Foundations of Web Programming

IT103

Rohan Fossé

avec les slides de Jean Rémy Falleri

General information

What is IT103?

A gentle introduction to **web programming**

Why is web programming important?

- *YouTube*
  - 2005 : funded in one night by three former PayPal employees
  - 2006 : sold **1,65 billion dollars** to Google
  - now : you watch it everyday, admit it 😉
- *Wikipedia*
  - 2001 : created by Jimmy Whales and Larry Sanger on a collaborative model
  - 2015 : contains more than **40 millions articles** in 300 languages (Britannica: 40 000 articles)
  - now : it saved the day of many homework, isn’t it?

You, now

- **Familiar user** of many web applications (Snapshat, Twitter, …)
- **Rookie coder** 👩‍💻 in at least a real language (C, Python)
- Familiar with **basic networking** and knowledge of the **internet infrastructure** (IP, TCP, …)

You, after IT103

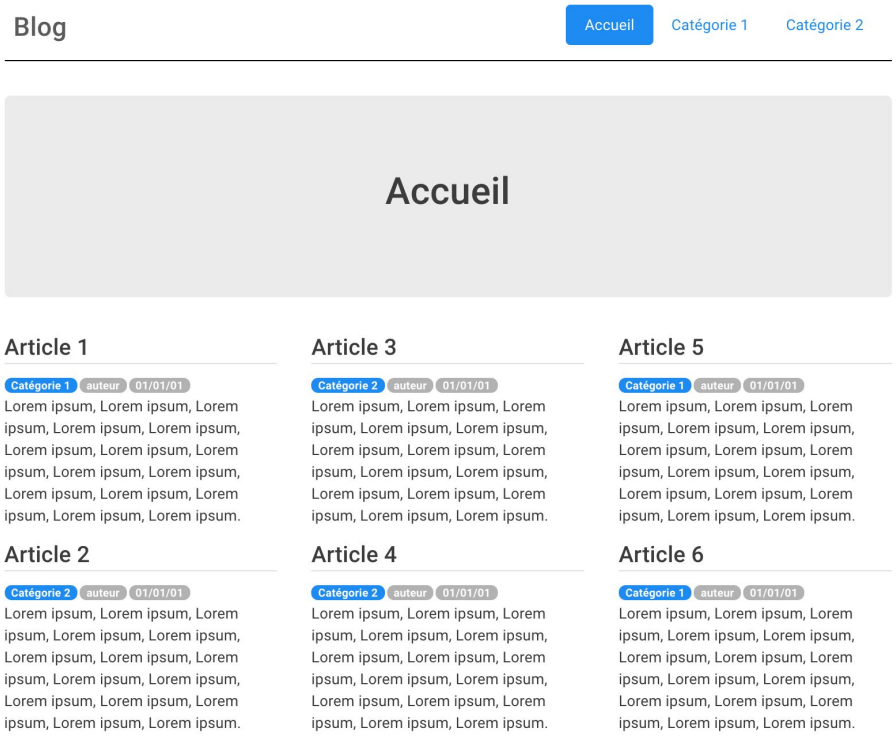
- Knowledge of the **main concepts and technologies** used in web programming
- Knowledge of the **main web applications' architectures**
- Able to create a **simple responsive web** application (desktops and phones ready) involving **server side processing** and **database storage**

What we do **not** learn

- We do not learn how to code
- We do not learn advanced client side programming (such as VueJS / React)
- You won’t be a web programming wizard by taking just one course (only years of experience will have this effect)

IT103 in practice

1. Client-side programming
   a. HTML
   b. CSS
2. Server-side programming
   a. Databases
   b. PHP



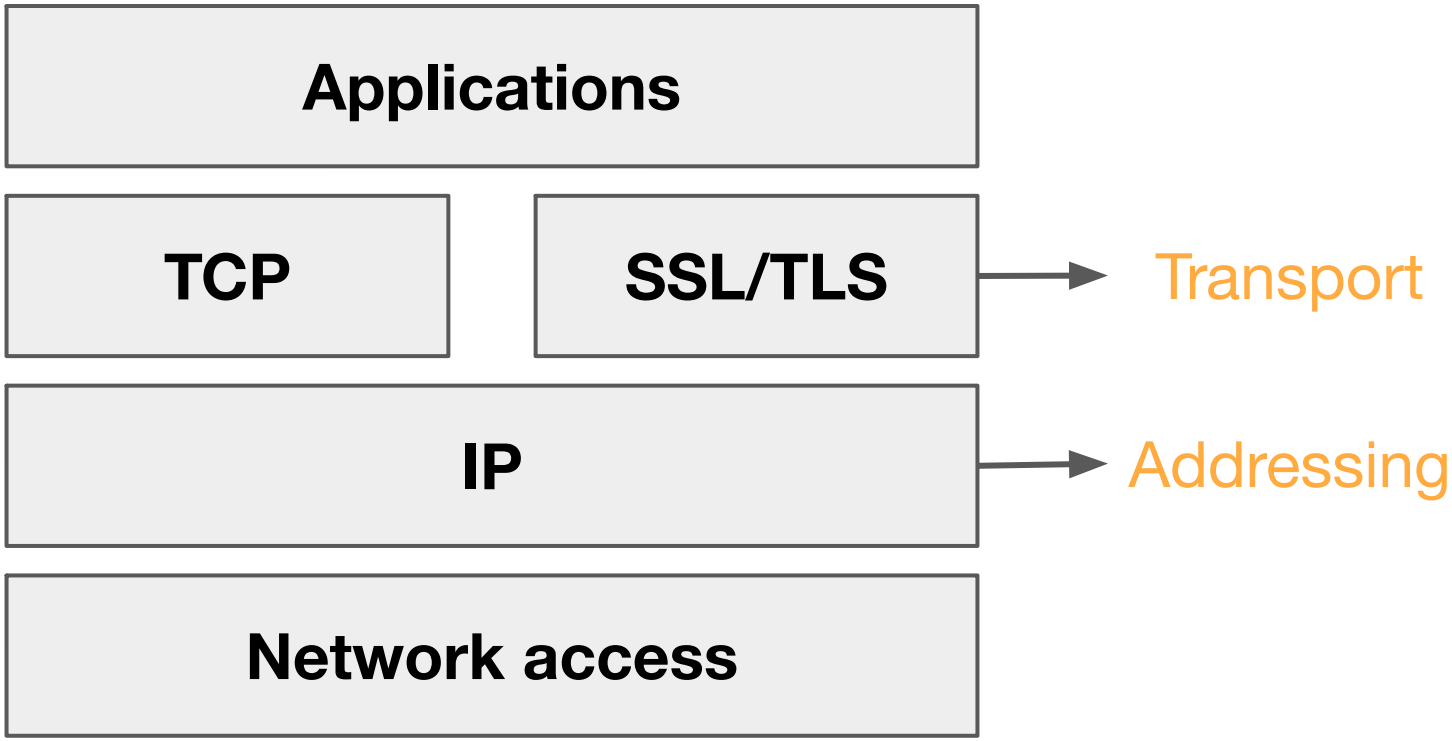
Lectures and evaluation

- Introduction lecture in an amphitheater
- Subsequent lectures in groups, inside machine classrooms
  - Tight integration between concepts and manipulation
  - Better for questions!
- Evaluation
  - MCQ (1/2)
  - Project (1/2)

Introduction

What is internet?

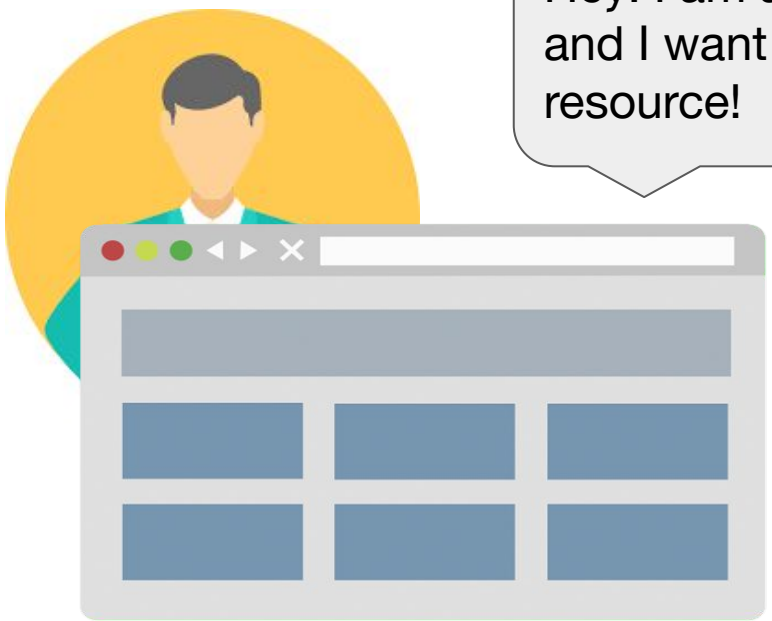
Wikipedia : a global system of interconnected computer networks



What is the web?

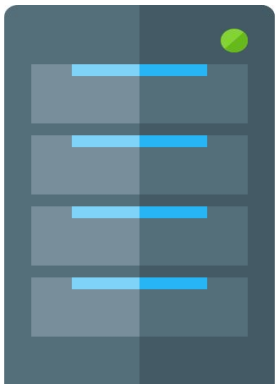
- Wikipedia: **an information system where documents [...] are accessible over internet**
- Relying on a **client-server** architecture
- Mainly standardized by the **W3C** consortium
  - HTML, CSS, …
- Other important technologies standardized by the **IETF**
  - HTTP, TCP, ...
- W3C and IETF technologies are implemented in open-source or industrial programs
  - Browsers (Firefox, Chrome, …)
  - Web servers (Apache, Nginx, …)

The **client-server** architecture



Hey! I am a client and I want a resource!

Yo! I am a server and I have plenty of resources



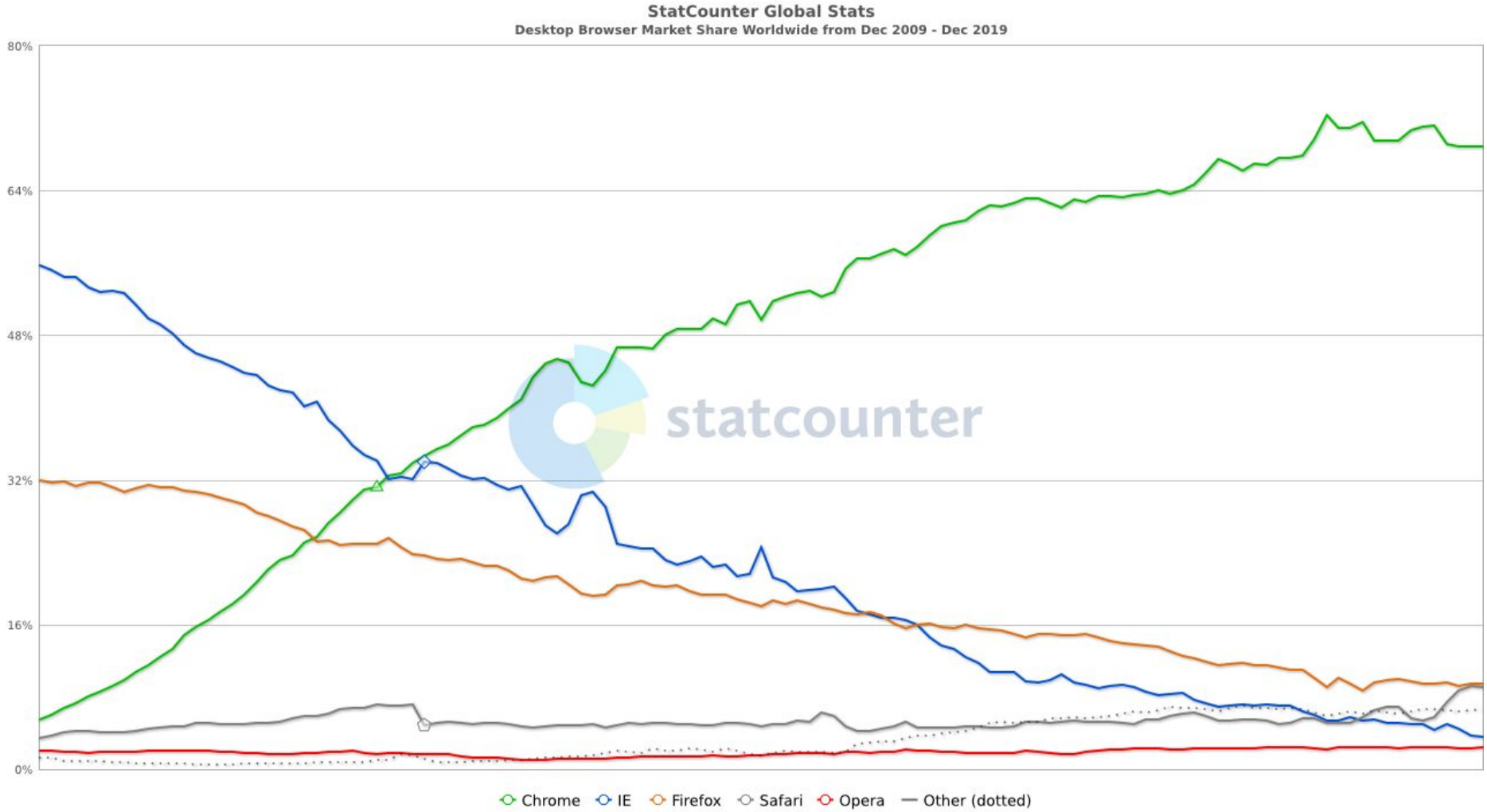
Client (Browsers)

(Web) Server

What the heck is a browser?

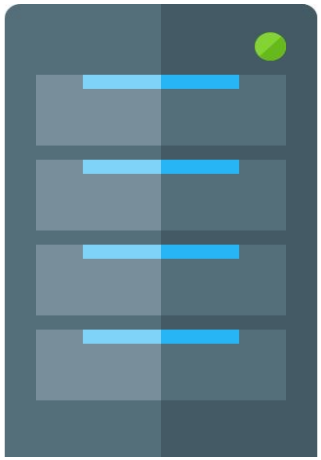
- Native program that allows a user to transparently access web resources
- Also, it can execute arbitrary code (JavaScript code only)
- Nowadays, shipped by default in most desktop operating systems
  - Safari on Mac OS
  - Edge on Windows
  - Firefox on Linux distributions
  - But the more famous is Chrome 😊

Browsers in practice

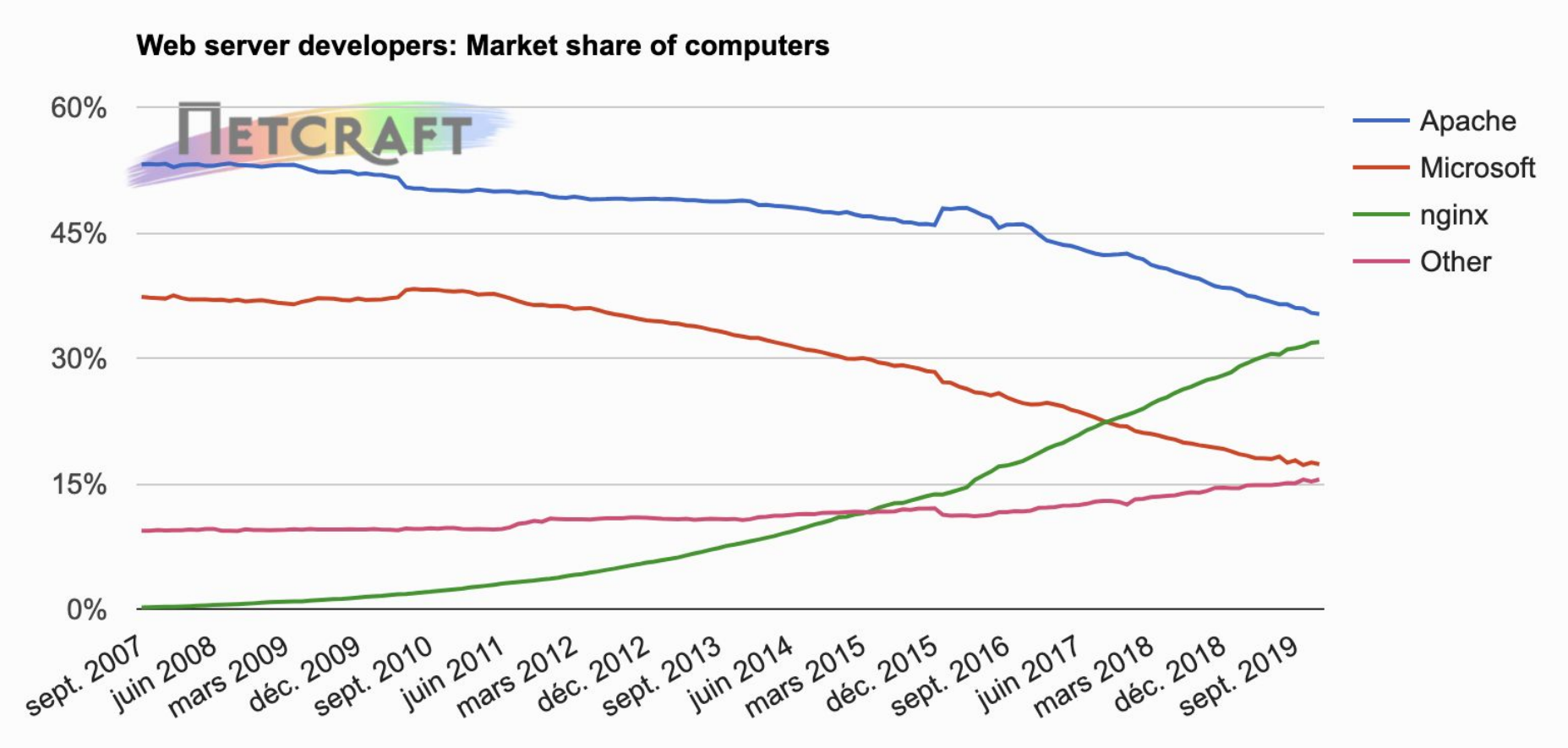


What the heck is a web server?

- A web server is nothing more than a normal machine connected to internet
- However, it has a special program, always running that listens to every incoming TCP connections
- And replies accordingly
- If you want, your laptop can become a web server : install Apache



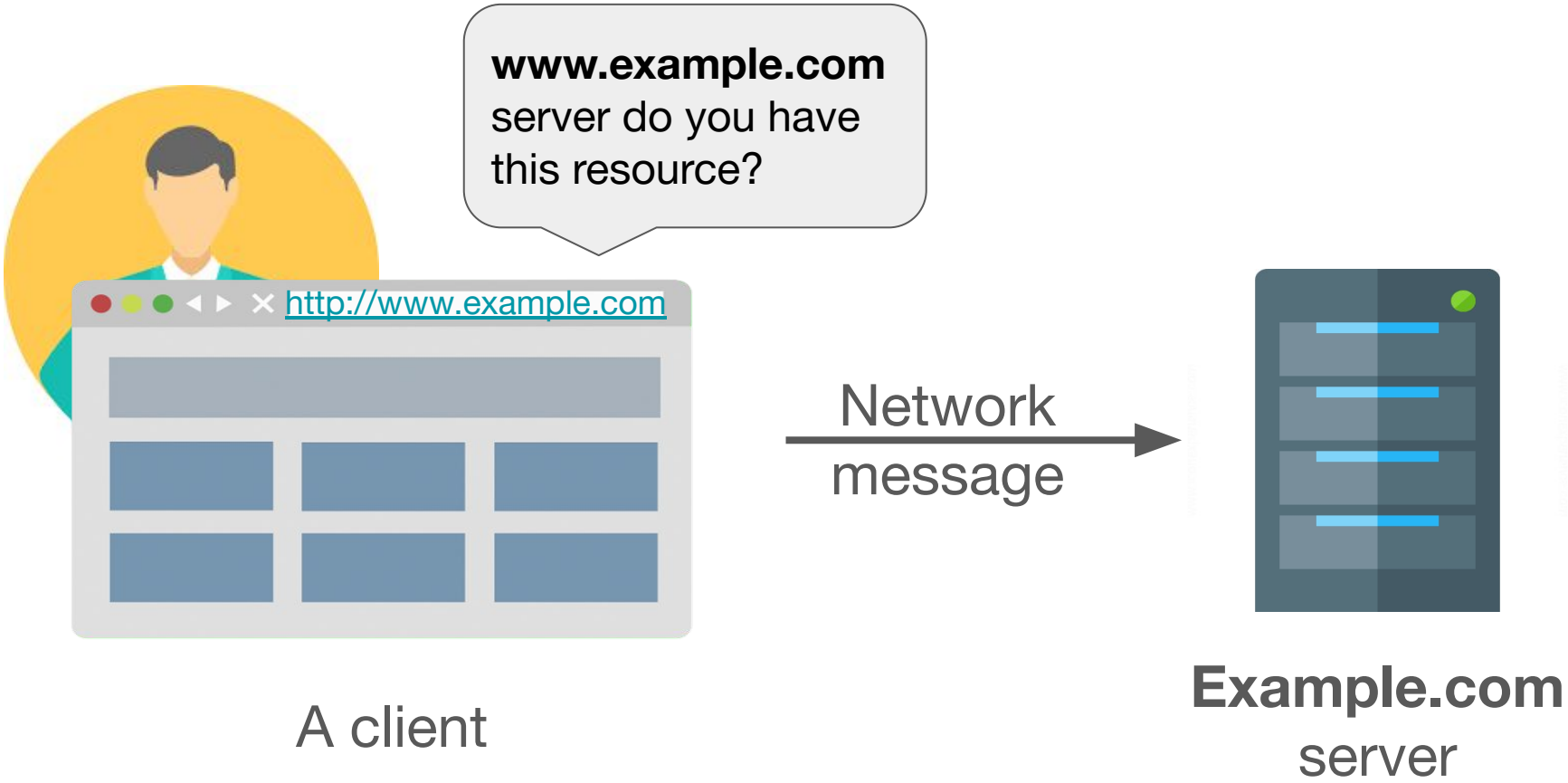
Web servers in practice



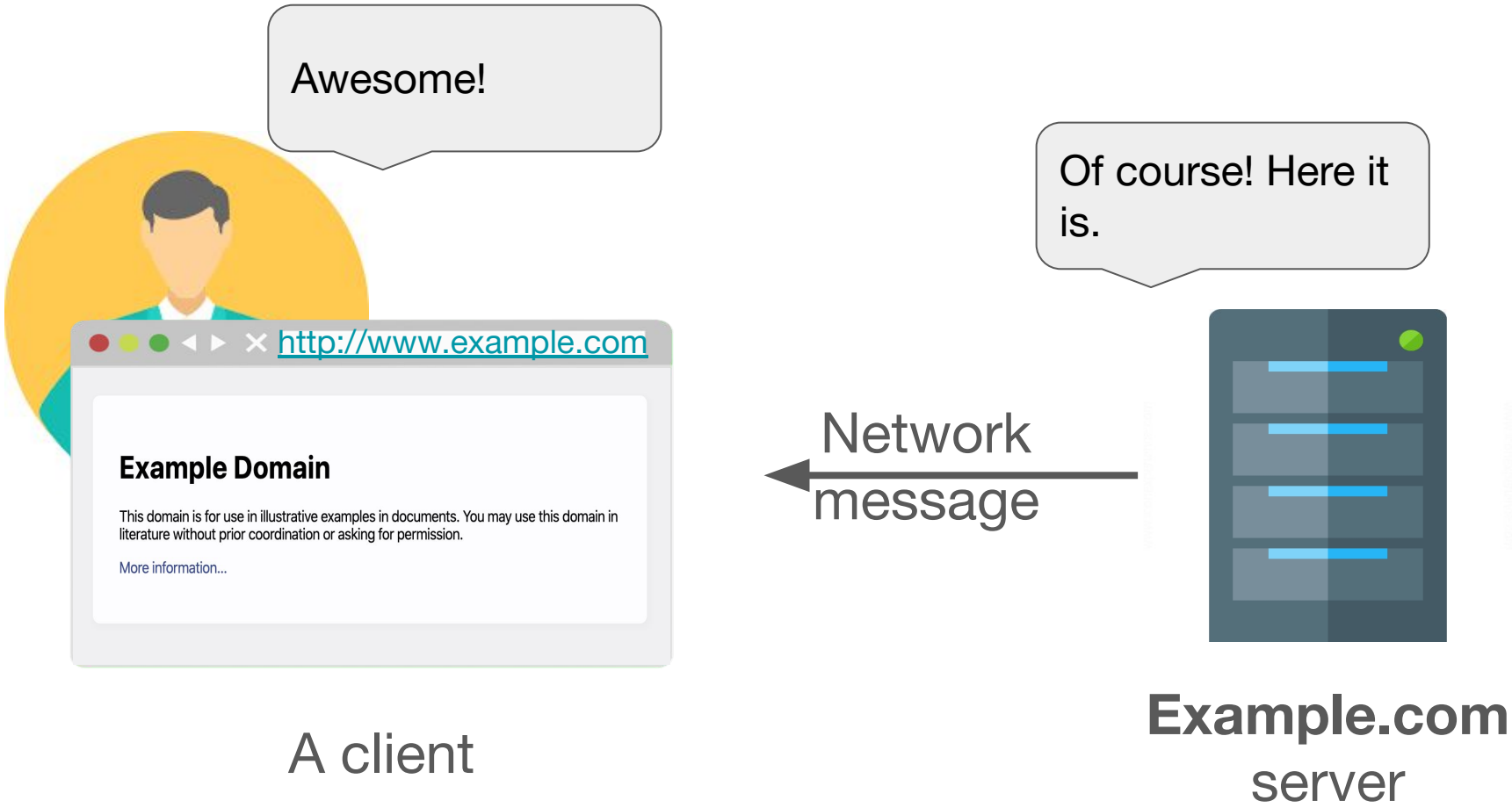
A web application **in a nutshell**

A client

A web application **in a nutshell**



A web application **in a nutshell**



A step back

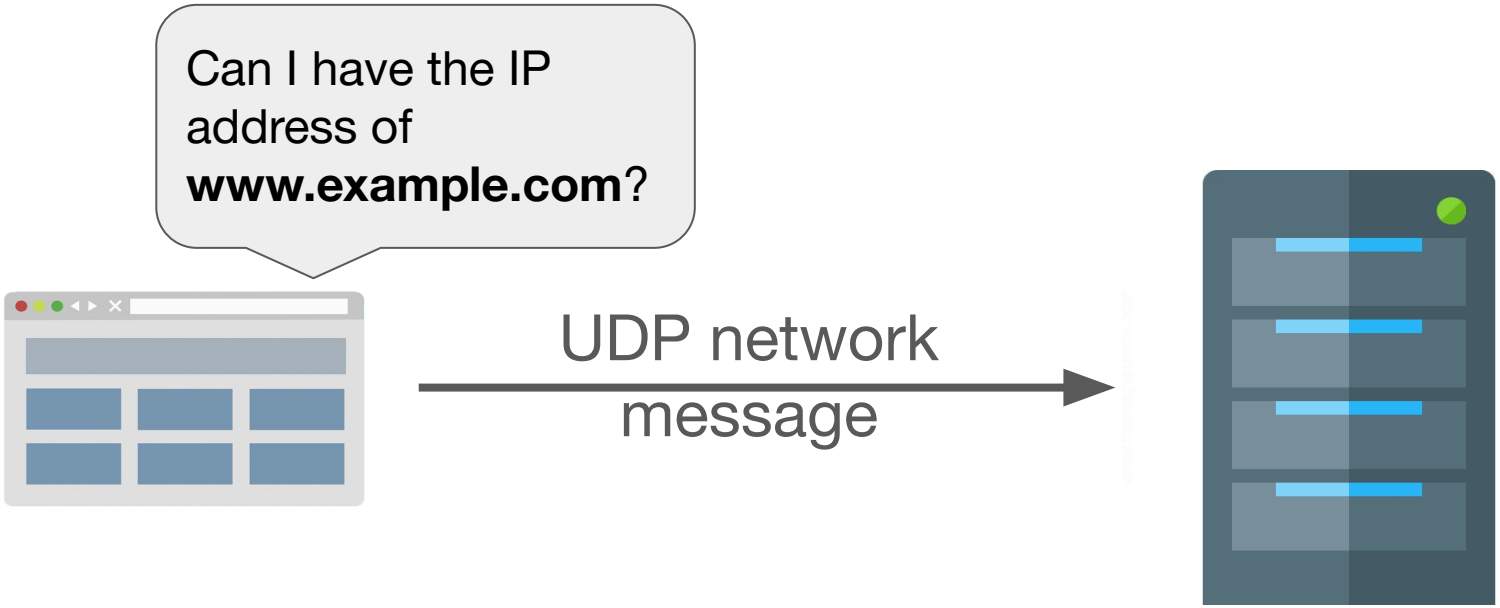
This was a rather handwavy explanation!
All started by entering **http://www.example.com** in the browser **what is this**?

A Uniform Resource Locator (URL)

http://www.example.com: no user, no password, no port (in this case the default 80 port is used), no url-path (in this case the default resource will be retrieved)

But wait www.example.com is not an IP address! How am I going to establish a network connection?

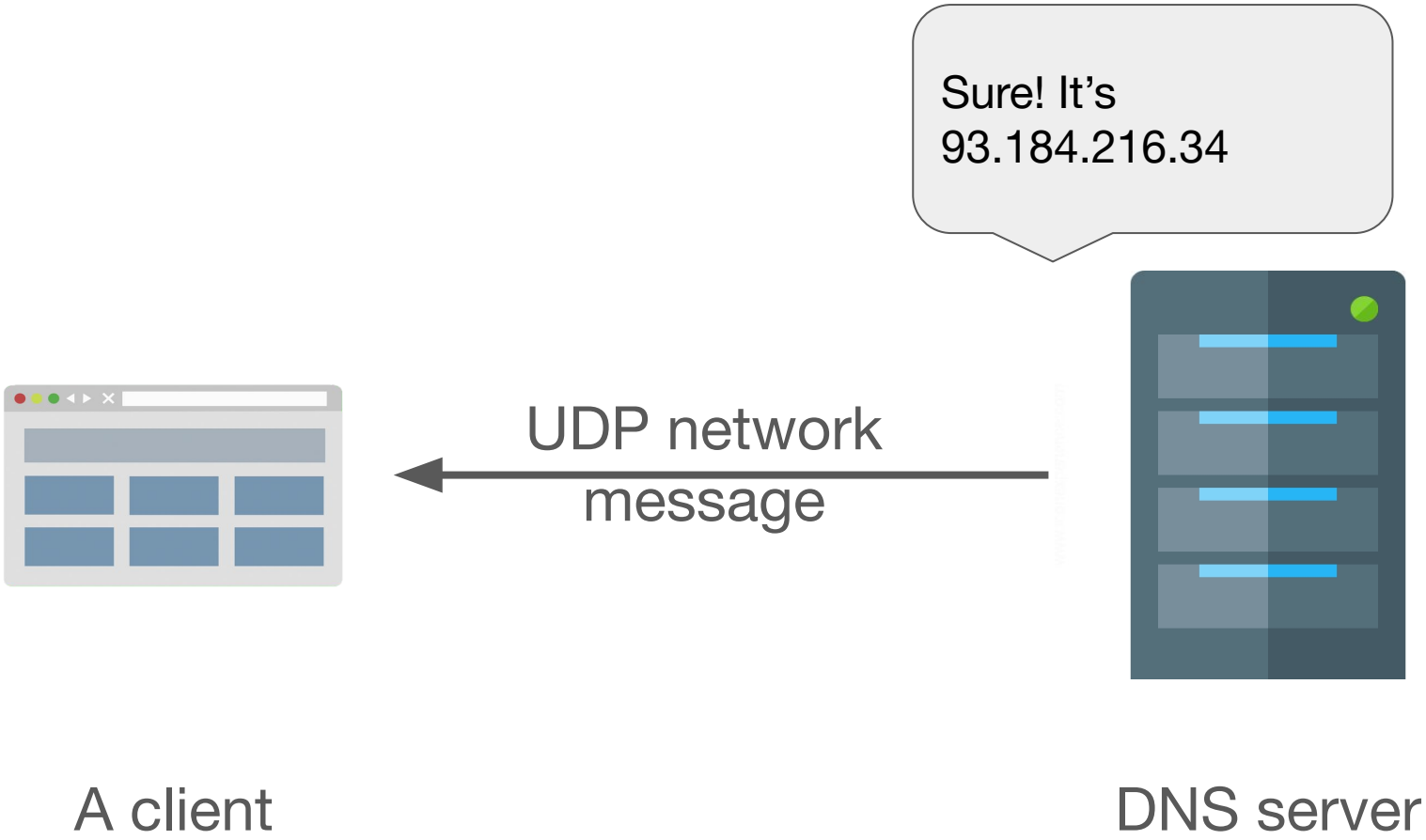
Domain Name System (DNS)



A client

DNS server

Domain Name System (DNS)



Under the hood

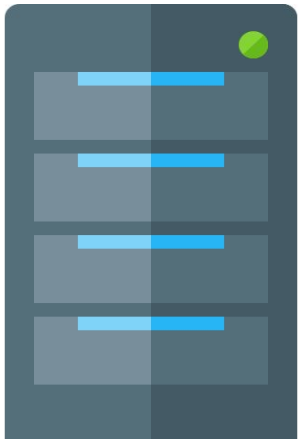
```
~ dig www.example.com

; <<>> DiG 9.10.6 <<>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32190
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4000
;; QUESTION SECTION:
;www.example.com.                IN      A

;; ANSWER SECTION:
www.example.com.        86176   IN      A       93.184.216.34

;; Query time: 8 msec
;; SERVER: 89.2.0.1#53(89.2.0.1)
;; WHEN: Tue Jan 14 09:35:32 CET 2020
;; MSG SIZE  rcvd: 60
```

IP address of www.example.com

Back to the resource exchange

How does the client tell the server that it wants the default resource?

A client

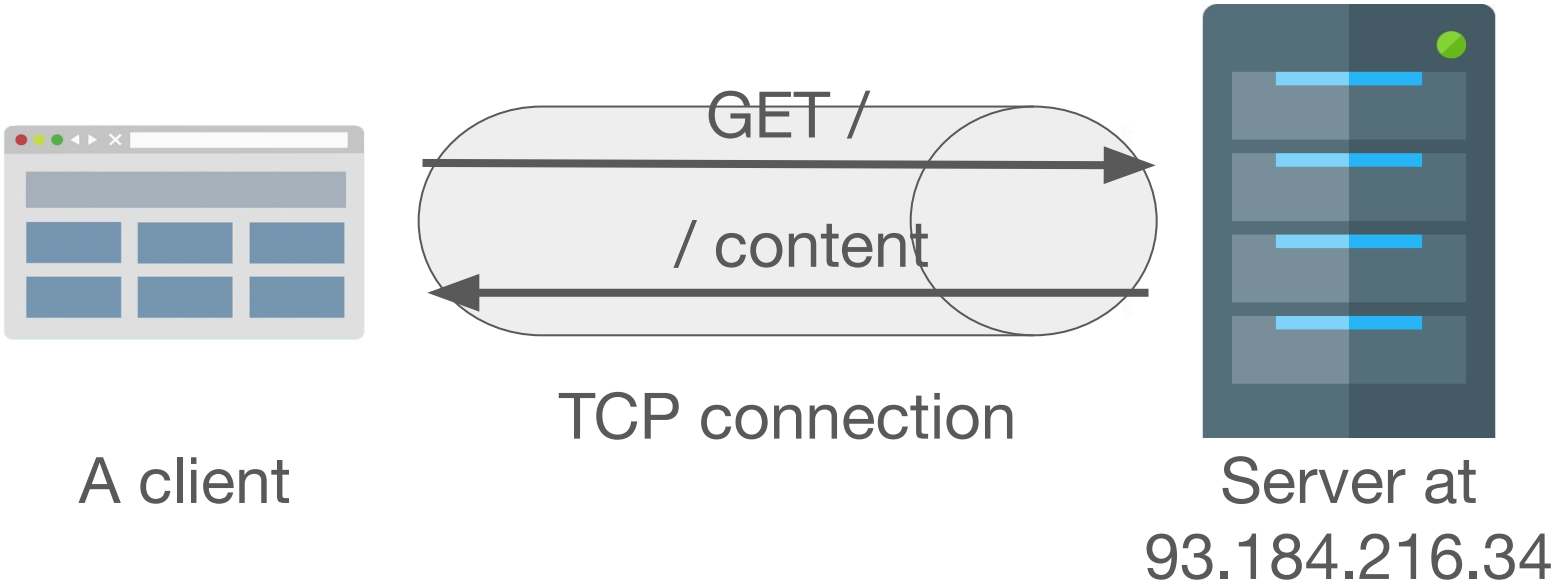


Server at
93.184.216.34

Via a dedicated protocol : http://www.example.com

Hypertext Transfer Protocol (HTTP)

- Document exchange protocol based upon TCP
- Relying on a **request-response** model
  - Client sends request to server
  - Server sends response to client
- Several types of requests : **GET** to retrieve a resource



1. fish /Users/falleri (fish)

```
~ telnet www.example.com 80
Trying 2606:2800:220:1:248:1893:25c8:1946...
Connected to www.example.com.
Escape character is '^]'.
GET /index.html HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Mon, 11 Jan 2016 13:40:59 GMT
Etag: "359670651"
Expires: Mon, 18 Jan 2016 13:40:59 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (ewr/144C)
Vary: Accept-Encoding
X-Cache: HIT
x-ec-custom-error: 1
Content-Length: 1270

<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
        margin: 5em auto;
        padding: 50px;
```

Telnet client for raw TCP connections

Client's request

Headers

Server's response

Data

1. telnet /Users/falleri (telnet)

```
~ telnet www.example.com 80
Trying 93.184.216.34...
Connected to www.example.com.
Escape character is '^]'.
GET /index.html HTTP/1.1
Host: www.example.com
Accept-Encoding: gzip

HTTP/1.1 200 OK
Content-Encoding: gzip
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html
Date: Tue, 27 Mar 2018 09:25:45 GMT
Etag: "1541025663+gzip"
Expires: Tue, 03 Apr 2018 09:25:45 GMT
Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT
Server: ECS (dca/53DB)
Vary: Accept-Encoding
X-Cache: HIT
Content-Length: 606
```

Headers have drastic effects!

Example Domain

Non sécurisé www.example.com

dev travaux recherche enseignement loisirs admin

Example Domain

This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.

More information...

HTML

Elements Console Network 1

View: Group by frame Pre

Filter Hide data URLs

All XHR JS CSS Img Media Font Doc WS Manifest Other

Name

www.example.com

1 requests | 935 B transf...

Headers Preview Response

General

Request URL: http://www.example.com/

Request Method: GET

Status Code: 200 OK

Remote Address: 93.184.216.34:80

Referrer Policy: no-referrer-when-downgrade

Response Headers view source

Cache-Control: max-age=604800

Content-Encoding: gzip

Content-Length: 606

Content-Type: text/html

Date: Tue, 27 Mar 2018 09:28:59 GMT

Etag: "1541025663+gzip"

Expires: Tue, 03 Apr 2018 09:28:59 GMT

Last-Modified: Fri, 09 Aug 2013 23:54:35 GMT

Server: ECS (dca/5327)

Vary: Accept-Encoding

X-Cache: HIT

Request Headers view source

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8

Accept-Encoding: gzip, deflate

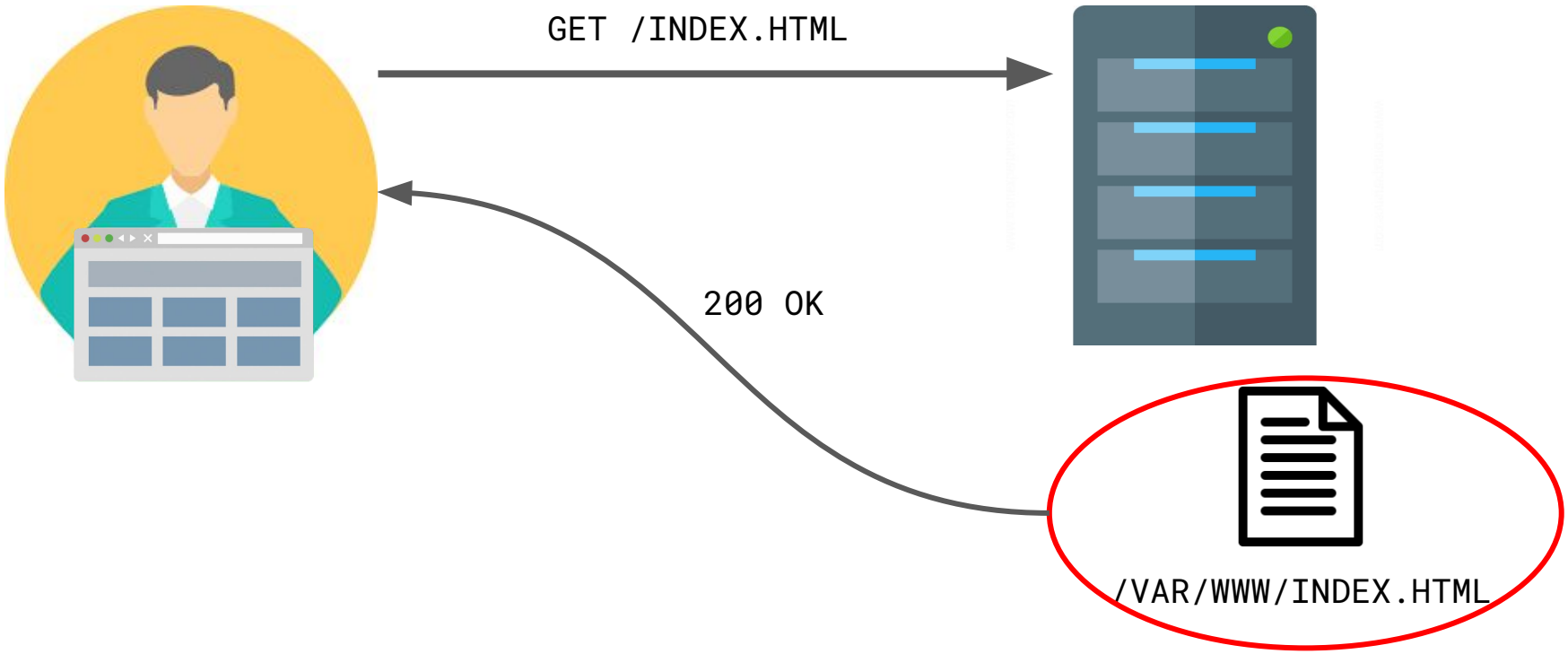
Accept-Language: fr,en-US;q=0.9,en;q=0.8

Cache-Control: max-age=0

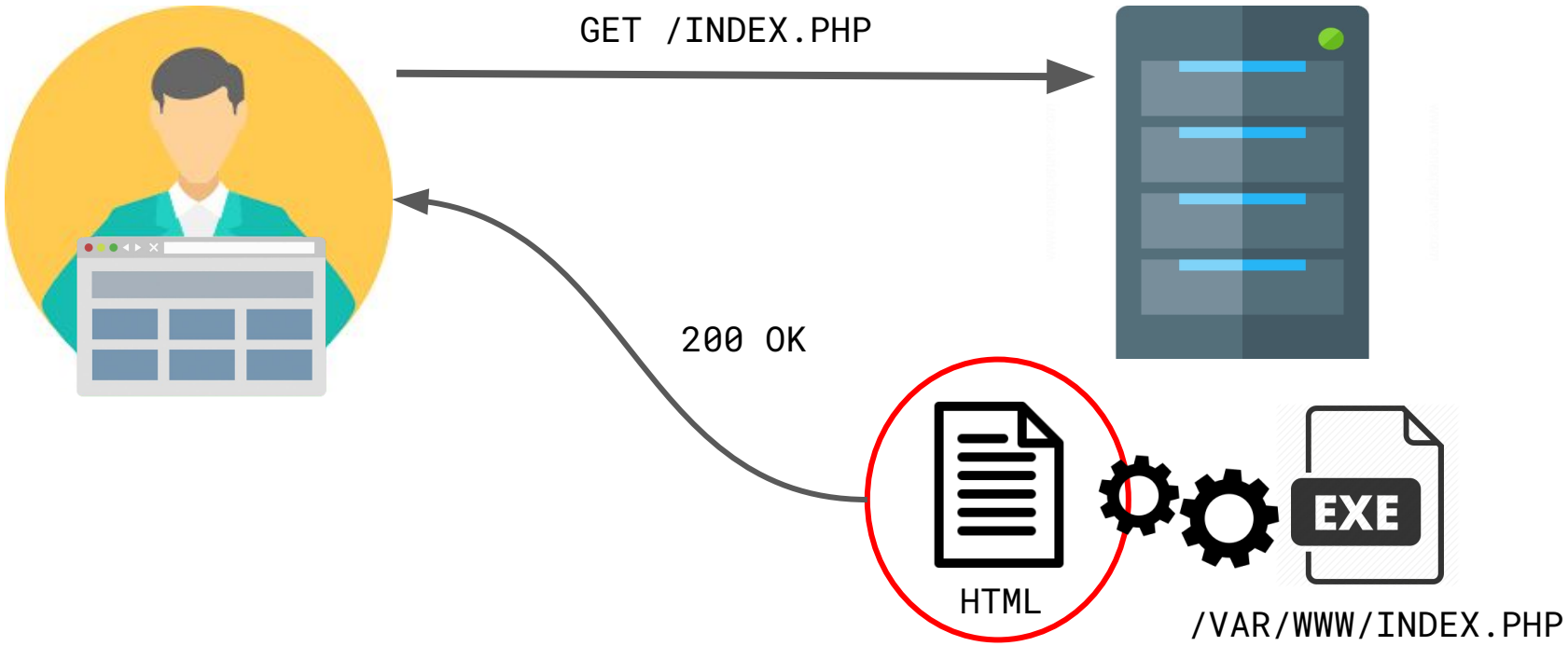
Web applications

- Client-server applications running through the web
- Users interact with them using a browser
- Competitive advantage : no client deployment!
- Major drawbacks :
  - Web GUIs are not so great
  - Severe cost and technical challenges w.r.t. servers
  - Works often poorly when the network is down

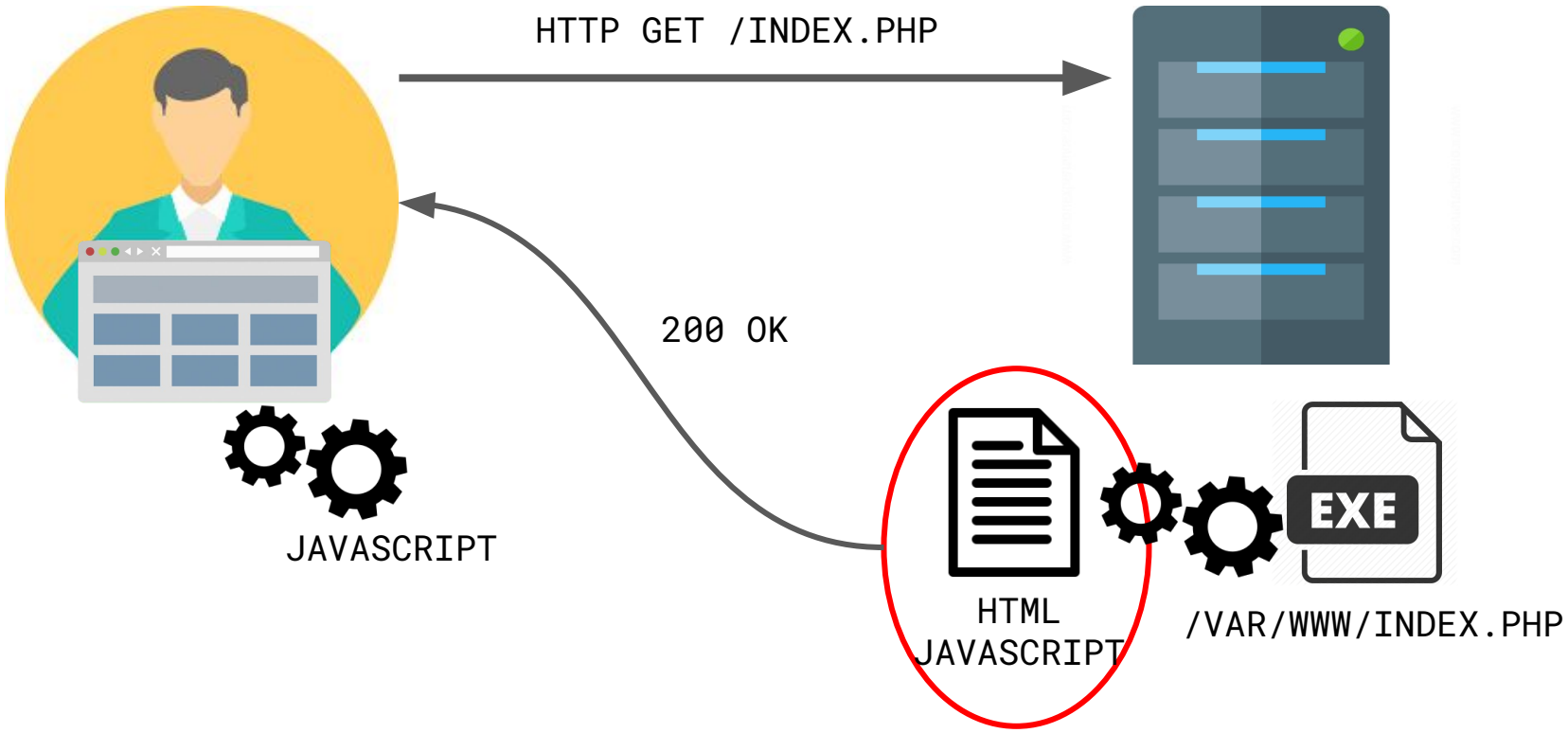
Static web applications



Server-dynamic web applications



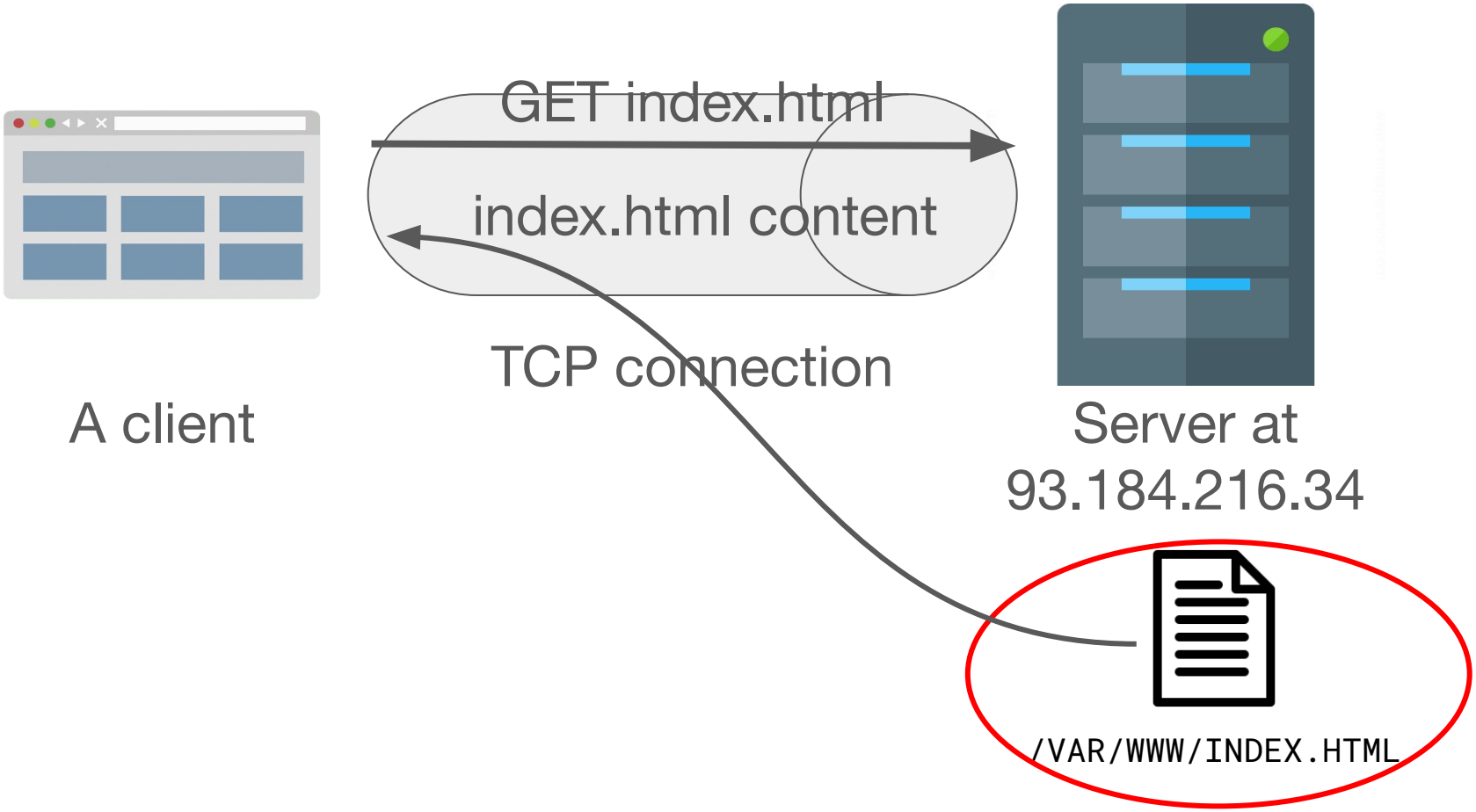
Server and client-dynamic web applications



HTML

Previously in IT103

A **static** web application



Take home message

Mastering static web applications **is the same as** mastering resources that are placed on a web server

- HTML resources (a logical document) **today**
- CSS resources (aesthetic properties) **next episode**
- Some binary resources

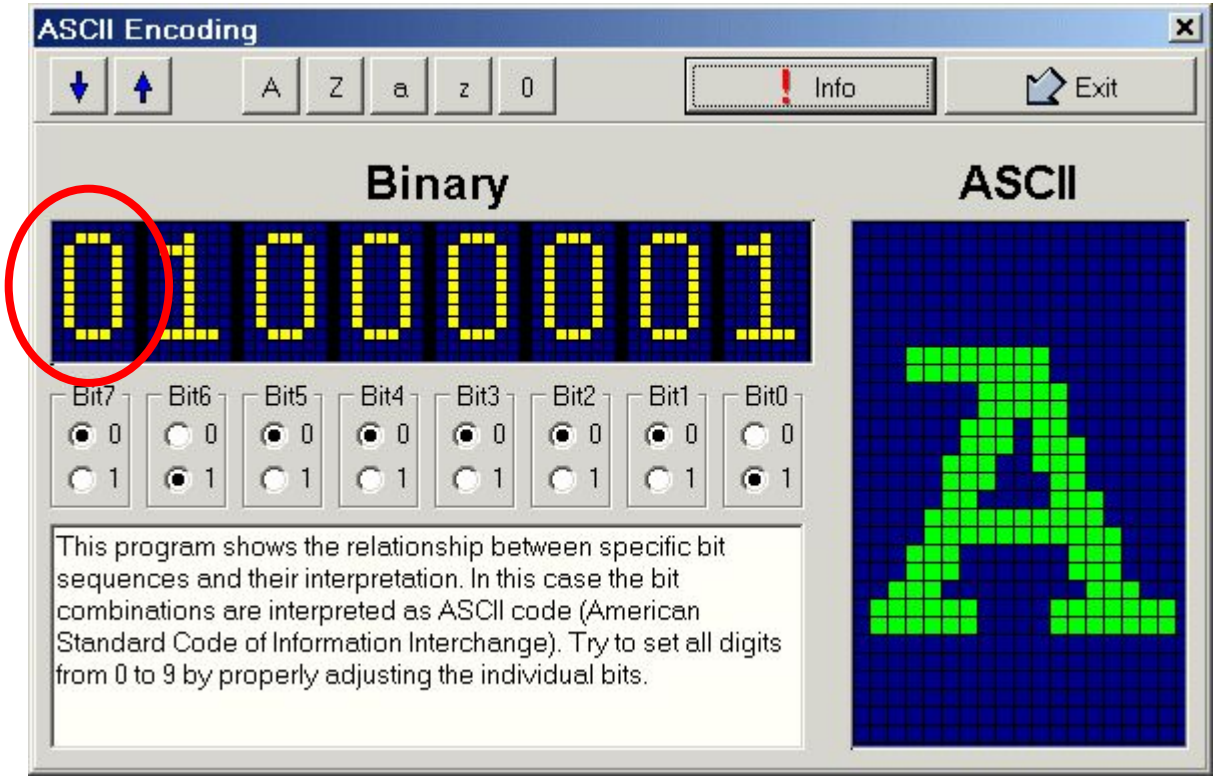
Before digging deeper, let’s get back to a more boring resource : **a text resource**

Plain old text

- Computer memories store **sequences of 0 and 1** (bits) this is not text
- Then how to make text out of bits?
- We need a technique to encode/decode text characters to/from bits
- Decoded characters are shown to the user using images installed in the OS : fonts
- OK! So what is **011011000110111101101100** ?

The ASCII table

useless



Plain old text

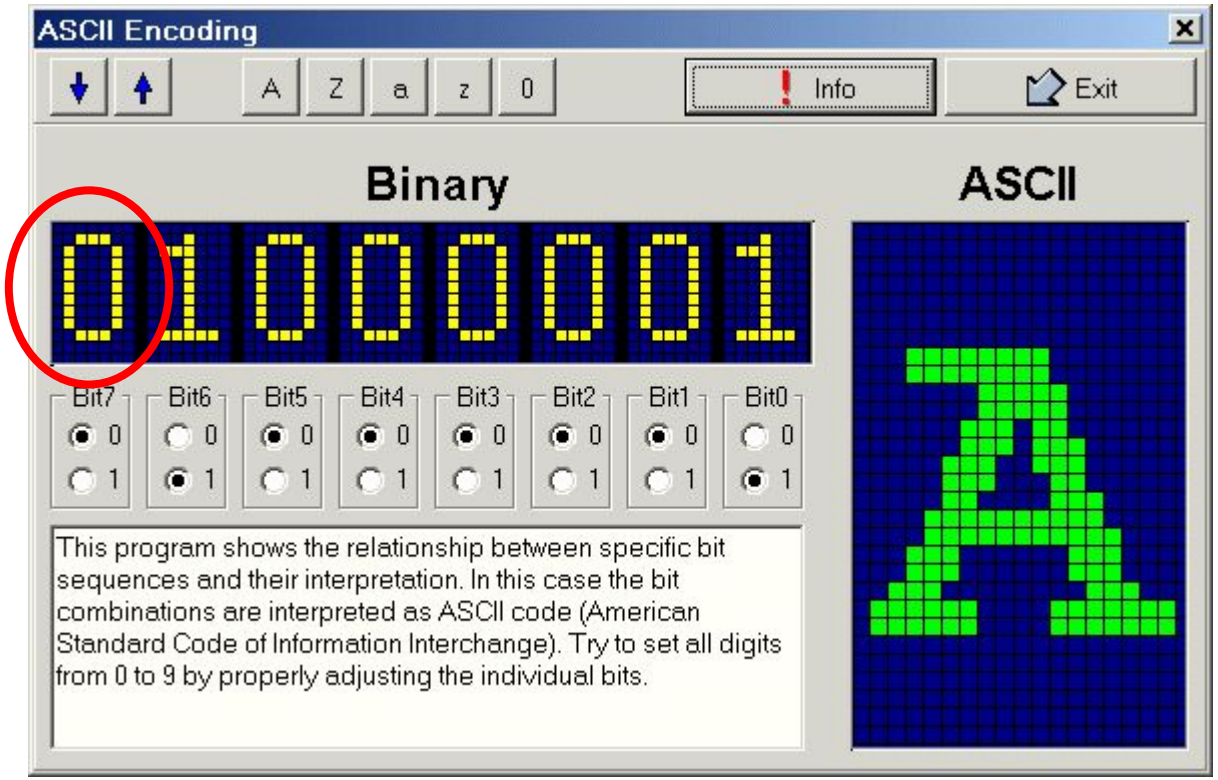
```
01101100 01101111 01101100
```

L O L

Problem: 7 bits are 128 values, far less than all possible text characters!

In the hell of the ISO-* tables

Let's use this damn bit!



Yay! Extra 128 characters! One encoding/decoding table per language though 😢

The UTF tables

character	encoding	bits
A	UTF-8	01000001
A	UTF-16	00000000 01000001
A	UTF-32	00000000 00000000 00000000 01000001
あ	UTF-8	11100011 10000001 10000010
あ	UTF-16	00110000 01000010
あ	UTF-32	00000000 00000000 00110000 01000010

Variable-length text characters, using the last bit!
Nearly perfect solution, UTF-8 is 👑

Why this fuss about text?

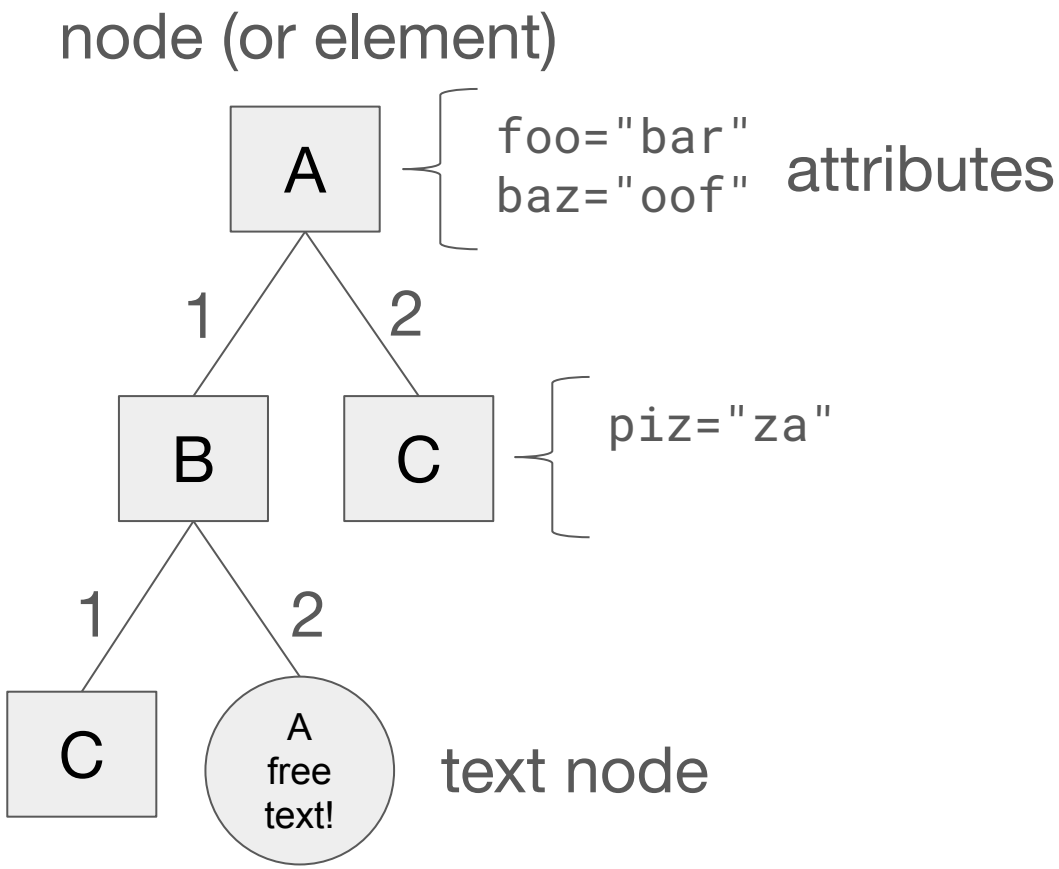
HTML resources contains primarily text, so you have to know how it works unless you like showing ⍰ to the users

- You'll need to know what "kind" of text your editor produces
- You'll have to tell the browser which table to use to decipher your text

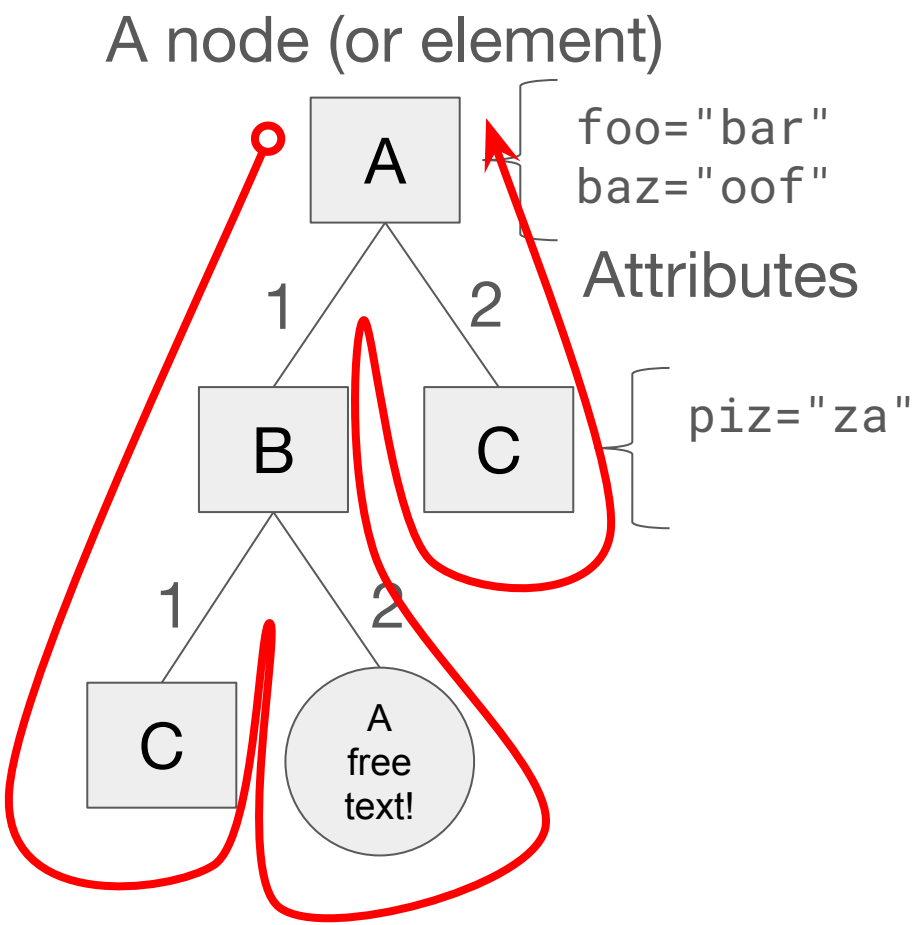
Now: Hypertext Markup Language (HTML)

- We just saw how to encode text characters into a sequence of bits
- Similarly, HTML encodes a tree into a text (i.e. a sequence of text characters)
- Before presenting HTML, I will present the more general eXtended Markup Language (HTML is a special case of XML)
- You'll learn one language for free, how cool is that?

A sample XML tree



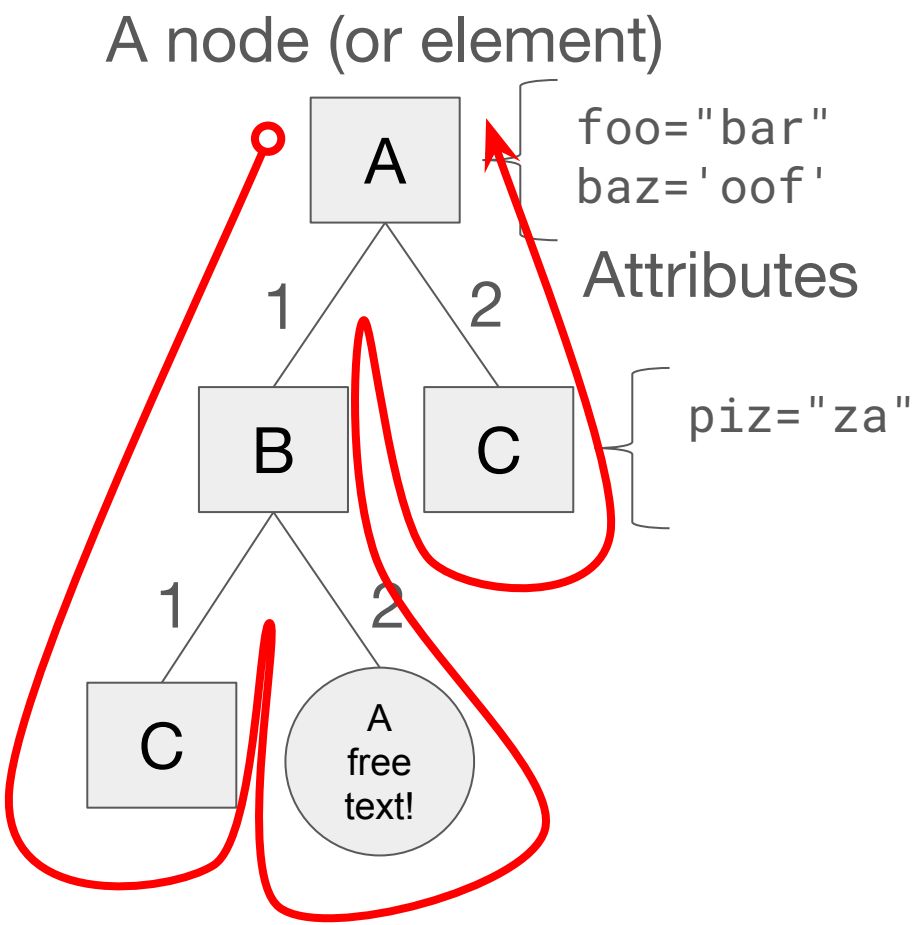
XML tree traversal



Rules :

- When entering a node, output a opening tag (<a>) with attributes
- When exiting a node output a closing tag ()
- For free text, just recopy the free text

XML tree traversal



XML code :

```
<a foo="bar" baz='oof'>
    <b>
        <c>
        </c>
    A free text!
    </b>
    <c piz="za"/>
</a>
```

Free text white-spaces peculiarities

Original text:

It·is···an·awesome·text!↵
↵
␣indented text!

Parsed text:

It·is·an·awesome·text!
·indented·text!

Don’t put too much effort in formatting your free text
😉

XML/HTML entities and comments

- Trouble ahead : imagine your free text contains **`<`**
- You have entities that are of the form **`<`**
  - **` `**
  - **`&`**
  - **`>`**
- You can put comments using the following weird syntax
  `<!-- awesome comment -->`

XML superpower

- Awesome language to define a user-format without having the burden of writing a parser
- You want to store a list of students in a text file?

```
<students>
   <student id="1">
        <first_name>Joe</first_name>
        <last_name>Bar</last_name>
    </student>
</students
```

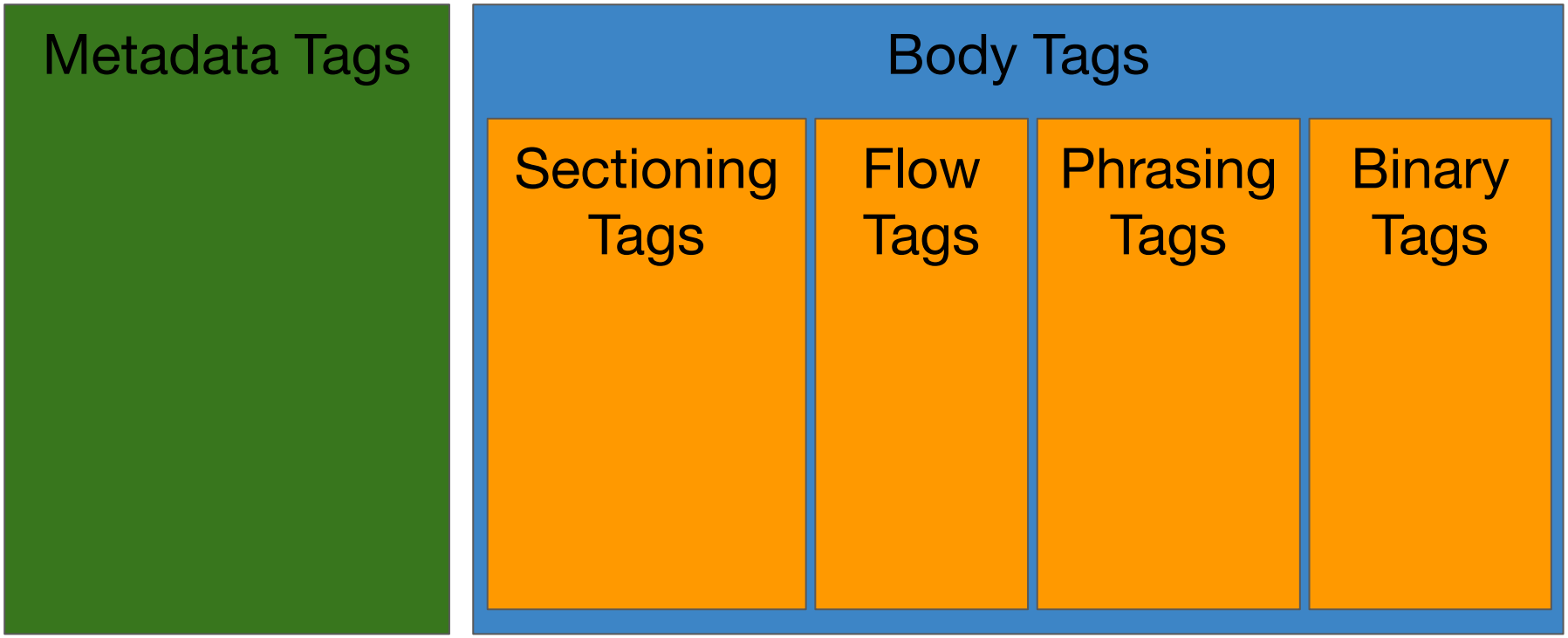
Nice! But what about damn HTML?

- HTML is just a particular case of XML where you don't get to choose nor the node labels neither the attributes
- In fact XHTML is the particular case of XML, HTML has one particularity
- Some tags, which are known to be leaf tags, do not need closing tags (i.e.
)
- In the remainder we will focus on HTML 5 (beware of outdated online doc! protip: <font> no longer exists 😉)

A HTML skeleton

```
<!DOCTYPE html><!-- HTML5 document -->
<html>
    <head>
        <!-- metadata -->
    </head>
    <body>
        <!-- content -->
    </body>
</html>
```

Categories of HTML tags



Go into <head>

Go into <body>

Metadata tags, the best-of

- `<title>Browser tab’s title not the real title</title>`
- `<meta>`
  - `<meta charset="utf-8">`
  - **Perfect example of a tag without closing tag because HTML knows it has no children**
- `<script src="mycode.js"></script>`
- `<style>`
- `<link href="style.css" rel="stylesheet">`

Body tags

The four categories goes from the most abstract tags (indicating the structure of the resources) to the most low-level tags. The order is:

1. Sectioning
2. Flow
3. Phrasing
4. Binary

Sectioning tags, the best-of

- `<header>`
- `<footer>`
- `<section>`
- `<article>`
- `<aside>`
- `<div>`

Flow tags, the best-of

- `<p>a paragraph</p>`
- `<a href="http://www.google.fr">Google!</a>`
- `<ul><li>a bullet</li><li>an other</li></ul>`
- `<table><tr><td>line1 col1</td></tr></table>`
- `<h1>..<h6>`
- `<div>`

Phrasing tags, the best-of

- `<em>`
- `<strong>`
- `<mark>`
- `<span>`

Binary tags, the best-of

- `<img src="picture.jpg">`
- `<audio src="sound.mp3">`
- `<video src="movie.mp4">`

Now, your turn to work!

- Go make the blog!
- When using a tag for the first time, read the doc!
- Use the W3C validator frequently
- Set-up correctly your text editor
- It’s not pretty? We don’t care! We’ll get to that next time

CSS

Last episode’s blog

Titre du blog

Section A

Post 1

- Date: d
- Auteur: a

Contenu

Post 2

- Date: d
- Auteur: a

Contenu

Section B

Post 1

- Date: d
- Auteur: a

Contenu

Ugly 🤮

How do I turn

This

Titre du blog

Section A

Post 1

- Date: d
- Auteur: a

Contenu

Post 2

- Date: d
- Auteur: a

Contenu

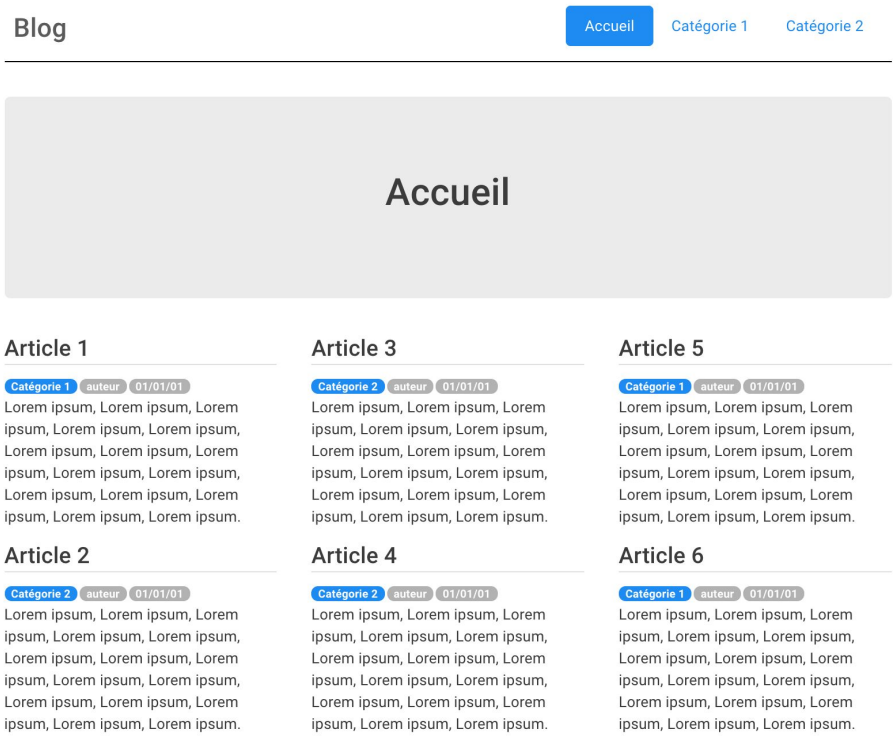
Section B

Post 1

- Date: d
- Auteur: a

Contenu

Into this?



Cascading Style Sheets

A **CSS rule** has a **selector** and contains multiple **declarations** (here one):

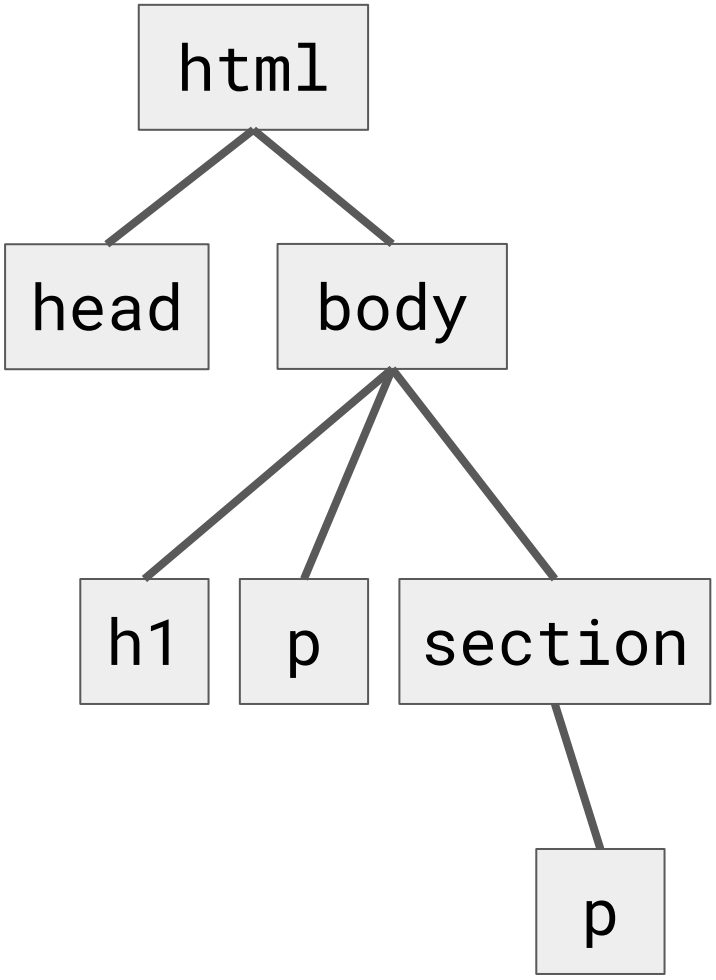
```
selector {
   property: value;
}
```

How does that works?

```
selector {
   property: value;
}
```

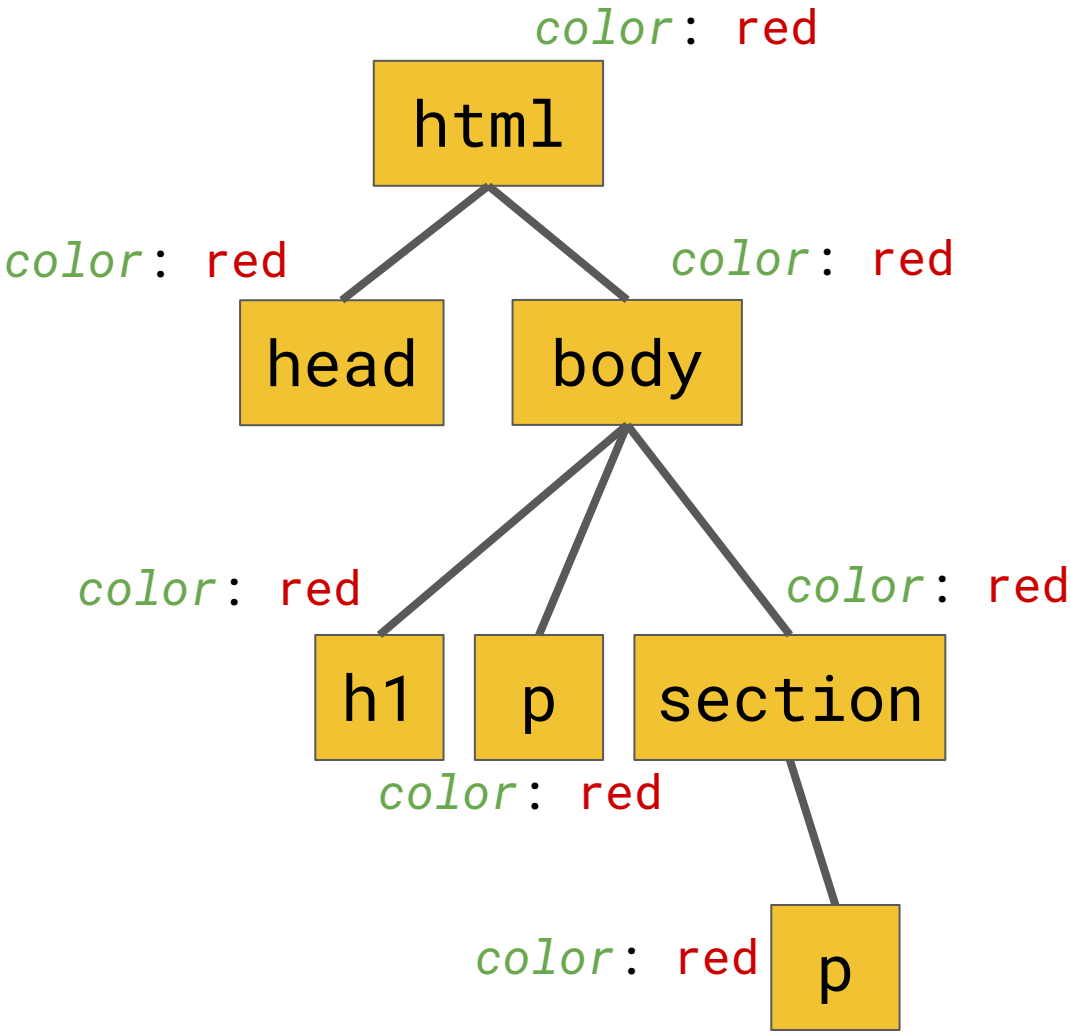
The selector selects a subset of the HTML tree's nodes and apply the declaration to them

Declaration have graphical meaning that will be applied by the browser



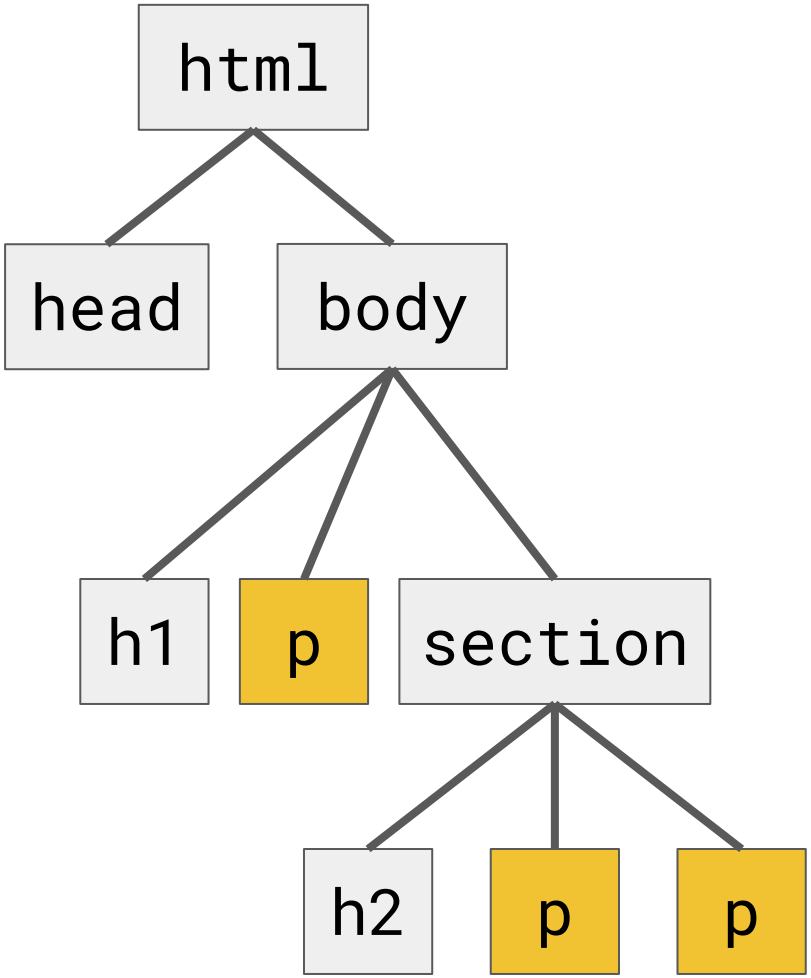
First example with the *joker* selector

```
* {
  color: red;
}
```



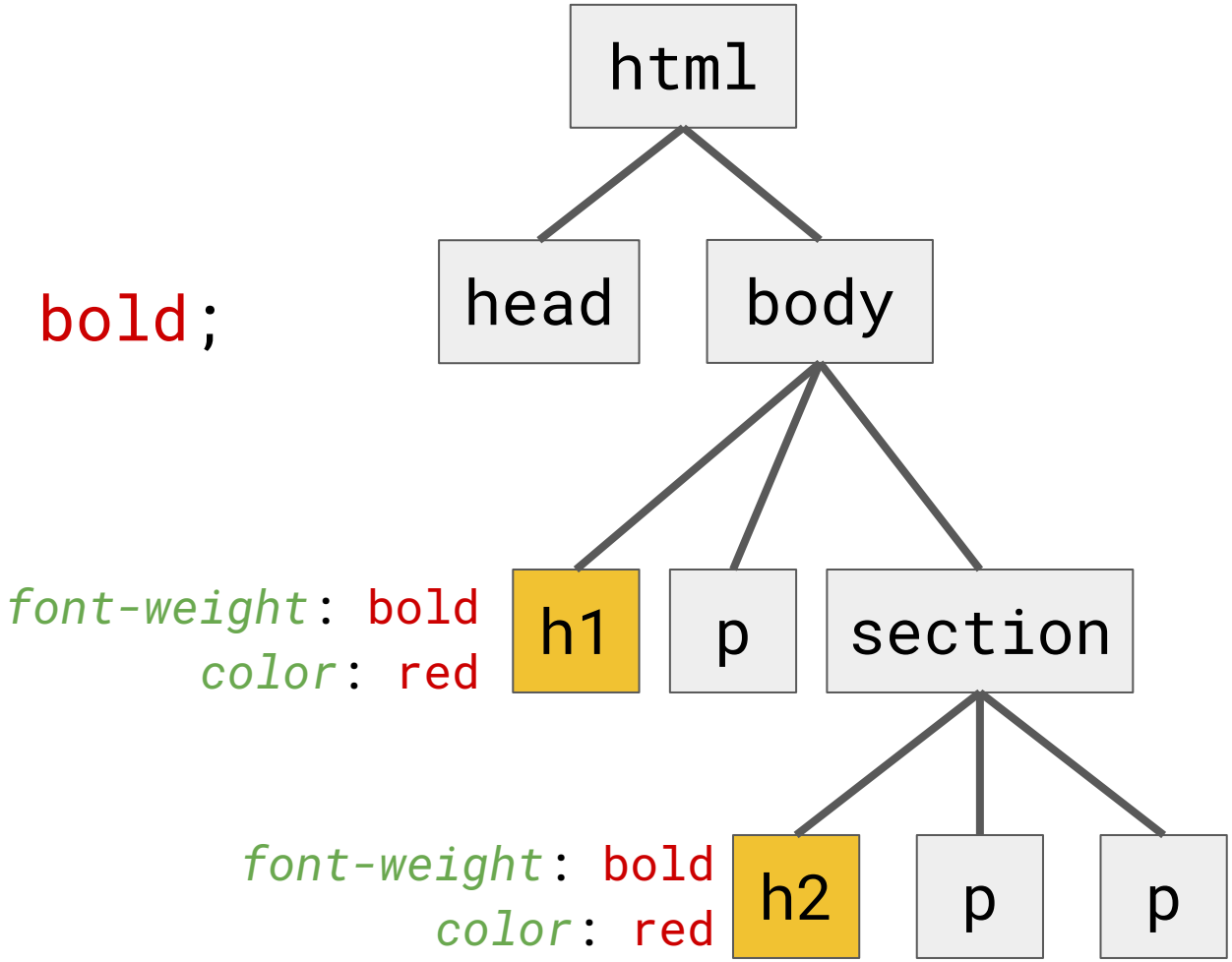
The *tag* selector

```
p {
   color: red;
}
```



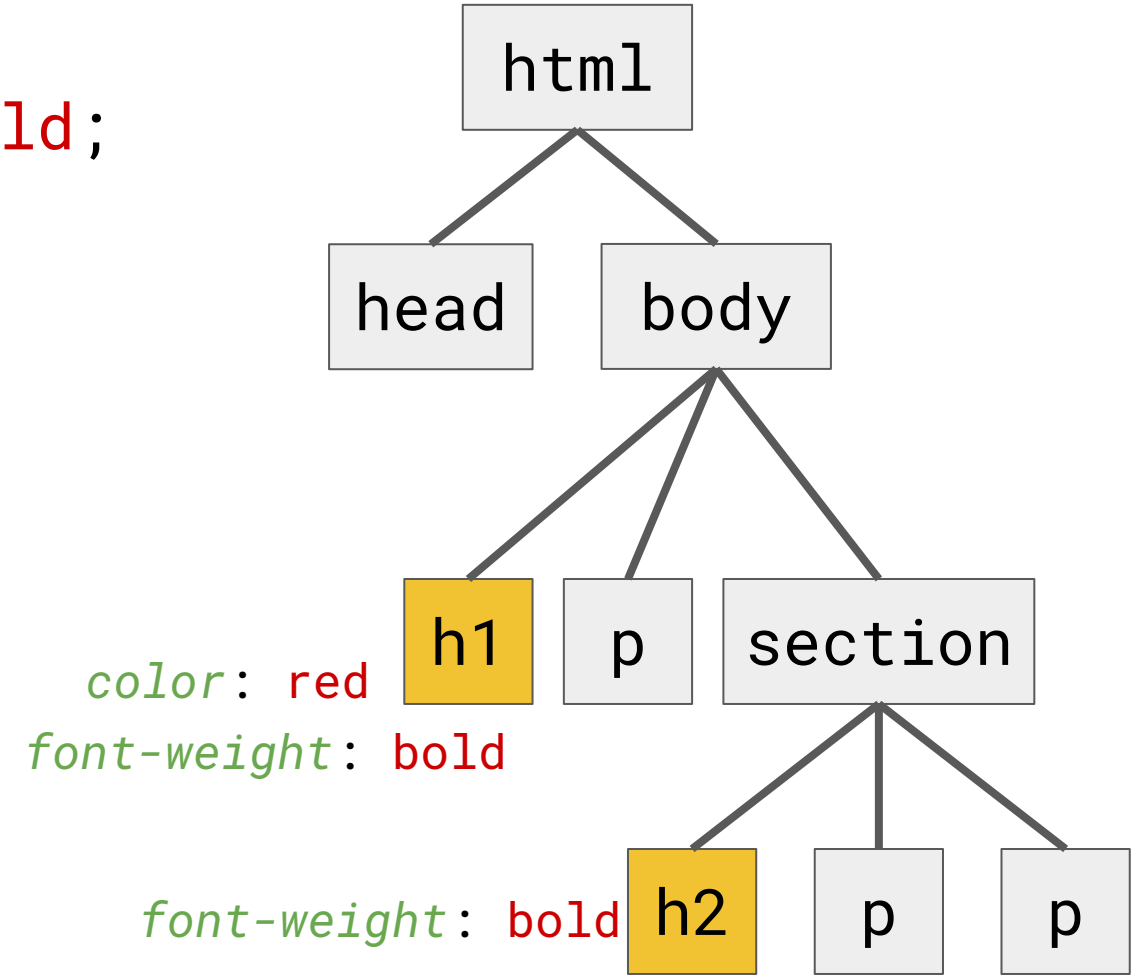
Selector union

```
h1, h2 {
   font-weight: bold;
   color: red;
}
```



Multiple rules

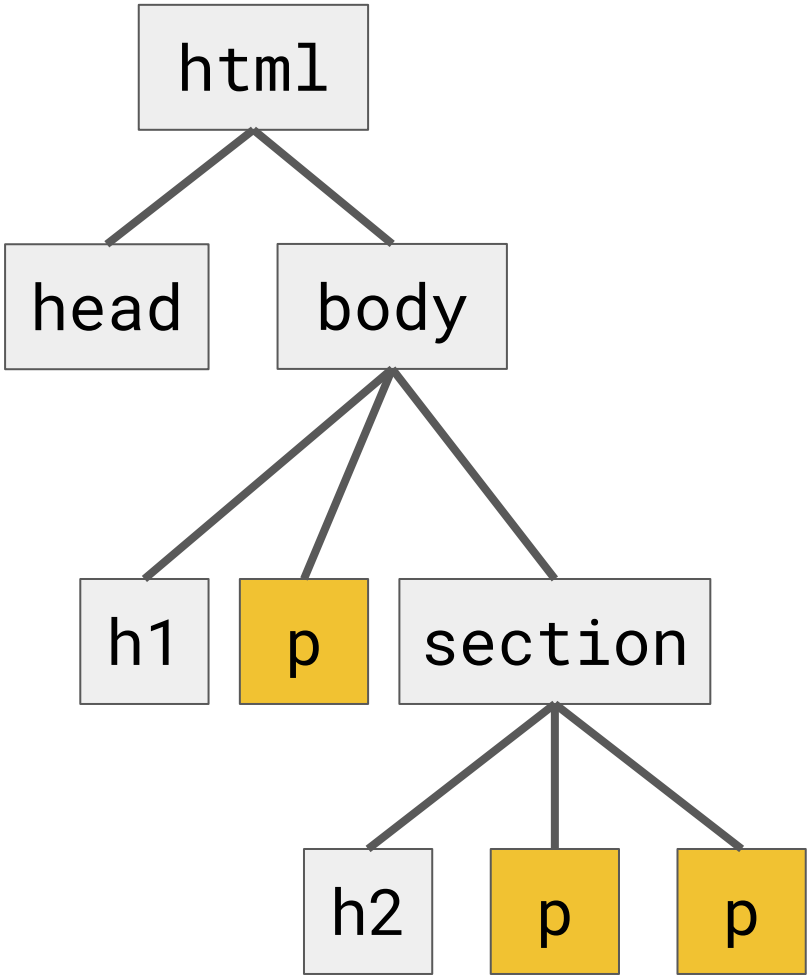
```
h1, h2 {
   font-weight: bold;
}

h1 {
   color: red;

}
```



The parent-child selectors

Selects all paragraphs that are descendants of a body

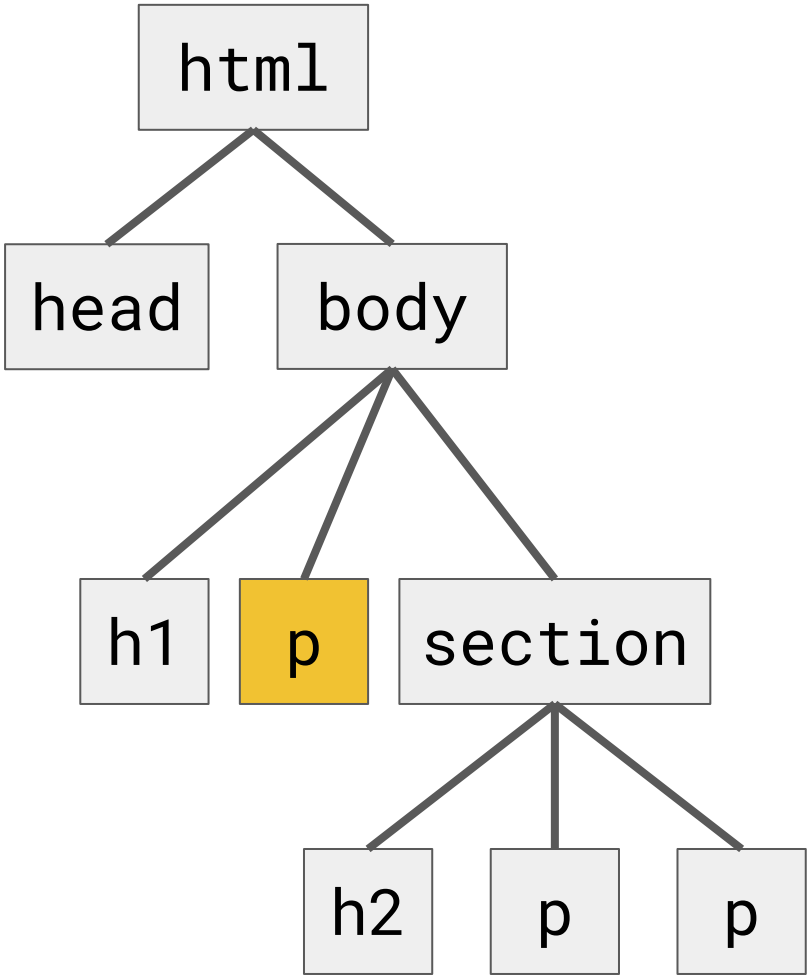
```
body p {
   color: red;
}
```



The parent-child selectors

Selects all paragraphs that are direct children of a body

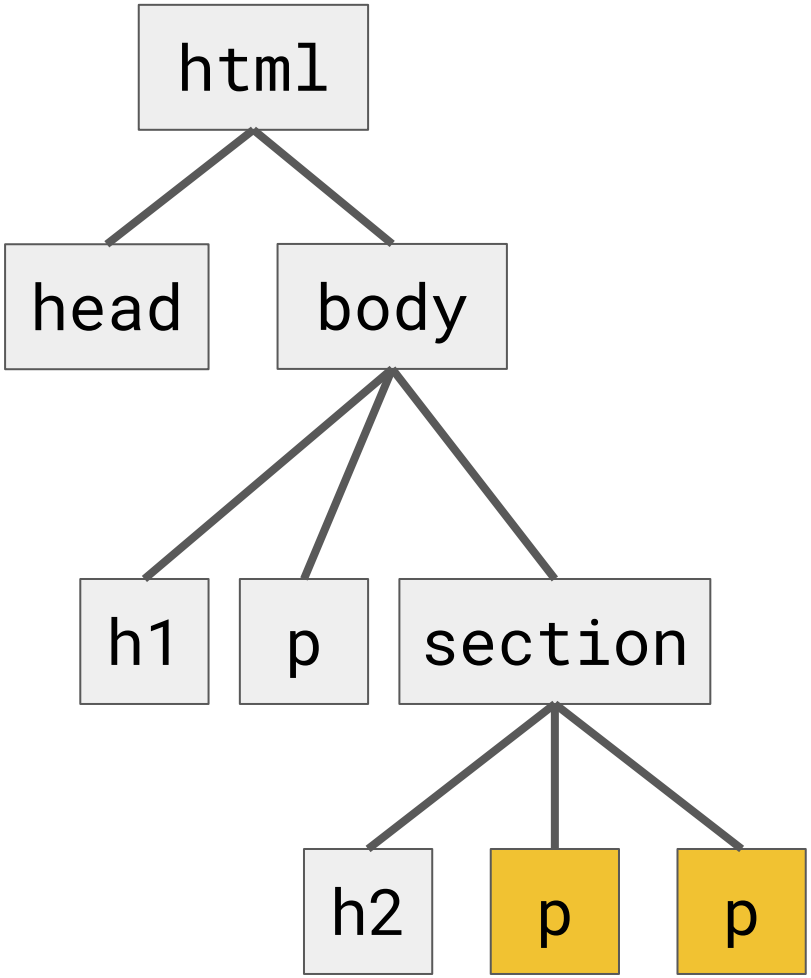
```
body > p {
   color: red;
}
```



The sibling selectors

Selects all paragraphs that are (right) siblings of a h2

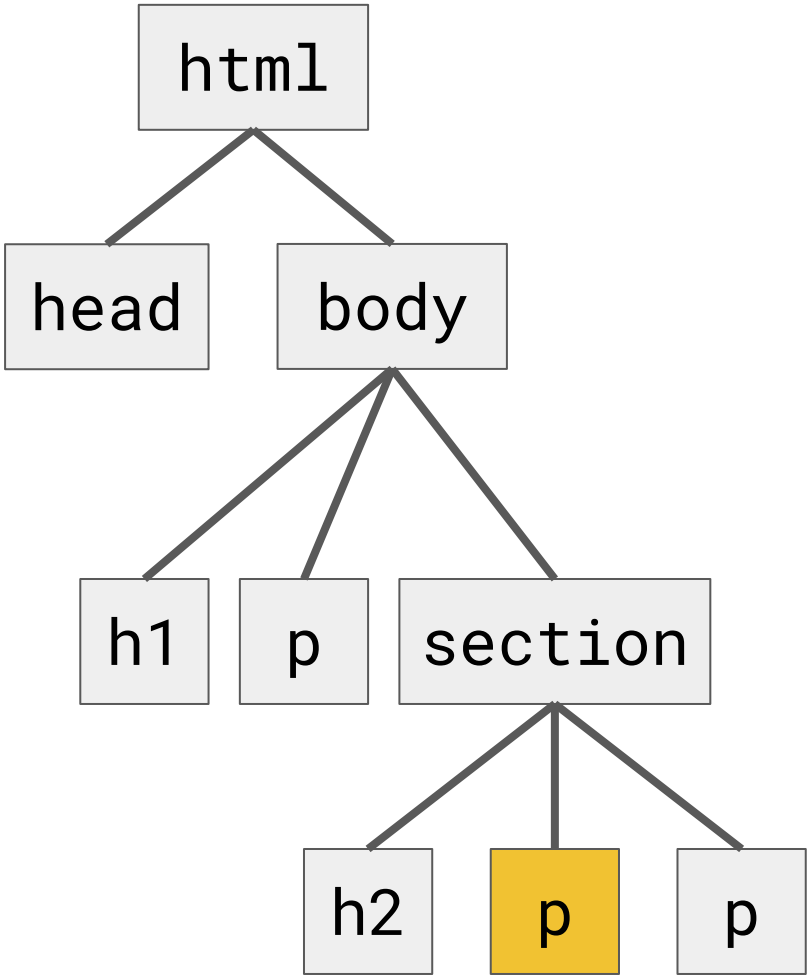
```
h2 ~ p {
   color: red;
}
```



The sibling selectors

Selects all paragraphs that are direct (right) siblings of a h2

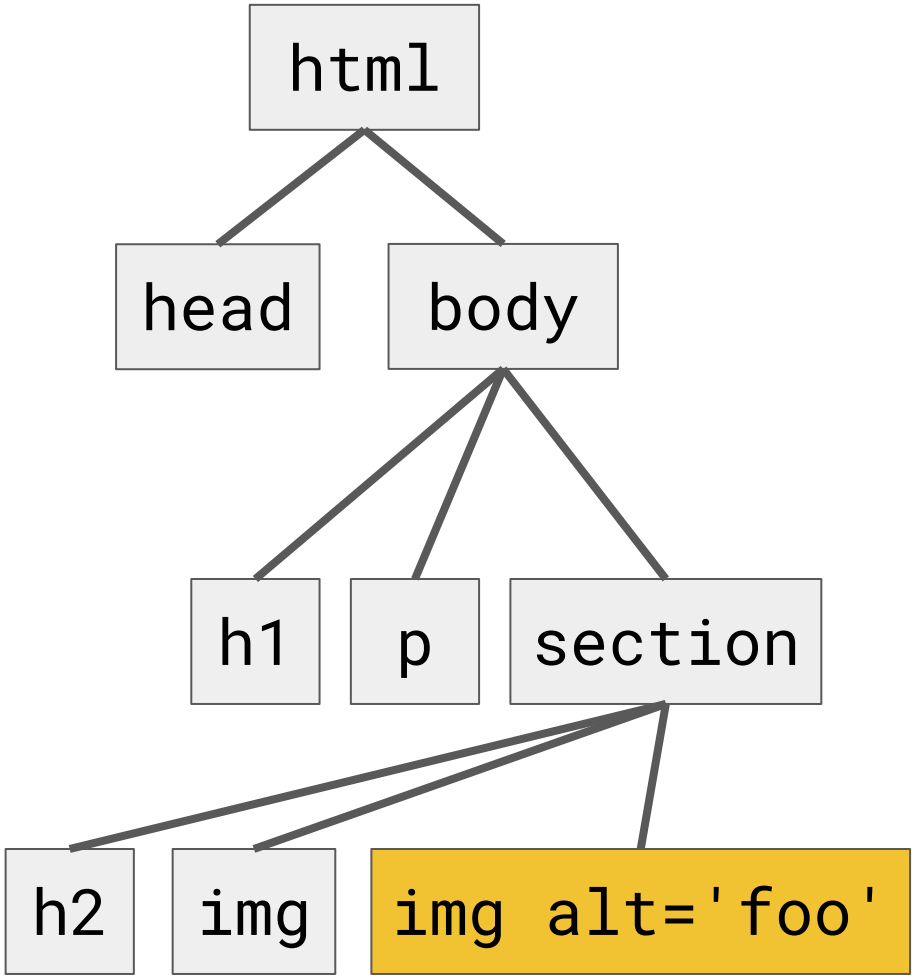
```
h2 + p {
   color: red;
}
```



Attribute-based selection

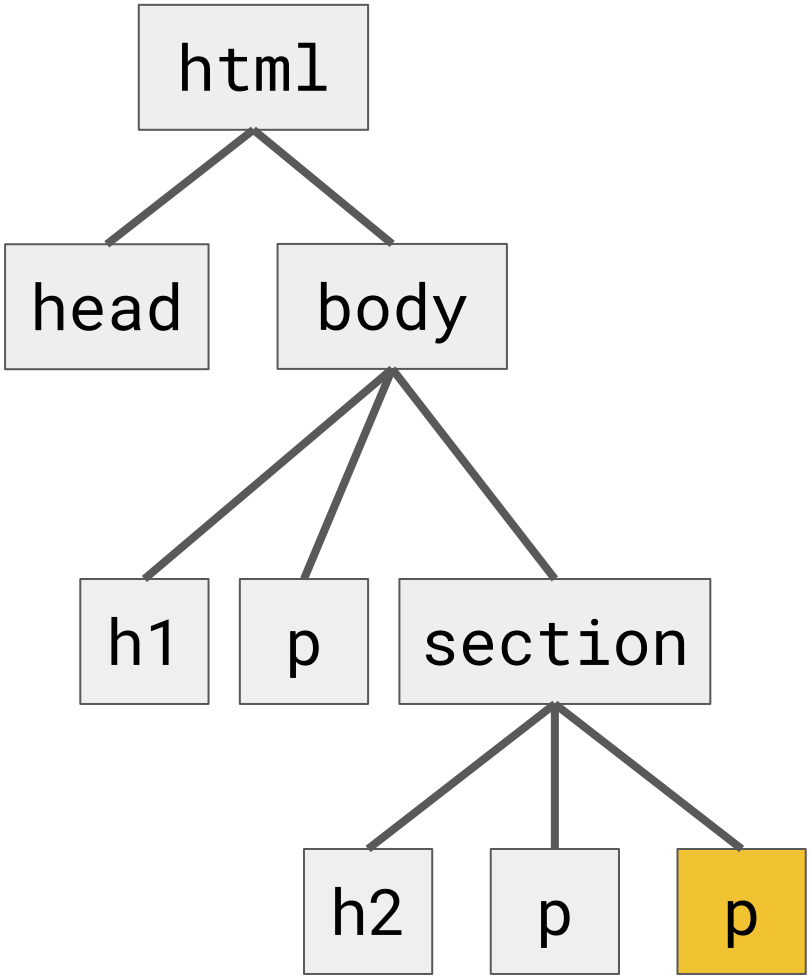
Selects all paragraphs that are direct children of a body

```
img[alt='foo'] {
   color: red;
}
```



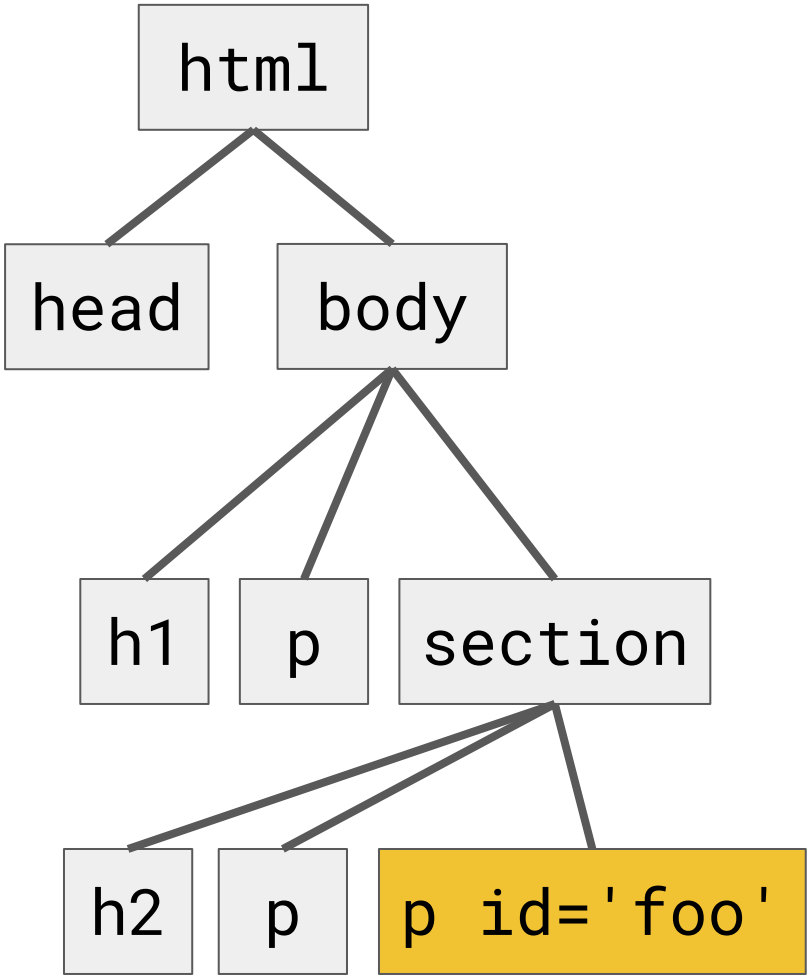
ID-based selection

What if I want just to select this paragraph? It's kind of boring (and dangerous) to write a selector for it



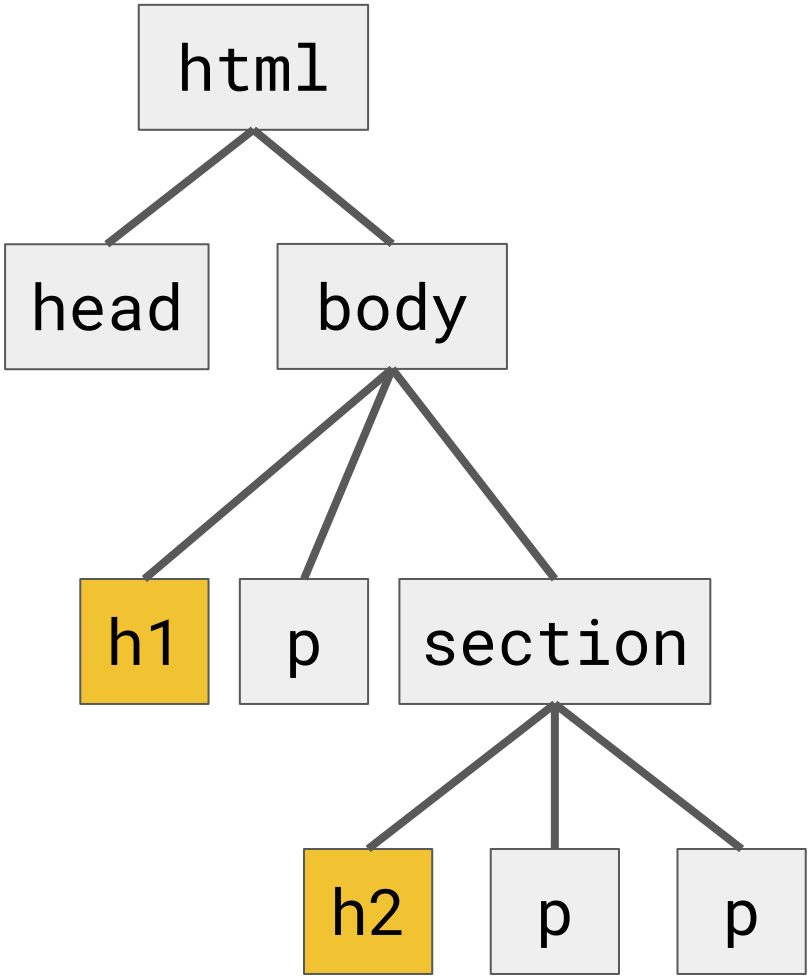
ID based selection

```
#foo {
   color: red;
}
```



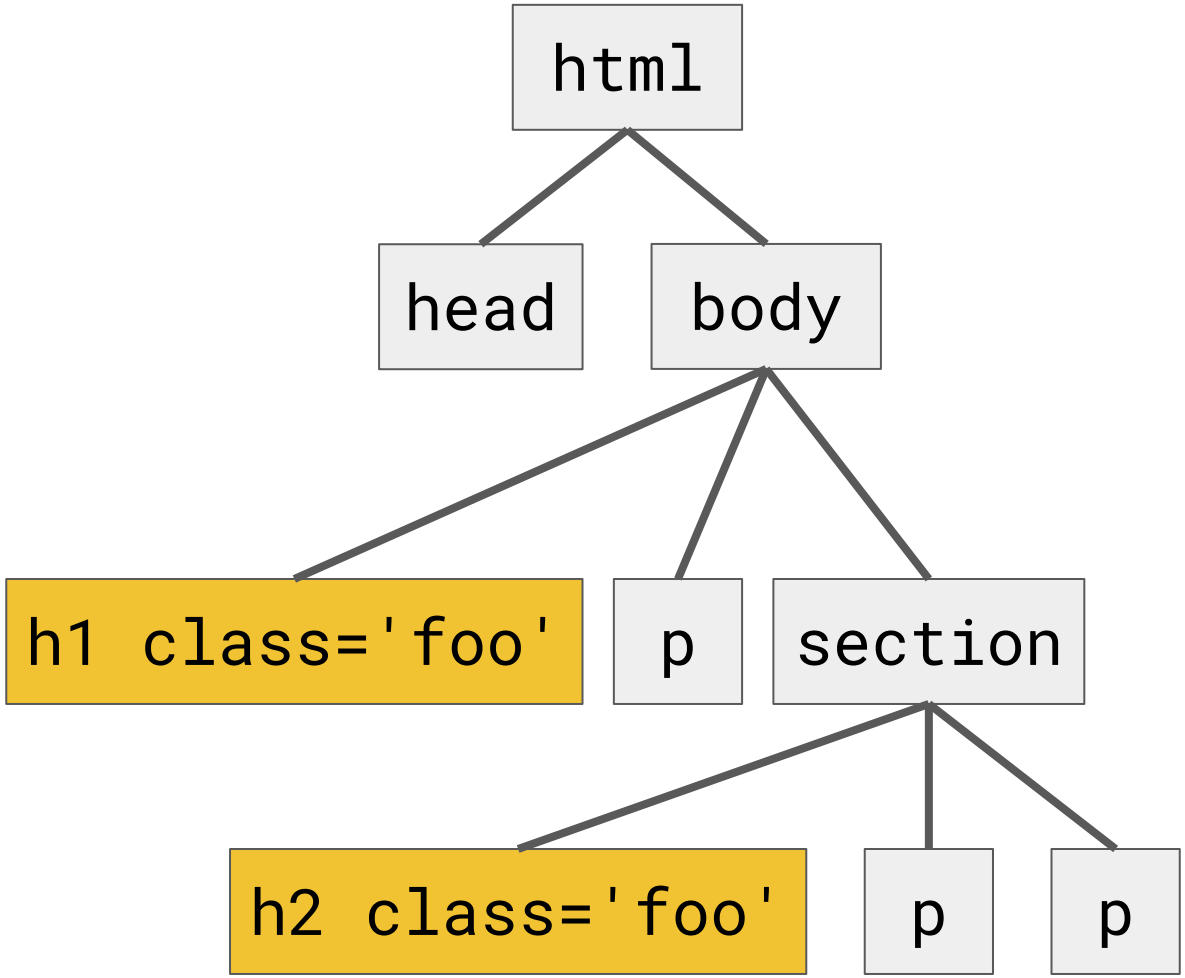
Class-based selection

What if I want just to select these nodes together? OK I can always use selector union, but if the group is large it will quickly become booooring!



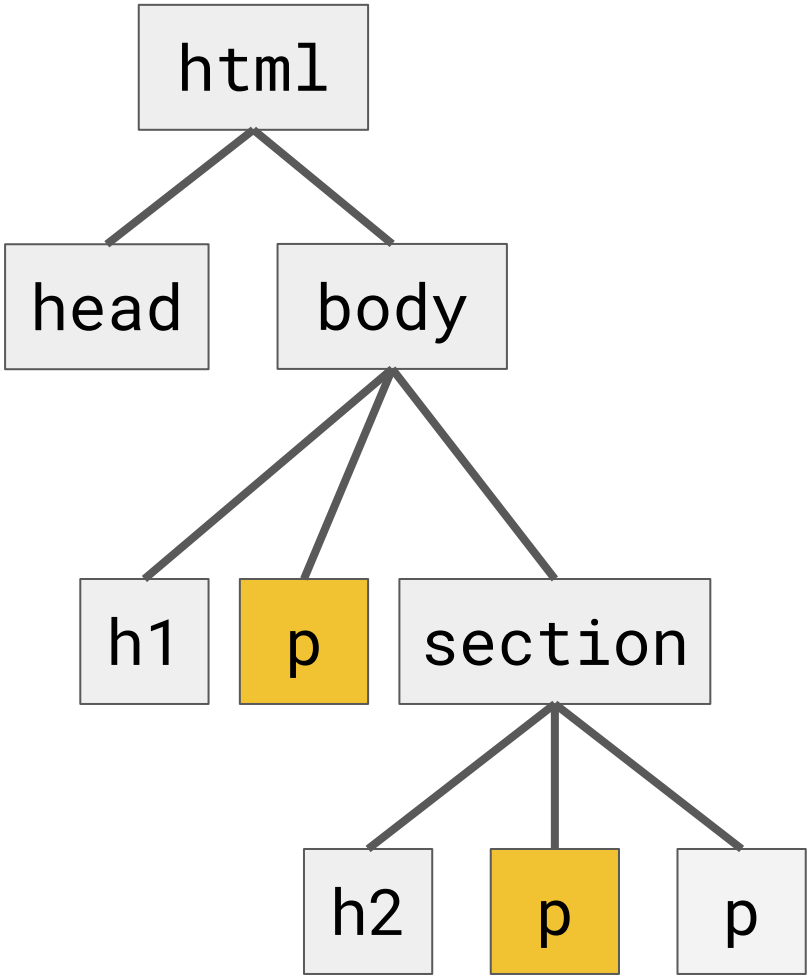
Class-based selection

```
.foo {
  color: red;
}
```

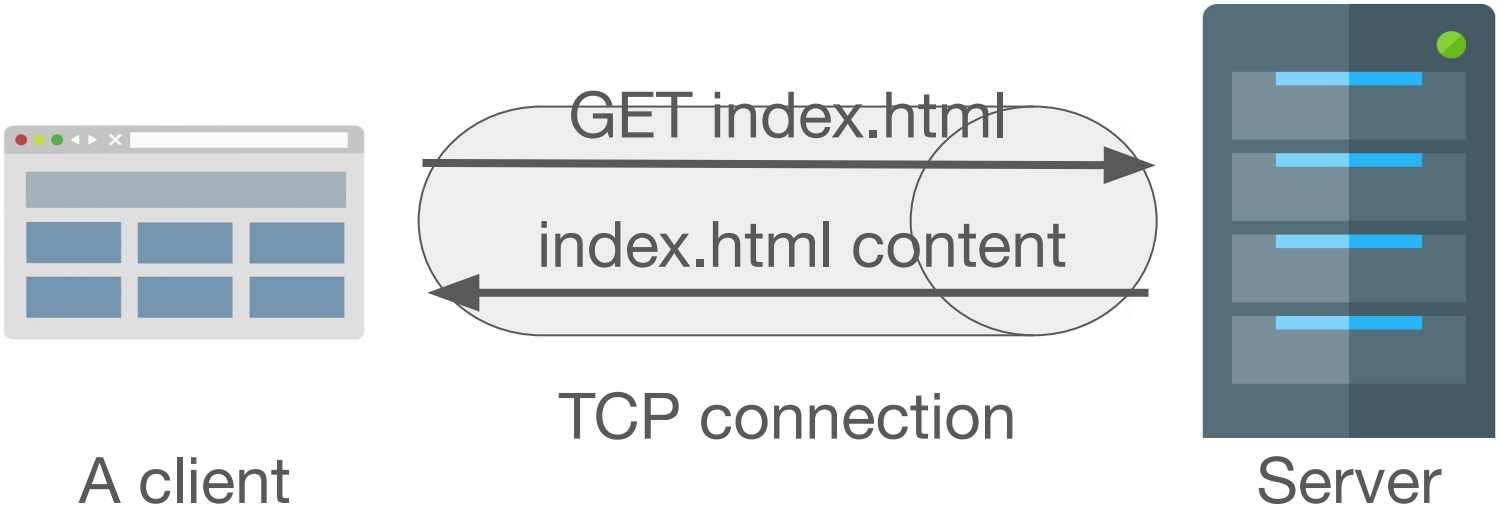


Pseudo class selection

```
p:first-of-type {
   color: red;
}
```



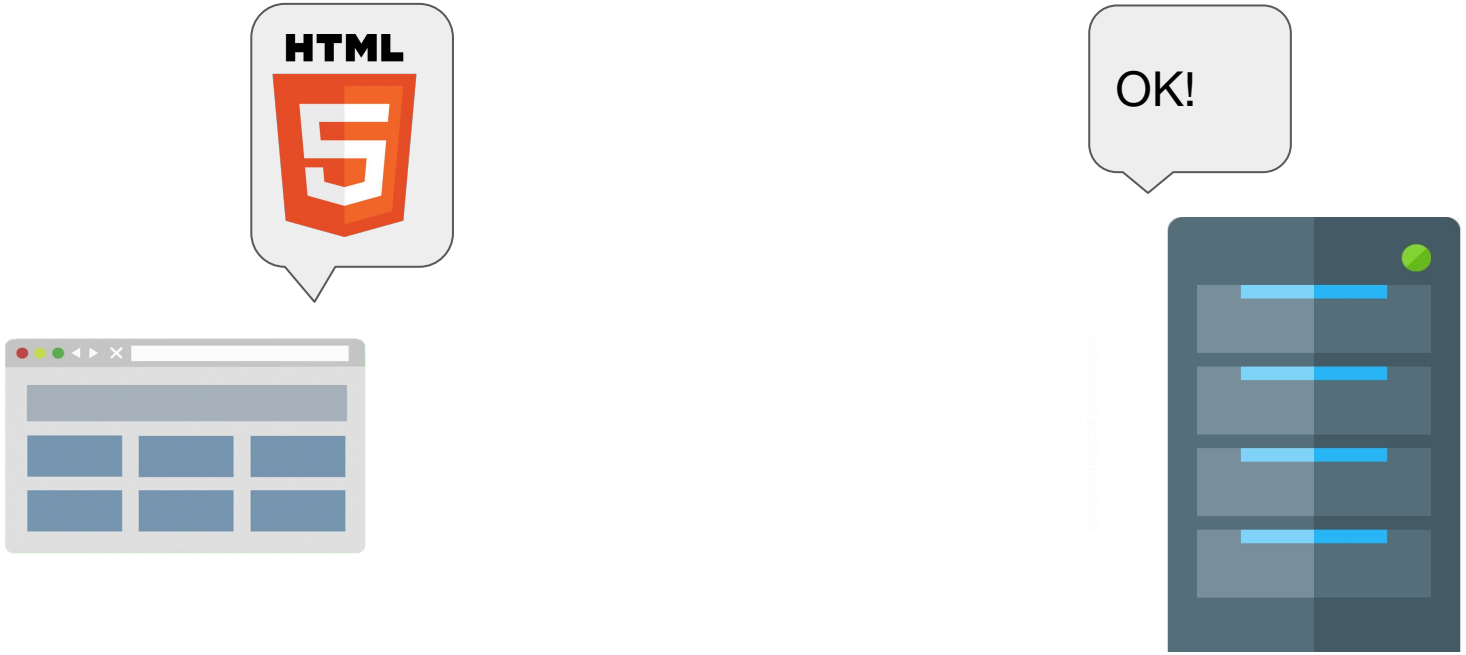
Neat! But how I give CSS to a HTML resource?



But wait! What's going on when there is an image in the page? It's not part of the content!

A more realistic resource exchange

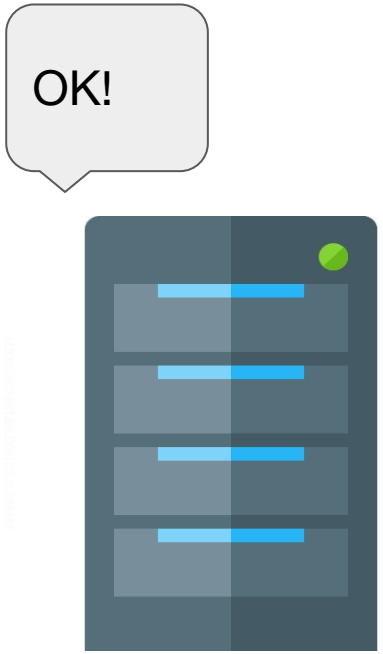
In fact:



When the browser receives a HTML resource, it scans it and asks to the server all embedded resources

Download of embedded resources

In fact:



When the browser receives a HTML resource, it scans it and asks to the server all embedded resources

Back to CSS inclusion, the king’s way

```
<! doctype html>
<html>
    <head>
        <link href='my.css' rel='stylesheet'>
    </head>
    <body>
        <p>Yay</p>
    </body>
</html>
```

Back to CSS inclusion, the quick way

```
<! doctype html>
<html>
    <head>
        <style>h1 { color: red; }</style>
    </head>
    <body>
        <p>Yay</p>
    </body>
</html>
```

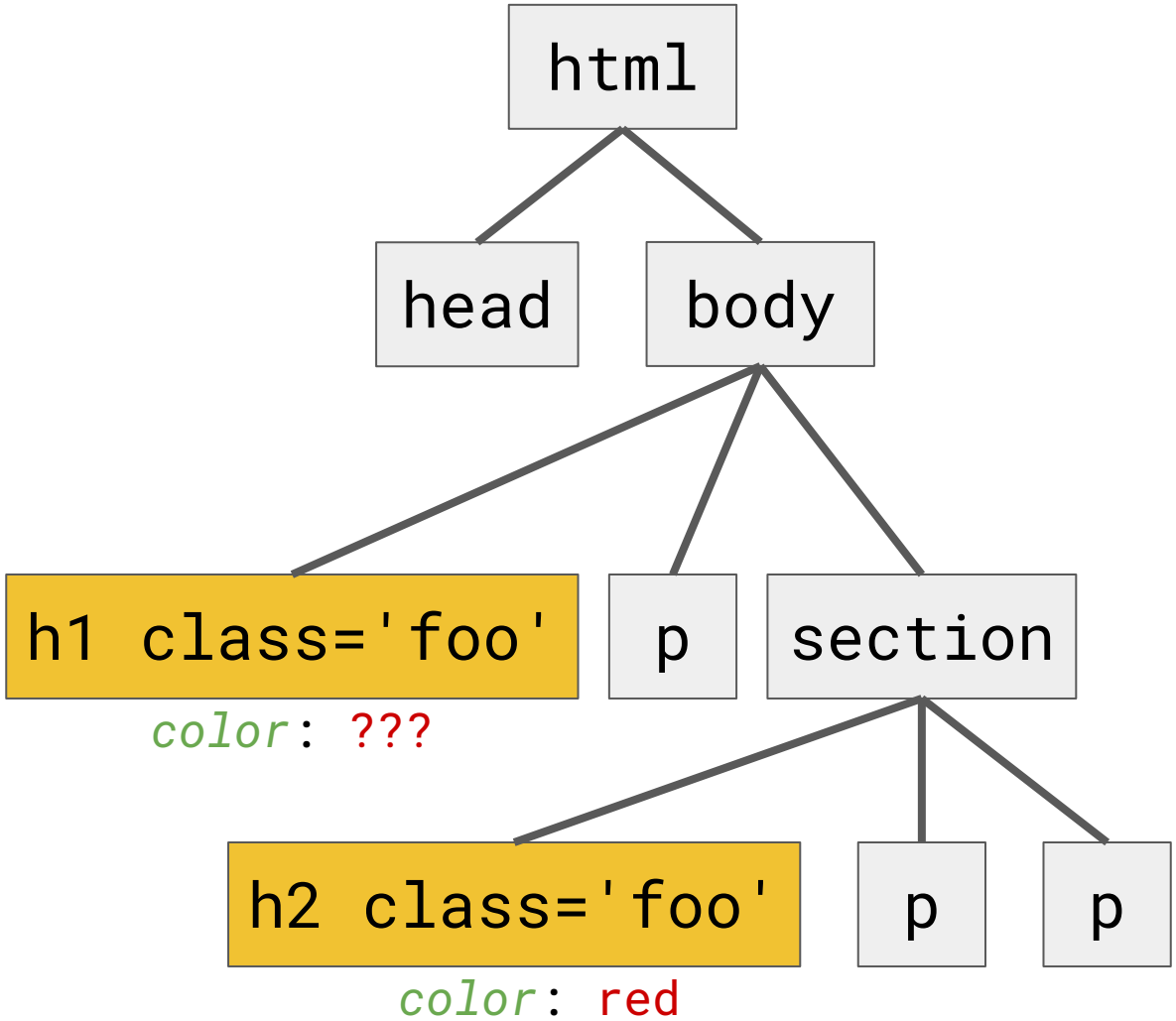
Back to CSS inclusion, the dirty way

```
<! doctype html>
<html>
    <head>
    </head>
    <body>
        <p style='color: red; font-weight:
bold'>Yay</p>
    </body>
</html>
```

Back to CSS rules with a 🤯 example

```
.foo {
   color: red;
}
h1 {
   color: blue;
}
```

Who wins ⚔️ ?



CSS specificity

- Each declaration has a four-dimensional specificity vector coming from its selector
- *First (right) dimension*: number of tags in the selector (i.e. `body > html` has **[0, 0, 0, 2]**)
- *Second dimension*: number of classes or attributes in the selector (i.e. `body + p.foo` has **[0, 0, 1, 2]**)
- *Third dimension*: number of ids in the selector (i.e. `body > #foo` has **[0, 1, 0, 1]**)
- *Fourth dimension*: 1 if the declaration comes from a style attribute (`<a style='color: red;'>` has **[1, 0, 0, 0]**)

CSS specificity comparison

- When two conflicting declarations (i.e. `color: red;` and `color: blue;`) are given via two selectors: fight!
- The corresponding specificity vectors are compared left to right
- As soon as one has a greater value in the i-th dimension, it wins! Example: **[0, 0, 3, 2] > [0, 0, 2, 4]**
- In case of egality, last defined rule wins (yuck)!
- To give priority to a loser declaration, you can use:

```
h1 {
   color: red !important;
}
```

Quick poll

Who wins?

- `body #foo p.bar h1`
- `body #foo #baz`
- `*`

Quick poll

Who wins?

- `body #foo p.bar h1` [0, 1, 1, 3]
- **`body #foo #baz` [0, 2, 0, 1]**
- `*` [0, 0, 0, 0]

I still don’t know are things are displayed!

OK let’s dig into that now. First thing to know is that there are **block** elements and **inline** elements

For instance how do you think the following HTML will be displayed?

```
<h1>Hello World!</h1>
<p>Yay it’s an <em>awesome</em> text
paragraph!</p>
```

Result

Hello World!

Yay it’s an *awesome* text paragraph!

How come the h1 is alone on this line whereas awesome is in the same line as the p’s text?

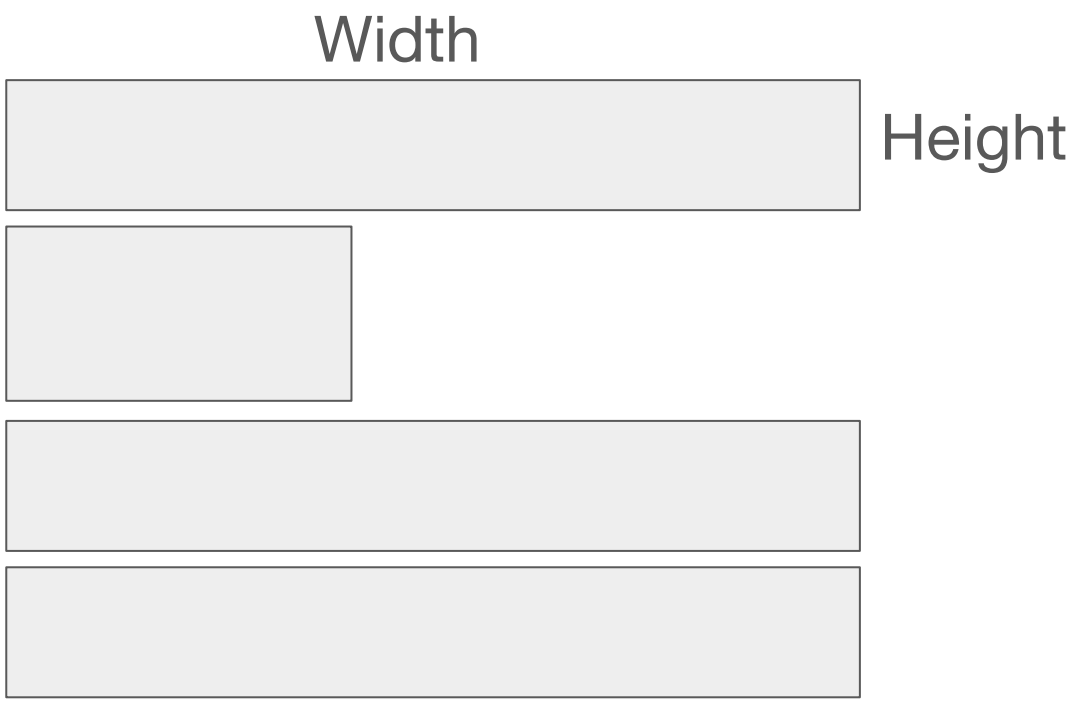
Block and inlines

Because

- **`h1`** and **`p`** are **block** elements (as all sectioning and flow tags are)
- **`em`** is an **inline** element (as all phrasing tags are)

Block elements

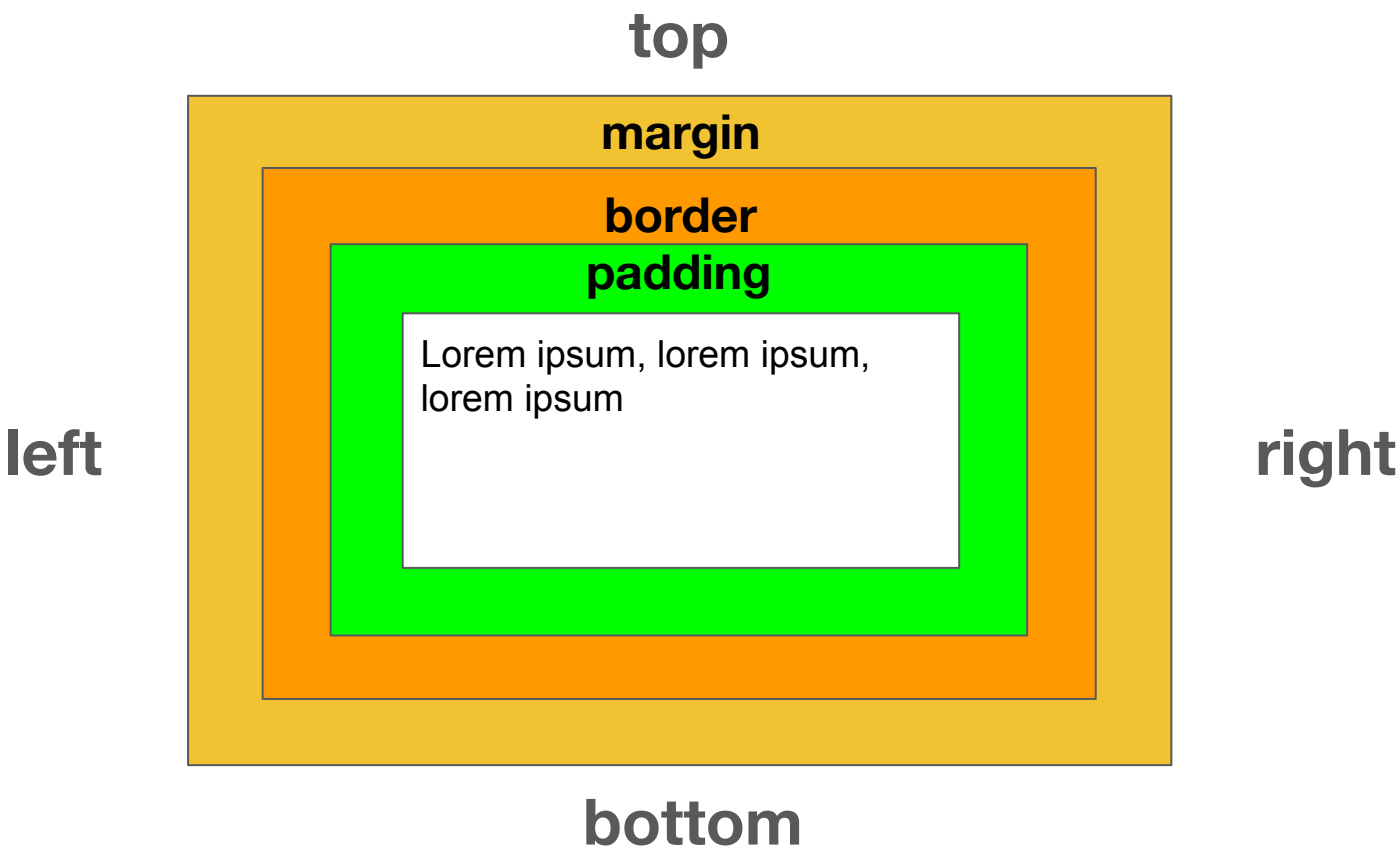
- Flows from top to bottom, alone on their lines
- Can have a width, a height and a custom position
  - `width: 200px; height: 20%;`
- Example



Inline elements

- Flows from left to right, automatically going to a new line
- Have automatic width and height and no custom position
- Cannot have children
- Example:

Tweaking the size of block elements



Inline elements have only left and right margin/padding

Margin, padding and border properties

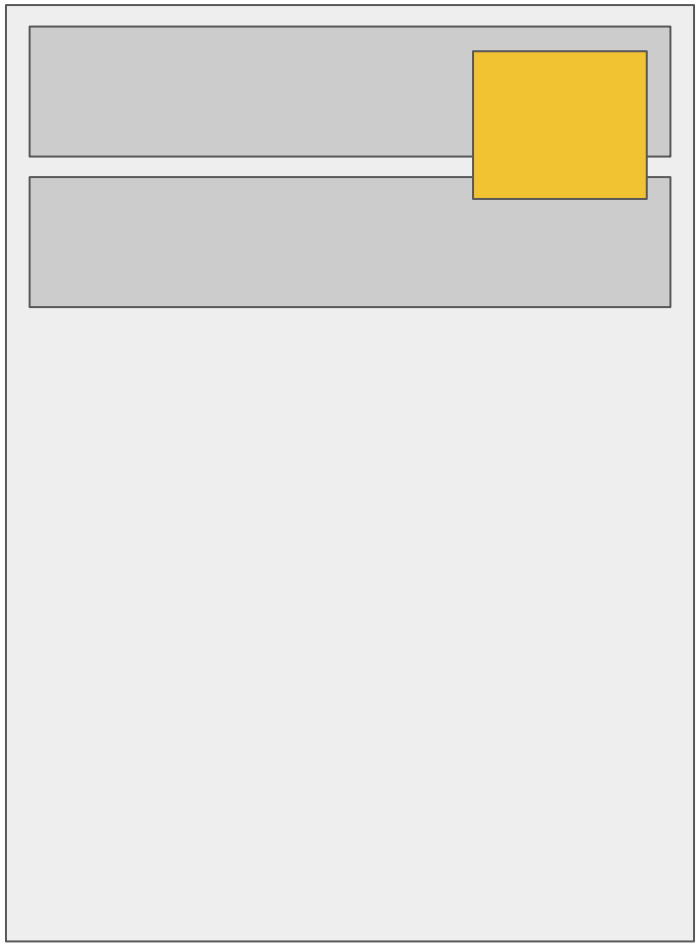
Margin (same for padding):

- `margin: 2px;`
- `margin: 1em;`
- `margin: 50%;`
- `margin: auto;`
- `margin-top: 1px;`
- `margin: 1px 2px;`

Border:

- `border: 1px solid red;`
- `border-top: 1px solid red;`
- `border-width: 3px;`
- `border-style: dotted;`
- `border-color: red;`
- `border-top-width: 3px;`

And what if I want this?

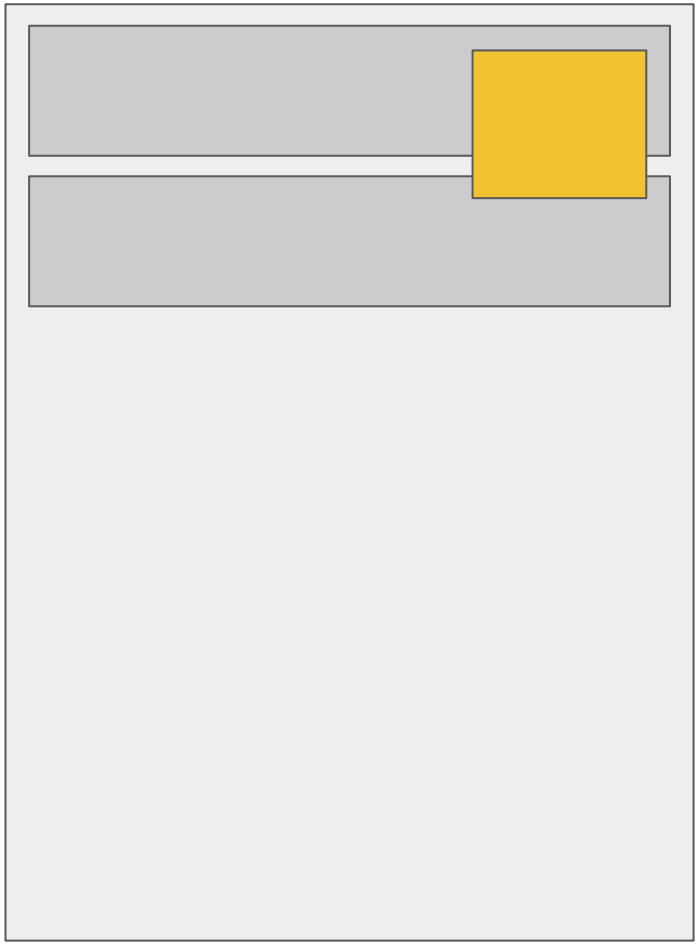


Positioned block

Blocks can have custom positions, not following the classic rules previously presented

- `position: static;` default one (already explained)
- `position: absolute;` these blocks are positioned w.r.t. to the whole page
- `position: fixed;` these blocks are positioned w.r.t. to the browser's window
- `position: relative;` these blocks are positioned w.r.t. to their parent
- `position: sticky;` hard to explain, but fun! Test it

Example of a positioned block



```
#mydiv {
  position: fixed;
  top: 10px;
  right: 10px;
  z-index: 10;
}
```

Multi-column layouts

How the hell do I do this 🤔

Historial solution : inline-block

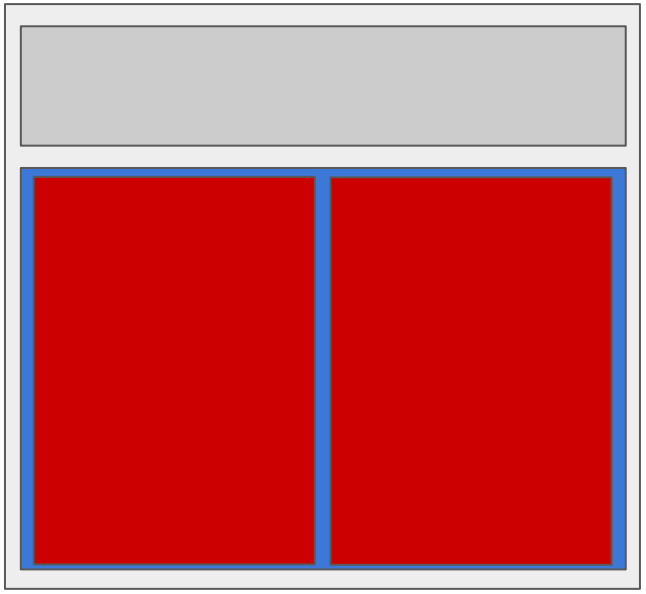
- Elements with `display: inline-box;` can go side by side (as inline ones)
- They can also have a custom size / position
- Best of both worlds

Example

There is a sneaky trick to make this works! will you find it?

```
#left {
  display: inline-block;
  width: 50%;
  margin: 0;
  padding: 0;
  background-color: red;
}

#right {
  display: inline-block;
  width: 50%;
  margin: 0;
  padding: 0;
  background-color: blue;
}
```

Multi-column layouts in the new age: flexbox



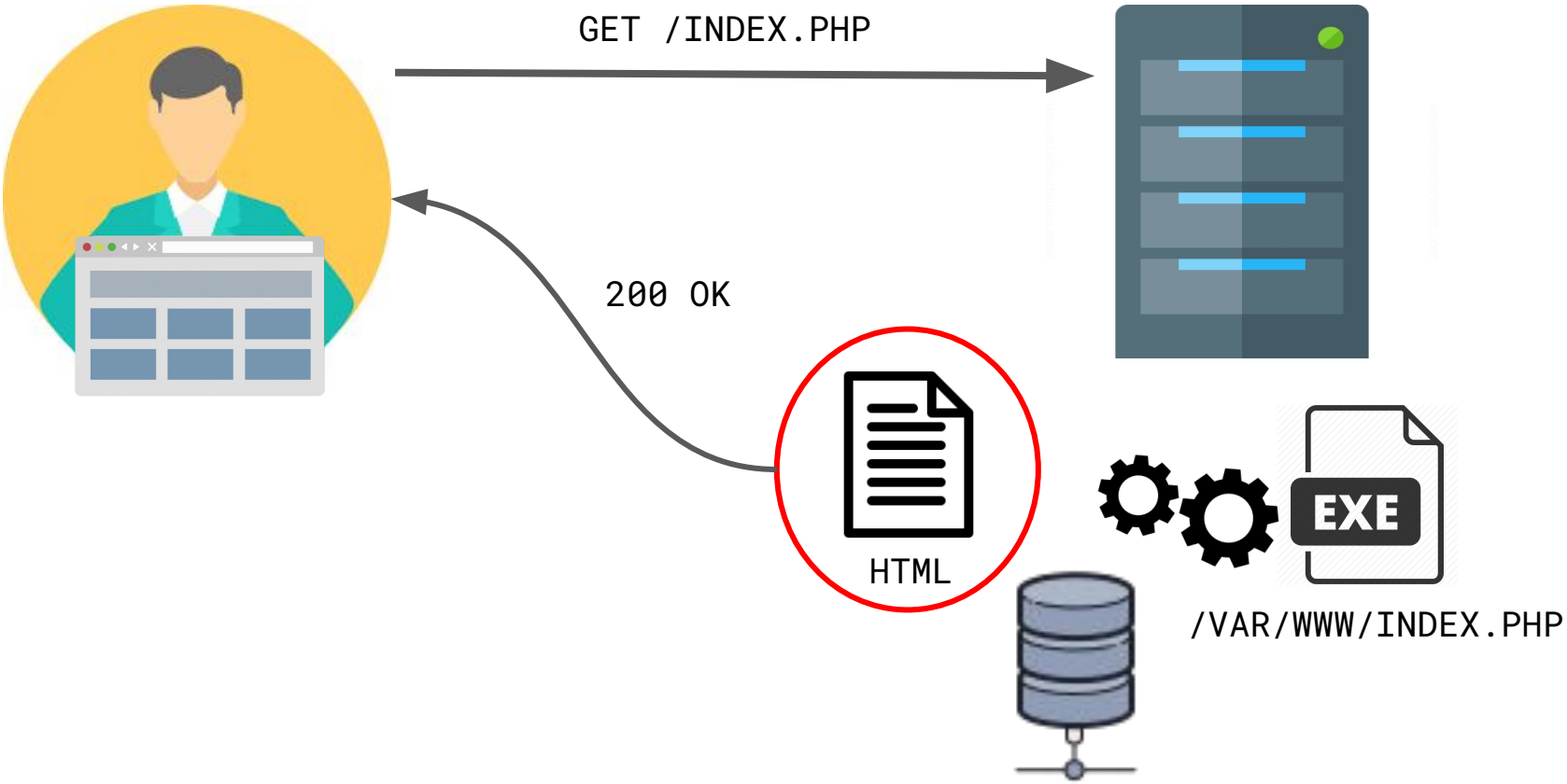
```
#container {
 display: flex;
 background-color: blue;
}

.column {
 flex: 50%;
 background-color: red;
}
```

Go make your blog a beauty

- Use the CSS we learned to improve the design of the blog developed previously
- Try to change fonts, colors
- Try to use a columned layout
- Try to put a title bar
- **Validate constantly your CSS**
- **Use the browser inspector to debug it**

PHP

Server-dynamic web applications



Why do we need server-dynamic web applications?

Imagine a web application of type **search engine**

- You won’t be able to a-priori build web resources that give answers to every possible search (infinite space)
- But given a set of keywords, you can search for matches on-the-fly and generate a resource returned to the client
- Of course the content to be searched cannot fit in RAM it needs to be stored (persisted) somewhere
- Such content is usually stored in a database

What more with server-dynamic web applications?

Imagine the Facebook web application

- First time you hit the URL bar and enters Facebook's URL
- You end up on the login form and enter your credentials
- This gives you back your wall
- If you hit refresh your browser will issue the **very same request**
- But now you directly get your wall, **the server remembers you!**
- HTTP cannot remember your, but **what about the scripts running on the server?**

Classical concerns for server-dynamic applications

- You need to know how to **produce HTML** resources for your clients
- You need to know how to **pass parameters** to your scripts
- You need to know how to **simulate a persistent connexion** with your clients
- You need to know how to **integrate with a persistent storage system**

Drawbacks of server-dynamic web applications

- You need to know **one more language** than can be executed by web servers
  - In this course, we use PHP
- You will be subject to **many more security threats** due to the server-side code execution
- Your servers will experience a drastic **increase of CPU charge** and **memory consumption**
- You can no longer test your application without a **running server**

PHP

- Interpreted language specially targeting web applications
  - no compilation
  - executed line by line
  - slow 🐌
- Dynamic typing
- Garbage collected
- Many fancy built-in types (dynamic and associative arrays)
- Mature language successfully used in popular web applications (Facebook, Wordpress, Drupal)

How do I put a PHP script in my application?

In a nutshell, it's very simple just drop a **script.php** file on your server and access it through it's regular URL via your browser (http://mymachine.org/script.php)

You often use ssh or webdav to access your server's filesystem

The web server recognizes that it's PHP (using the extension) and executes it automatically

The *hello world* script

```
<?php
  echo "<!doctype html><html><head>";
  echo "<meta charset=\"utf-8\">";
  echo "</head><body><p>";
  echo "Hello World!";
  echo "</p></body></html>";
?>
```

You've discovered how to generate HTML for your clients!

It’s ugly 🤮🤮🤮

The fact that we have to produce HTML to the client makes program **hard to read**

Indeed, your HTML is now contained in PHP strings given as arguments for `echo`

Can we do a little better?

Hell yes!

The *hello world* revisited

```
<?php $msg = "Hello World!"; ?>
<!doctype html>
<html>
  <head><meta charset="utf-8"></head>
  <body><p>
  <?php echo "  " . $msg; . "\n" ?>
  </p></body>
</html>
```

Lines outside PHP zones are replaced by calls to echo

PHP script quirks

```
<!doctype html>
<html>
  <head><meta charset="utf-8"></head>
  <body><p>
  <?php if ($display == true) { ?>
    <em>Hello!</em>
  <?php } ?>
  </p></body>
</html>
```

That works because *Hello!* is in fact replaced by a echo

PHP collections

```
$strings = array("a", "b", "c");
$strings = ["a", "b", "c"];
array_push($strings, "d");

foreach ($string as $val) {
  echo $val;
}
```

PHP collections are associative arrays

```
$infos = array("age" => "12", "id" => "Joe");
$infos = ["age" => "12", "id" => "Joe"];

foreach ($infos as $key => $value) {
  echo $key;
  echo $value;
}
```

PHP functions

```
function add($a, $b) {
  return $a + $b;
}

echo add(2,5);
```

PHP includes

```
// execute instructions of other_script.php
include "other_script.php";
// fails if other_script.php doesn't exist
require "other_script.php";
// do nothing if not the first include of the
script
require_once "other_script.php";
```

You can also use includes to reuse HTML fragments that are duplicated across resources!

Back to our search engine use case

- Remember you have to supply the searched text to the search script
- The searched text must go from the client to the server
- It needs to go through the network and thus must be part of a HTTP request
- How does it work?

In case of a GET request

- Remember it's the main request type of browsers
- You do not have many options
  - You cannot send content to the server
  - The headers are set by the browser so you cannot use them
  - Only one thing remain: **the URL**
- `http://mysite.com/script.php?p1=v1&p2=v2`
- parameters are passed via the query part of the URL
- You have to put links such as `<a href="script.php?p1=v1&p2=v2">Link</a>` in your page to pass parameters

Obtaining GET URL parameters in PHP

```
// parameters are automatically injected in a
global associative array
$_GET["p1"] // v1
$_GET["p2"] // v2

// You can use foreach to iterate on params
foreach ($_GET as $param => $value) {}

// You can check if a param has been supplied
if (isset($_GET["p1"])) {}
```

Parameters in GET URL : *summary*

- Easy to do for the developer
- Hard to do for users
- You **can** bookmark a page that has parameters
- Very well suited when you have **integer** parameters
- **Cannot accommodate large** parameter values
  - Imagine you want to give a long text as a parameter
  - The URL quickly becomes huge
  - Imagine now you want to send an image!
  - You finally receive a `414 Request-URI Too Long`

Parameters in POST requests

- POST requests seems a much more attractive fit to pass parameters as they can send content to the server
- However browsers do not issue post requests by default
- How do I force a browser to issue a POST request (with my parameters in the content)?
- **Solution: use a form**

Forms

```
<form action="script.php" method="post">
  <input name="p1" type="text">
  <input name="p2" type="text">
  <input type="submit">
</form>
```

Will display **two text input boxes and a *submit* button** in your page. When clicked all **data will be pass to `script.php`** inside the content of a **POST request** (basically the content is exactly the same as query string of the URL but inside the content of the request)

The POST request example

```
POST script.php HTTP/1.1
Host: mydomain.com

p1=v1&p2=v2
```

Obtaining POST parameters in PHP

```
// parameters are automatically injected in a
global associative array
$_POST["p1"] // v1
$_POST["p2"] // v2

// You can use foreach to iterate on params
foreach ($_POST as $param => $value) {}

// You can check if a param has been supplied
if (isset($_POST["p1"])) {}
```

Parameters in POST request : *summary*

- Hard (boring at least) to do for developers
- Easy to manipulate for users
- You **cannot** bookmark a page that has parameters (damn form resend!)
- **Can accommodate large** parameter values
  - Long texts
  - Even whole binary files

PHP parameters oddity

Imagine this form

```
<form action="script.php?id=12"
method="post">
  <input name="age" type="text">
  <input type="submit">
</form>
```

When submitted, `script.php` receives params both in the URL and in the request content

```
$_GET["id"] // 12
$_POST["age"] // value entered in the form
```

Back to the Facebook use case

- Remember your scripts have to memorize if they have already seen a given client
- How to do that?

By passing a parameter!

- Imagine the server code issue an id the first time it encounters a new client and somehow manage to send it to him
- The server also keep a dedicated memory zone associated to each id
- In order to be recognized the client then pass it's id as a parameter to the server in all subsequent requests
- The server can therefore access the client dedicated memory zone
- Basically, it's what is called a ***session***

How a server is going to pass a param to clients?

- Remember HTTP requests are issued by the client, and HTTP response are made by the server
- The parameter **must be part of the response**
- Which part of the response can I use?

The easy *cookie* way

- Remember HTTP requests can have headers
- Two headers manage cookies, a key-value list located in your browser and associated with a domain
- The server send `SetCookie: PHPSESSID=12345;` in the response headers
- The client's browser stores `PHPSESSID=12345` and automatically adds a `Cookie: PHPSESSID=12345;` header in all subsequent requests to the same domain
- **Problem solved!**

The messy *URL-rewriting* way

- Imagine that your client **has turned off cookies**
- We have no other choice than passing the session id **in the response content**
- But we have to find a way that force the client to **send it back!**
- Solution : **URL rewriting**
- Just before generating the HTML resource PHP engine looks for relative links inside the resource and **append a `PHPSESSID=id` parameter to each link's URL**
- Therefore, whichever link the client will click on, the **session id will be sent back to the server**

Passing the session id : *summary*

- Cookies are more reliable : less work if doing also HTTP requests via JavaScript
- Cookie are more secured : any person can see the session id on the client screen and hijack the session
- BTW how do you think that a session ids should be computed
  - A sequence of natural numbers (1, 2, …, n) ?
  - Something else?

Using a session in PHP

```
session_start(); // before any call to echo:
create a session id and send it to the client

// after calling session_start you can access
a $_SESSION associative array initially empty

$_SESSION["logged"] = true;

// the content of this array will be
conserved across requests. You can imagine
that you have one array per client
```

Destroying a session

```
session_start(); // even if you want to
destroy a session, you have to start it first

session_destroy(); // at this point the
session array for the client has been dropped
```

Database

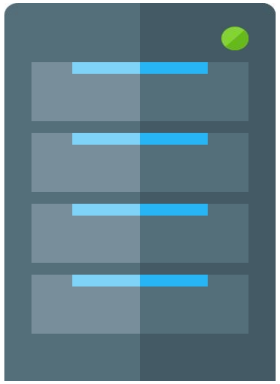
Data is important

- The data is one of the main important aspects of web applications
- Therefore it cannot just be recorded on a plain file carelessly
  - Concurrent modifications
  - A search would require an entire traversal of the file

Database systems

Hey! I am a client and I want to manage data

Yo! I am the database server and I have your data!



Client

(Database) Server

Database systems

- Have a similar architecture to web applications
  - Client want data, server has data
- The clients and the server communicate through a TCP connection
  - Except this time the language is not HTTP but a language to manipulate data
  - And the connexion stays open until the client is finished

ACID guarantees

- **ATOMIC** either a set of modifications (transaction) is applied (commit) or nothing is applied (rollback)
- **CONSISTENT** a database only go from a sound state to another sound state
- **INDEPENDENT** two transactions do not influence each others or the database run them sequentially
- **DURABLE** once applied, a transaction result have a permanent effect on the data

Relational databases in a nutshell

Sample database

Persons

ID	NAME	AGE	COUNTRY
...	...	...	...
...	...	...	...

Cars

ID	CAR	SPEED	PRICE
...	...	...	...
...	...	...	...

...

...	...	...
...	...	...
...	...	...

Relational databases

- Are just a bunch of named excel spreadsheets
- Columns have a **name** and a **type**
  - **INT** a fixed-size integer
  - **VARCHAR** a fixed size text
  - **DATE** a date
  - **TEXT** a variable size text
  - **BLOB** a variable size binary sequence
- Each line represent **an entity**
- Usually relational databases are manipulated using the standard **SQL** language

Storing authors and categories of the blog

Authors

ID	PSEUDO	MAIL
1	Joe	joe@mail.com
2	Bob	bob@mail.com

Categories

ID	TITLE
1	Sport
2	Personal

SQL code

```sql
CREATE TABLE `authors` (
 `id` int(11) NOT NULL,
 `pseudo` varchar(200) NOT NULL,
 `mail` varchar(200) NOT NULL
)        ENGINE=InnoDB        DEFAULT        CHARSET=utf8mb4;

ALTER TABLE `authors`
 ADD PRIMARY KEY (`id`),
 ADD UNIQUE KEY `pseudo` (`pseudo`),
 ADD UNIQUE KEY `mail` (`mail`),
 MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

BTW what is the ID column?

- You often have to discriminate a particular line in a table (example one line in the authors table is ONE author)
- You therefore have to rely on an unique value in a cell of the line : **PRIMARY KEY**
- Sometimes a particular cell describing the data can do the trick : for the author table the mail
- But remind yourself that then you'll have to use this value as a key everywhere you need this kind of data
- In web applications, we prefer passing integer parameters, therefore entities have dedicated ID columns (plus users might change mail addresses)

Storing the posts

Posts

ID	TITLE	DATE	AUTHOR	CONTENT
1	France wins!	12/01/1998	Joe	...
2	France loses!	12/01/2016	Bob	...

Problem with authors

- You can put there a pseudo that do not correspond to one of the author table
- Even if you put a correct pseudo, imagine that Joe want to change pseudo, you have to change it everywhere in the post table
- This lead to **CONSISTENCY** problems
- Solution **USING A FOREIGN KEY**
- The underlying assumption is that you have a **published by** relation between **authors** and **post** entities (one author can publish multiple articles, a given article is only published by one author)

Storing the posts revisited

Posts

ID	TITLE	DATE	ID_AUTHOR	CONTENT
1	France wins!	12/01/1998	1	...
2	France loses!	12/01/2016	2	...

The database ensures that values in ID_AUTHOR are legal!

SQL code

```sql
CREATE TABLE `posts` (
 `id` int(11) NOT NULL,
 `titre` text NOT NULL,
 `date` date NOT NULL,
 `id_auteur` int(11) NOT NULL,
 `contenu` text NOT NULL
)        ENGINE=InnoDB        DEFAULT        CHARSET=utf8mb4;

ALTER TABLE `posts`
 ADD  CONSTRAINT  `posts_ibfk_1`  FOREIGN  KEY  (`id_auteur`)
REFERENCES `auteurs` (`id`);
```

Note on the design choice

- Authors can publish an unbounded number of articles
- Therefore we cannot put this information on the author table (it requires an unbounded number of columns)
- Articles are published by only one author
- Therefore we can place this information on the article table using a single column
- **How to deal with posts being part of several categories while categories can contains several posts?**

The N*M problem

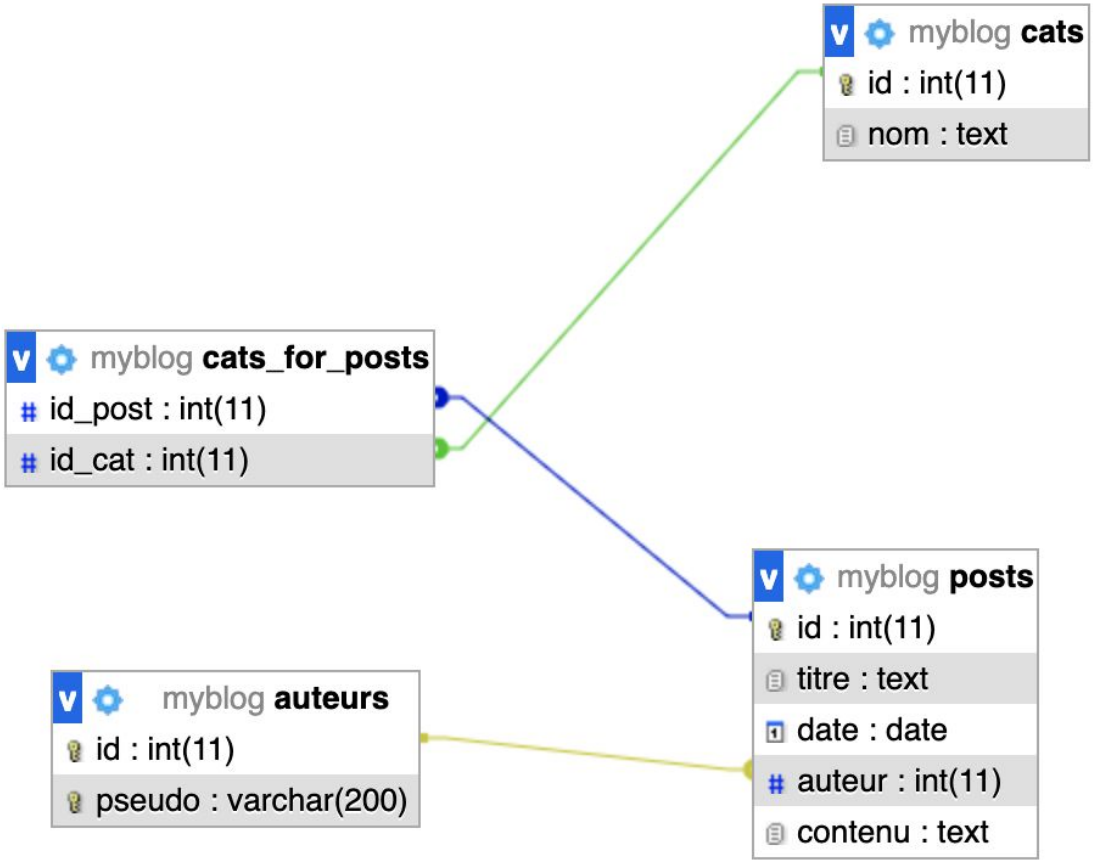
- Categories can contain an unbounded number of posts therefore we cannot put this information in the categories table
- Posts can be part of an unbounded number of categories therefore we cannot put this information in the posts table
- Where the hell to we put this information?
- **In a dedicated table that encodes this relation!**

The cat for post table

Cats for posts

ID_POST	ID_CAT
1	1
1	2
2	2

The big picture



Write data into the tables

```
INSERT INTO `auteurs` (`id`, `pseudo`) VALUES (NULL, 'joe');
INSERT INTO `cats` (`id`, `nom`) VALUES (NULL, 'sport');
INSERT INTO `posts` (`id`, `titre`, `date`, `auteur`,
`contenu`) VALUES (NULL, 'premier post!', '2019-04-01', '1',
'c\'est un super post!')
INSERT INTO `cats_for_posts` (`id`, `nom`) VALUES (NULL,
'sport');
INSERT INTO `cats_for_posts` (`id_post`, `id_cat`) VALUES
('1', '1')
```

Read data from the tables

```
SELECT * FROM `cats`
SELECT (`titre`,`contenu`) FROM `posts`
SELECT (`pseudo`) from `auteurs` WHERE `id`=1
SELECT * FROM `posts` WHERE `date` = ‘2019-04-01’ AND
`pseudo=1
SELECT * FROM `posts` WHERE `auteur` = ( SELECT `id` FROM
`auteurs` WHERE `pseudo` = ‘joe’)
SELECT posts.id as p_id, `auteurs`.id as a_id, titre,
contenu, pseudo  FROM `posts` INNER JOIN `auteurs` ON
`posts`.id_auteur = `auteurs`.id WHERE `pseudo` = ‘joe’
```

Delete data from the table

```
DELETE FROM `posts` WHERE `id` = 1
```

Update data from the table

```
UPDATE posts SET titre = 'toto', date = '2019-04-01' WHERE
id = 1
```

Databases from PHP

```
<?php
$mysqli = mysqli_connect("db.com", "user", "passwd", "db");
if (mysqli_connect_errno($mysqli)) {
  echo "Echec lors de la connexion à MySQL : " .
mysqli_connect_error();
}

$res = mysqli_query($mysqli, "SELECT * FROM posts");
$row = mysqli_fetch_assoc($res);
echo $row['contenu'];
mysqli_free_results($res);
mysqli_close($mysqli);
?>
```