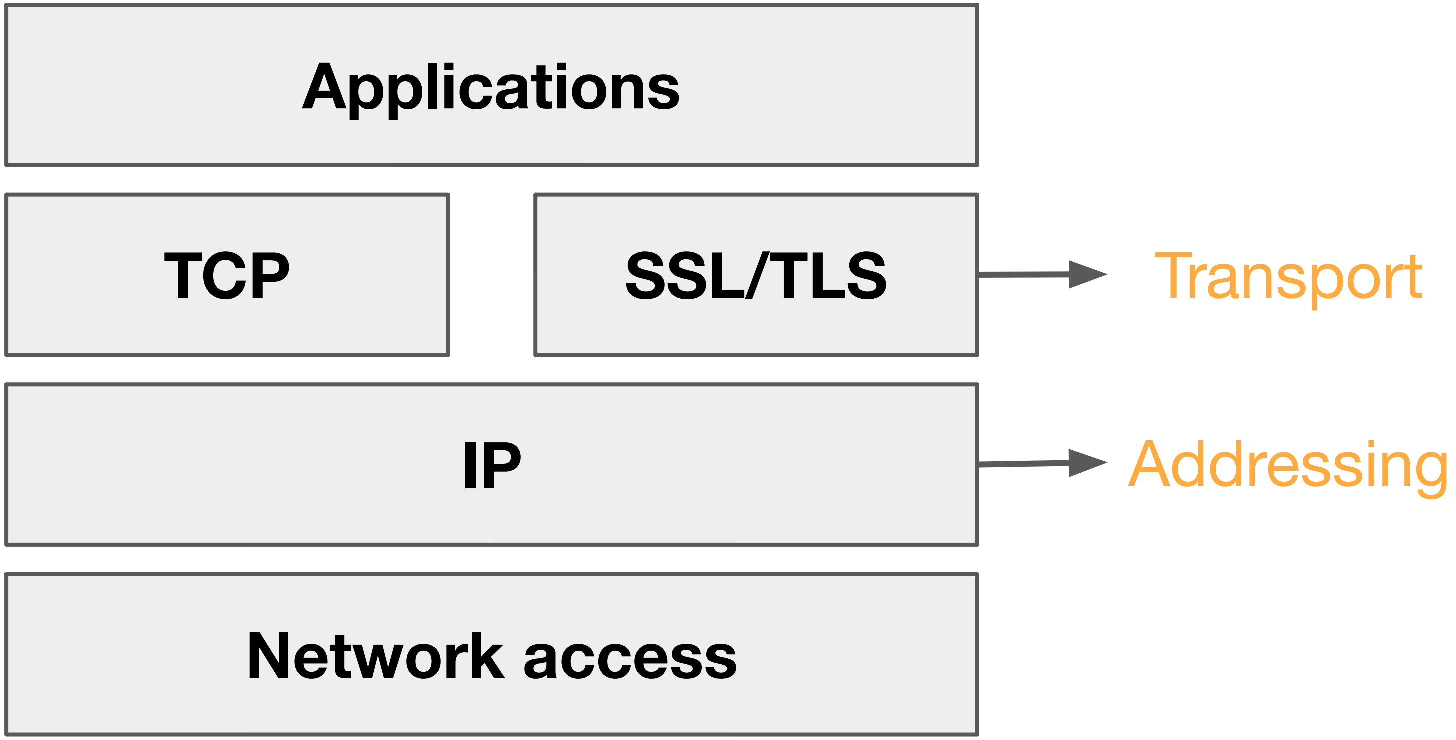
Introduction

What is internet?

Wikipedia : a global system of interconnected computer networks



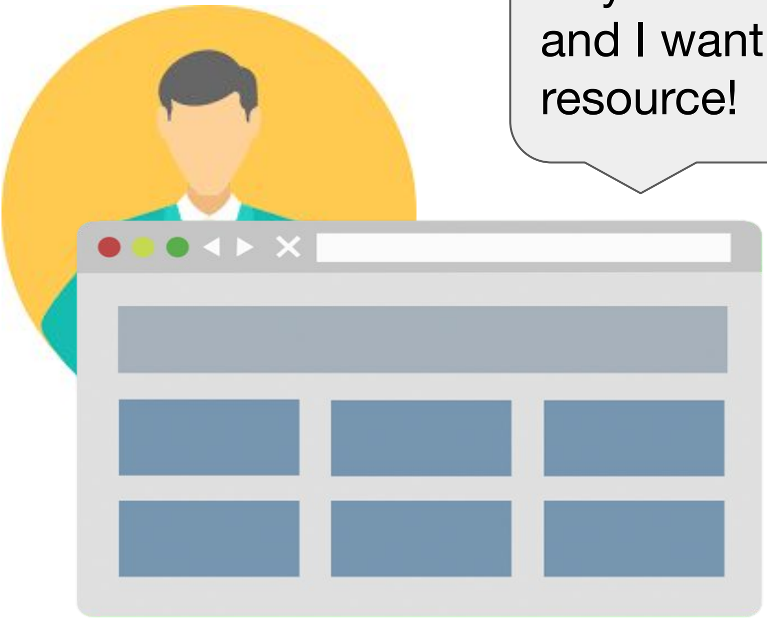
What is the web?

- 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, …)

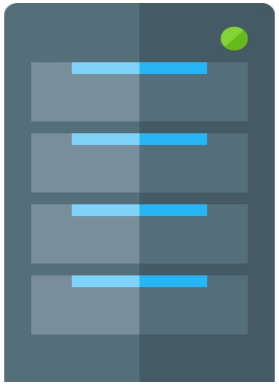
The **client-server** architecture

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

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



Client (Browsers)

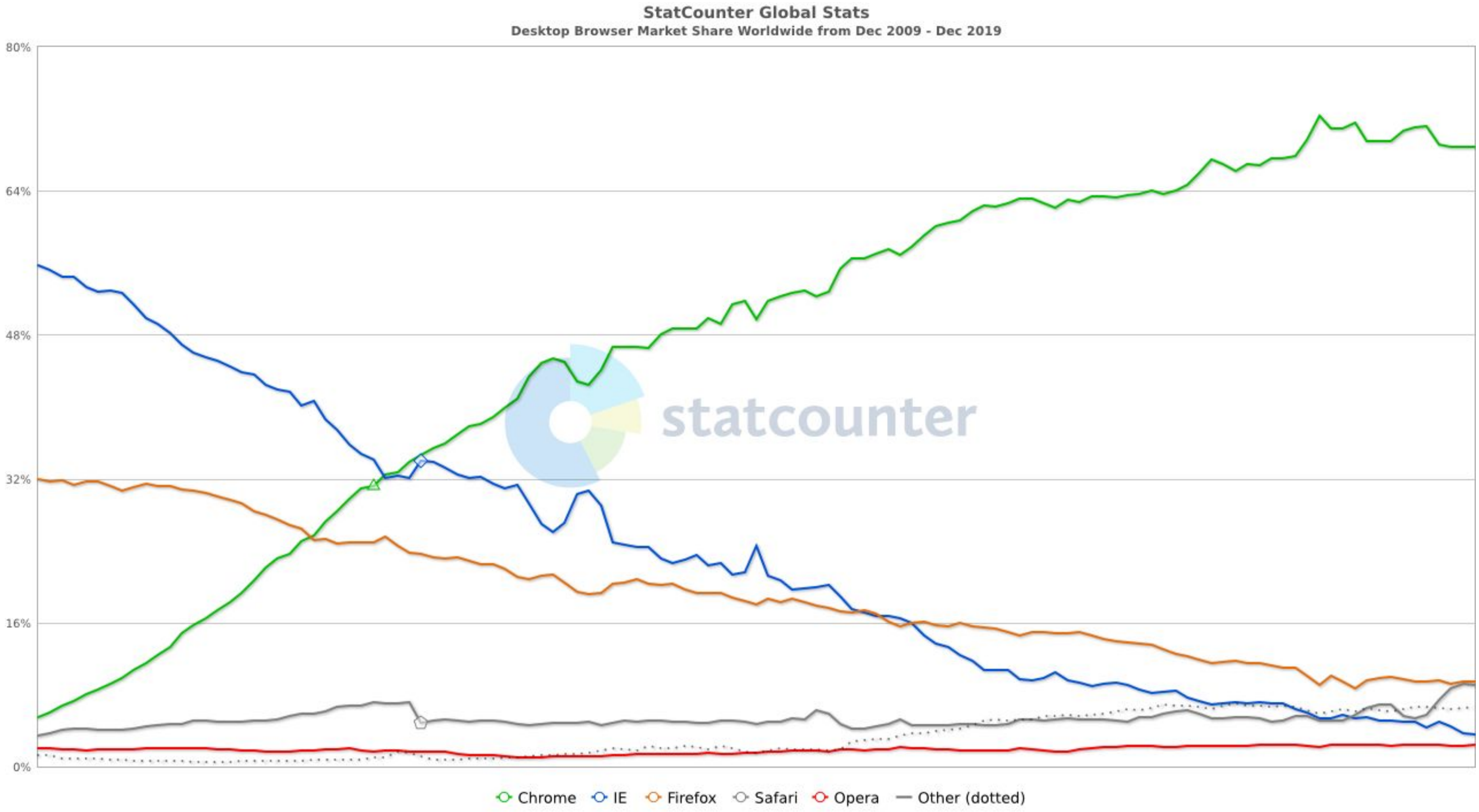


(Web) Server

What the heck is a browser?

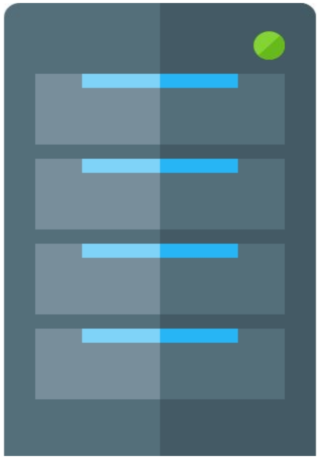
- Native program that allows a user to transparently access web resources
- Also, it can execute arbitrary code (JavaScript code only)
- Nowadays, shipped by default in most desktop operating systems
  - Safari on Mac OS
  - Edge on Windows
  - Firefox on Linux distributions
  - But the more famous is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

Browsers in practice

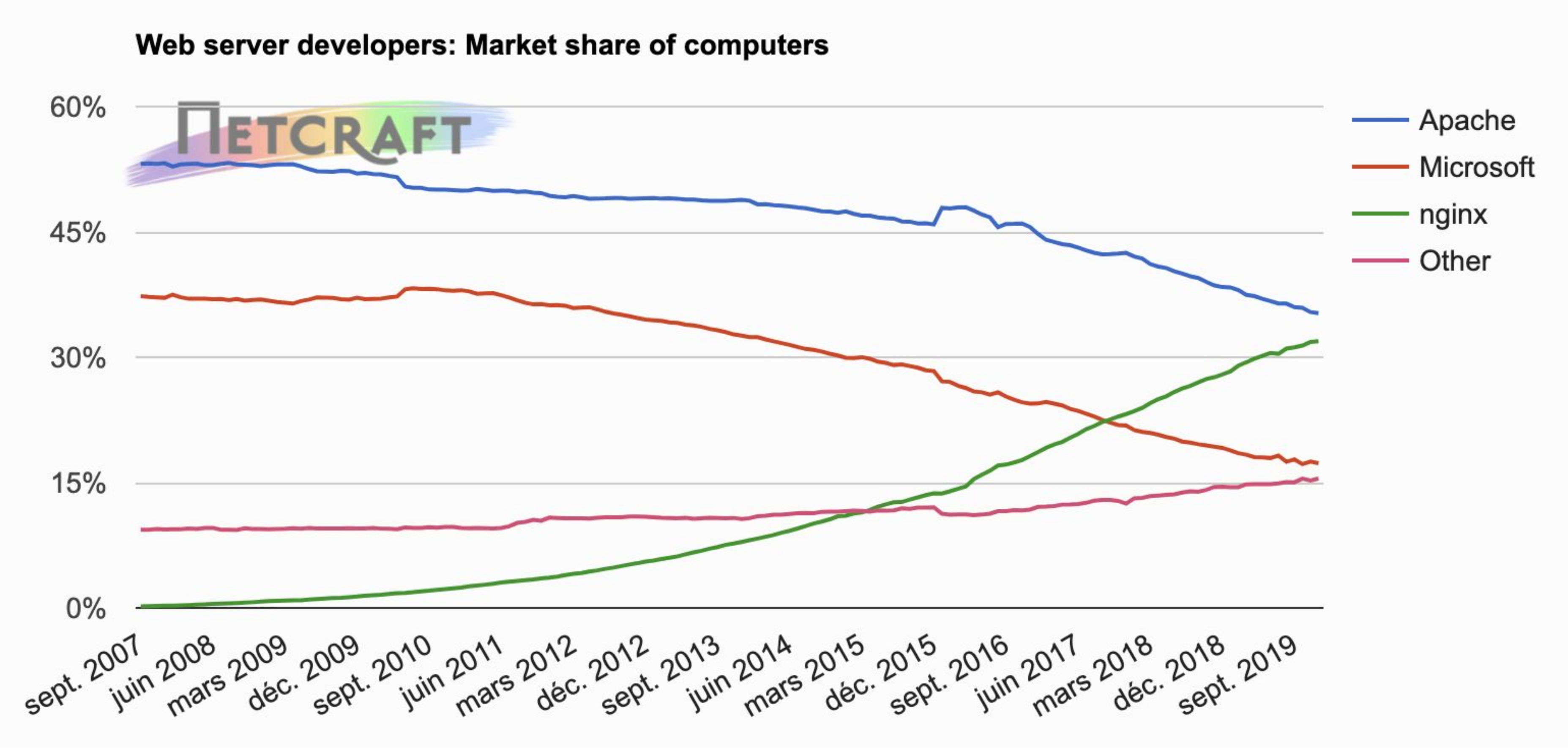


What the heck is a web server?

- 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

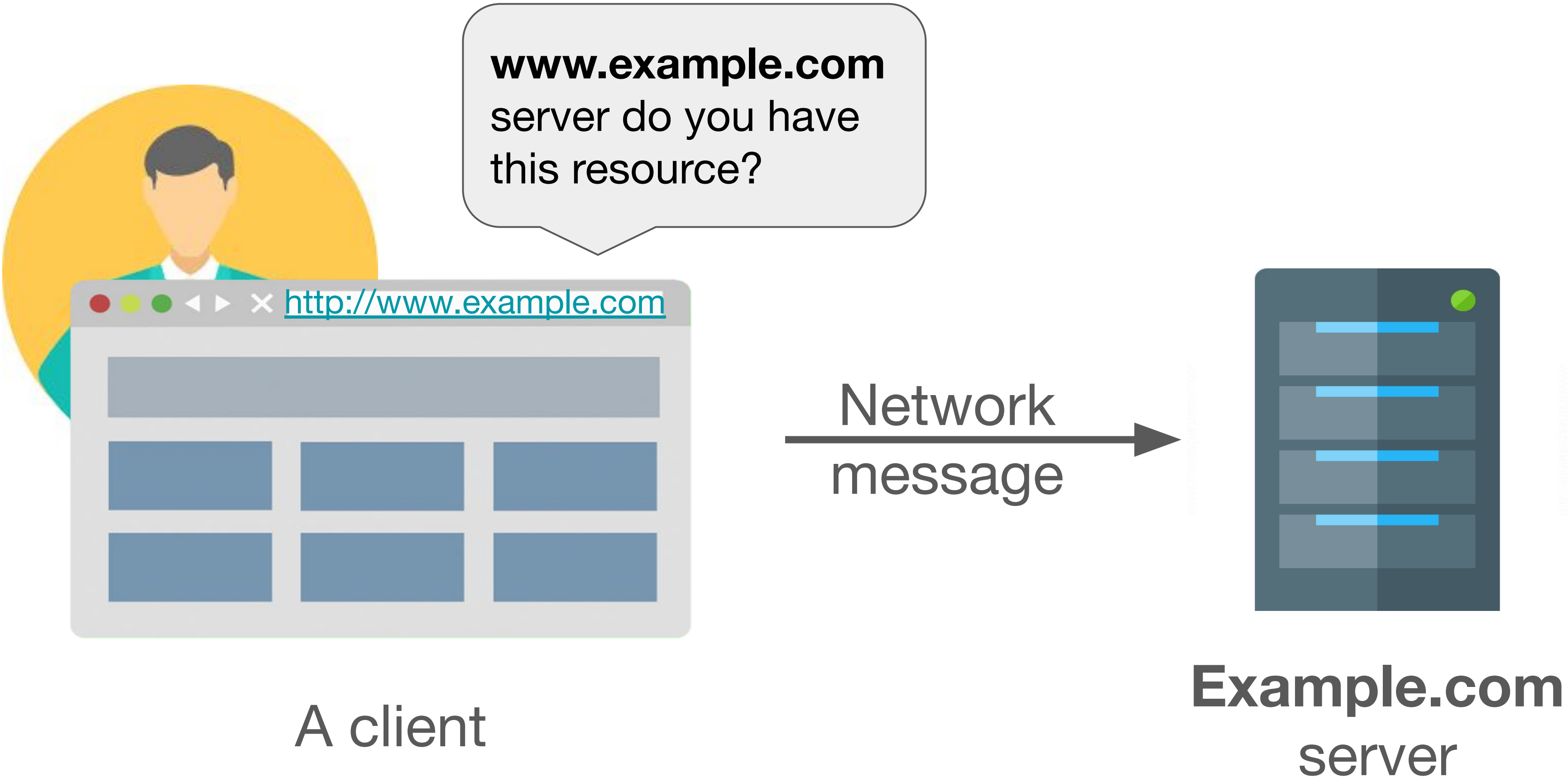


A web application **in a nutshell**

Hey web browser! Can you retrieve for me the marvelous **http://www.example.com** homepage?
http://www.example.com

A client

A web application **in a nutshell**



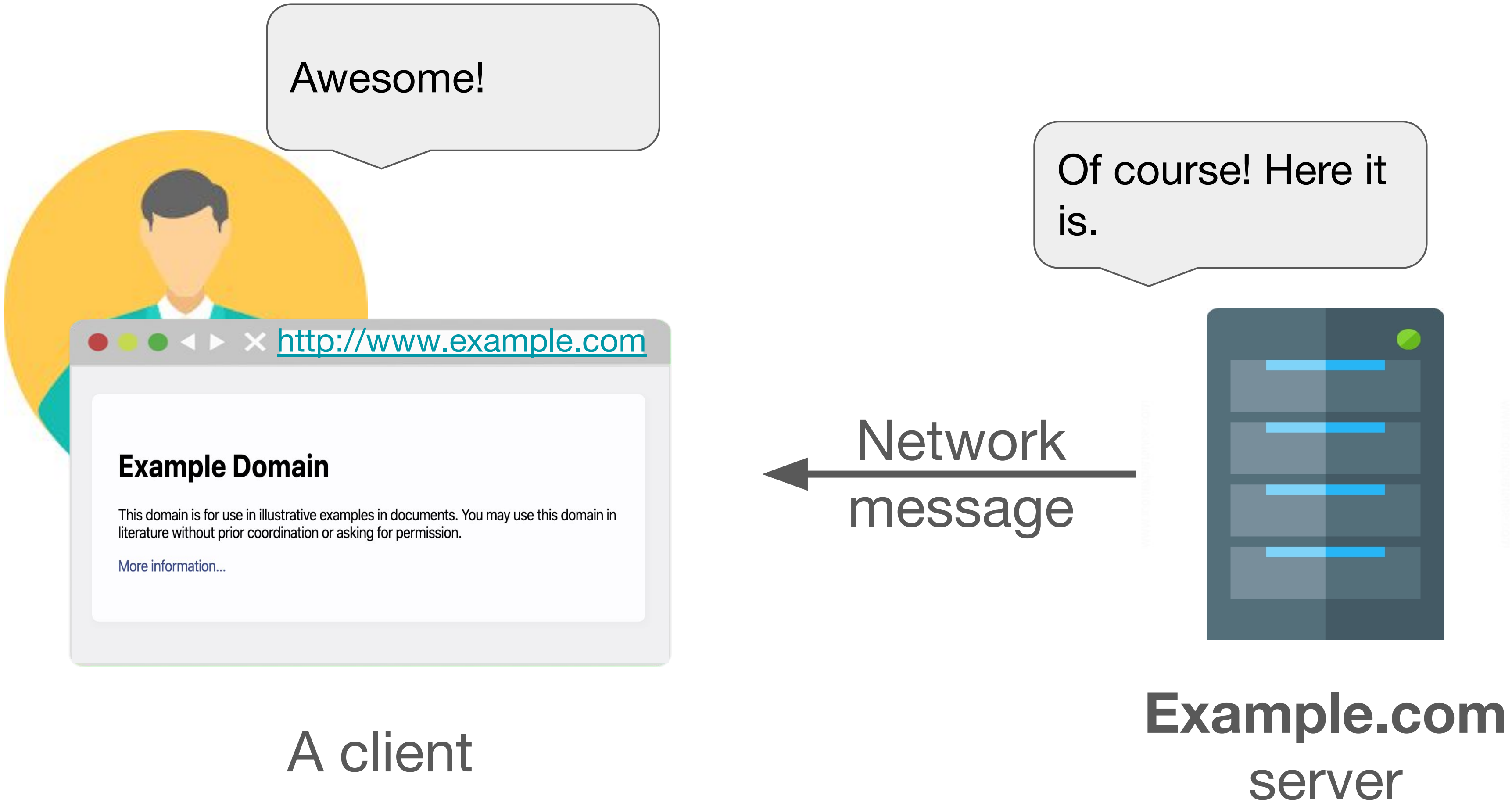
www.example.com server do you have this resource?

Network message

A client

Example.com server

A web application **in a nutshell**



Awesome!

http://www.example.com

Example Domain

This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.

More information...

Of course! Here it is.

Network message

A client

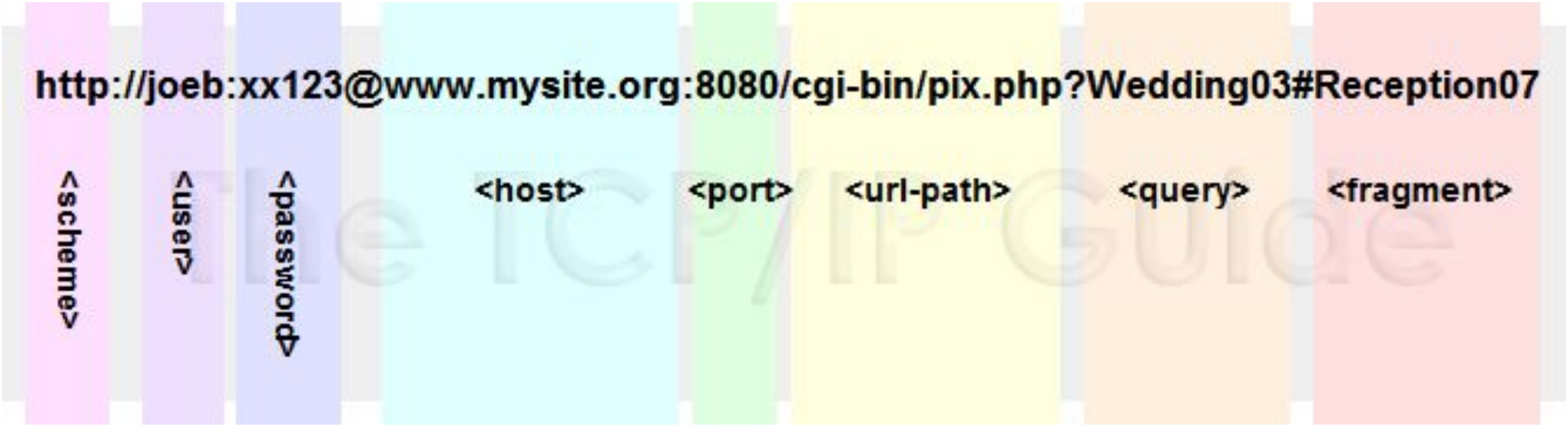
Example.com server

A step back

This was a rather handwavy explanation!

All started by entering **http://www.example.com** in the browser **what is this**?

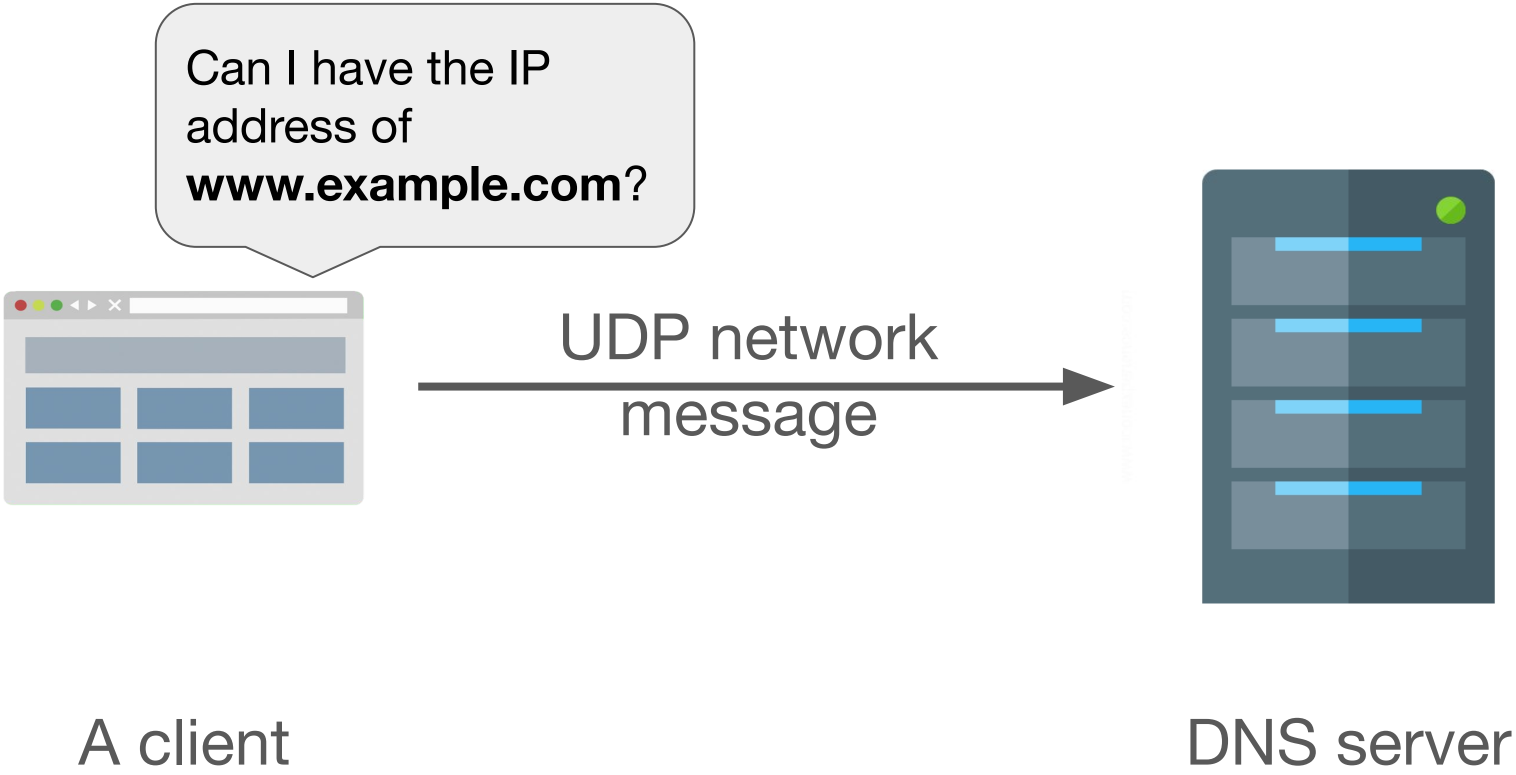
A Uniform Resource Locator (URL)



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)



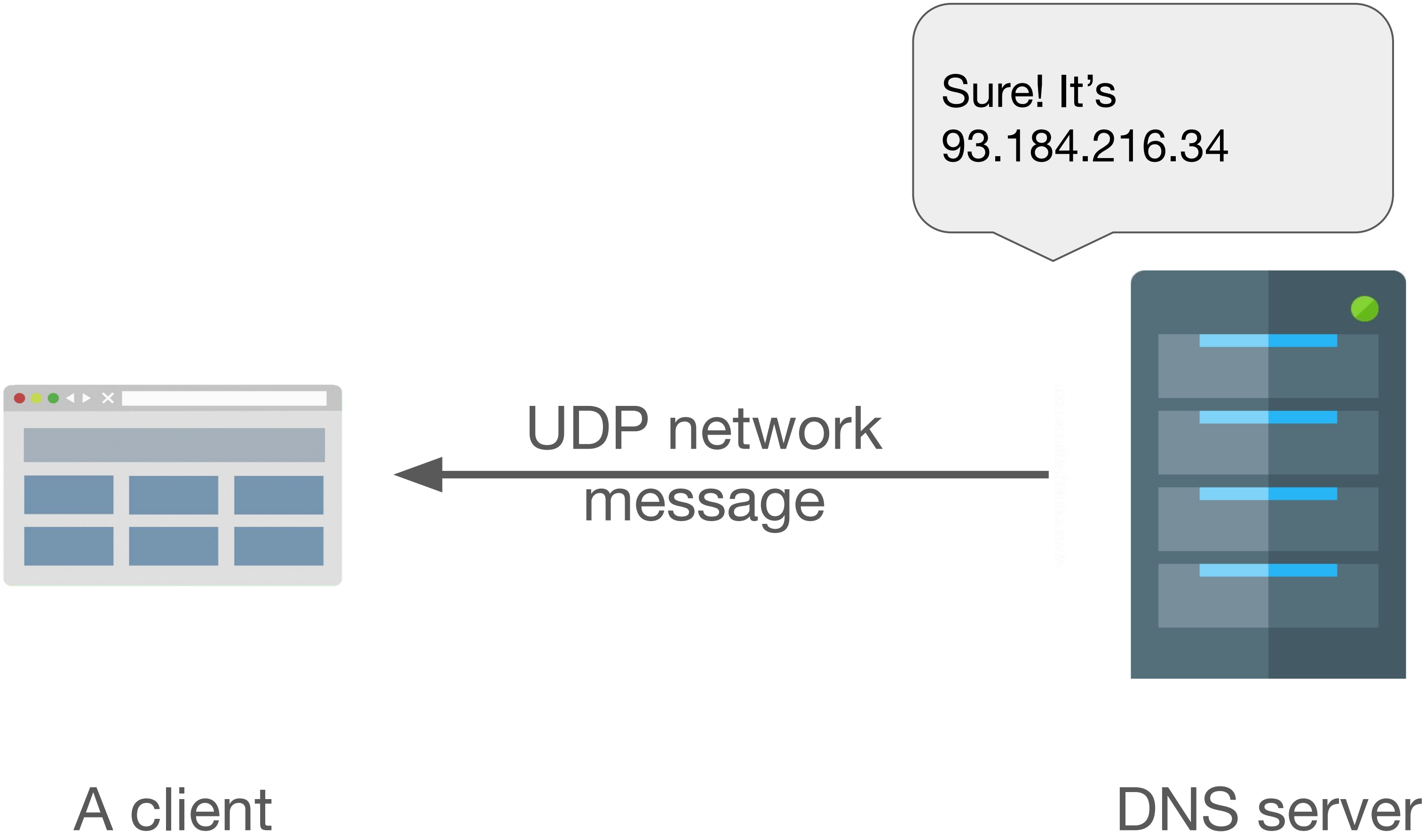
Can I have the IP address of **www.example.com**?

UDP network message

A client

DNS server

Domain Name System (DNS)



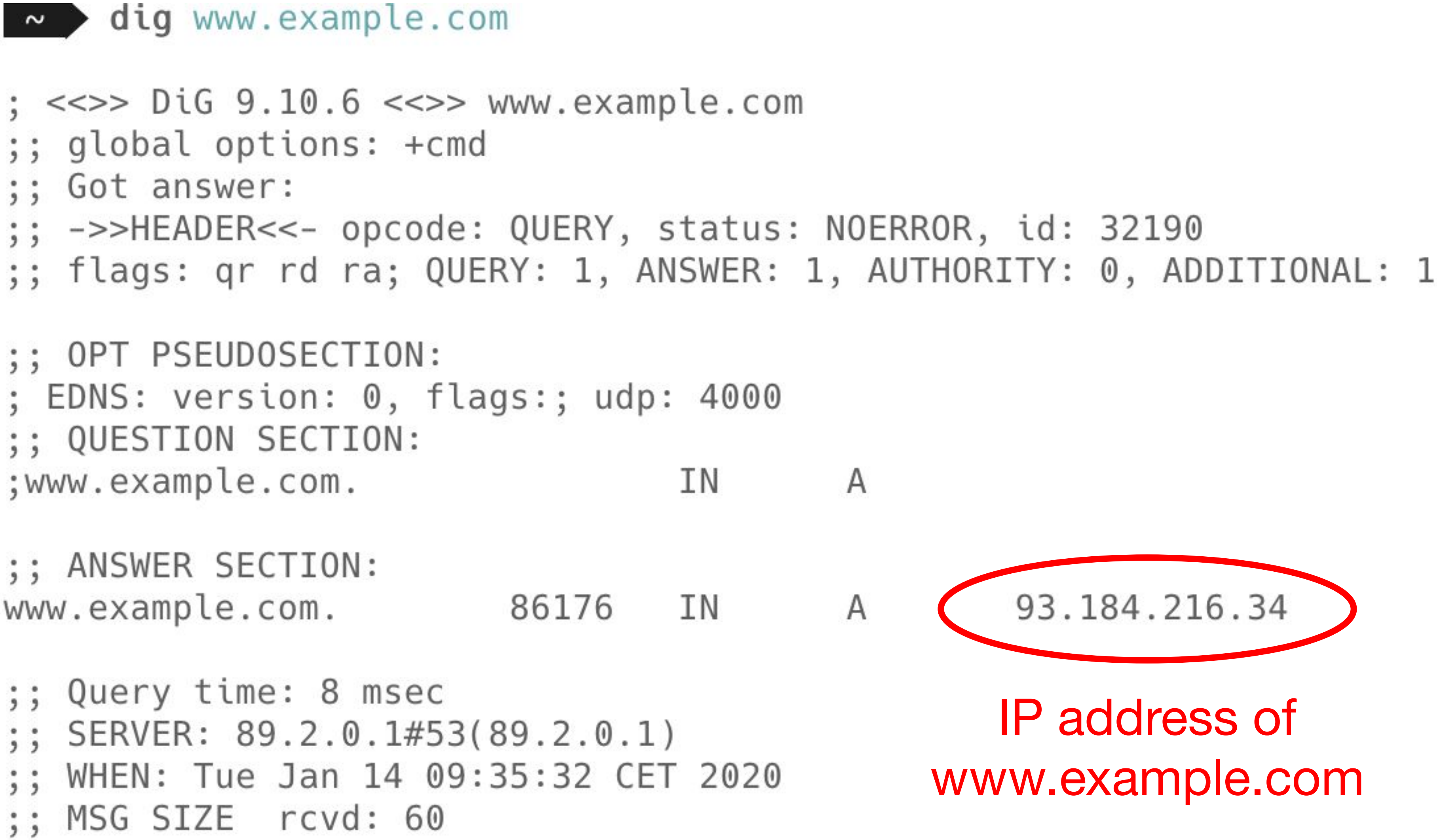
Sure! It’s 93.184.216.34

UDP network message

A client

DNS server

Under the hood



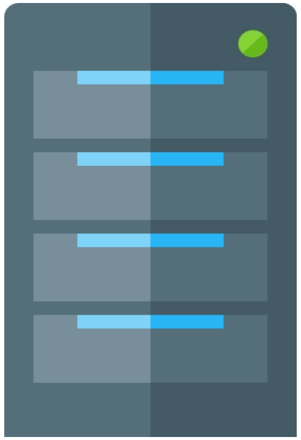
```
~ dig www.example.com

; <<>> 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

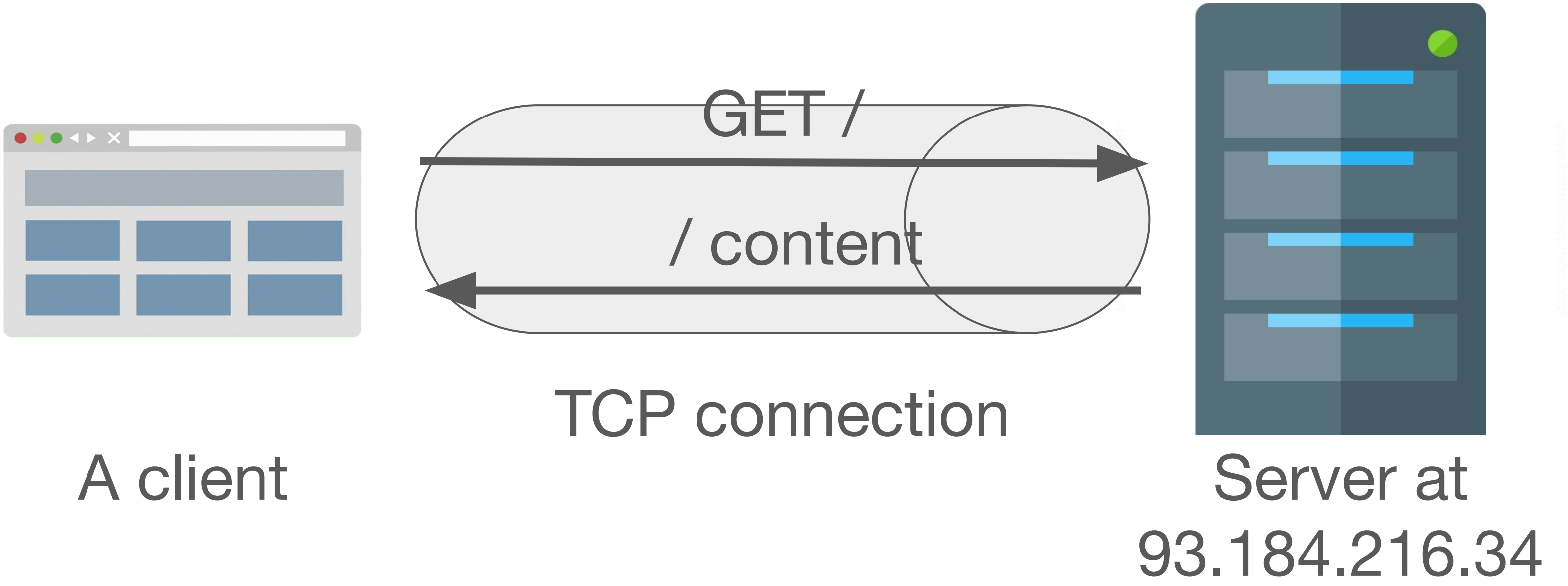


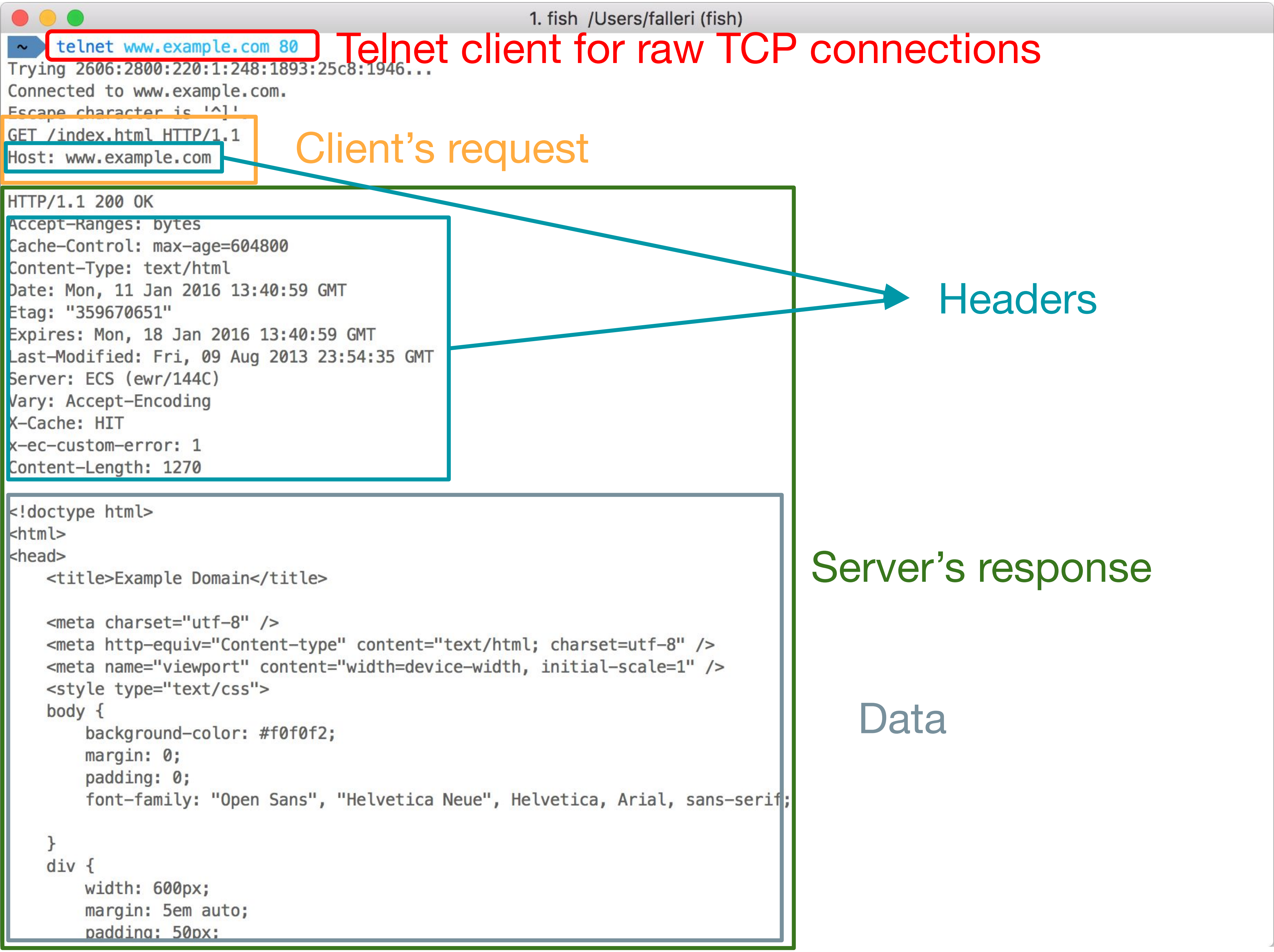
Server at 93.184.216.34

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

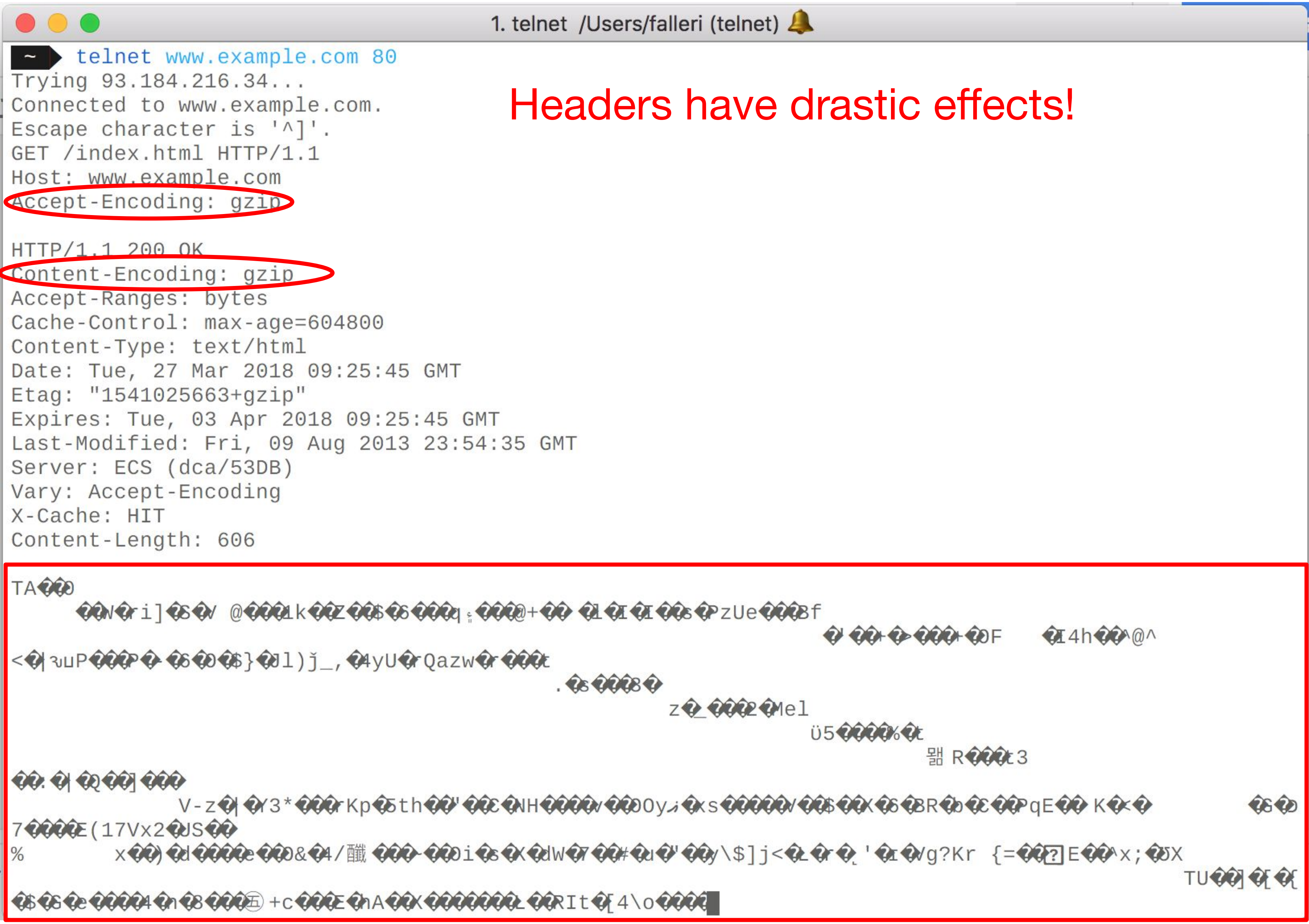
Hypertext Transfer Protocol (HTTP)

- 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



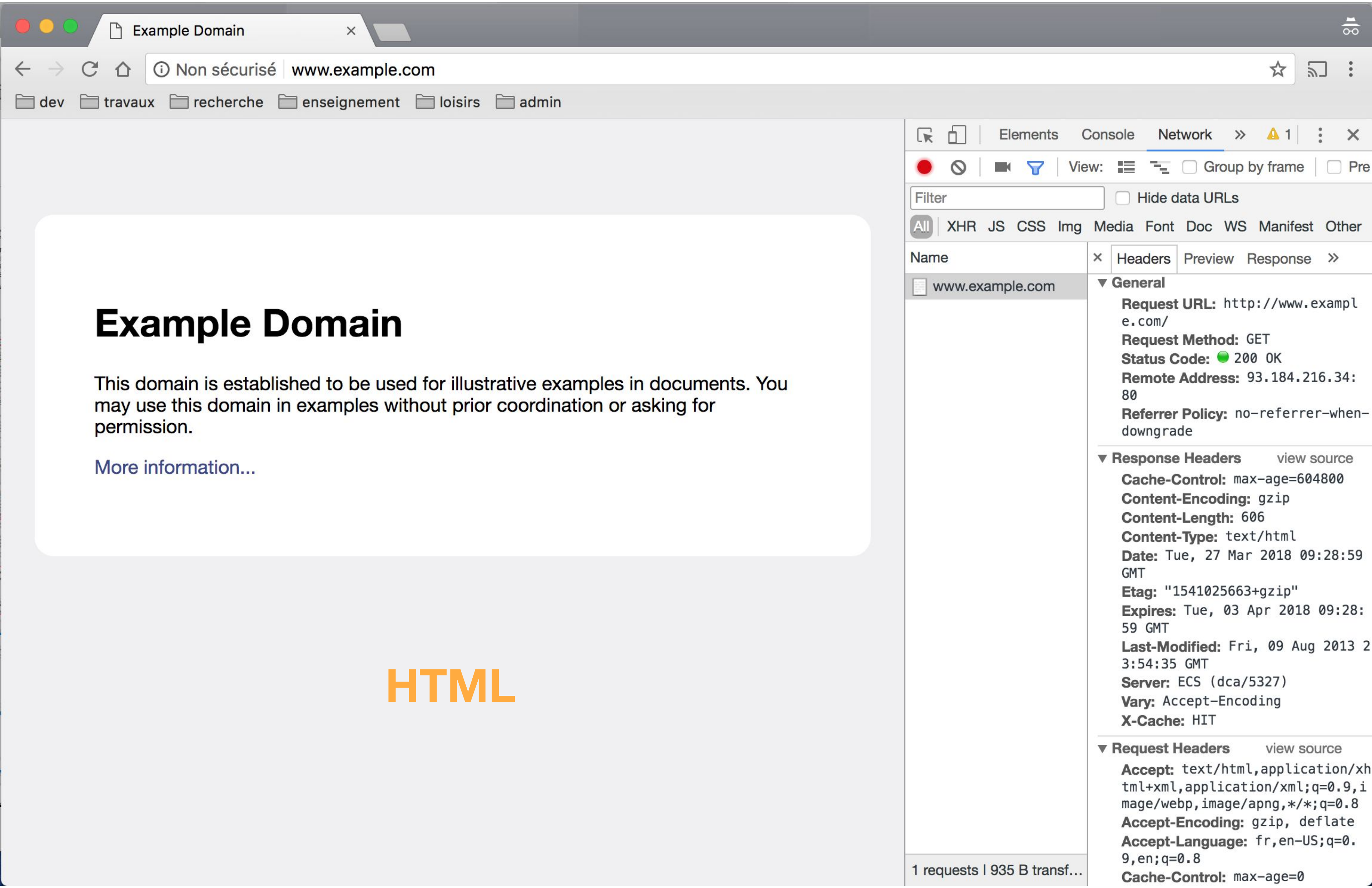

1. fish /Users/falleri (fish)
Telnet client for raw TCP connections
Client’s request
Headers
Server’s response
Data

```
~ 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;
```



Headers have drastic effects!

```
~ 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

TA��0
     ��N�ri]�S�V @���1k��Z��$�6���q ���@+�� �l�I�I��s�PzUe���Bf
                                                               � ��+�>���+�0F    �I4h��^@^
<�|ɿuP���P�-�6�0�$}�Jl)ǰ_,�4yU�rQazw�r���t
                                          .�s���8�
                                                  z�_���2�Mel
                                                             Ü5����%�t
                                                                      뫪 R���t3
��:�|�Q��]���
             V-z�|�Y3*���rKp�5th��"��C�NH����v��0Oy�xs�����V��$��X�6�BR�b�C��PqE�� K�<�      �G�D
7����E(17Vx2�JS��
%       x��)�d����e��0&�4/�later ���-��0i�s�X�dW�7��#�u�"��y\$]j<�L�r� '�r�Vg?Kr {=��?E��^x;�5X
                                                                                          TU��]�[�{
�$�G�e����4�n�8���㊄+c���E�hA��X�������L��RIt�[4\o����
```



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...

▼ 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

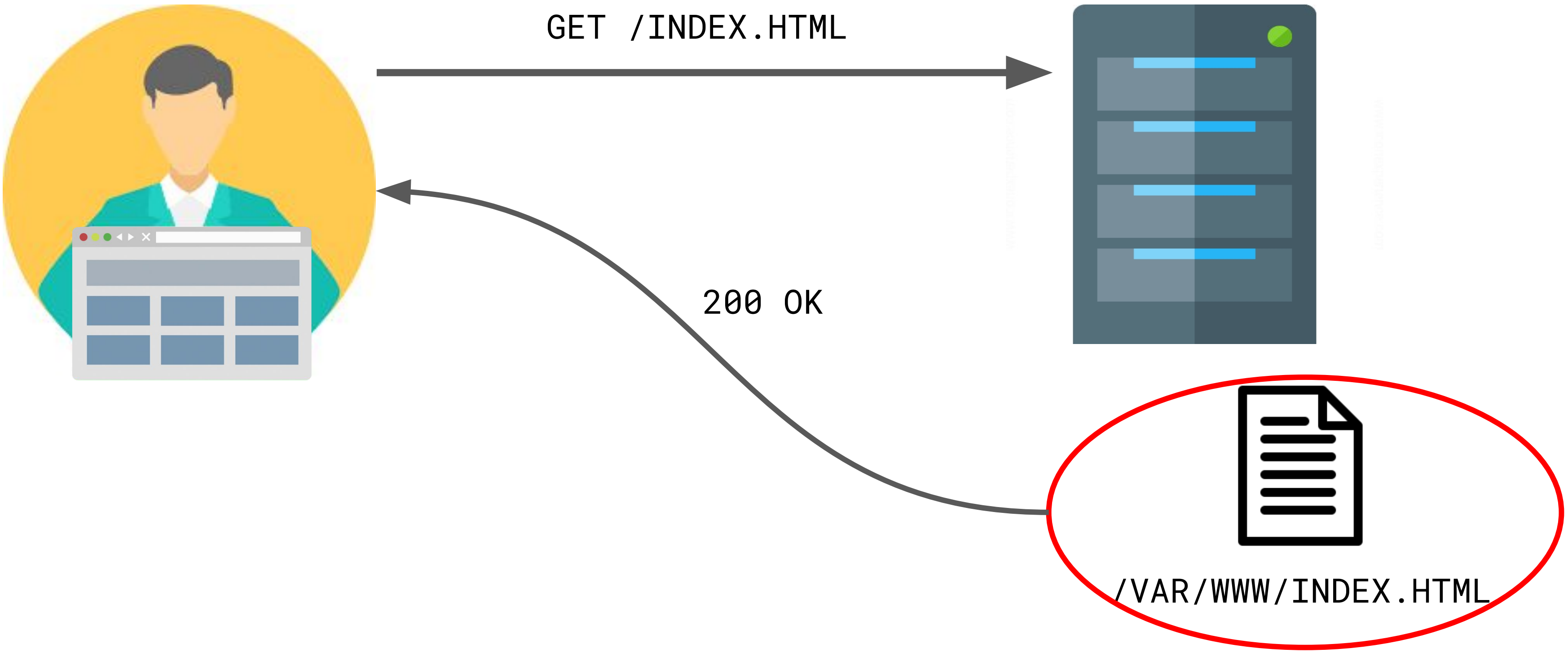
Cache-Control: max-age=0

HTML

Web applications

- Client-server applications running through the web
- Users interact with them using a browser
- Competitive advantage : no 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

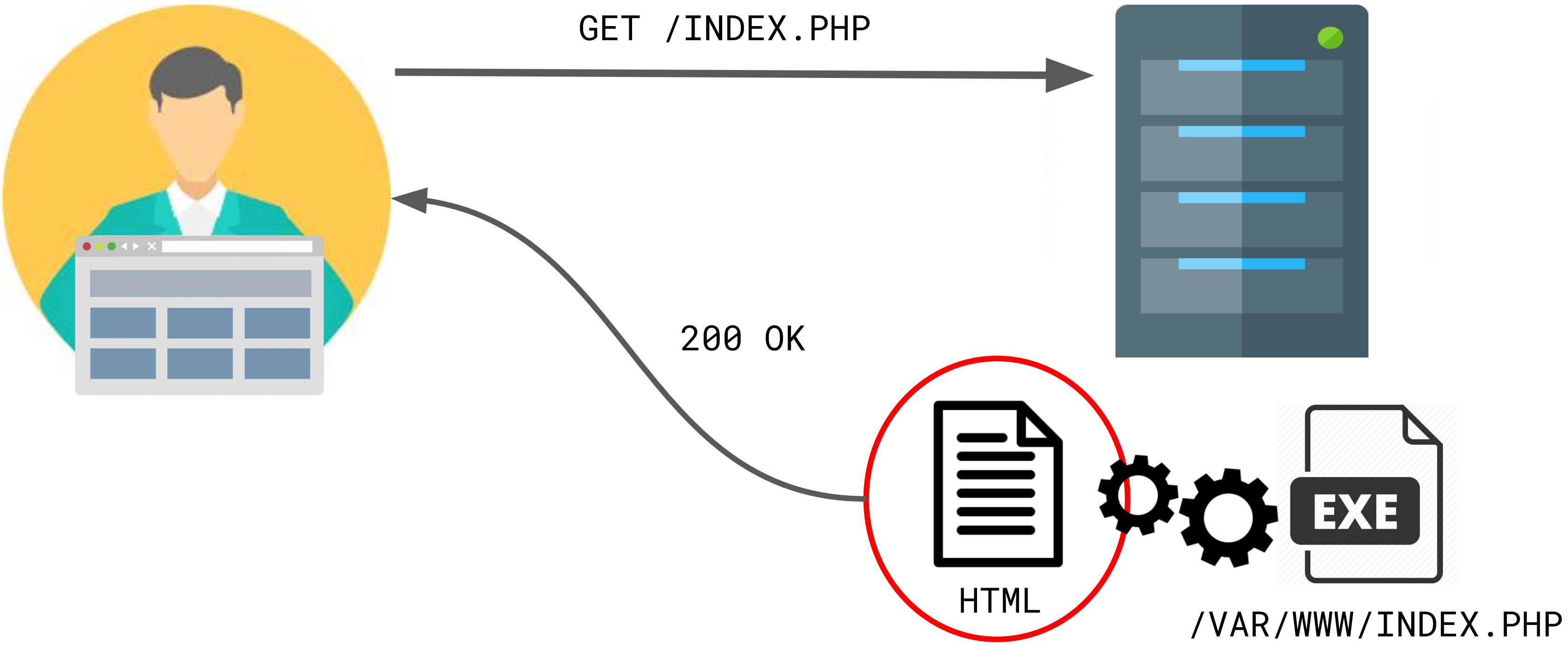


GET /INDEX.HTML

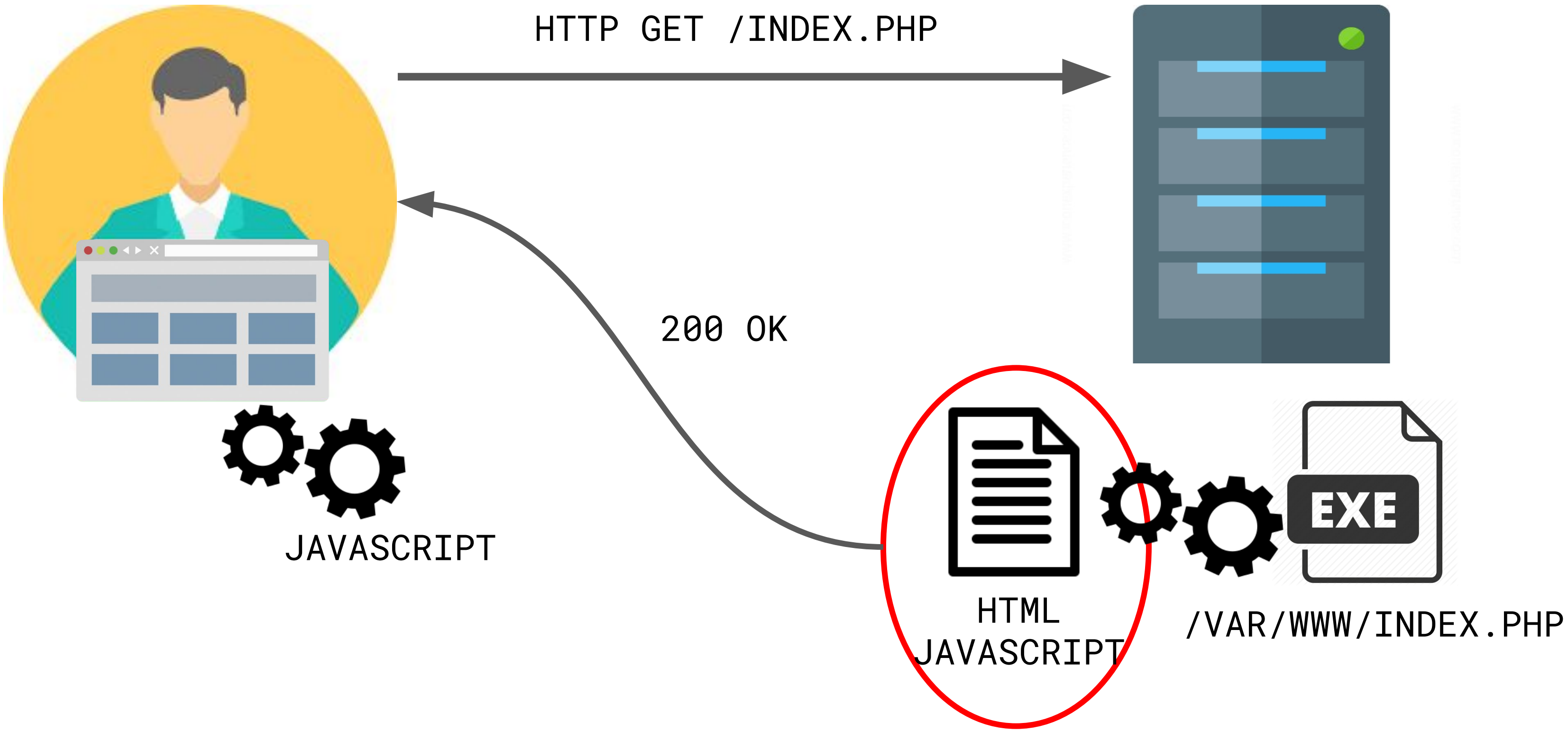
200 OK

/VAR/WWW/INDEX.HTML

Server-dynamic web applications



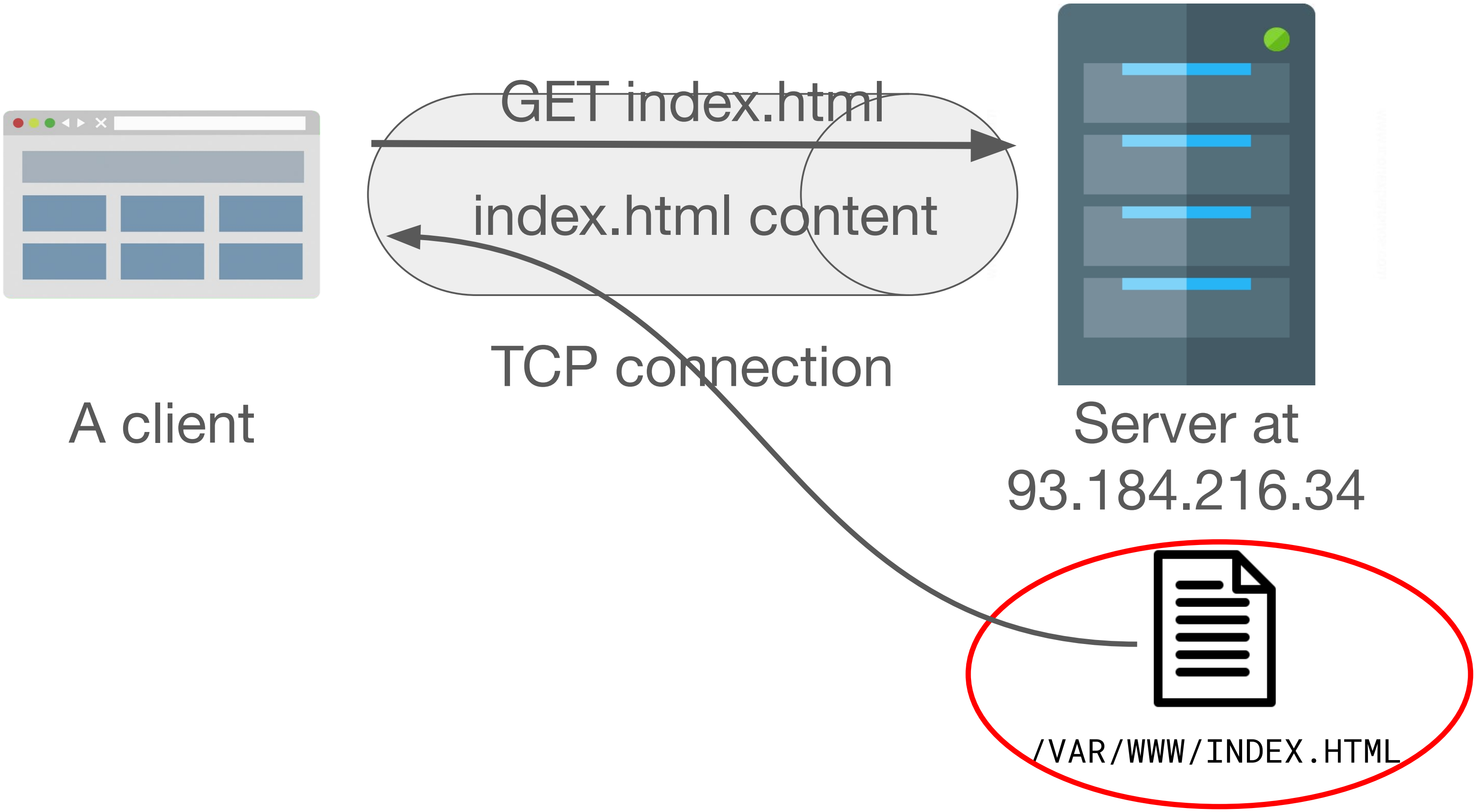
Server and client-dynamic web applications



HTML

Previously ...

A **static** web application



GET index.html

index.html content

TCP connection

A client

Server at 93.184.216.34

/VAR/WWW/INDEX.HTML

Take home message

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

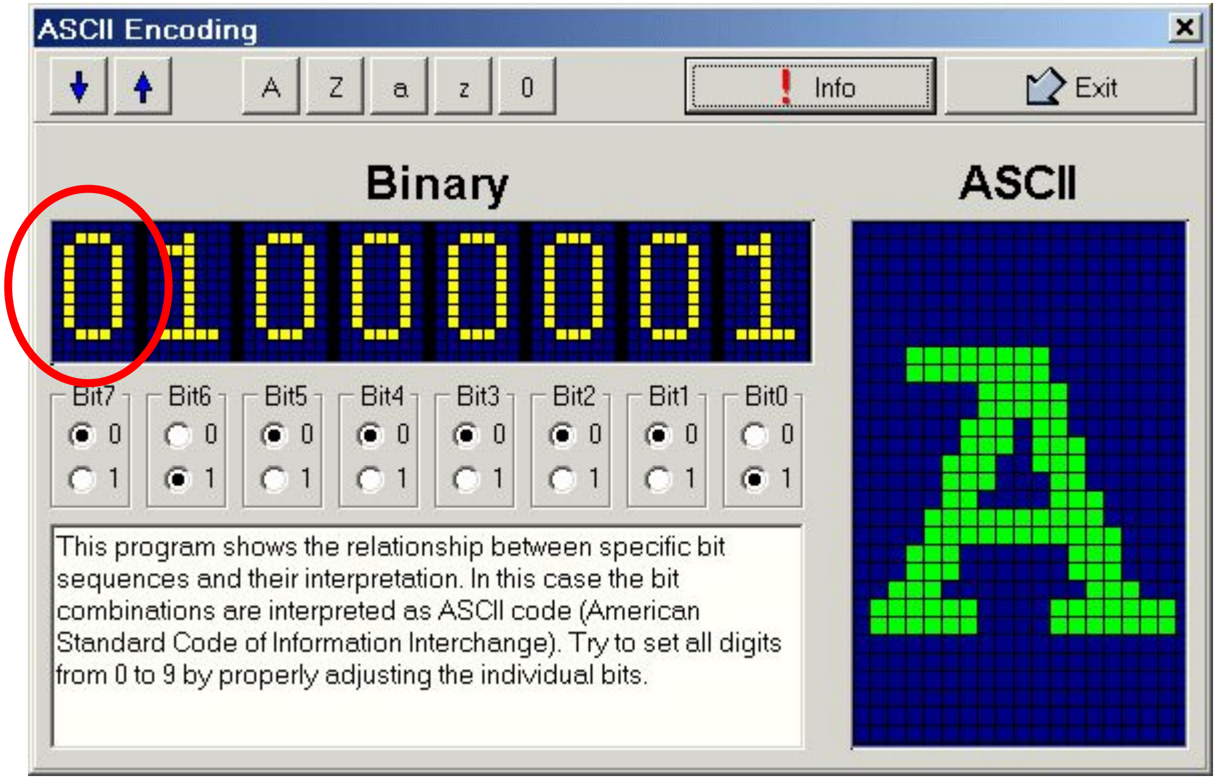
- HTML resources (a logical document) **today**
- CSS resources (aesthetic properties) **next episode**
- Some binary resources

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

Bit7: ● 0 ○ 1
Bit6: ○ 0 ● 1
Bit5: ● 0 ○ 1
Bit4: ● 0 ○ 1
Bit3: ● 0 ○ 1
Bit2: ● 0 ○ 1
Bit1: ● 0 ○ 1
Bit0: ○ 0 ● 1

This program shows the relationship between specific bit sequences and their interpretation. In this case the bit combinations are interpreted as ASCII code (American Standard Code of Information Interchange). Try to set all digits from 0 to 9 by properly adjusting the individual bits.

Plain old text

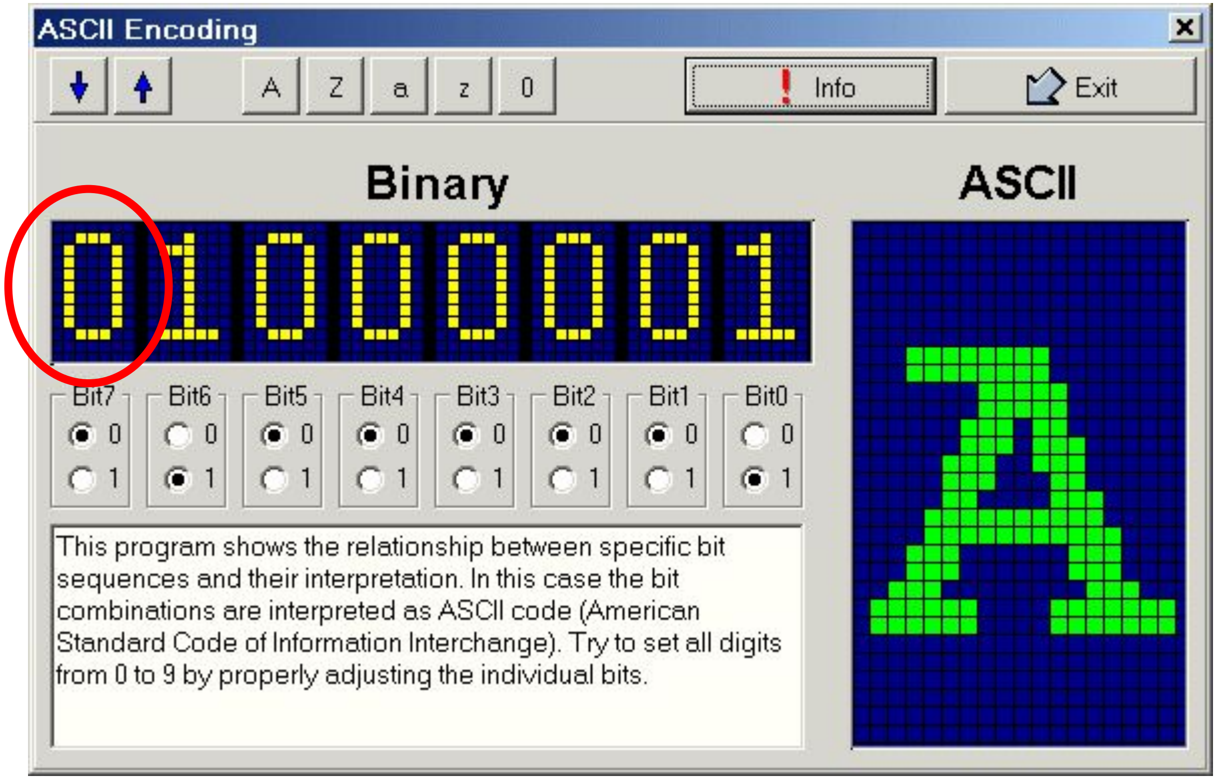
01101100 01101111 01101100

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!

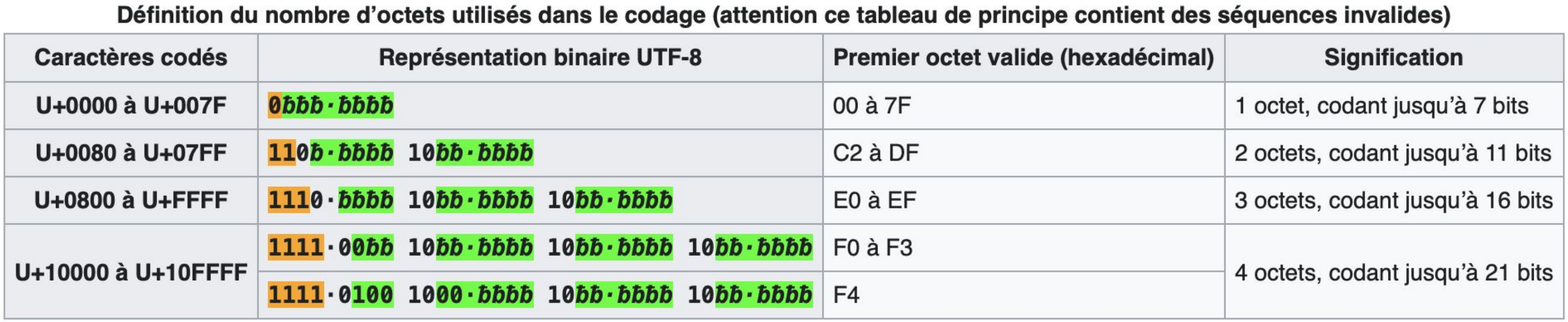


Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
● 0	○ 0	● 0	● 0	● 0	● 0	● 0	○ 0
○ 1	● 1	○ 1	○ 1	○ 1	○ 1	○ 1	● 1

This program shows the relationship between specific bit sequences and their interpretation. In this case the bit combinations are interpreted as ASCII code (American Standard Code of Information Interchange). Try to set all digits from 0 to 9 by properly adjusting the individual bits.

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

The UTF tables



Définition du nombre d’octets utilisés dans le codage (attention ce tableau de principe contient des séquences invalides)

Caractères codés	Représentation binaire UTF-8	Premier octet valide (hexadécimal)	Signification
U+0000 à U+007F	0bbb·bbbb	00 à 7F	1 octet, codant jusqu’à 7 bits
U+0080 à U+07FF	110b·bbbb 10bb·bbbb	C2 à DF	2 octets, codant jusqu’à 11 bits
U+0800 à U+FFFF	1110·bbbb 10bb·bbbb 10bb·bbbb	E0 à EF	3 octets, codant jusqu’à 16 bits
U+10000 à U+10FFFF	1111·00bb 10bb·bbbb 10bb·bbbb 10bb·bbbb	F0 à F3	4 octets, codant jusqu’à 21 bits
	1111·0100 1000·bbbb 10bb·bbbb 10bb·bbbb	F4	

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

Why this fuss about text?

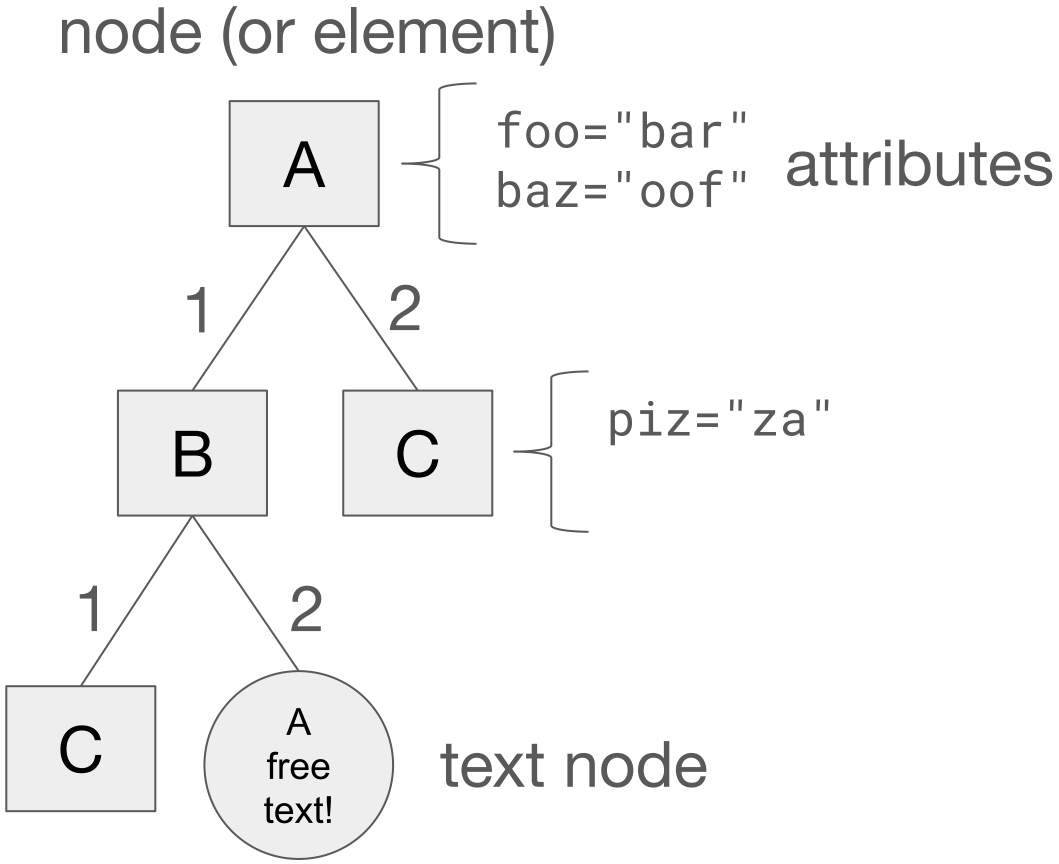
HTML resources contains primarily text, so you have to know how it works unless you like showing � to the users

- 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

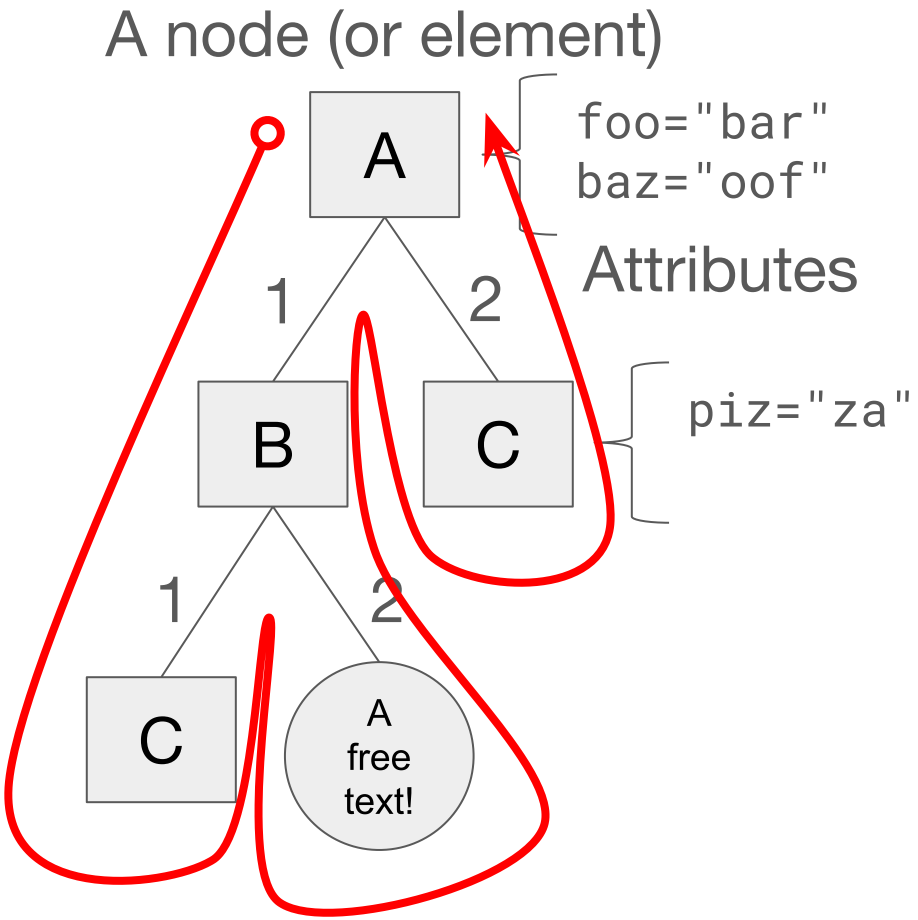
Now: Hypertext Markup Language (HTML)

- 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

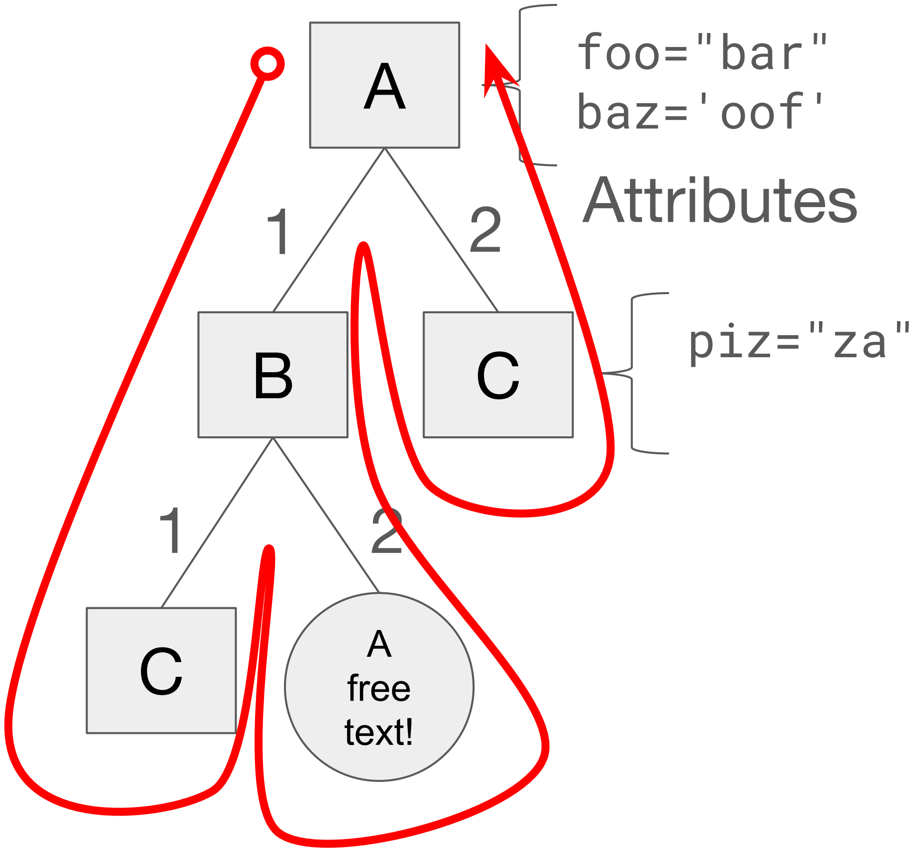


Rules :

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

XML tree traversal

A node (or element)



foo="bar"
baz='oof'

Attributes

piz="za"

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 **`&lt;`**
  - **`&nbsp;`**
  - **`&amp;`**
  - **`&gt;`**
- 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
```

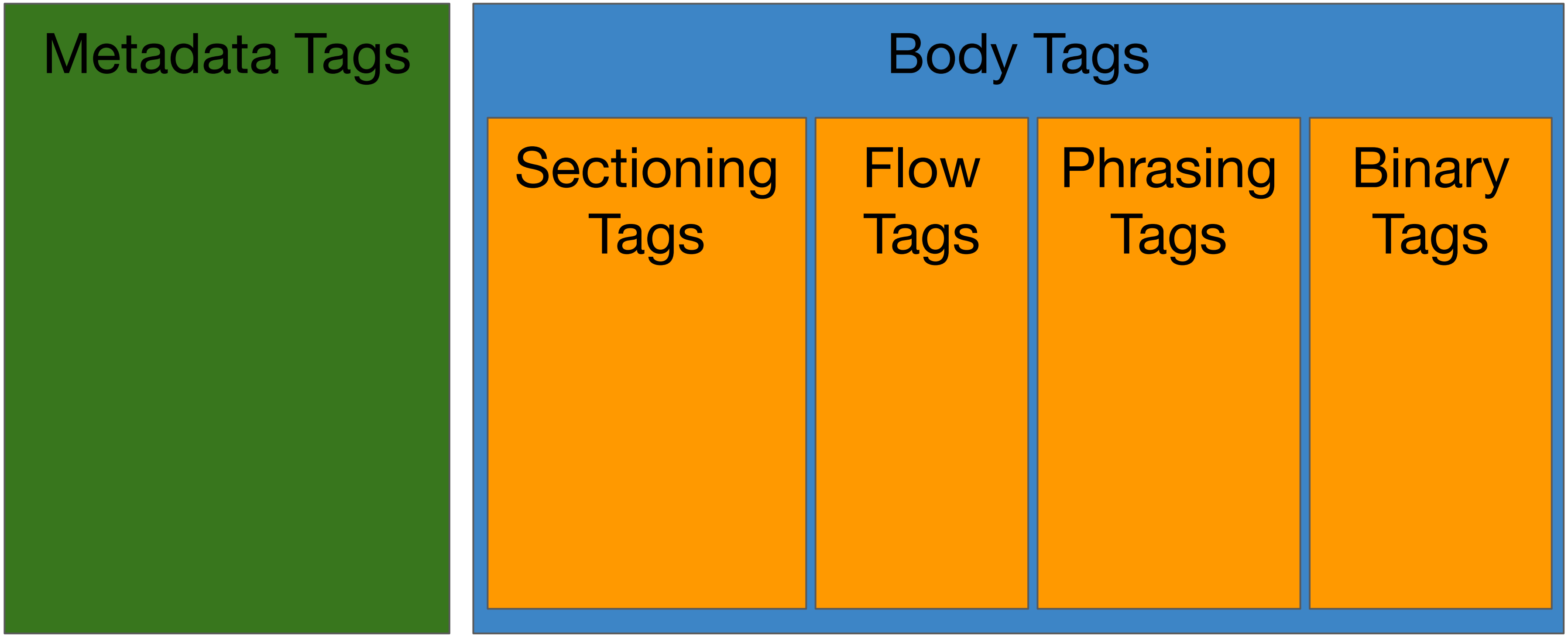
Nice! But what about damn HTML?

- 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. <br>)
- 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



Metadata Tags

Body Tags

Sectioning Tags

Flow Tags

Phrasing Tags

Binary 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

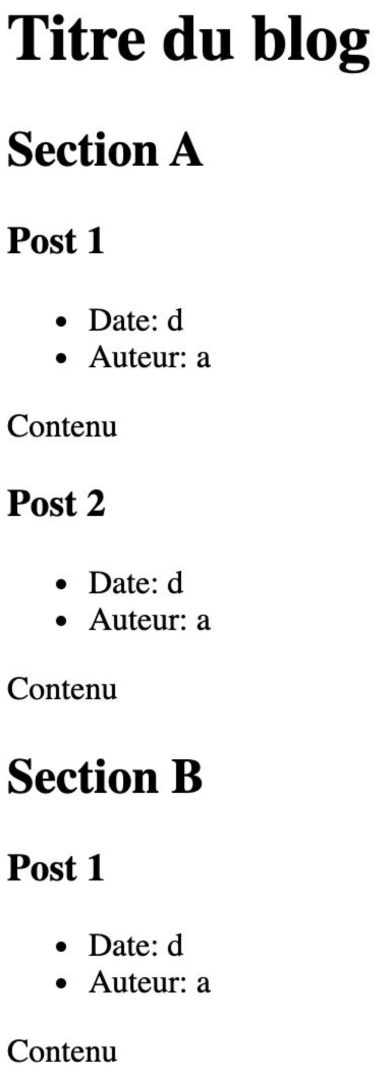
- `<em>`
- `<strong>`
- `<mark>`
- `<span>`

Binary tags, the best-of

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

CSS

My blog



Titre du blog

Section A

Post 1

- Date: d
- Auteur: a

Contenu

Post 2

- Date: d
- Auteur: a

Contenu

Section B

Post 1

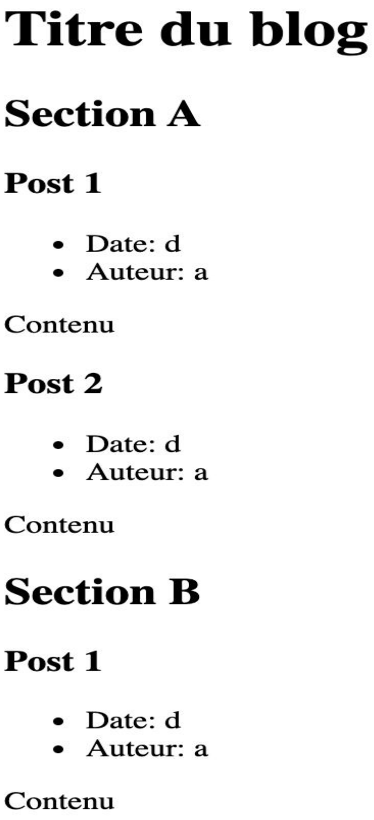
- Date: d
- Auteur: a

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

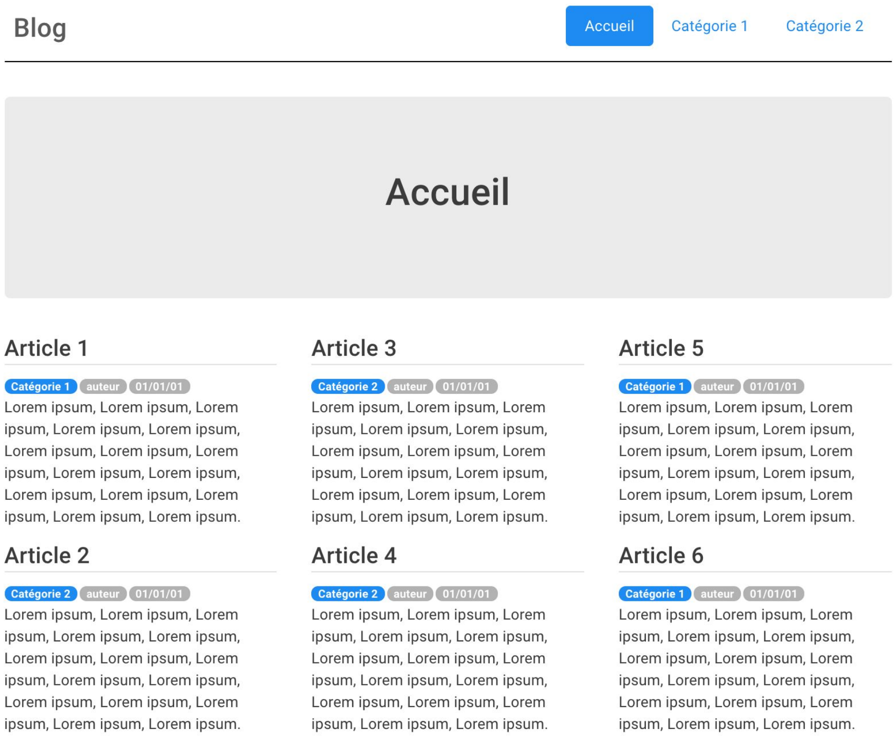
Section B

Post 1

- Date: d
- Auteur: a

Contenu

Into this?



Accueil

Article 1

Catégorie 1 auteur 01/01/01

Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum.

Article 2

Catégorie 2 auteur 01/01/01

Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum.

Article 3

Catégorie 2 auteur 01/01/01

Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum.

Article 4

Catégorie 2 auteur 01/01/01

Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum.

Article 5

Catégorie 1 auteur 01/01/01

Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum.

Article 6

Catégorie 1 auteur 01/01/01

Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum, Lorem ipsum.

Cascading Style Sheets

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

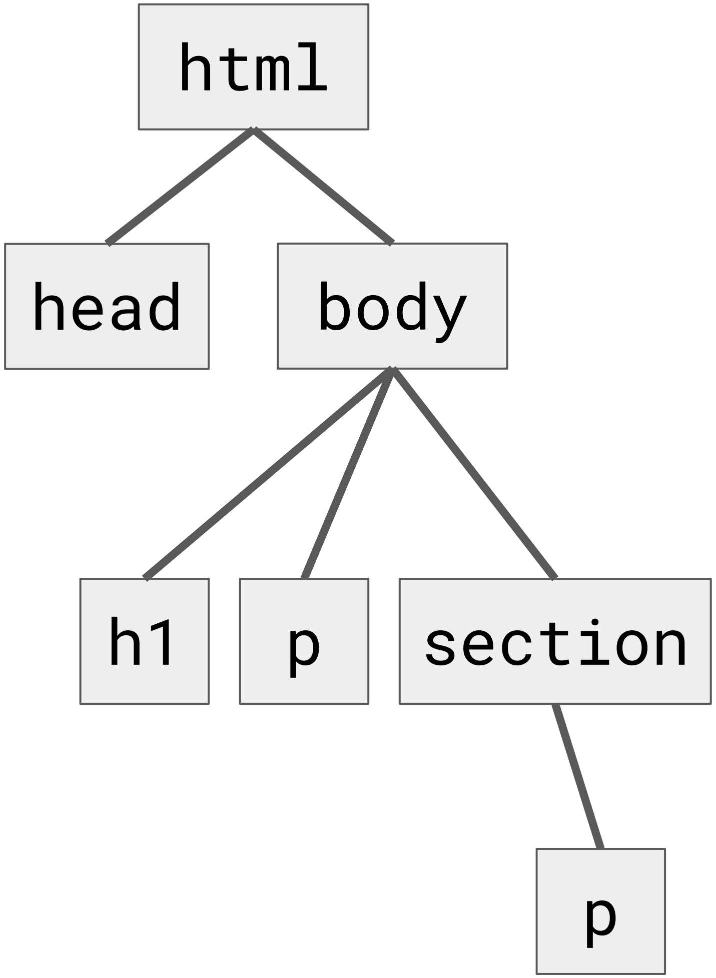
```css
selector {
    property: value;
}
```

How does that works?

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

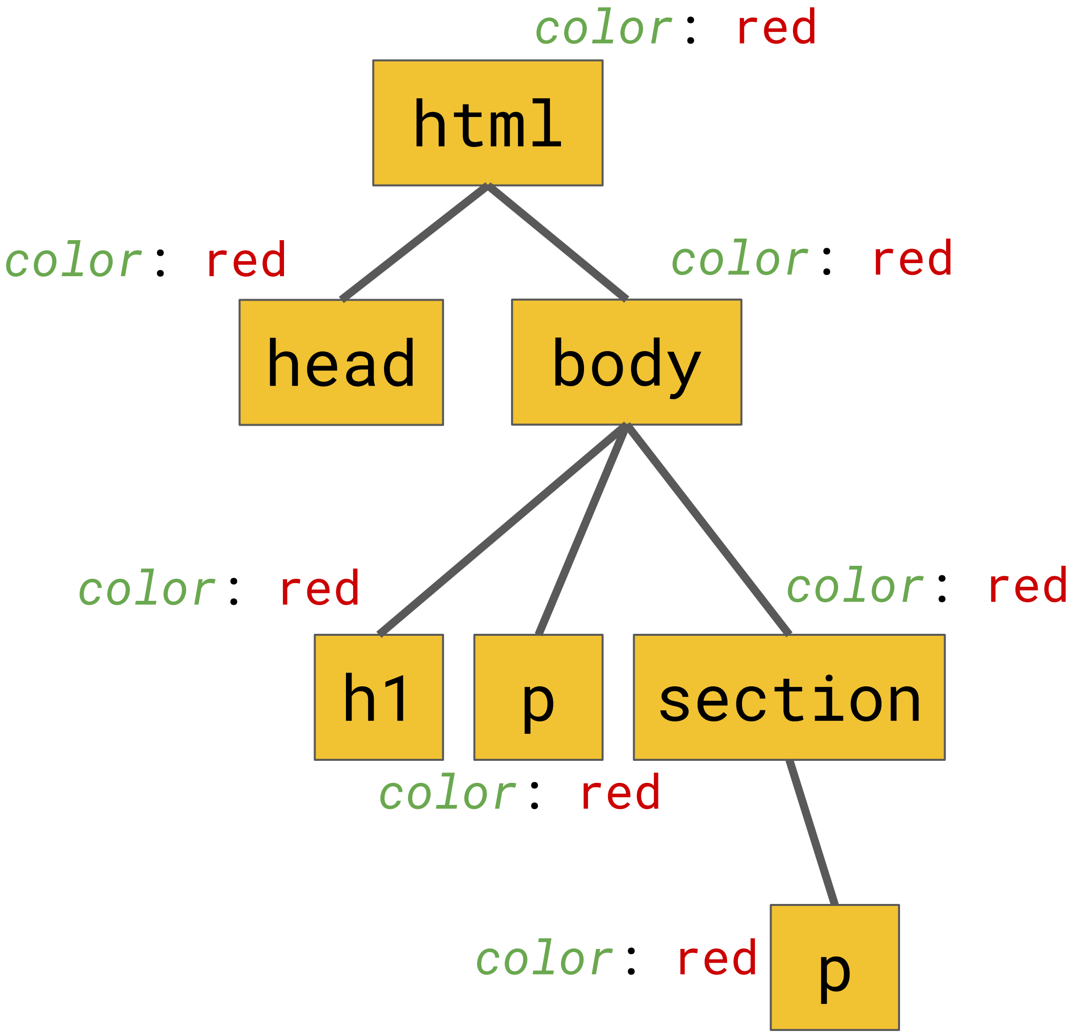
The selector selects a subset of the HTML tree’s nodes and apply the declaration to them

Declaration have graphical meaning that will be applied by the browser



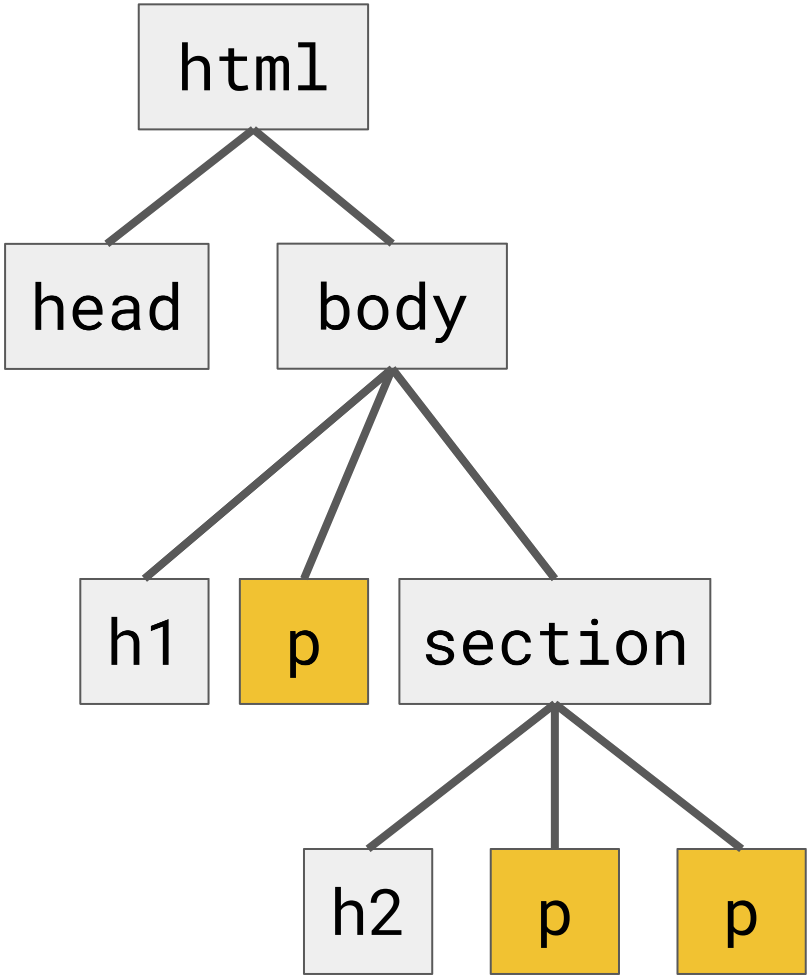
First example with the *joker* selector

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



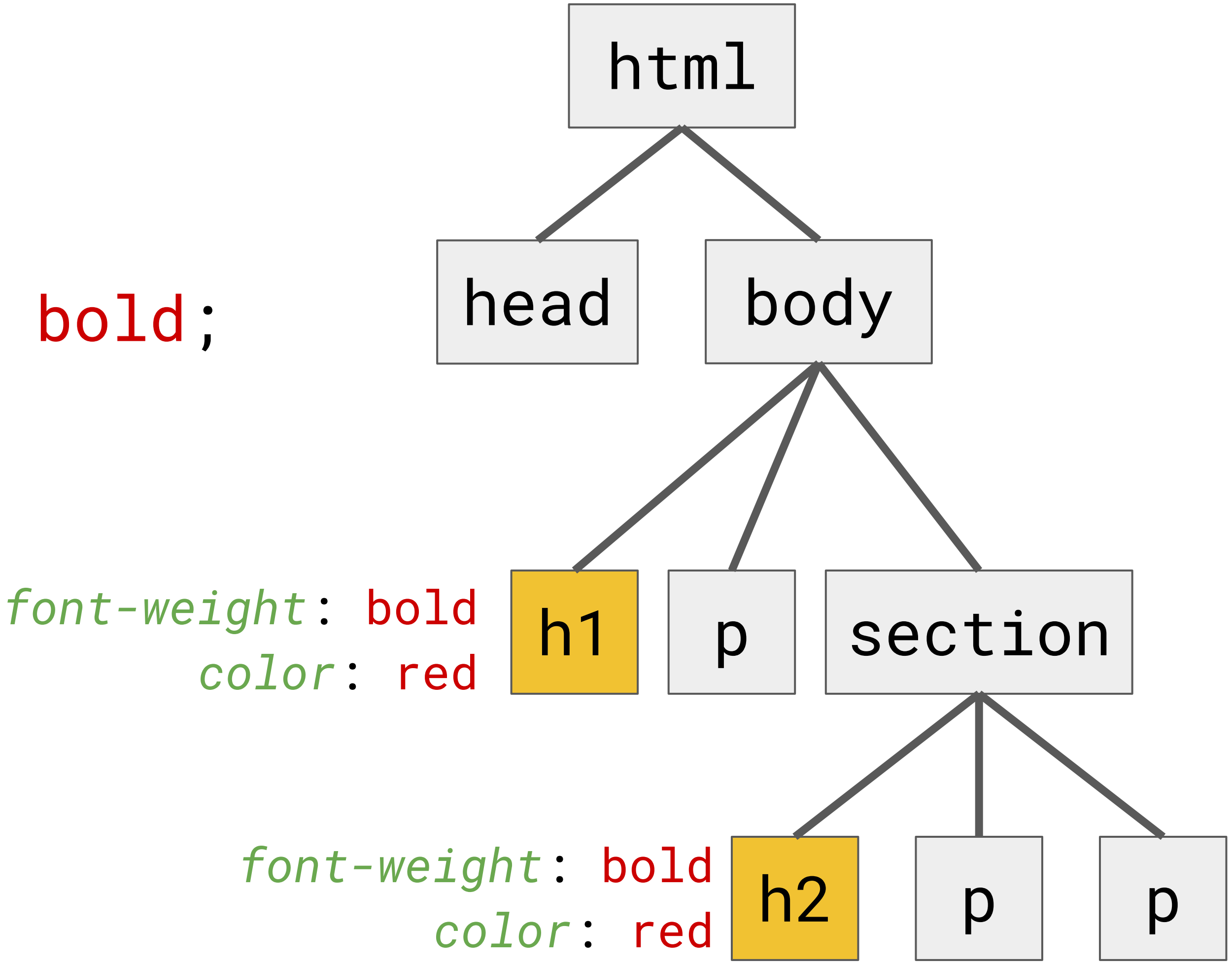
The *tag* selector

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



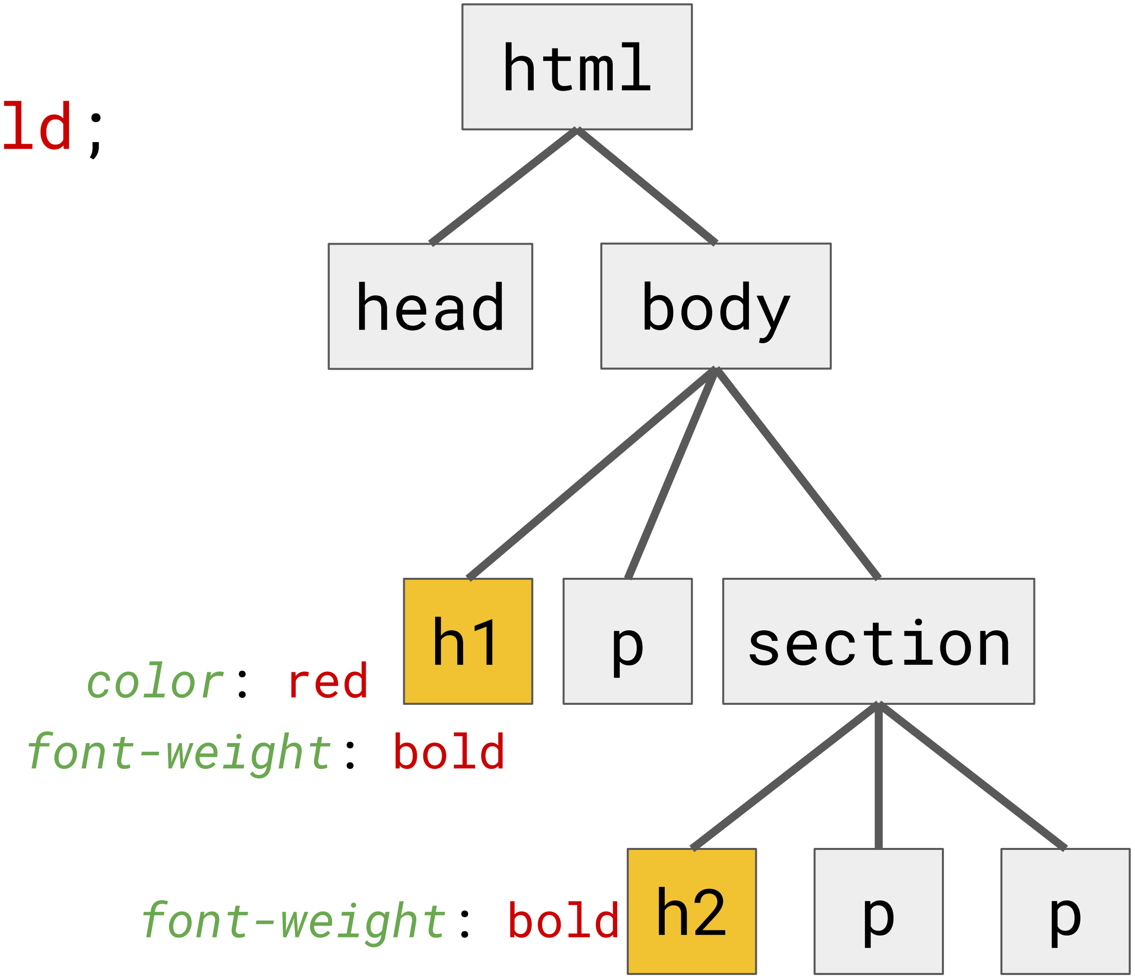
Selector union

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



Multiple rules

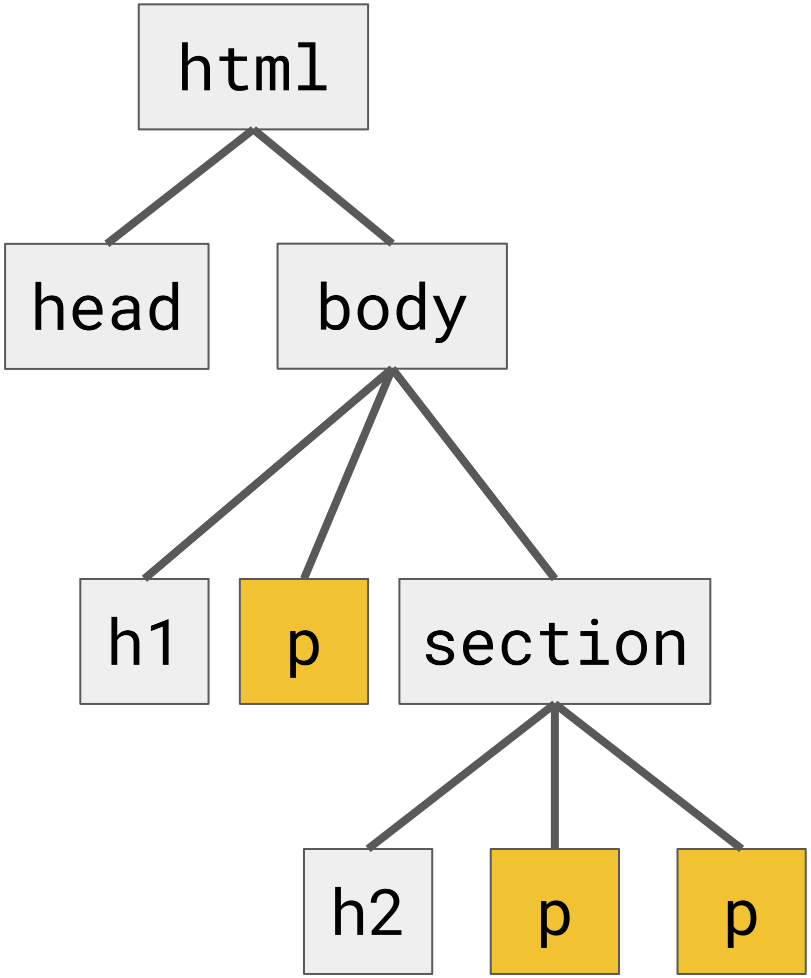
```
h1, h2 {
    font-weight: bold;
}

h1 {
    color: red;

}
```



The parent-child selectors

Selects all paragraphs that are descendants of a body

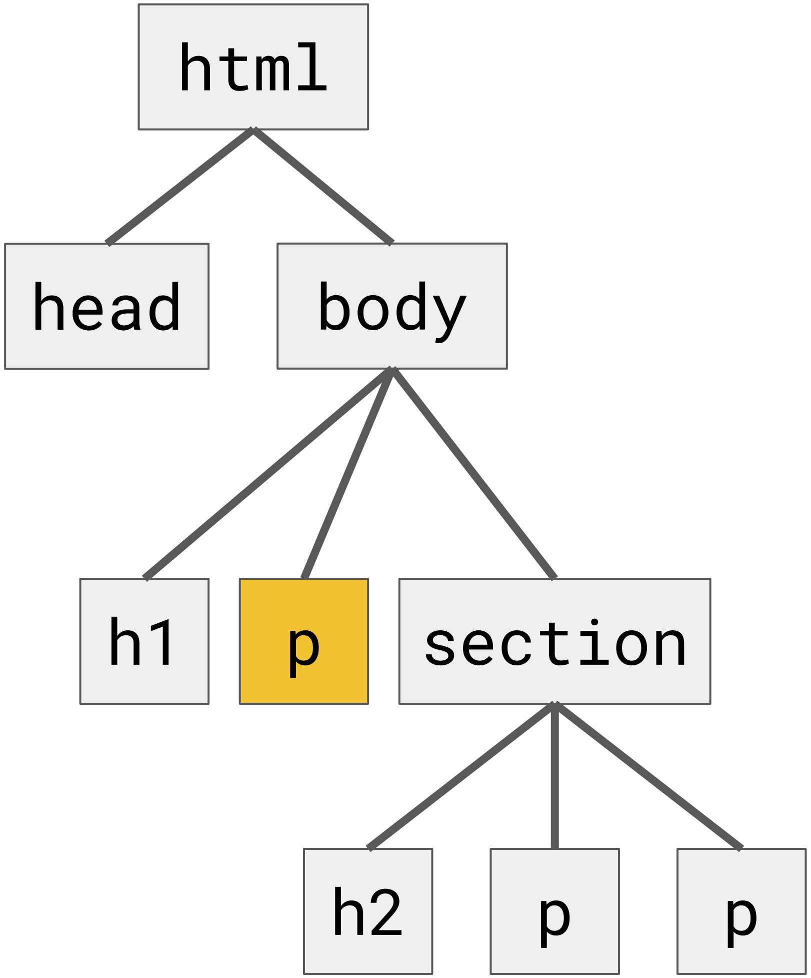
```
body p {
   color: red;
}
```



The parent-child selectors

Selects all paragraphs that are direct children of a body

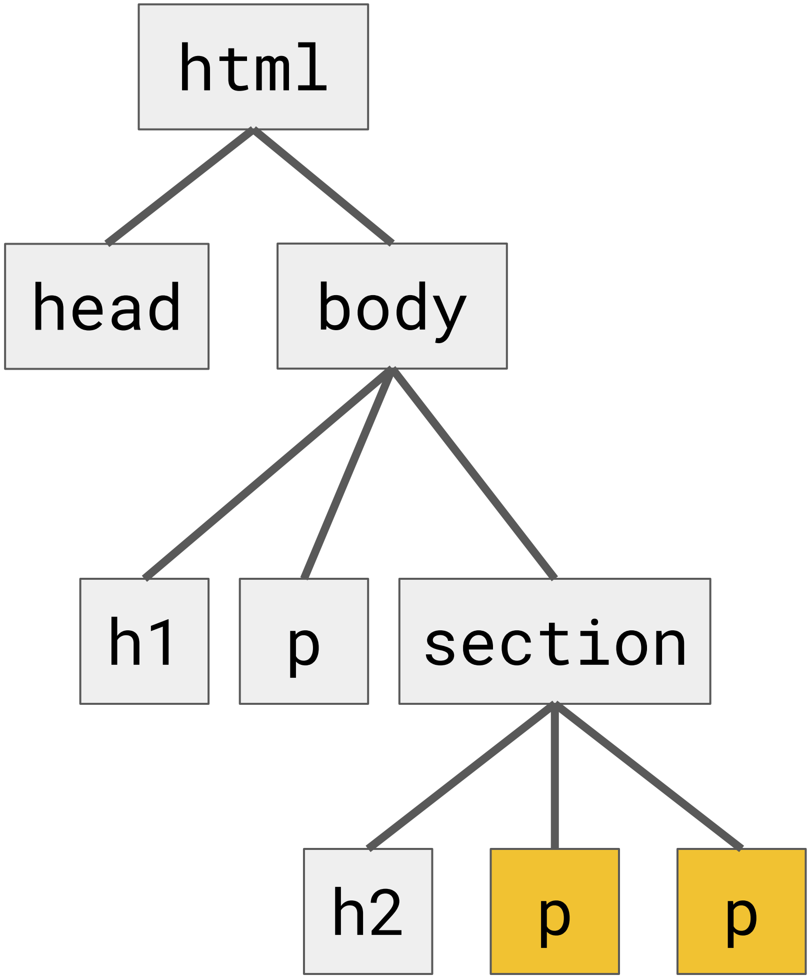
```
body > p {
    color: red;
}
```



The sibling selectors

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

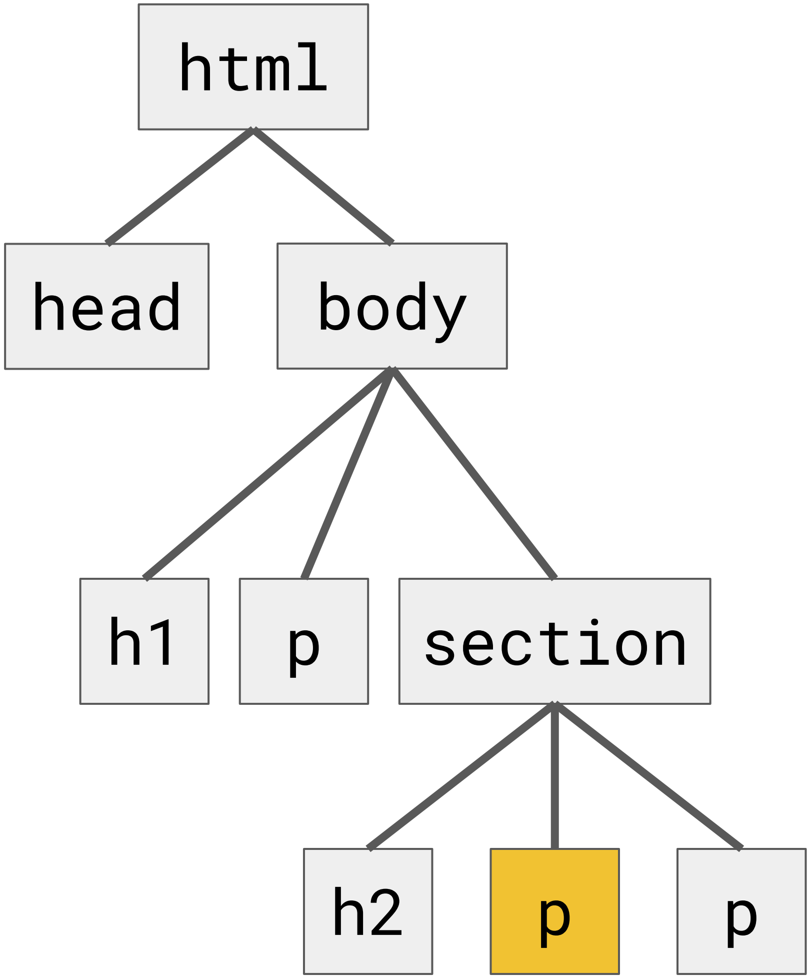
```
h2 ~ p {
   color: red;
}
```



The sibling selectors

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

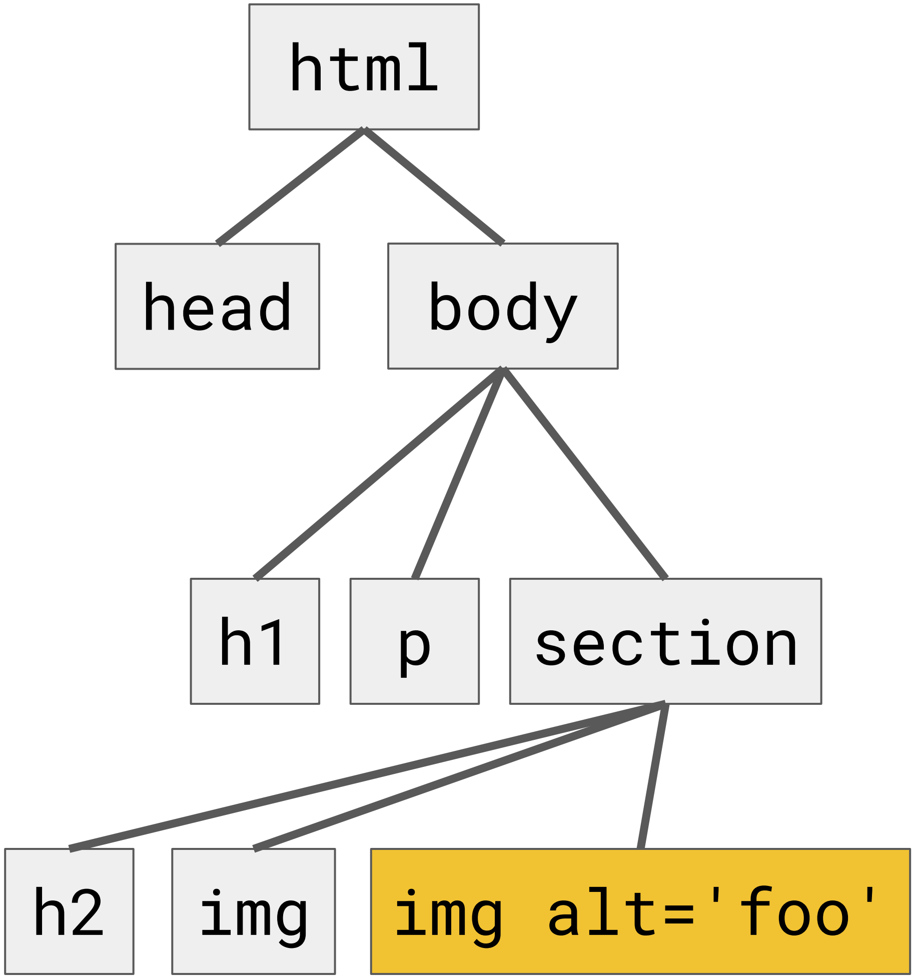
```
h2 + p {
    color: red;
}
```



Attribute-based selection

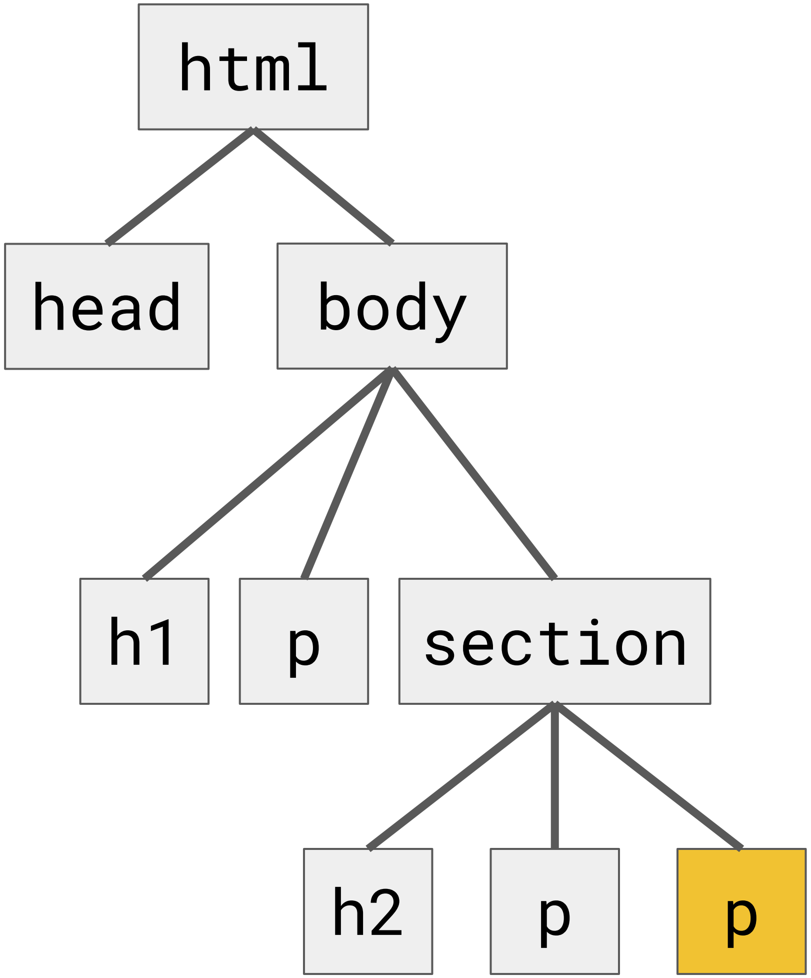
Selects all paragraphs that are direct children of a body

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



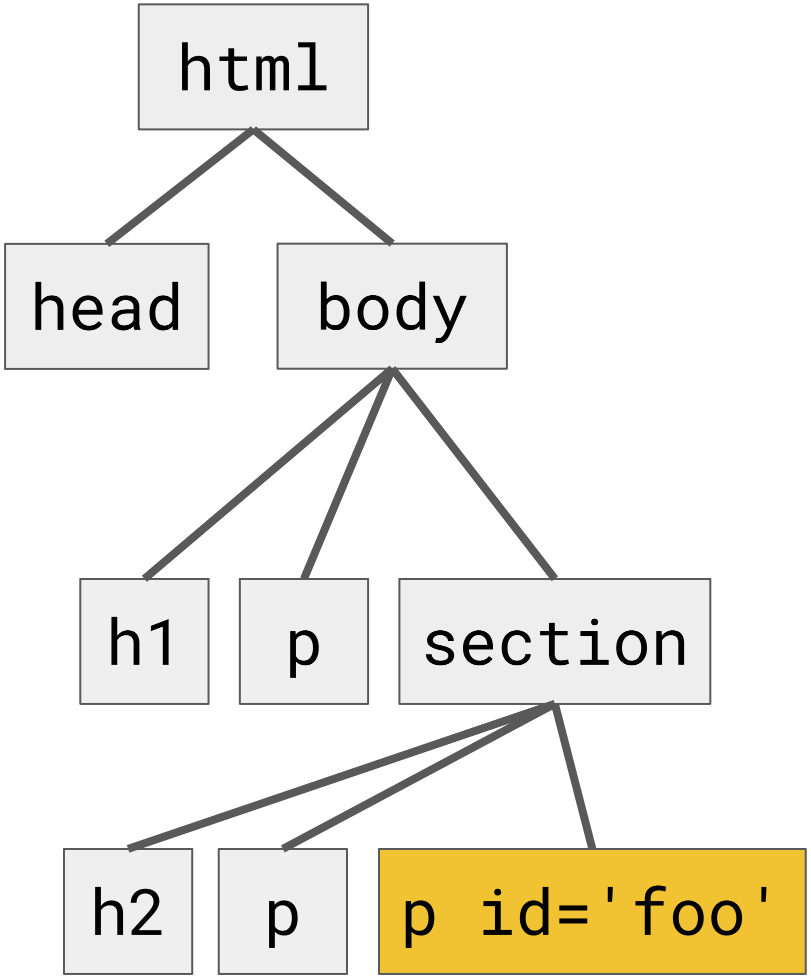
ID-based selection

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



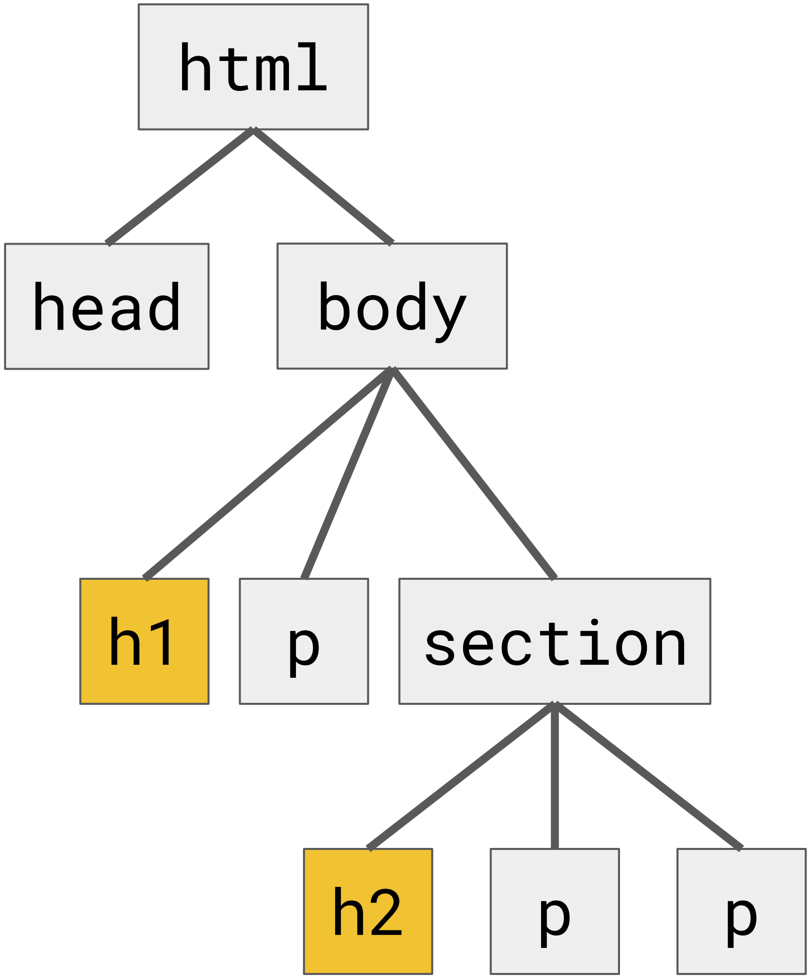
ID based selection

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



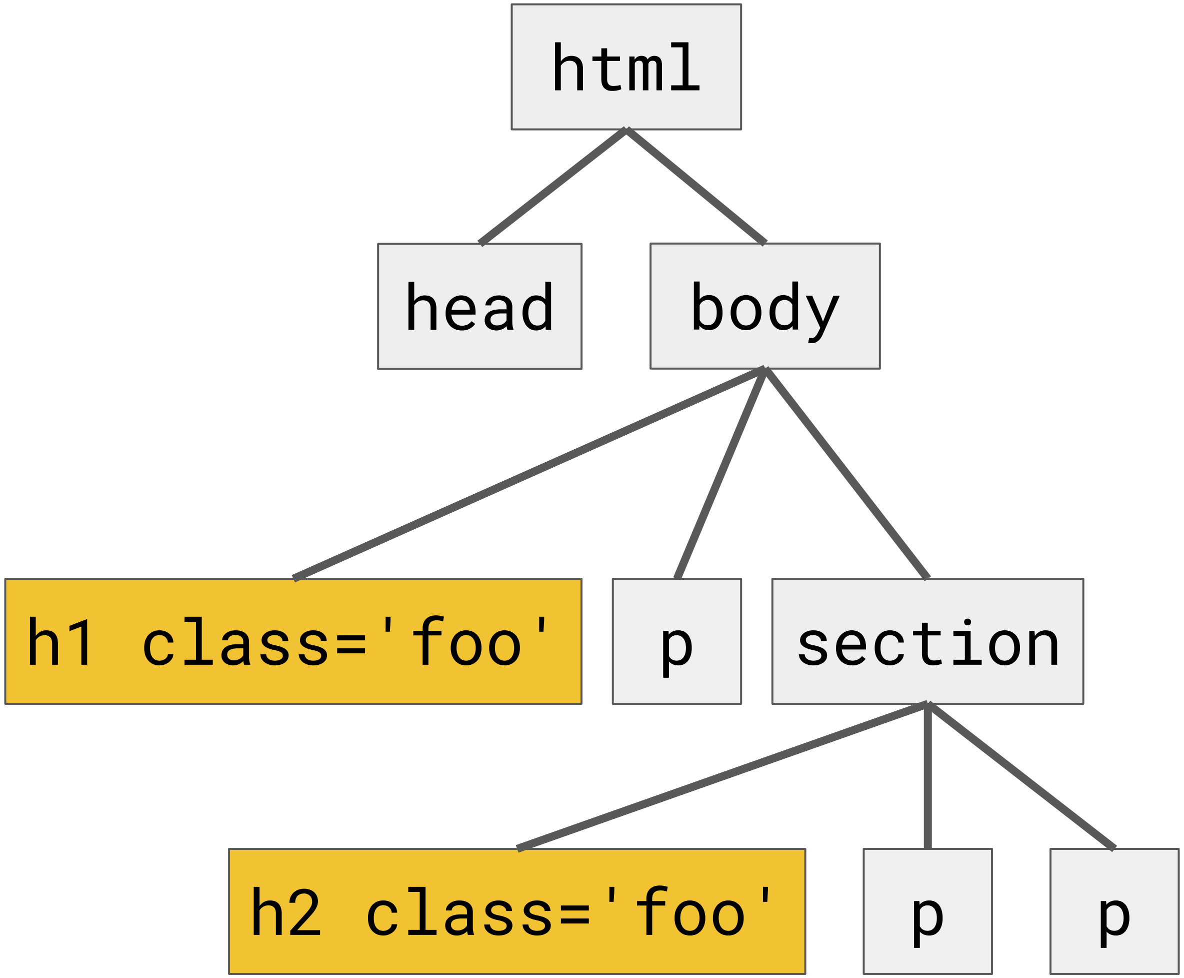
Class-based selection

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!



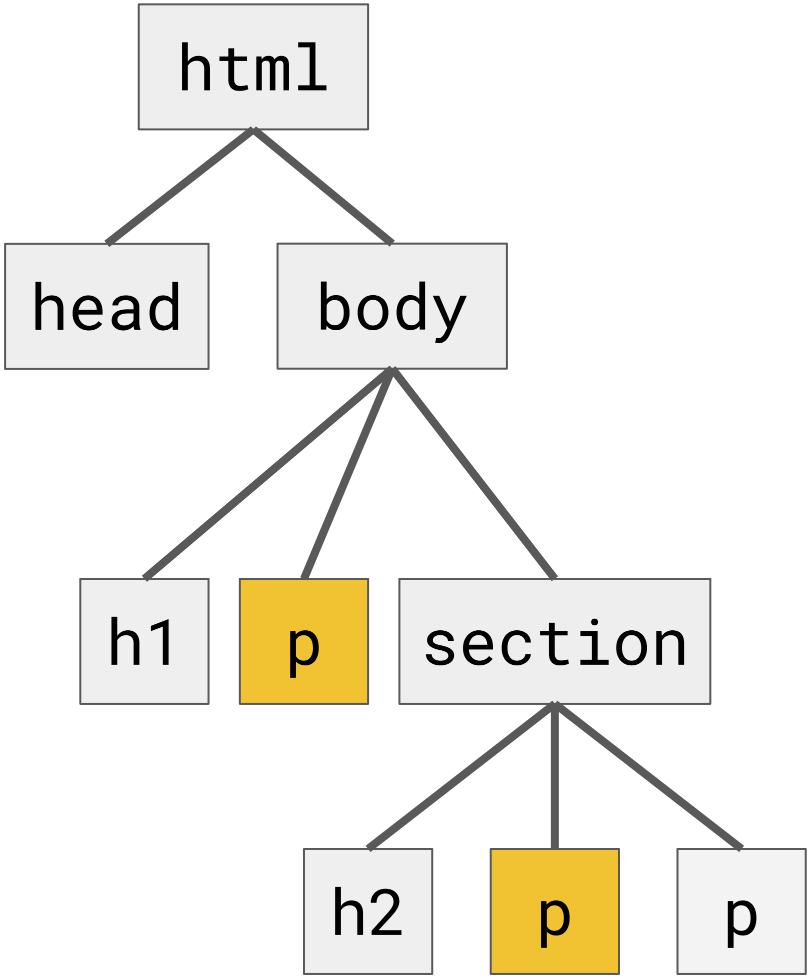
Class-based selection

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

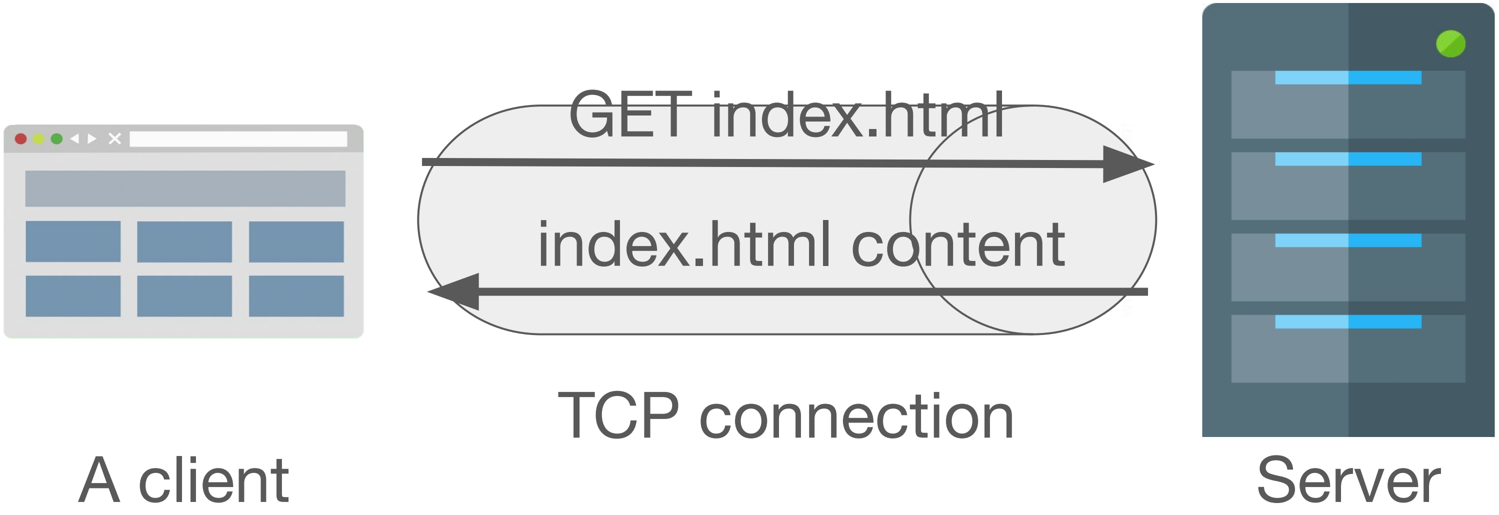


Pseudo class selection

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



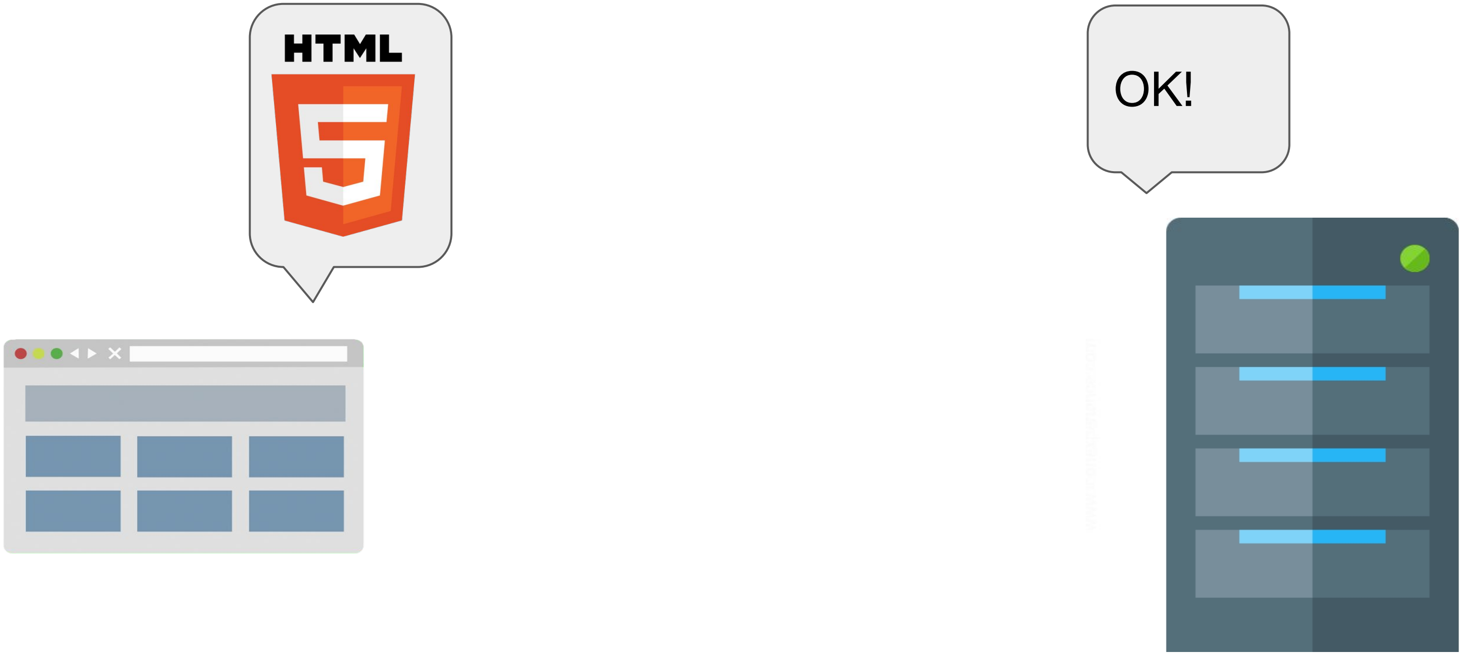
Neat! But how I give CSS to a HTML resource?



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>
```

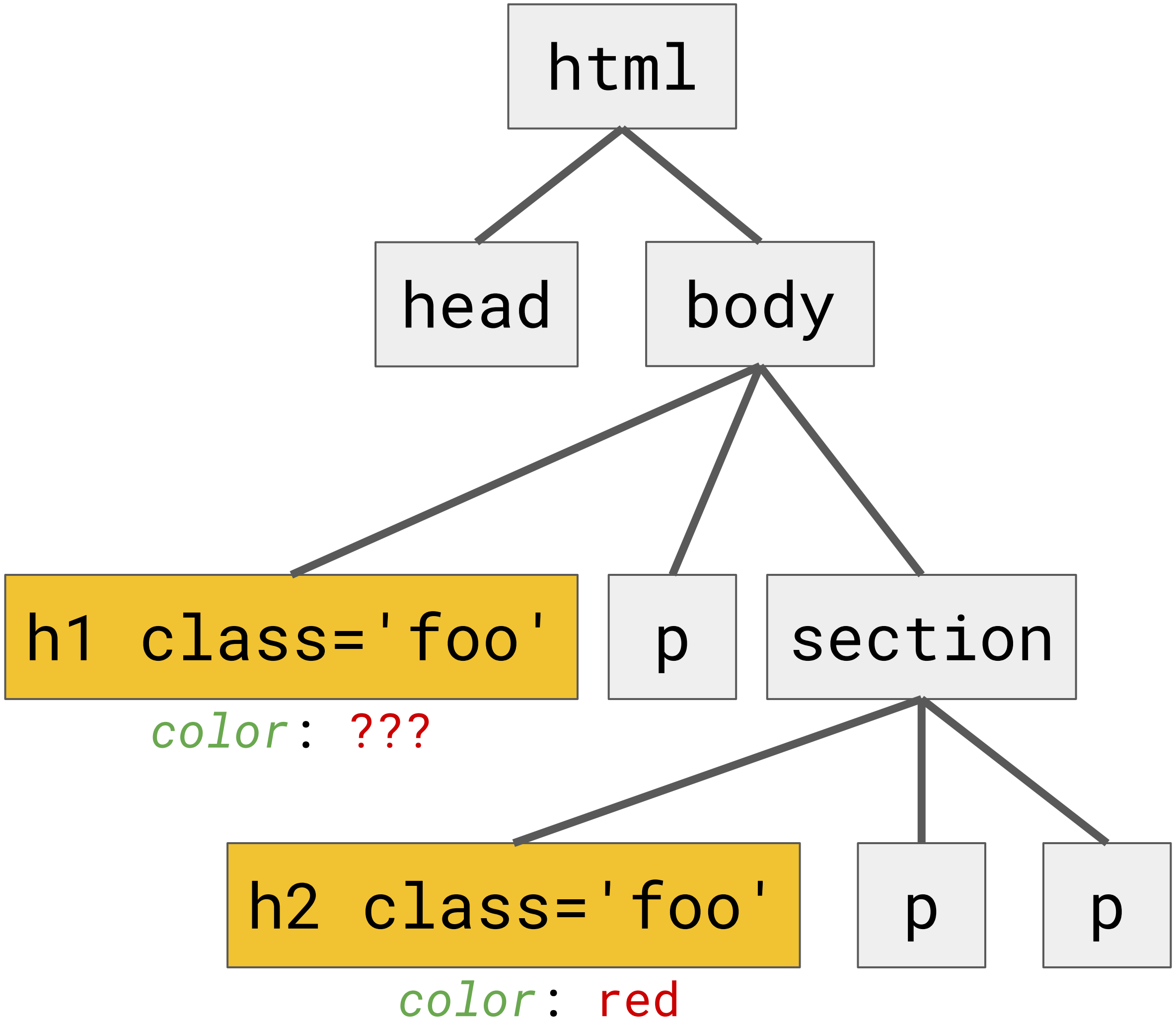
Back to CSS inclusion, the dirty way

```
<! 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?

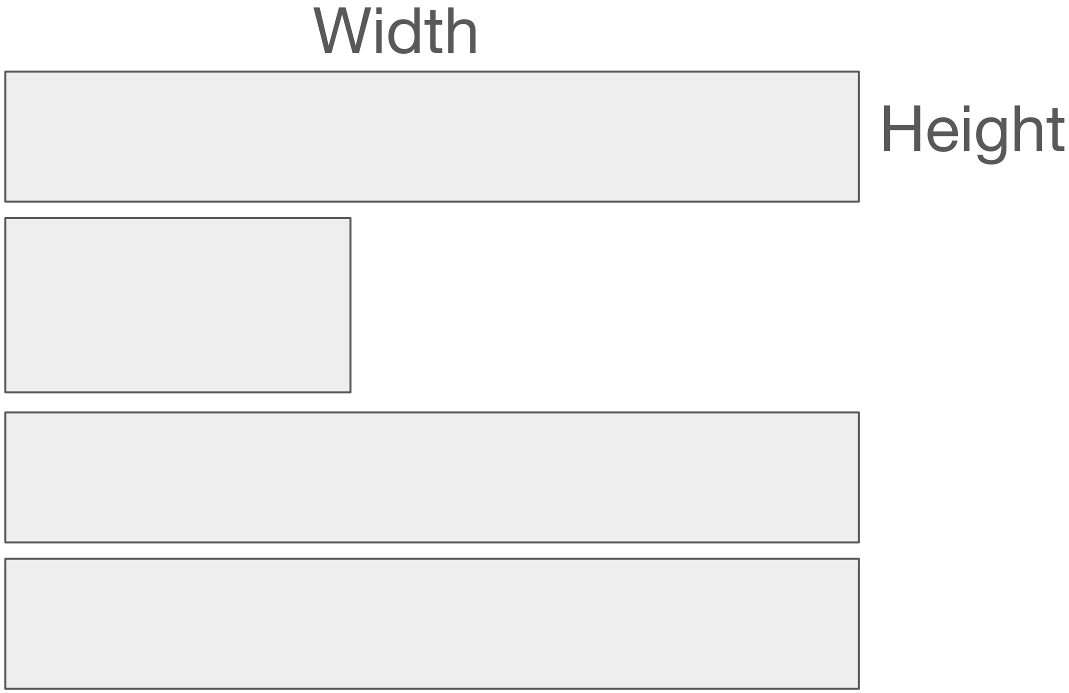
Block and inlines

Because

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

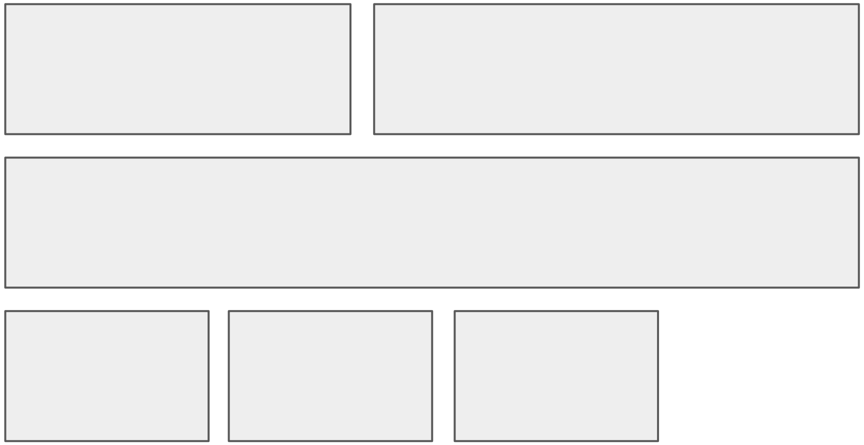
Block elements

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

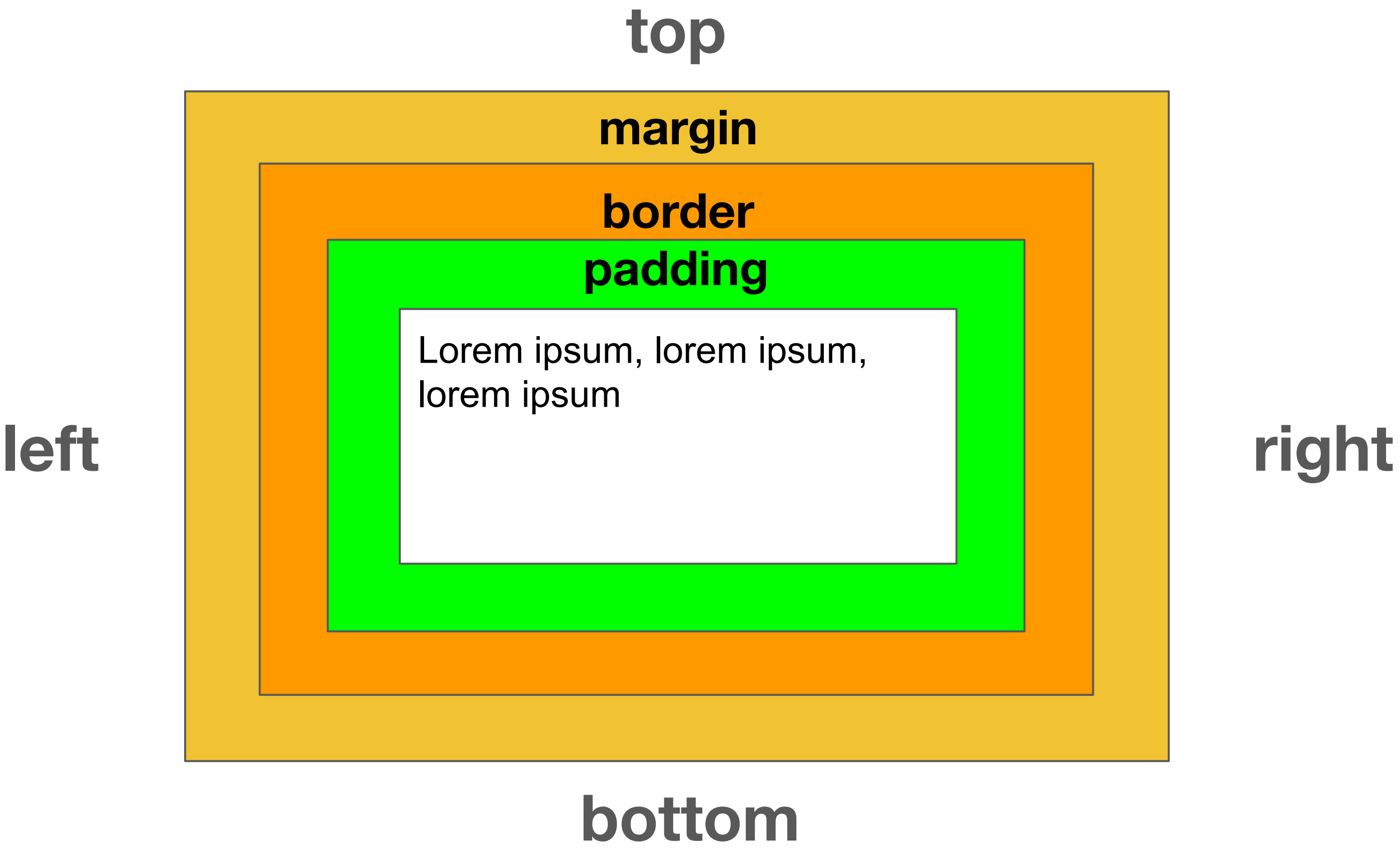


Inline elements

- 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

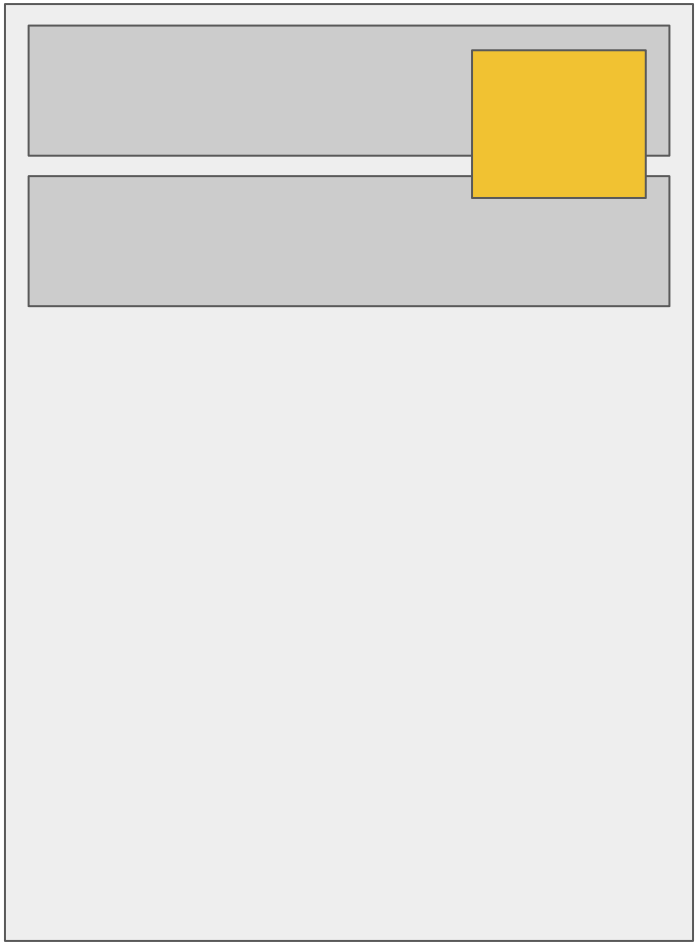
Margin (same for padding):

- `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?

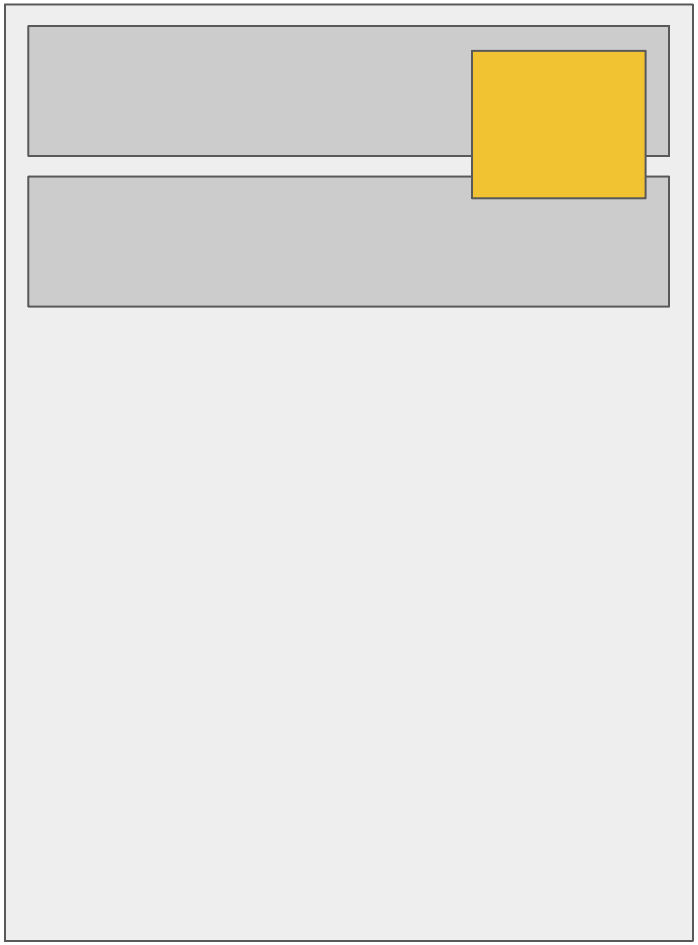


Positioned block

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

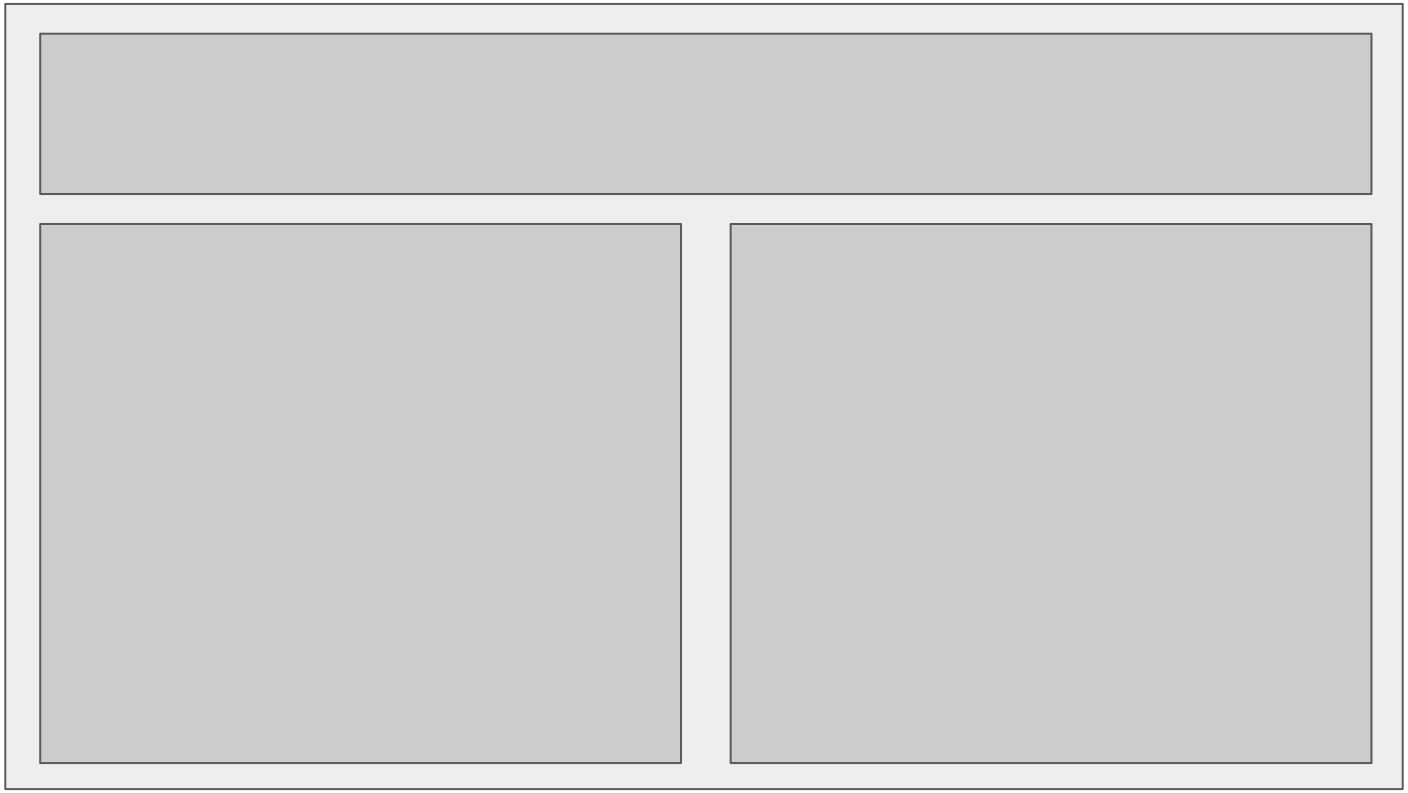
Example of a positioned block



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

Multi-column layouts

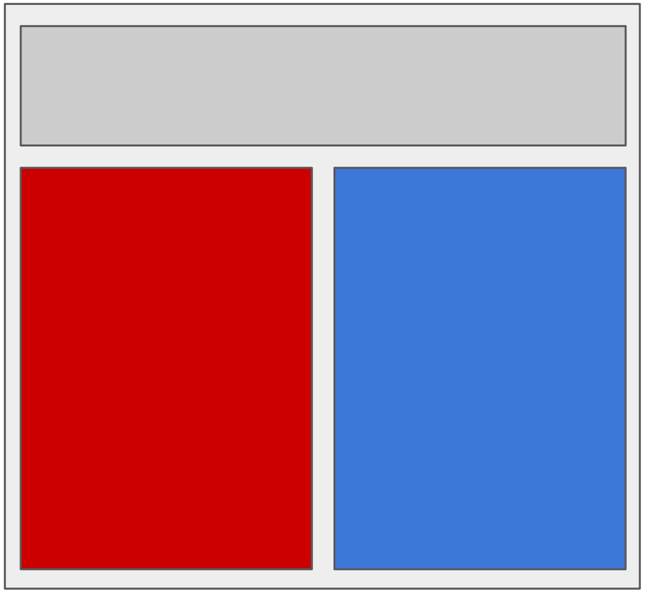
How the hell do I do this 🤔



Historial solution : inline-block

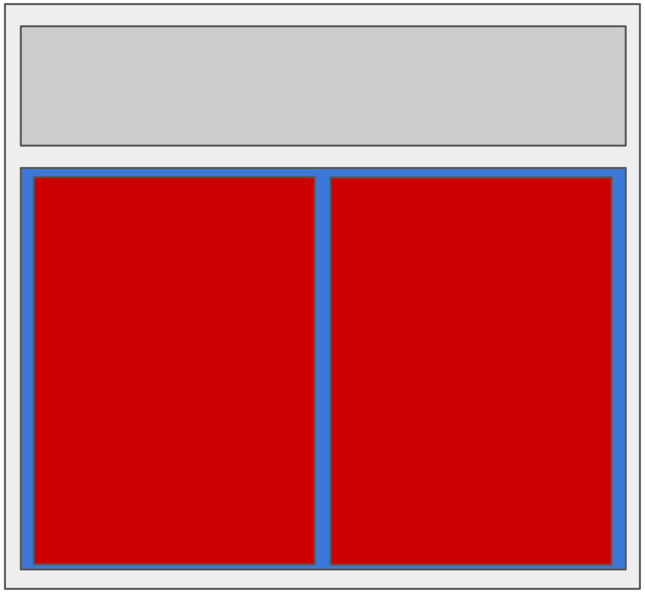
- Elements with `display: inline-block;` can go side by side (as inline ones)
- They can also have a custom size / position
- Best of both worlds

Example



```
#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**