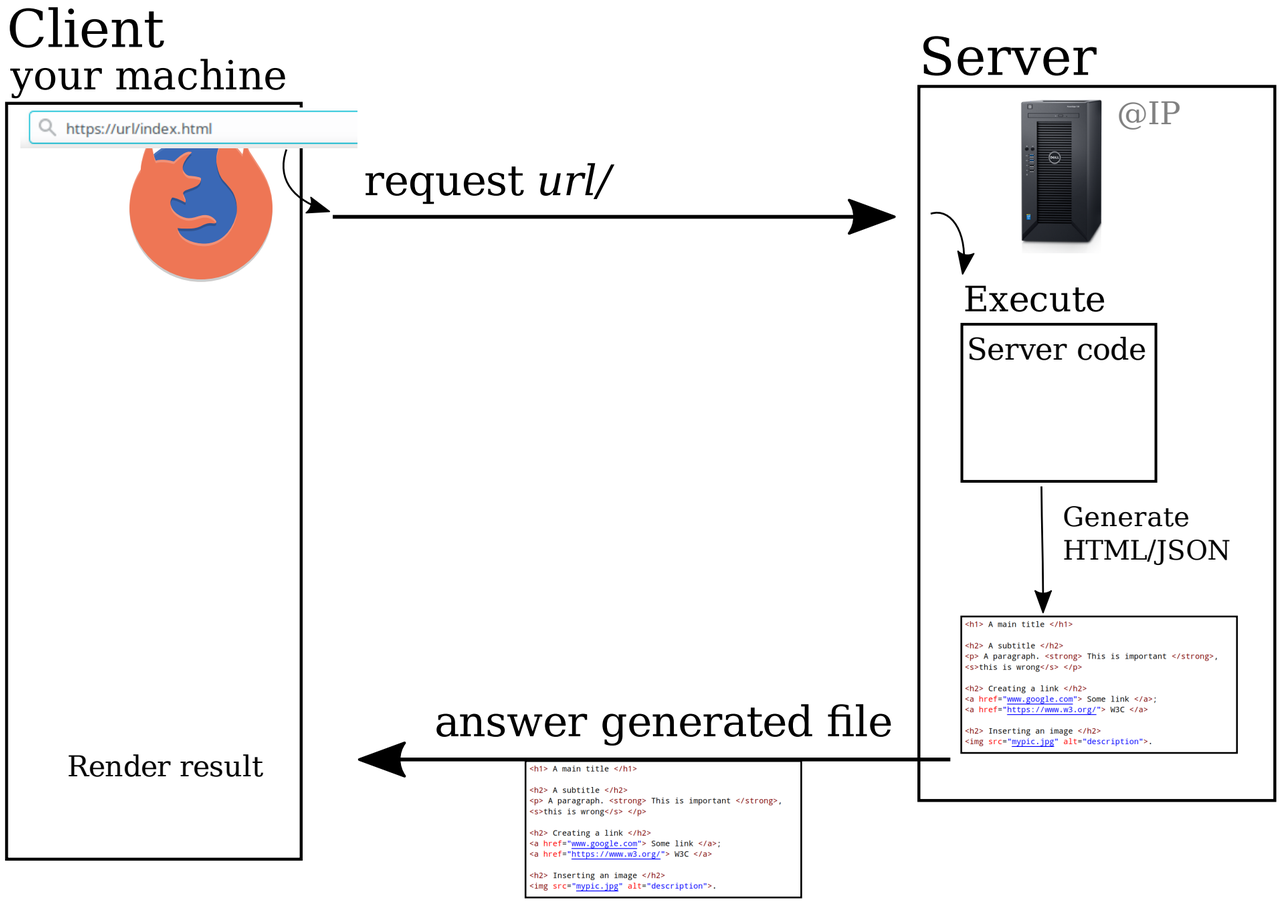


Server programming

1- Server receives a request from the client.

2- A code is executed on the server.

3- This code returns some data to the client (HTML page, JSON, ...)



Contacting a server

We already saw different ways to contact a server

- Direct URL entry in your browser
  - ex. you type cse104.org/your_directory
- Links from HTML pages
  - ex. `<a href="cse104.org/your_directory">`
- Fetch call from JavaScript
  - ex. fetch(cse104.org/your_directory).then(response)...

We saw that parameters can be sent with the URL

- url/?query?...

Server programming $\simeq$ Handling/Responding to these queries

Server Programming Language

Server Programming can use any languages (more variability than for client based)

-C++, Python, PHP, JavaScript, etc.

We are presenting an introduction to

- PHP: the standard, highly developed/available
- Node.js: the newcomer, developing fast

PHP - Introduction

PHP

PHP - Hypertext Preprocessor

- Scripting language
- Created in 1994 (initially as **P**ersonal **H**ome **P**age Tools)
- Last release: PHP 7 in 2015

Objectives

Automatize the creation of HTML pages

- File include, Variables, Loops
- Executed on the server (/JS executed on the client)

Principle of PHP interpreter:

Input: A PHP Script

Output: A HTML Page

Today: PHP is the most common language for server programming

(+) Very simple to be embeeded (/mixed) within HTML page

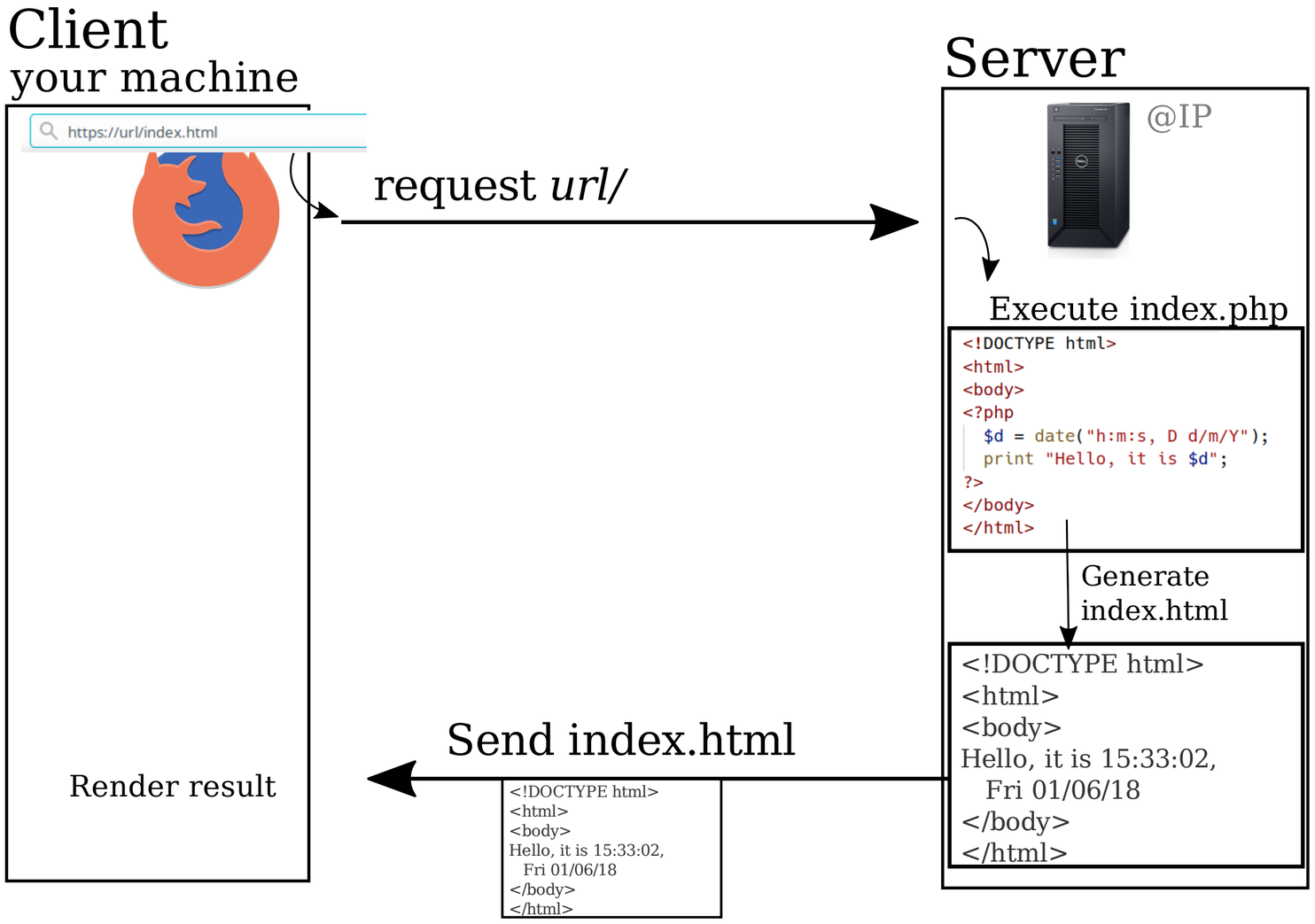
(-) Sometimes old and heavy for general client/server communication

First PHP Program

File index.php

```
<!DOCTYPE html>
<html>
<body>

<?php
  $d = date("H:i:s, D d/m/Y");
  print "Hello, it is $d";
?>

</body>
</html>
```

First PHP Program



First PHP Program - remarks

```
<!DOCTYPE html>
<html>
<body>

<?php
  $d = date("H:i:s, D d/m/Y");
  print "Hello, it is $d";
?>

</body>
</html>
```

- PHP code is included between the markups **<?php** and **?>**
- PHP code can be embedded within existing HTML
  - *Good practice: Write as much as possible HTML code. Add PHP only when needed*
- PHP code is not visible on the client browser! Only the resulting html [link]
- PHP code is executed at each query of the page on the server
- PHP variables are preceded by $
- The command *print* display its content in the resulting HTML

Executing PHP code

Need for Server

- PHP files cannot be read/interpreted directly by your browser
- PHP files are expected to be on a server

In normal/final conditions:

- PHP files are hosted on a server
  - ex. You upload your index.php file on cse104.org/yourDirectory/index.*php*
- You (the client) access the server from a URL
  - ex. You request cse104.org/yourDirectory/index.*html*

For development conditions: need to emulate a server on the local machine

Developing with PHP on your local machine

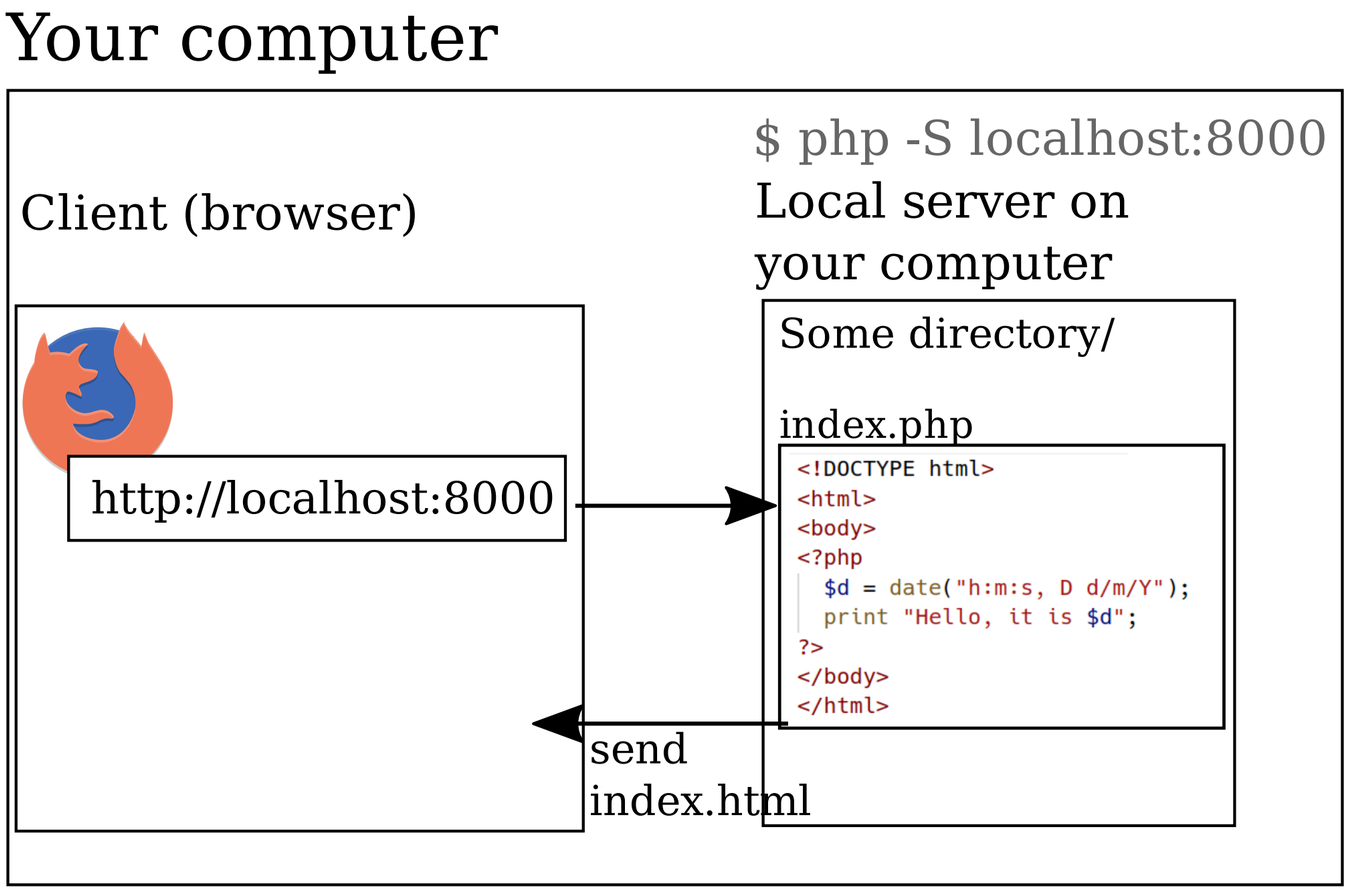
2 steps required

A. **Run a PHP interpreter** and a local server (serving the local machine)
See tutorials: Xampp / PHP

This run a server for the local machine, and interpret PHP files.
(the local machine is always called: **localhost**)

B. **Request the file** from your browser/client on **localhost**
Load your browser at the address http://localhost/

Developing with PHP on your local machine



PHP Code

Basic programming structures

- Variables: dynamically typed, preceded by $

  $a=5; $a=$a+8; print "$a";

- Arrays

  $a=[4,1,2]; a[15]="a string"; print "$a[0],$a[15]";

- For loop

  for($k=0; $k<10; $k++) {...}

- Foreach loop

  foreach($container as $value) {... $value ...};

  foreach($container as $key=>$value) {... $key ... $value ...};

Including PHP files

Can generate an HTML in aggregating several PHP files

- include("file.php"): Include "file.php" and emit a warning if file.php is not accessible.
- require("file.php"): Include "file.php" and stop the program if file.php is not accessible

```php
<?php
  require("header.php");
  require("navigation.php");
  require("body-home.php");
  require("footer.php");
?>
```

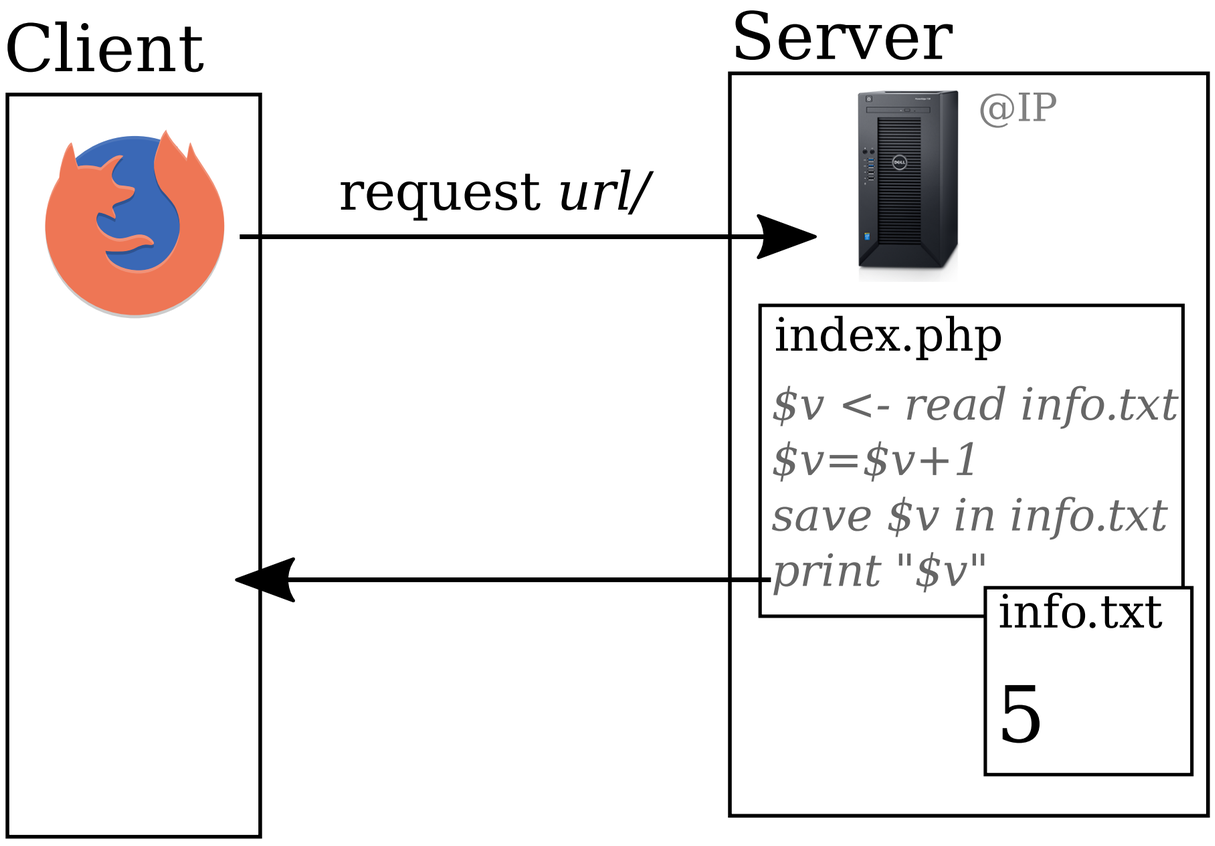
Reading/Writing files

file_get_contents : read content of a file
file_put_contents : write content to a file

Rem. Accessing files (on the client machine) in JavaScript is tricky (sandbox limitation for security issues).
Accessing files (on the server) in PHP is easy.
No security issue: only the server can access to its files
$\Rightarrow$ You can store/share any information in a permanent way through different users

Examples: Reading and drawing all images within directory, counting and storing the number of times a page has been viewed.

Reading/Writing files



GET/POST Parameters

Client can pass parameters to the server

GET method: pass parameter to the url

ex. google.com/search?q=sncf+strike

POST method: pass parameters implicitly (using HTML form, JS fetch).

Example using GET parameters on the server

GET parameters accessible through the global variable $_GET

```php
<?php
  $firstname = $_GET["firstname"];
  $lastname = $_GET["lastname"];
  print "<h1>Hello $firstname $lastname</h1>";
?>
```

Link: URL/?firstname=John&lastname=Doe

Sending parameters to the server using an HTML form

By default, forms send parameters using GET protocole

```
<form action="URL">
<ul>
<li>First Name : <input type="text" name="firstName"> </li>
<li>Last Name: <input type="text" name="lastName"> </li>
<li><input type="submit"></li>
</ul>
</form>
```

Example Link

To send a POST request - change the first line:

```
<form action="URL" method="post">
```

Node.js

Node.js

- A stand-alone JavaScript interpreter (doesn't need a browser).
- Mainly used for server program (can be used as general programming language).

Node.js / PHP

- Created in 2009: Node.js - recent
  Less common, but developing fast
- Allows the use of the same language (JS) on client and server
- Easy to develop general application (not focussed on HTML file generation)

Basic Node.js application

```
const express = require('express');
const app = express();

app.listen(8000, function () {
  console.log('Server listening on port 8000');
});

app.get('/',serverResponse);

function serverResponse(req,res) {
  res.send('Hello, this is a message from the server');
}
```

Use of *express.js* module: Efficient server development

Running node.js

- Install *express.j* module (only once)

```
npm install express
```

- Run the local server

```
node server.js
```

- Open the browser at the url *http://localhost:8000*

Hosting node.js project on a server

- Warning: Node.js project requires dedicated machine

  Usually not compatible with shared hosting

- Solution: Self-hosting, Virtual Private Server (VPS)

Using Node.js

Routing

Answering to URL interpreted as parameters

```
app.get('/',homeServerResponse);
app.get('/hello/:firstname/:lastname',helloResponse);


function homeServerResponse(req,res) {
  const response = `Home`;
  res.send(response);
}

function helloResponse(req,res) {
  const firstname = req.params.firstname;
  const lastname = req.params.lastname;
  const response = `Hello ${firstname} ${lastname}`;
  res.send(response);
}
```

Handlebars

Parameterizing/template HTML with variables

```
function helloResponse(req,res) {
  const firstN = req.params.firstname;
  const lastN = req.params.lastname;

  const placeholders = {firstname:firstN, lastname:lastN};
  res.render('file',placeholders);
}
```

file.handlebars

<body>

<h1>Hello {{firstname}} {{lastname}}</h1>

</body>

Returning JSON

```
// Return any JS object
res.send({arg1:value1,arg2:value2, ...});
```

Communicating with client: Fetch

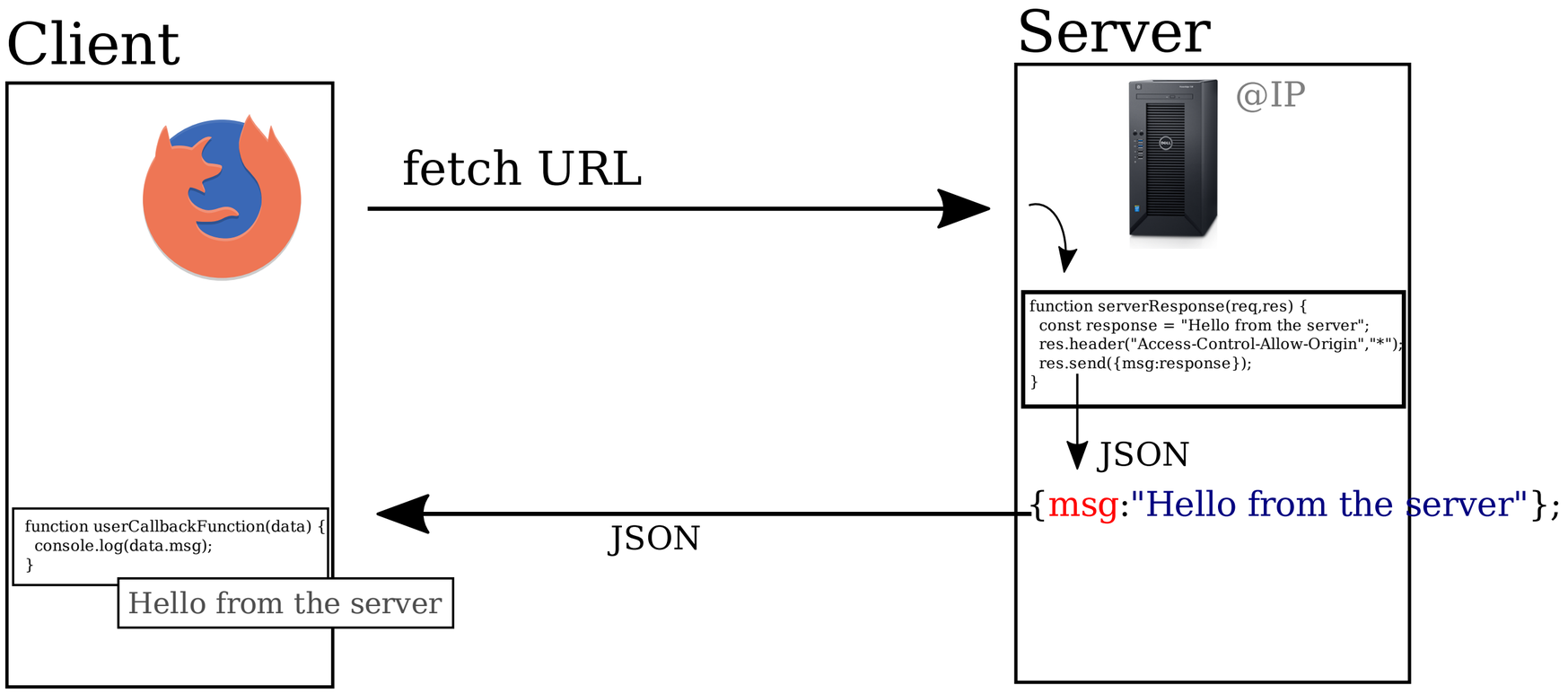
Server

```
function serverResponse(req,res) {
  const response = "Hello from the server";
  res.header("Access-Control-Allow-Origin","*");
  res.send({msg:response});
}
```

Client

```
fetch("http://localhost:8000/")
 .then(response=>{return response.json();})
 .then(userCallbackFunction);

function userCallbackFunction(data) {
  console.log(data.msg);
}
```

Communicating with client: Fetch

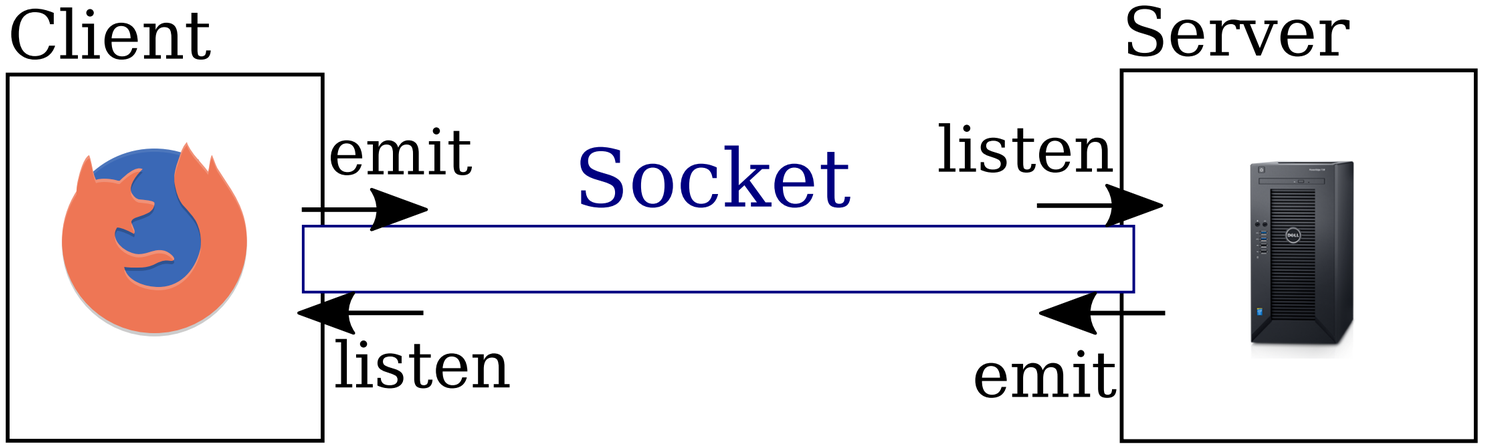


Communicating with client: Socket

Fetch allows the client to request a ressource from the server

But the server cannot contact the client

$\Rightarrow$ Soket: Allows bi-directional communication b/w client-server



Socket (module socket.io)

Server

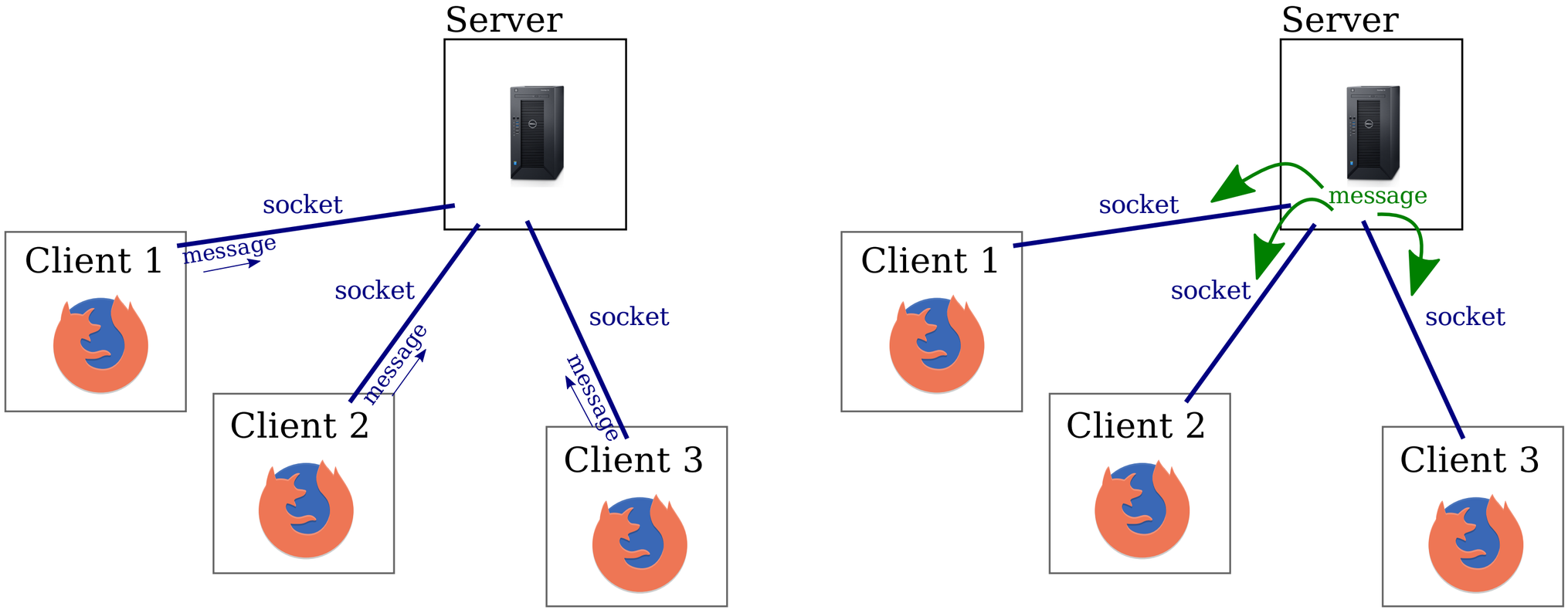
```
const io = require('socket.io')(http);
io.on('connection',onConnection);

function onConnection(socket) {
  socket.emit('message',"hello from server");
  socket.on('message',function(message){onMessage(message,socket)});
}
function onMessage(message,socket) {
  console.log('Server received message from client '+message);
}
```

Client

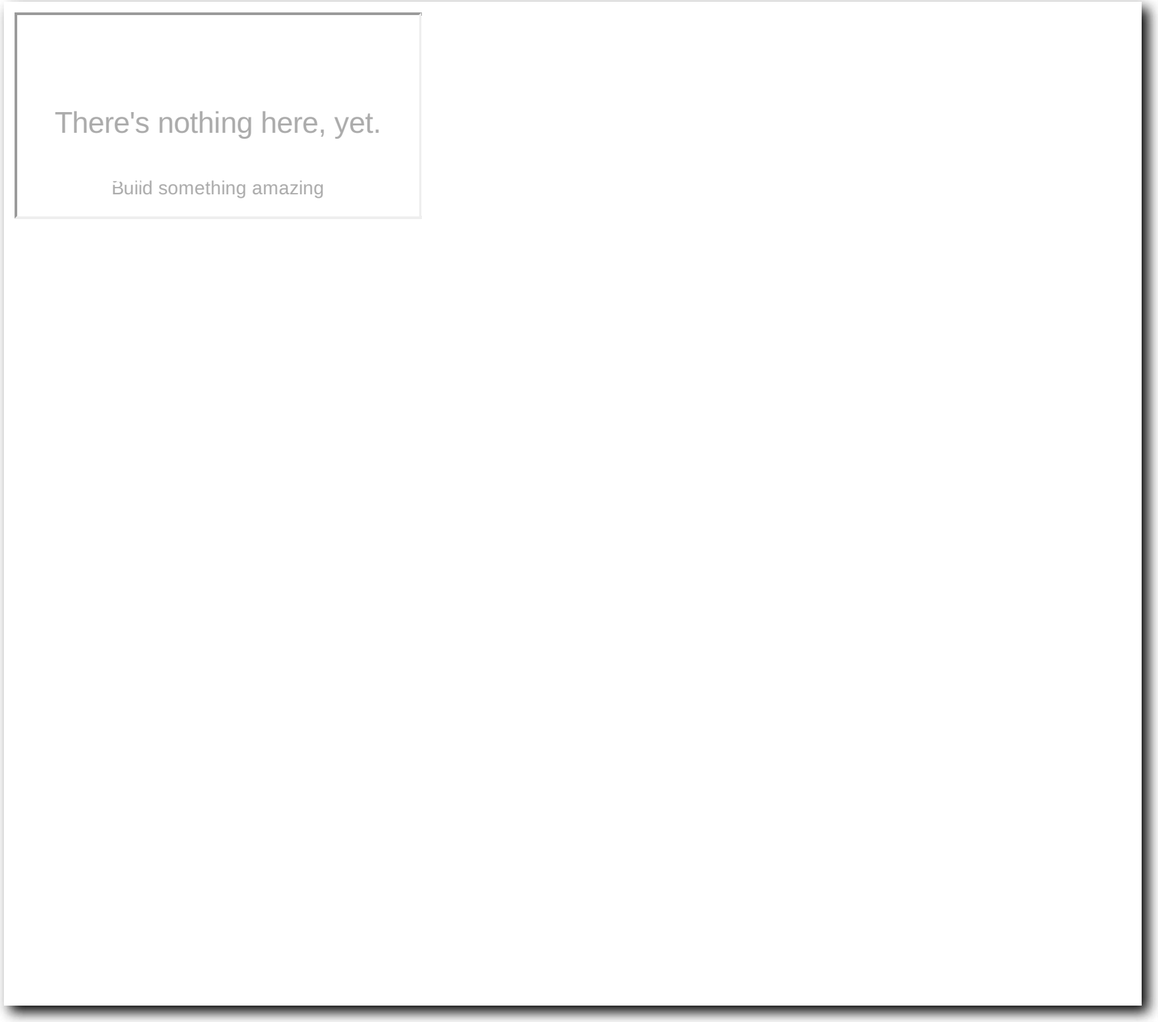
```
const socket = io.connect('http://localhost:8000');
socket.on('message',receiveMessageEvent);
socket.emit('message',"Hello from client");
function receiveMessageEvent(message) {
  console.log('Client received message from server: '+message);
}
```

Example: ChatBox

- Each client send their message to the server
- Server send back the message to every open sockets (broadcast)



Example: ChatBox



Link

```
function onMessage(message,socket) {
  // emit message to emitter client
  socket.emit('message',message);
  // emit message to other clients
  socket.broadcast.emit('message',message);
}
```