

Web Programming

What is a (World Wide) Web

- Web — a collection of documents (*) with **links** between them
(*) but also Web applications...
- World Wide — over a global network (Internet)
- Scale: `https://www.netcraft.com/blog/july-2026-web-server-survey`
- Viewed mostly in: `https://en.wikipedia.org/wiki/Usage_share_of_web_browsers`
But: `https://github.com/lwthiker/curl-impersonate`

What is a (World Wide) Web

- Web — a collection of documents (*) with **links** between them
(*) but also Web applications...
- World Wide — over a global network (Internet)
- Scale: `https://www.netcraft.com/blog/july-2026-web-server-survey`
- Viewed mostly in: `https://en.wikipedia.org/wiki/Usage_share_of_web_browsers`
But: `https://github.com/lwthiker/curl-impersonate`

What is a (World Wide) Web

- Web — a collection of documents (*) with **links** between them
(*) but also Web applications...
- World Wide — over a global network (Internet)
- Scale: `https://www.netcraft.com/blog/july-2026-web-server-survey`
- Viewed mostly in: `https://en.wikipedia.org/wiki/Usage_share_of_web_browsers`
But: `https://github.com/lwthiker/curl-impersonate`

What is the World Wide Web

- HTML documents
- over HTTP

HTML — HyperText Markup language

HTTP — HyperText Transport Protocol

HyperText: text, augmented with links to other documents

Maximally «hypertext» — Wikipedia (+ other wiki sites)

What is the World Wide Web

- HTML documents
- over HTTP

HTML — HyperText Markup language

HTTP — HyperText Transport Protocol

HyperText: text, augmented with links to other documents

Maximally «hypertext» — Wikipedia (+ other wiki sites)

What is the World Wide Web

- HTML documents
- over HTTP

HTML — HyperText Markup language

HTTP — HyperText Transport Protocol

HyperText: text, augmented with links to other documents

Maximally «hypertext» — Wikipedia (+ other wiki sites)

What is the World Wide Web

- HTML documents
- over HTTP

HTML — HyperText Markup language

HTTP — HyperText Transport Protocol

HyperText: text, augmented with links to other documents

Maximally «hypertext» — Wikipedia (+ other wiki sites)

What is the World Wide Web

- HTML documents
- over HTTP

HTML — HyperText Markup language

HTTP — HyperText Transport Protocol

HyperText: text, augmented with links to other documents

Maximally «hypertext» — Wikipedia (+ other wiki sites)

Behaviours

- Document — I read it
 - Form — I fill and submit it (maybe with a document back)
 - Application — I do something not close to above
-
- Course description — a document
 - Password change — a form
 - Site search — a form
 - OpenStreetMap — an application

Documents and forms work with HTML (+ CSS)

... can be improved — or made worse — with scripting

Web applications can't avoid JS

Behaviours

- Document — I read it
 - Form — I fill and submit it (maybe with a document back)
 - Application — I do something not close to above
-
- Course description — a document
 - Password change — a form
 - Site search — a form
 - OpenStreetMap — an application

Documents and forms work with HTML (+ CSS)

... can be improved — or made worse — with scripting

Web applications can't avoid JS

Behaviours

- Document — I read it
 - Form — I fill and submit it (maybe with a document back)
 - Application — I do something not close to above
-
- Course description — a document
 - Password change — a form
 - Site search — a form
 - OpenStreetMap — an application

Documents and forms work with HTML (+ CSS)

... can be improved — or made worse — with scripting

Web applications can't avoid JS

Background

- 1774 Electric telegraph prototype
- 1838 Practical telegraph deployment
- 1858 Operational transatlantic cable
- 1968 ARPANET multi-site deployment
- 1973 IP and IP addresses
- 1984 DNS
- 1986 NSFNET with TCP/IP, universities move there
- 1989-03-12 Tim Berners-Lee (CERN) creates Web

Background

- 1774 Electric telegraph prototype
- 1838 Practical telegraph deployment
- 1858 Operational transatlantic cable
- 1968 ARPANET multi-site deployment
- 1973 IP and IP addresses
- 1984 DNS
- 1986 NSFNET with TCP/IP, universities move there
- 1989-03-12 Tim Berners-Lee (CERN) creates Web

Background

- 1774 Electric telegraph prototype
- 1838 Practical telegraph deployment
- 1858 Operational transatlantic cable
- 1968 ARPANET multi-site deployment
- 1973 IP and IP addresses
- 1984 DNS
- 1986 NSFNET with TCP/IP, universities move there
- 1989-03-12 Tim Berners-Lee (CERN) creates Web

Early Web

- 1989-03-12 Tim Berners-Lee (CERN) creates Web
- 1989 Commercial connection to NSFNET allowed
- 1990
 - <http://info.cern.ch/hypertext/WWW/TheProject.html>
 - 1.5 Mbit/s transatlantic computer connection (CERN – U.Cornell)
 - Archie — first Internet search engine
- 1991 First web server in USA
- 1993
 - Mosaic — first graphical browser
 - CERN puts basic Web software into public domain

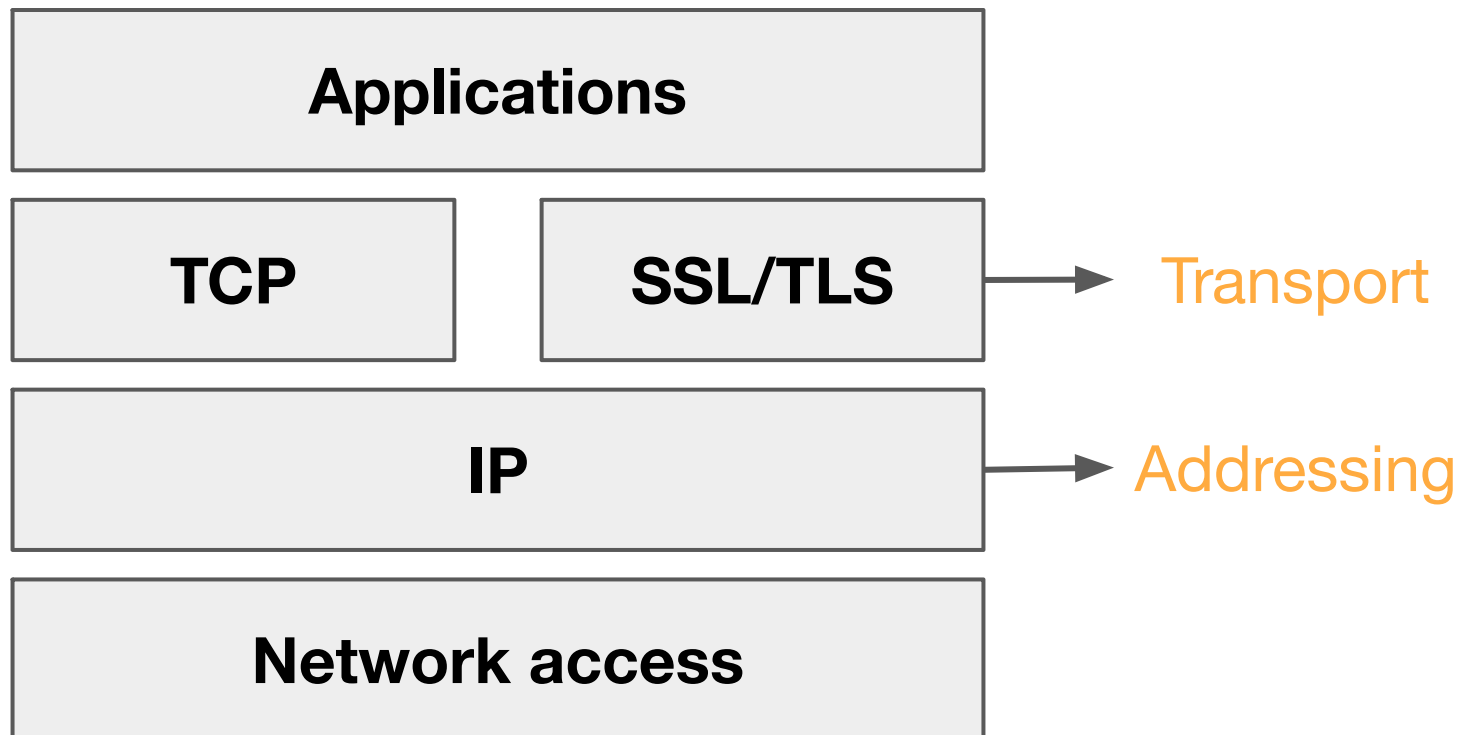
History of many websites — on Internet Archive

<https://archive.org/>

Introduction

What is internet?

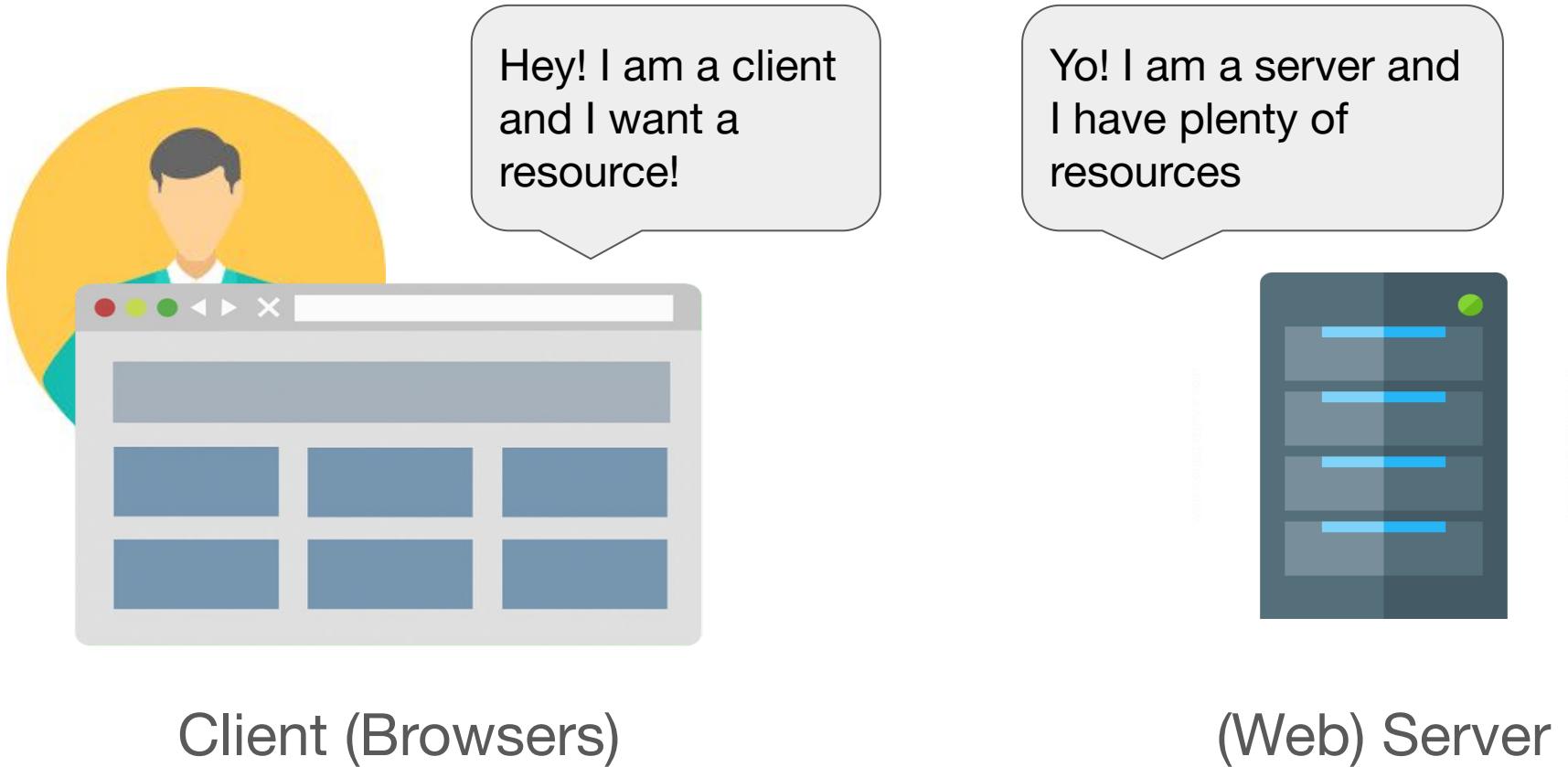
Wikipedia : a global system of interconnected computer networks



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



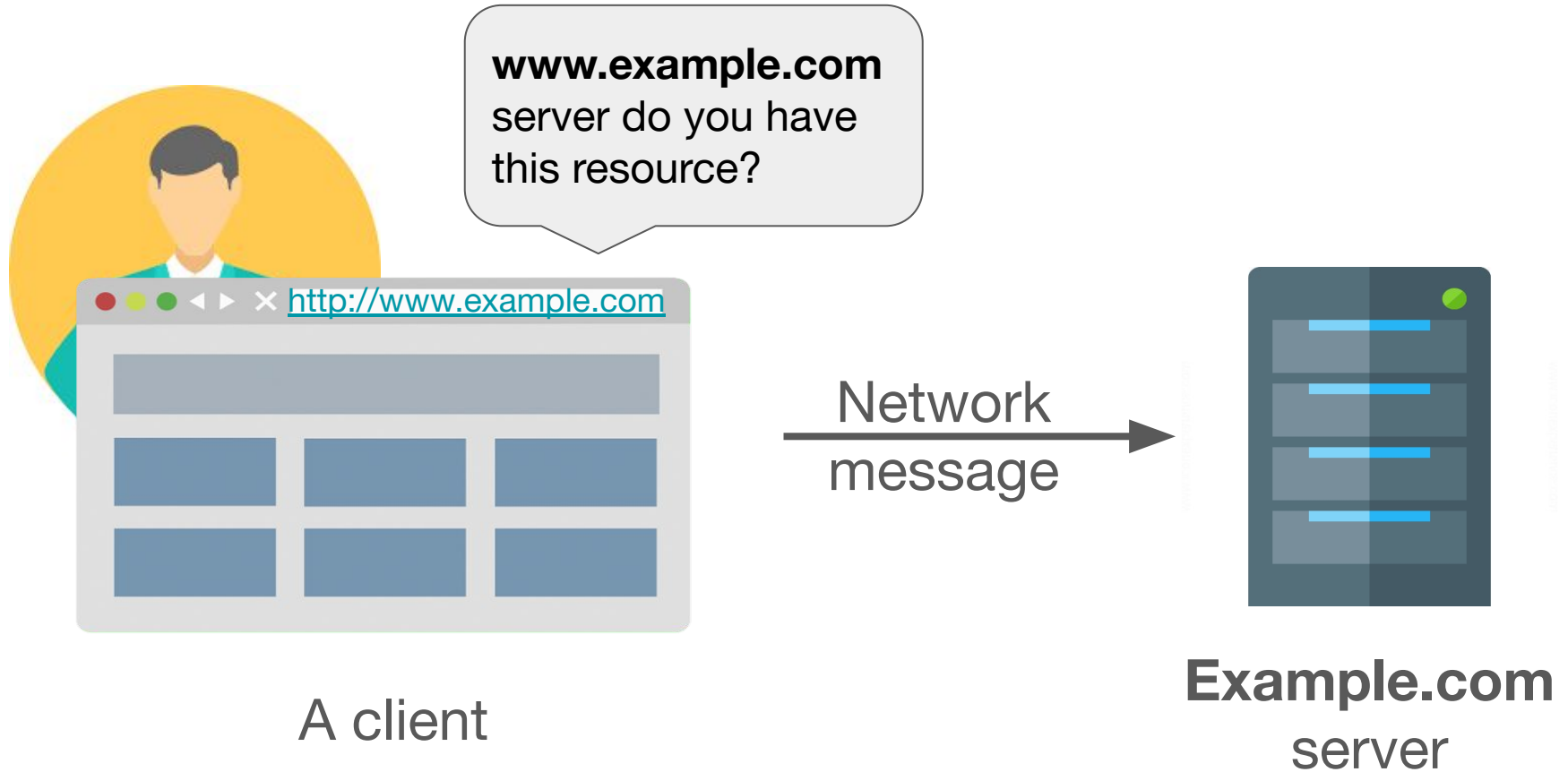
A web application in a nutshell

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

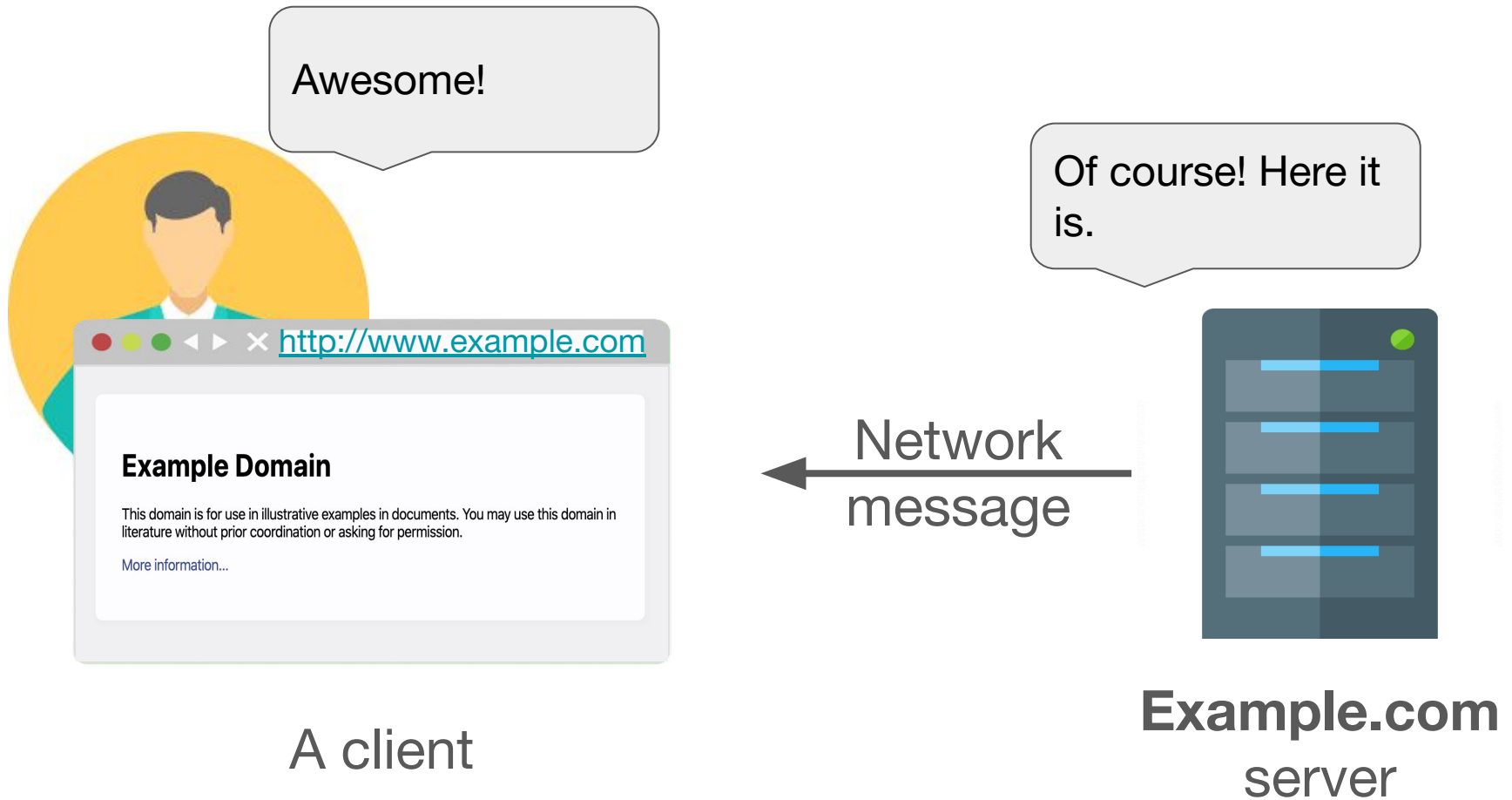


A client

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?**

Telnet client for raw TCP connections

```
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
```

Client's request

```
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
```

Headers

```
<!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;
```

Server's response

Data

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
```

TA000
00N0r i] 0S0/ @0001k00Z00\$06000q:0000+00 010I0I00s0PzUe000Bf
0'00+0'000+00F 0I4h00^@^
<0|3uP000P0 06000\$}001)j_,04yU0rQazw0r000t
.0s00080
z0_00020Me1
Ü50000%0t
뎡 R000t3
00.0|0Q00]000
V-z0|0r3*000rKp05th00'0000NH0000w0000y;0ks00000w00\$00K060BR0b0C00PqE00 K0<0 0600
70000E(17Vx20JS00
% x00)0d0000e000&04/醢000-000i0s0K0dw0700#0u0'00y\\$]j<0L0r0'0I0vg?Kr {=00?E00^x;05X
0\$060e000040n08000五+c000E0hA00K0000000L00Rit0f4\o0000

Example Domain

Non sécurisé | www.example.com

devtravauxrechercheenseignementloisirsadmin

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

ElementsConsoleNetwork

Filter

AllXHRJSCSSImgMediaFontDocWSManifestOther

Name	Headers	Preview	Response
www.example.com	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 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 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		

1 requests | 935 B transf...

Text encoding

We can transmit: bytes

We want to convey: text

What bytes mean?

Text encoding: character table

ASCII (1977/1986)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL



Changed or added in 1963 version



Changed in both 1963 version and 1965 draft

Text encoding: big character table

- ASCII not enough for French
- Of course there is a standard encoding for French — even more than one
- Latin-1 and Windows-1252 very similar but not same
- Let's have one big standard so no one dares to create another one

Unicode

Encode *a//* the text!

Unicode planes, and code point ranges used										[hide]
Basic		Supplementary								
Plane 0		Plane 1		Plane 2		Plane 3	Planes 4–13	Plane 14	Planes 15–16	
0000–FFFF		10000–1FFFF		20000–2FFFF		30000–3FFFF	40000–DFFFF	E0000–EFFFF	F0000–10FFFF	
Basic Multilingual Plane		Supplementary Multilingual Plane		Supplementary Ideographic Plane		Tertiary Ideographic Plane	unassigned	Supplementary Special-purpose Plane	Supplementary Private Use Area planes	
BMP		SMP		SIP		TIP	—	SSP	SPUA-A/B	
0000–0FFF	8000–8FFF	10000–10FFF	18000–18FFF	20000–20FFF	28000–28FFF			E0000–E0FFF	15: SPUA-A F0000–FFFFF 16: SPUA-B 100000–10FFFF	
1000–1FFF	9000–9FFF	11000–11FFF		21000–21FFF	29000–29FFF					
2000–2FFF	A000–AFFF	12000–12FFF	1A000–1AFFF	22000–22FFF	2A000–2AFFF					
3000–3FFF	B000–BFFF	13000–13FFF	1B000–1BFFF	23000–23FFF	2B000–2BFFF					
4000–4FFF	C000–CFFF	14000–14FFF	1C000–1CFFF	24000–24FFF	2C000–2CFFF					
5000–5FFF	D000–DFFF		1D000–1DFFF	25000–25FFF	2D000–2DFFF					
6000–6FFF	E000–EFFF	16000–16FFF	1E000–1EFFF	26000–26FFF	2E000–2EFFF					
7000–7FFF	F000–FFFF	17000–17FFF	1F000–1FFFF	27000–27FFF	2F000–2FFFF					

(but at what cost)

Assigned characters Unicode version 17.0

Plane	Allocated code points ^[a]	Assigned characters
0 BMP	65,520	55,718
1 SMP	31,872	28,869
2 SIP	61,536	61,513
3 TIP	13,440	13,429
14 SSP	368	337
15 SPUA-A	65,536	0 (by definition)
16 SPUA-B	65,536	0 (by definition)
Totals	303,808	159,866

A codepoint for every character? Not really.

🇫🇷 / 🇫🇷 — defined in Unicode, but as ligature of two codepoints

Still too many codepoints

Variable length encoding

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0yyyzzzz			
U+0080	U+07FF	110xxxxyy	10yyzzzz		
U+0800	U+FFFF	1110www	10xxxxyy	10yyzzzz	
U+010000	U+10FFFF	11110uvv	10vvwww	10xxxxyy	10yyzzzz

English: 1 byte per letter

French: 1–2 bytes per letter

ASCII, Latin-1, UTF-8 **all agree** on English text

All markup parts of HTML are safe from misinterpretation.

Variable length encoding

Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0yyyzzzz			
U+0080	U+07FF	110xxxxyy	10yyzzzz		
U+0800	U+FFFF	1110www	10xxxxyy	10yyzzzz	
U+010000	U+10FFFF	11110uvv	10vvwww	10xxxxyy	10yyzzzz

English: 1 byte per letter

French: 1-2 bytes per letter

ASCII, Latin-1, UTF-8 **all agree** on English text

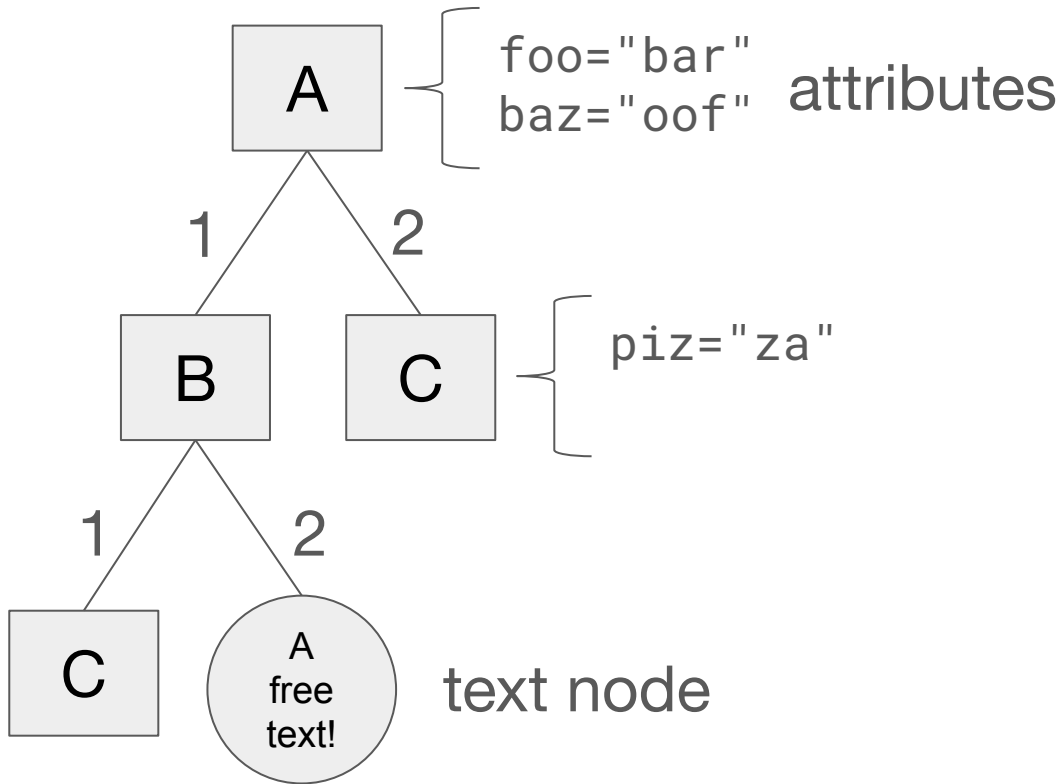
All markup parts of HTML are safe from misinterpretation.

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?

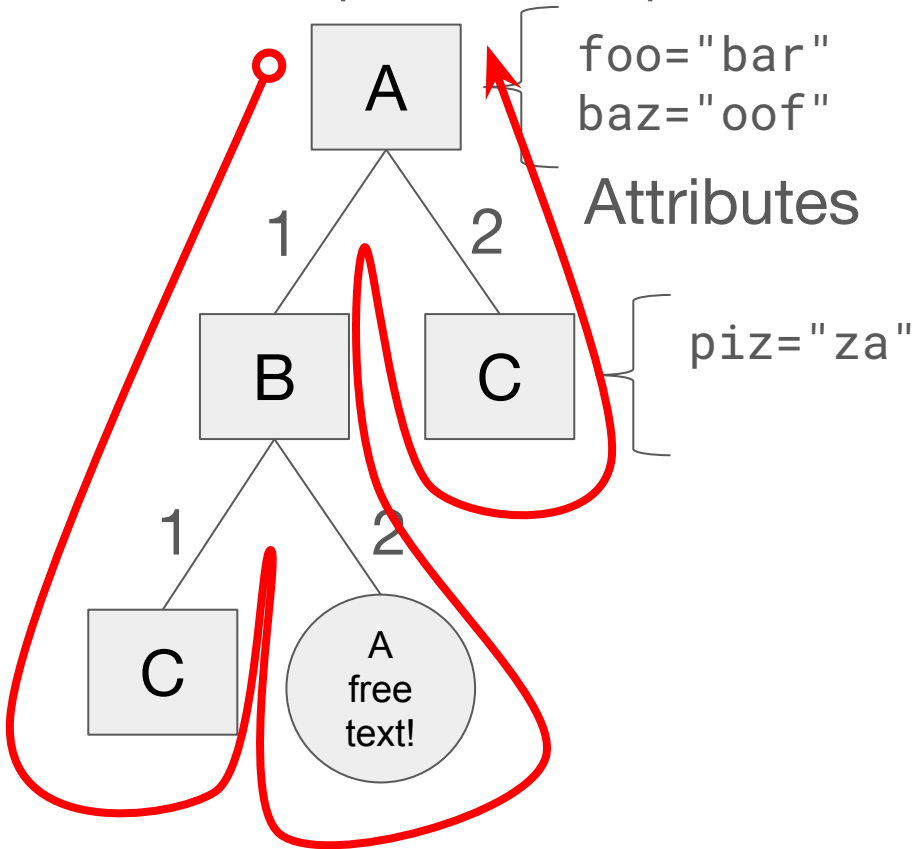
A sample XML tree

node (or element)



XML tree traversal

A node (or element)

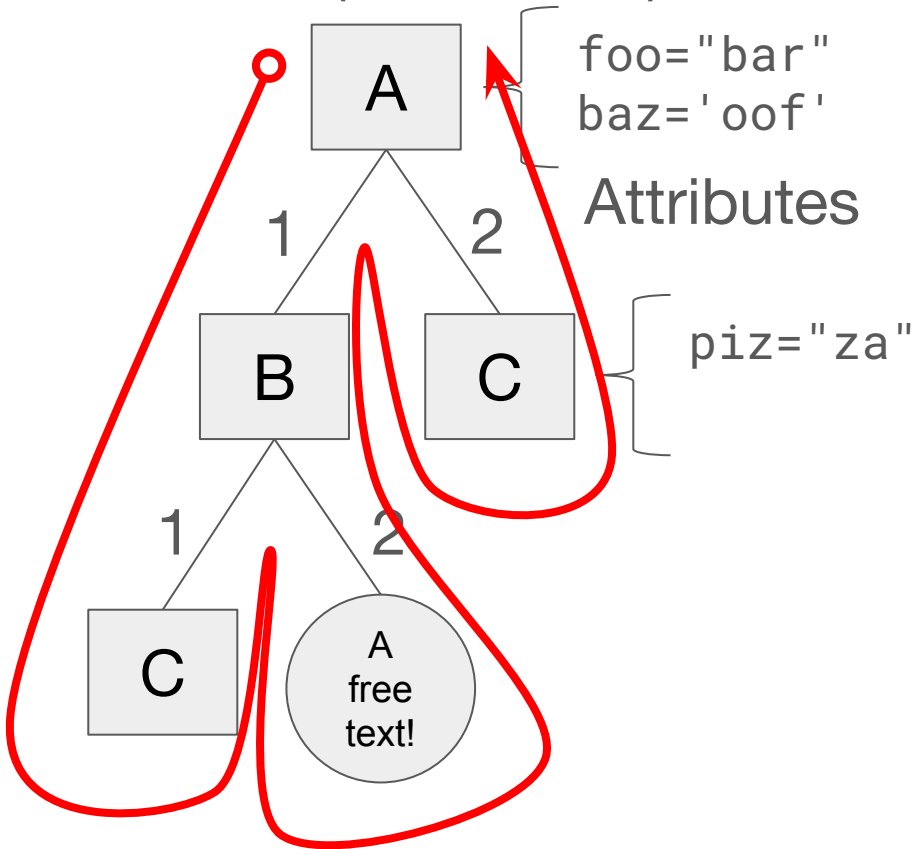


Rules :

- 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)



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. `<br>`) ... `<div><span></div>`
- 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

Go into <head>

Body Tags

Sectioning
Tags

Flow
Tags

Phrasing
Tags

Binary
Tags

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

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

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?

Blog

Accueil

Catégorie 1

Catégorie 2

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):

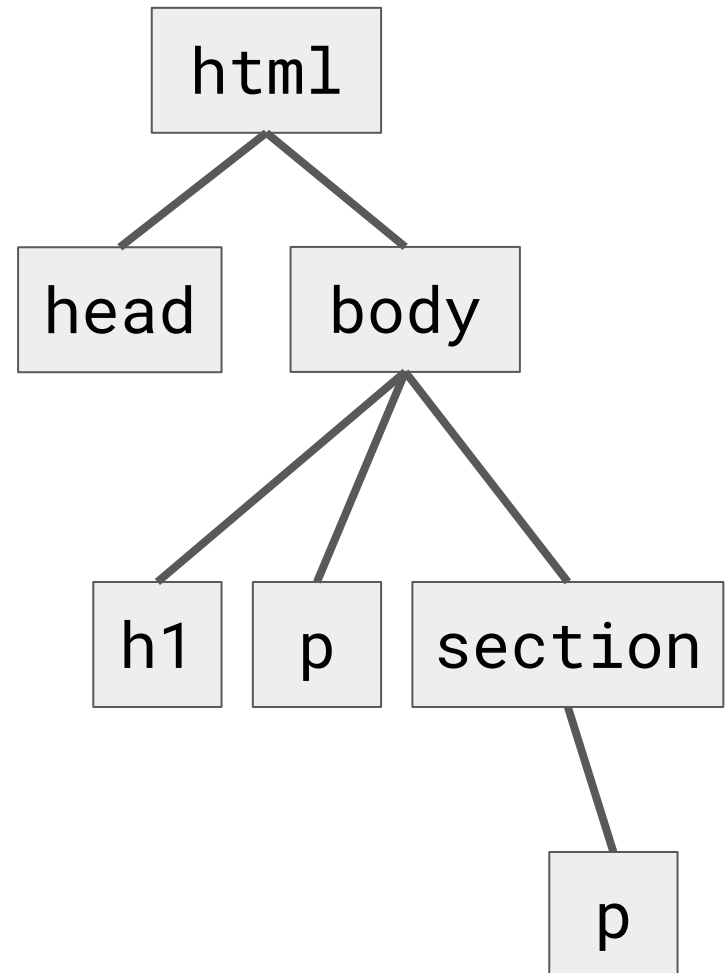
```
selector {  
    property: value;  
}
```

How does that works?

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

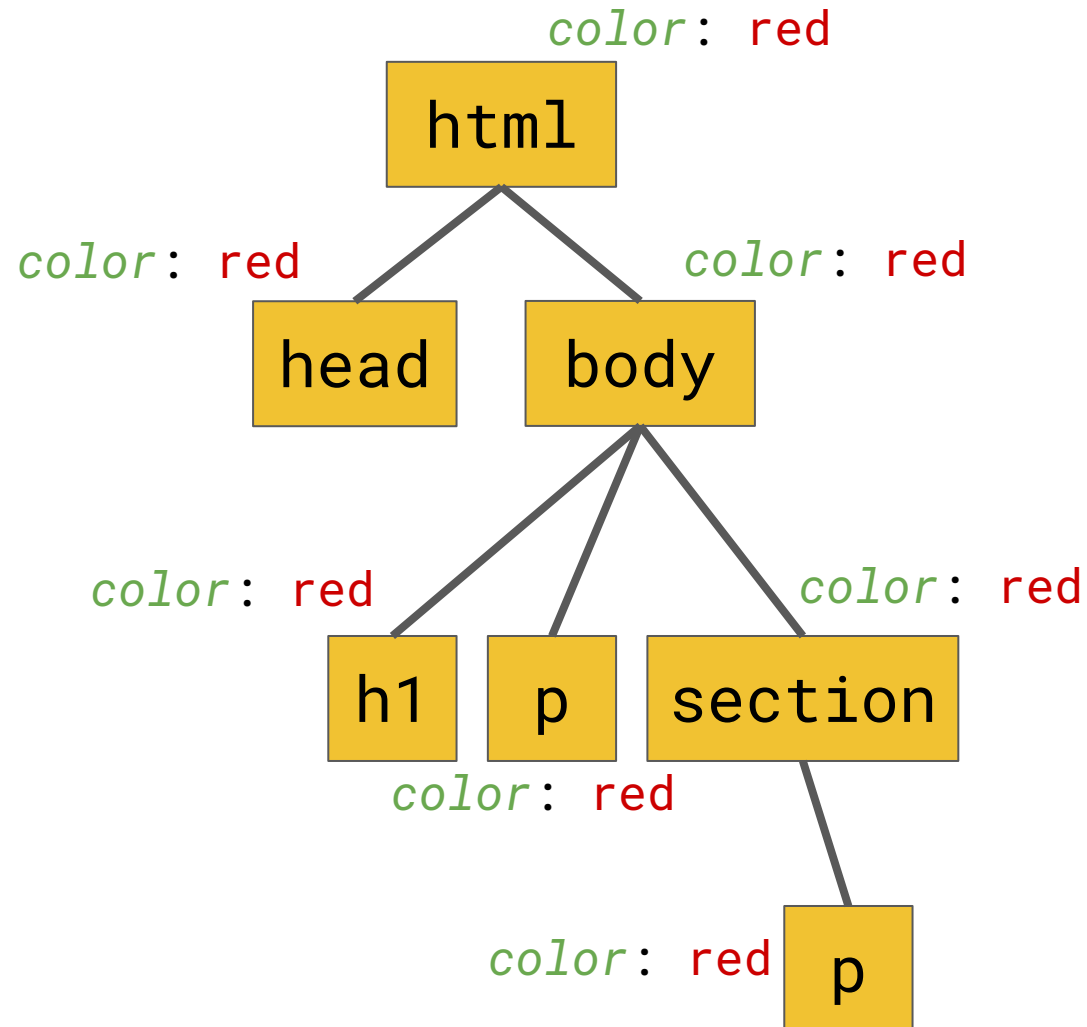
The selector selects a subset of the HTML tree's nodes and apply the declaration to them

Declaration have graphical meaning that will be applied by the browser



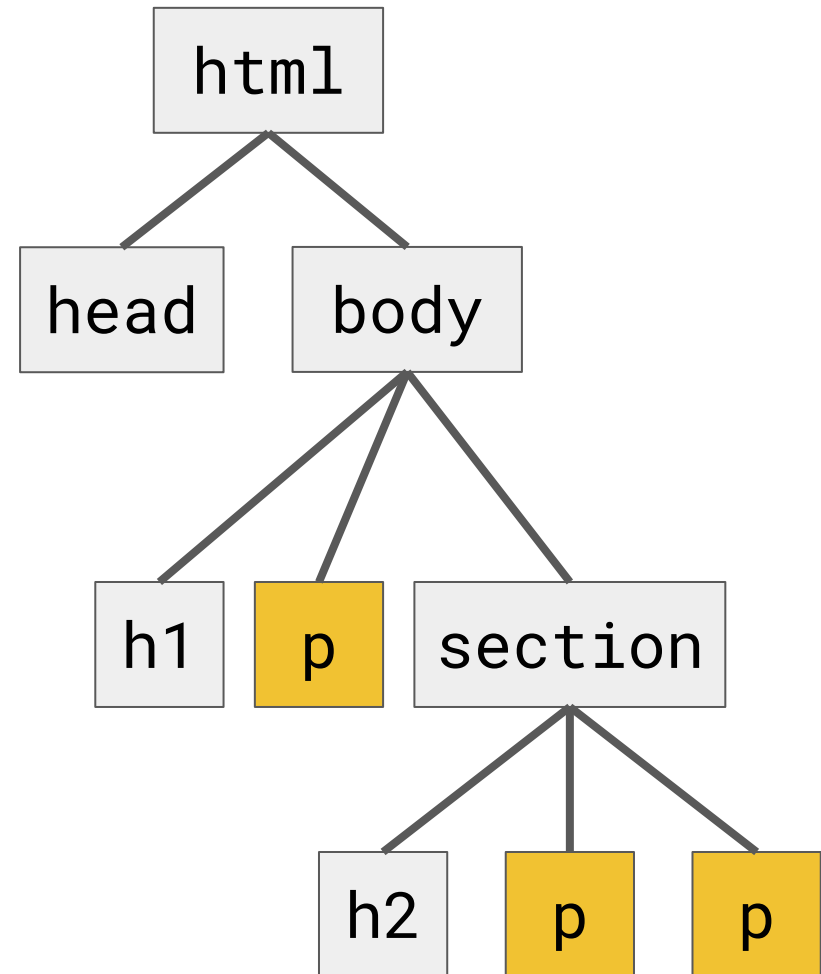
First example with the *joker* selector

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



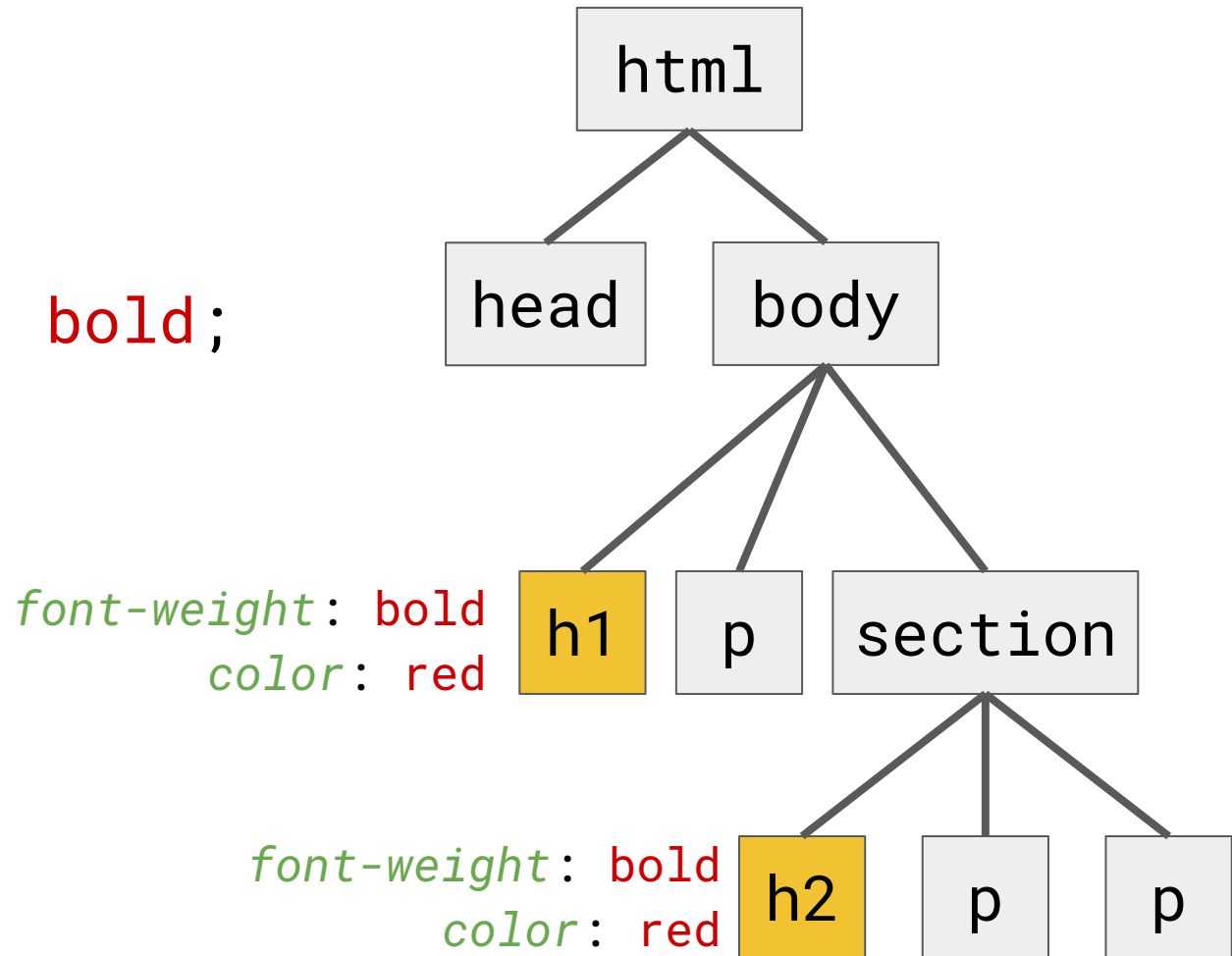
The *tag* selector

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



Selector union

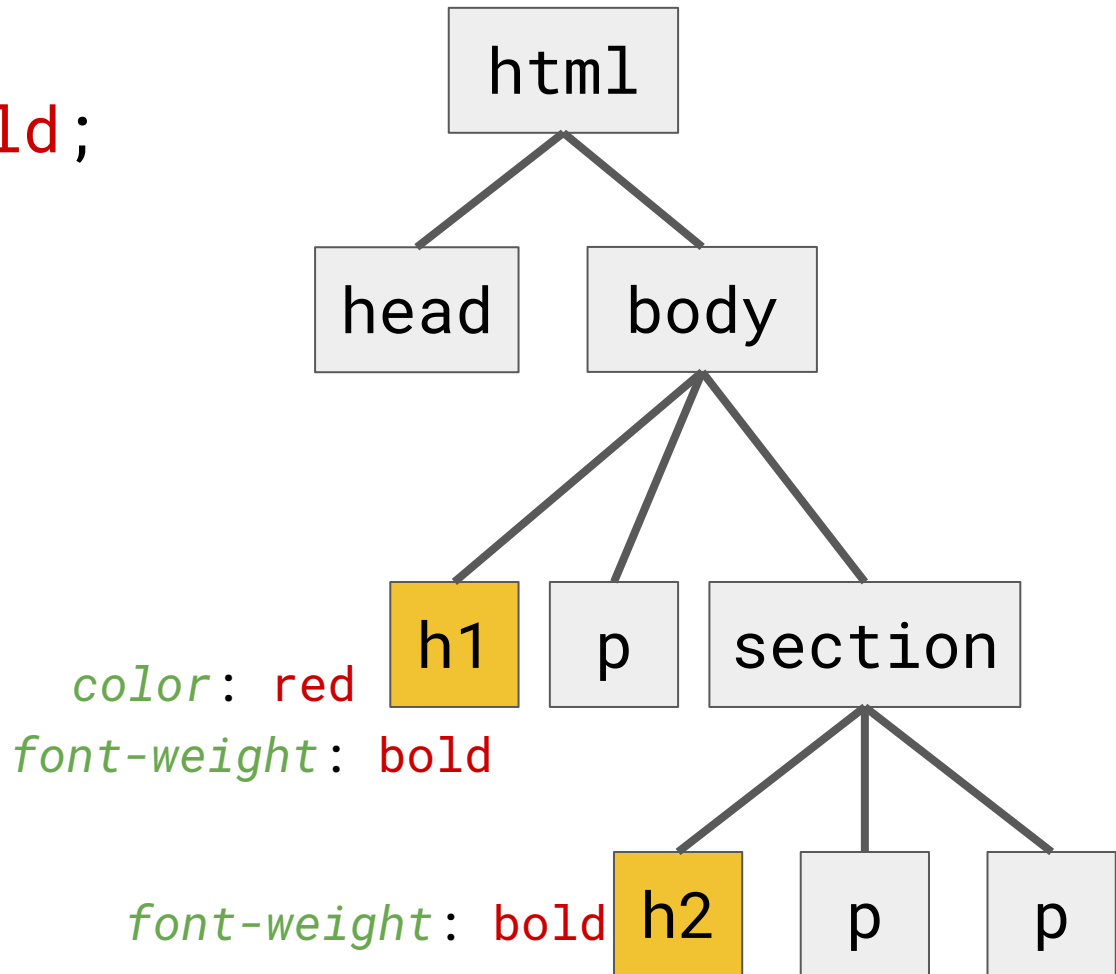
```
h1, h2 {  
  font-weight: bold;  
  color: red;  
}
```



Multiple rules

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

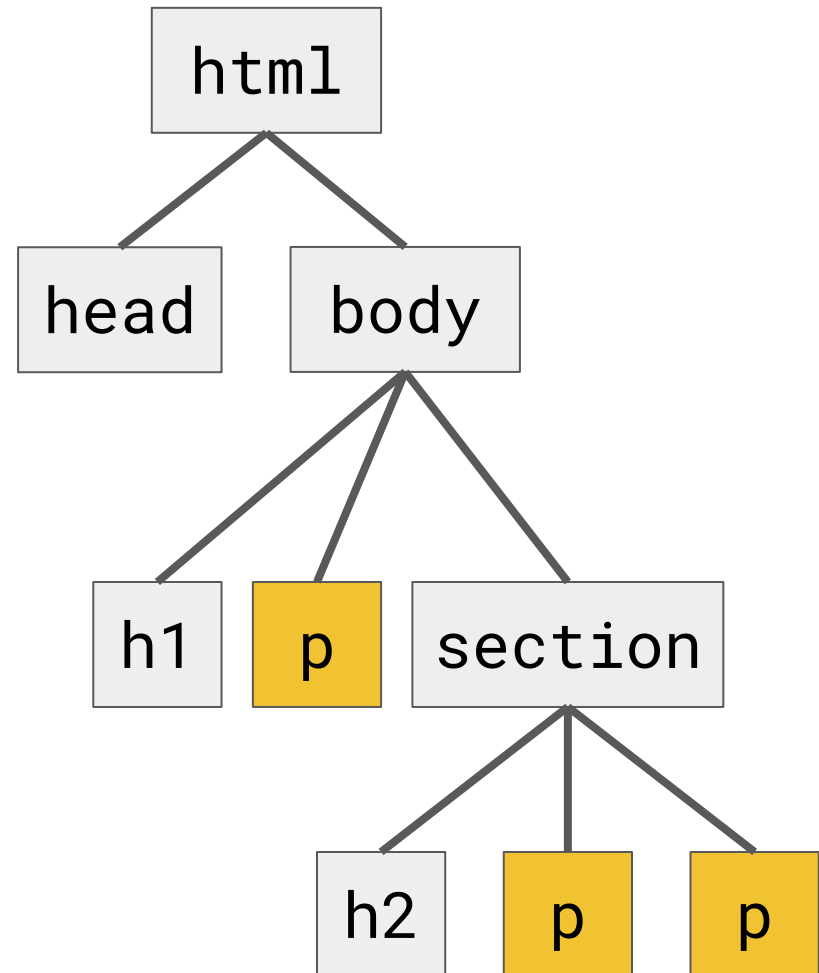
```
h1 {  
  color: red;  
}
```



The parent-child selectors

Selects all paragraphs that are descendants of a body

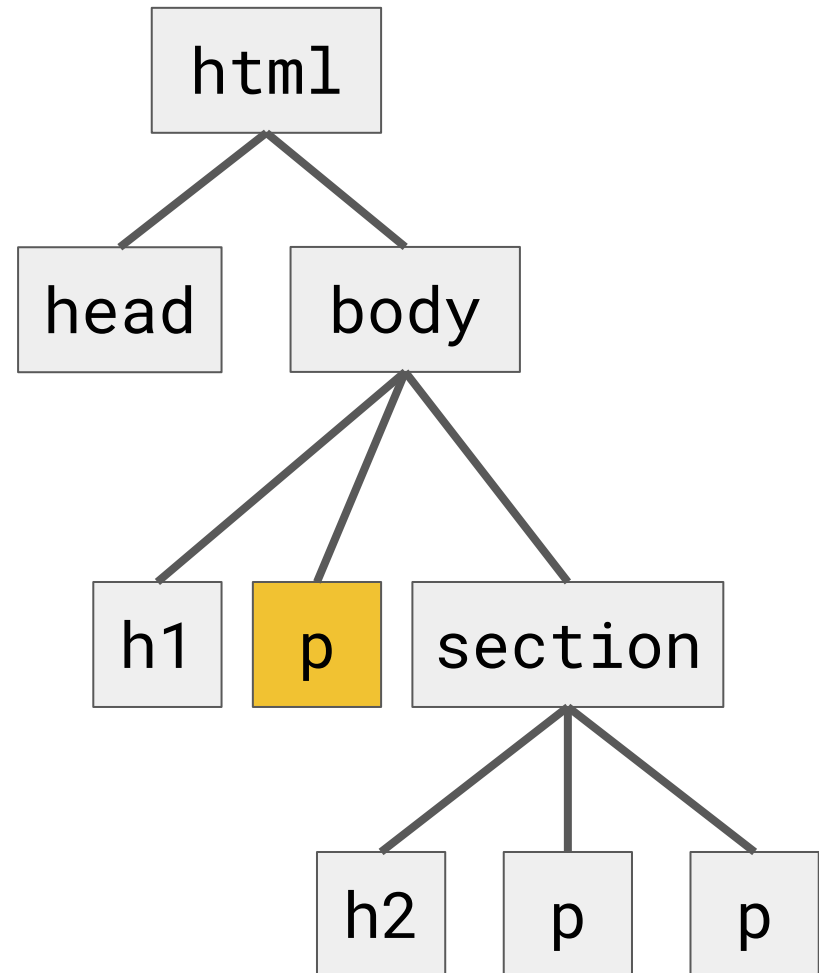
```
body p {  
  color: red;  
}
```



The parent-child selectors

Selects all paragraphs that are direct children of a body

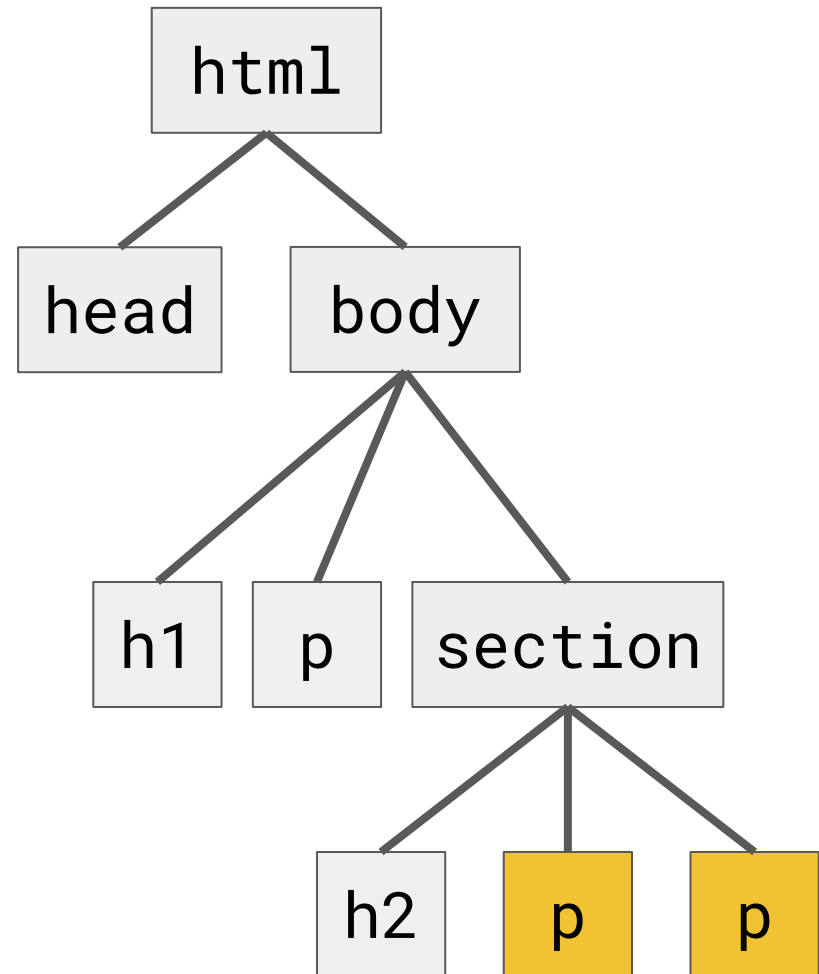
```
body > p {  
  color: red;  
}
```



The sibling selectors

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

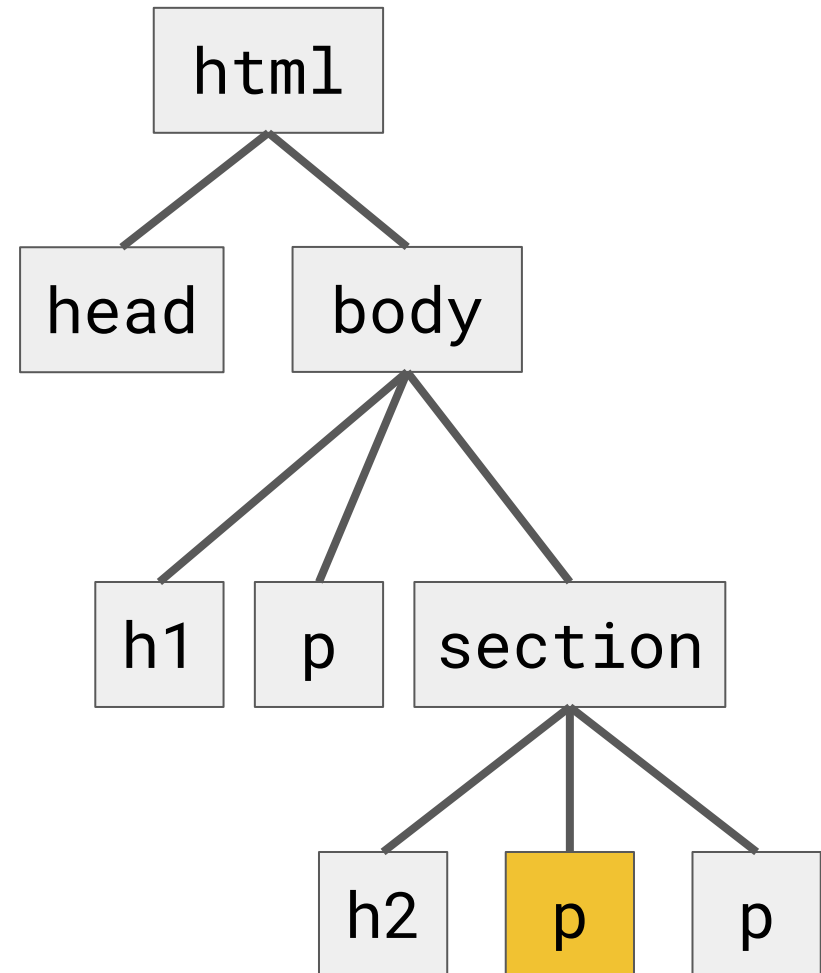
```
h2 ~ p {  
  color: red;  
}
```



The sibling selectors

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

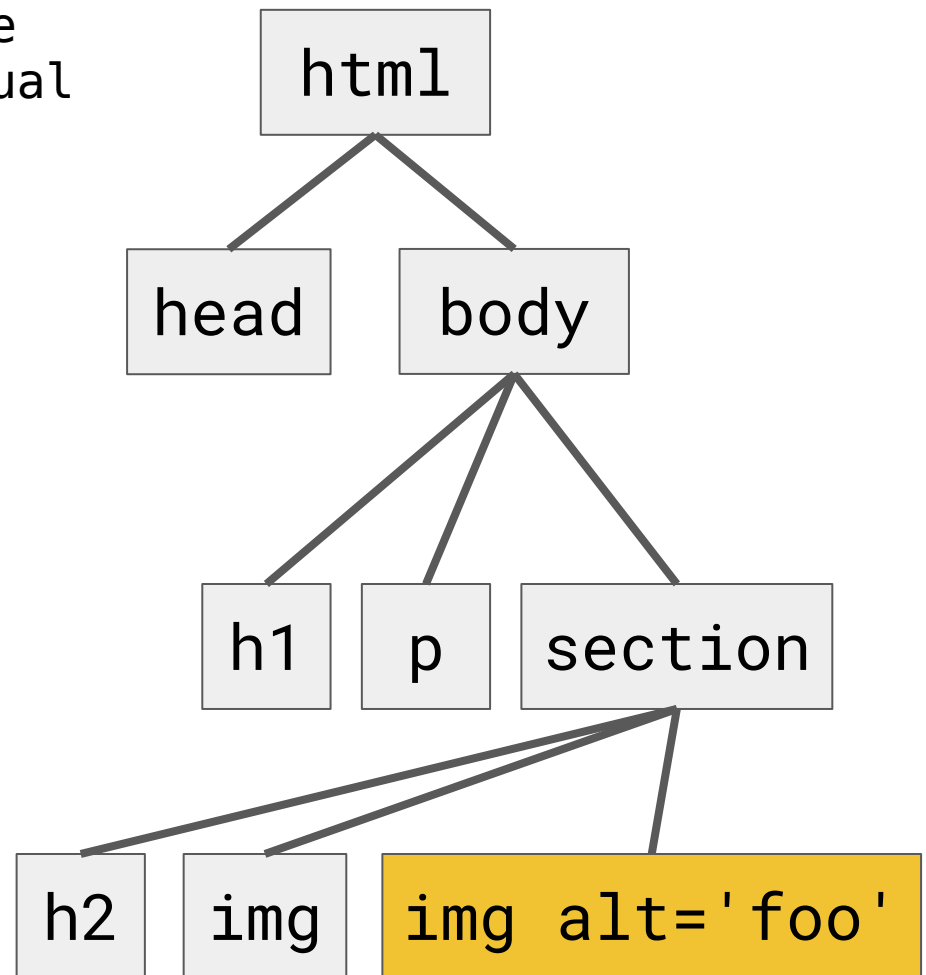
```
h2 + p {  
  color: red;  
}
```



Attribute-based selection

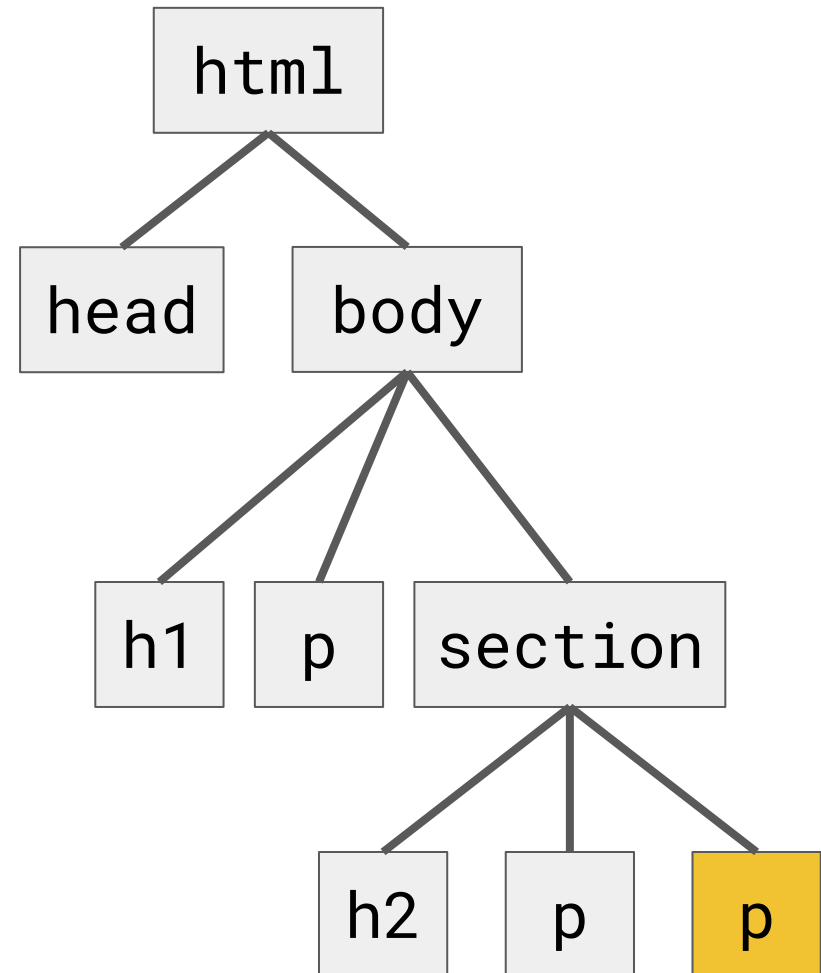
Selects all elements that have «alt» attribute with value equal to «foo»

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



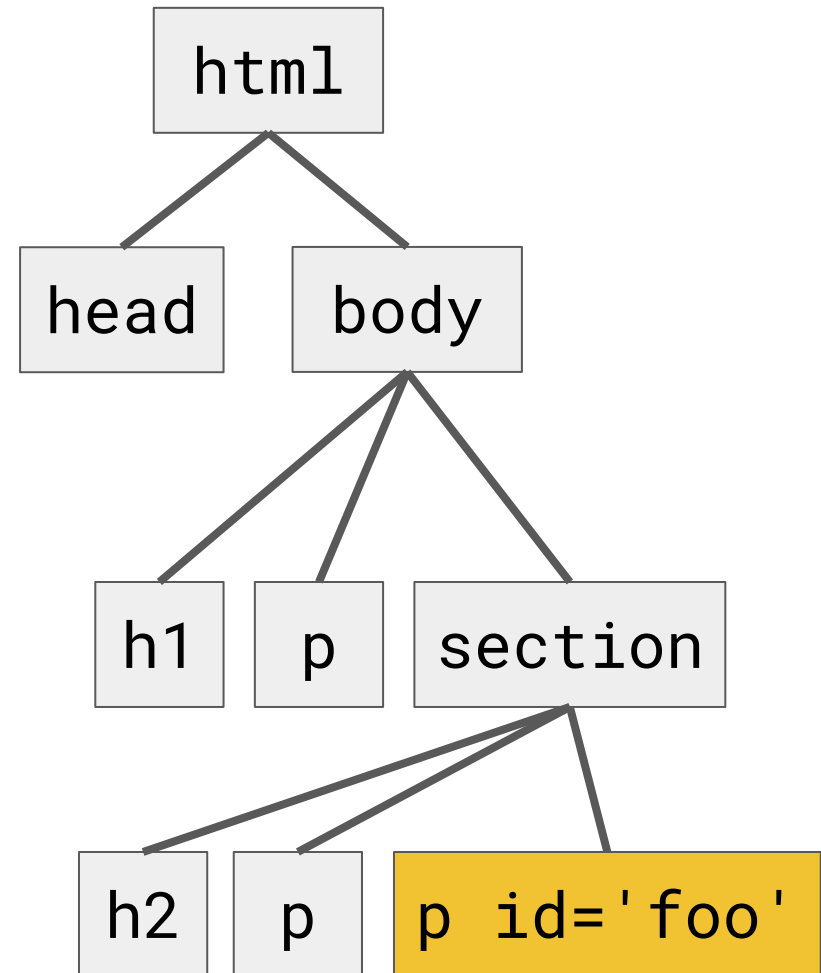
ID-based selection

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



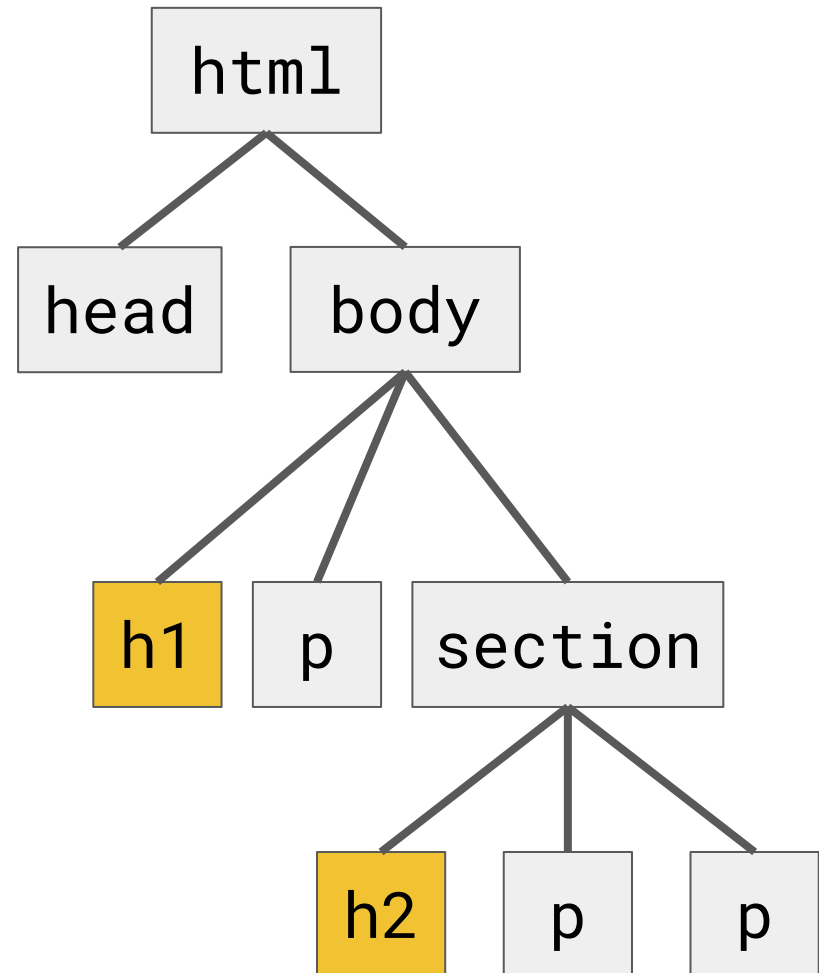
ID based selection

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



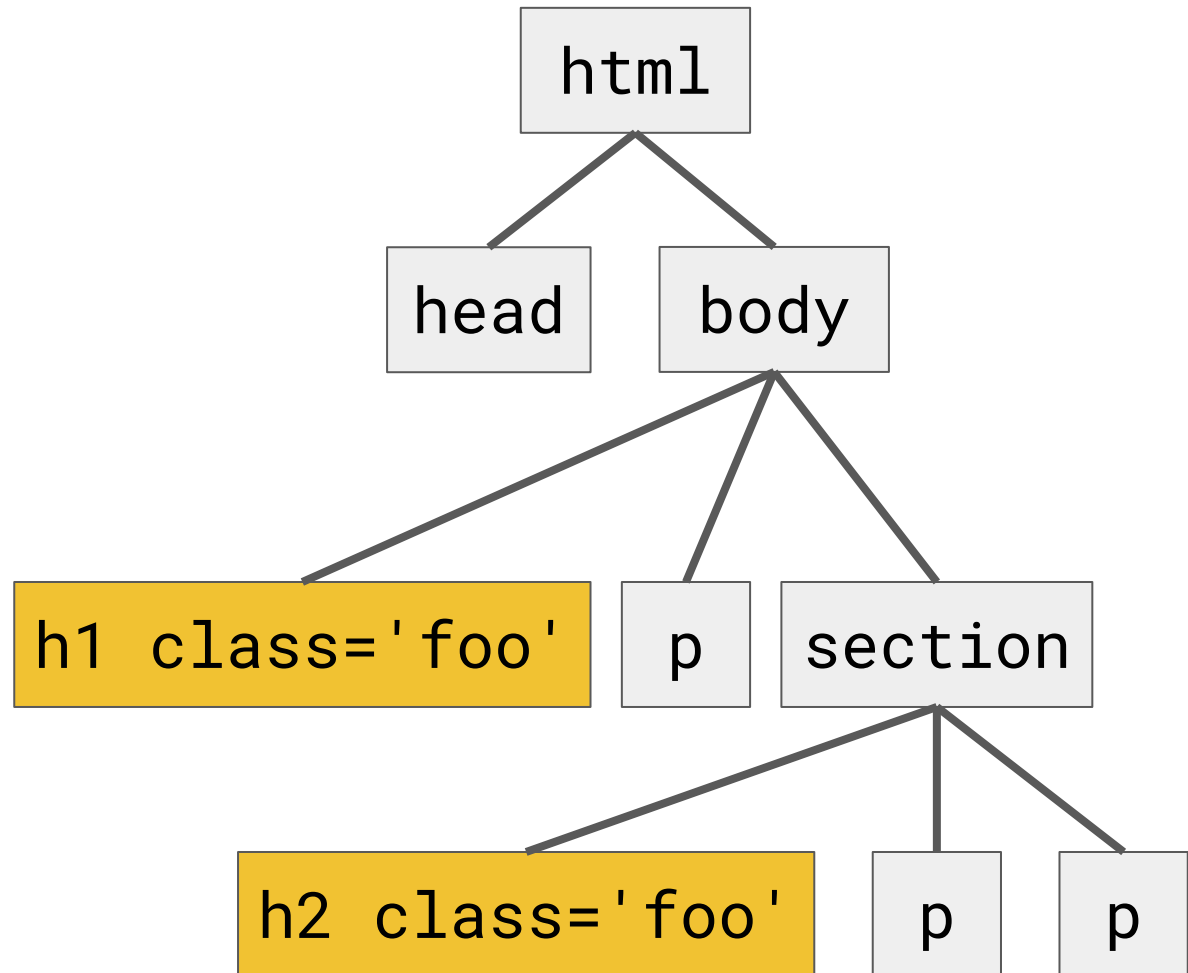
Class-based selection

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!



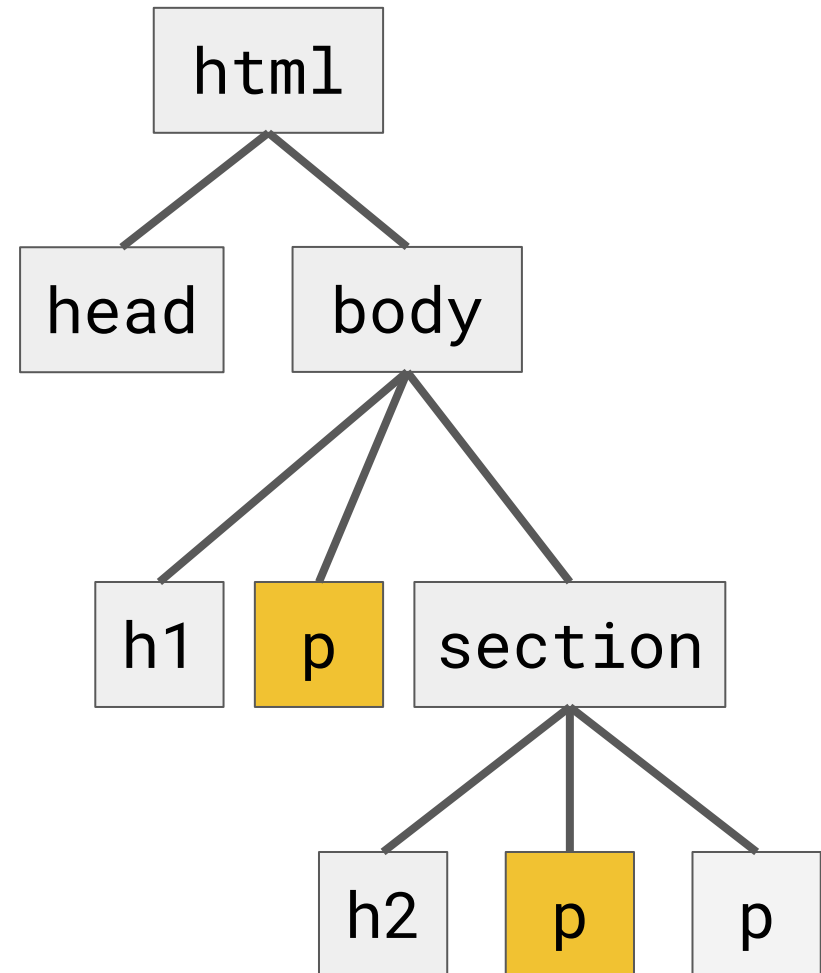
Class-based selection

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

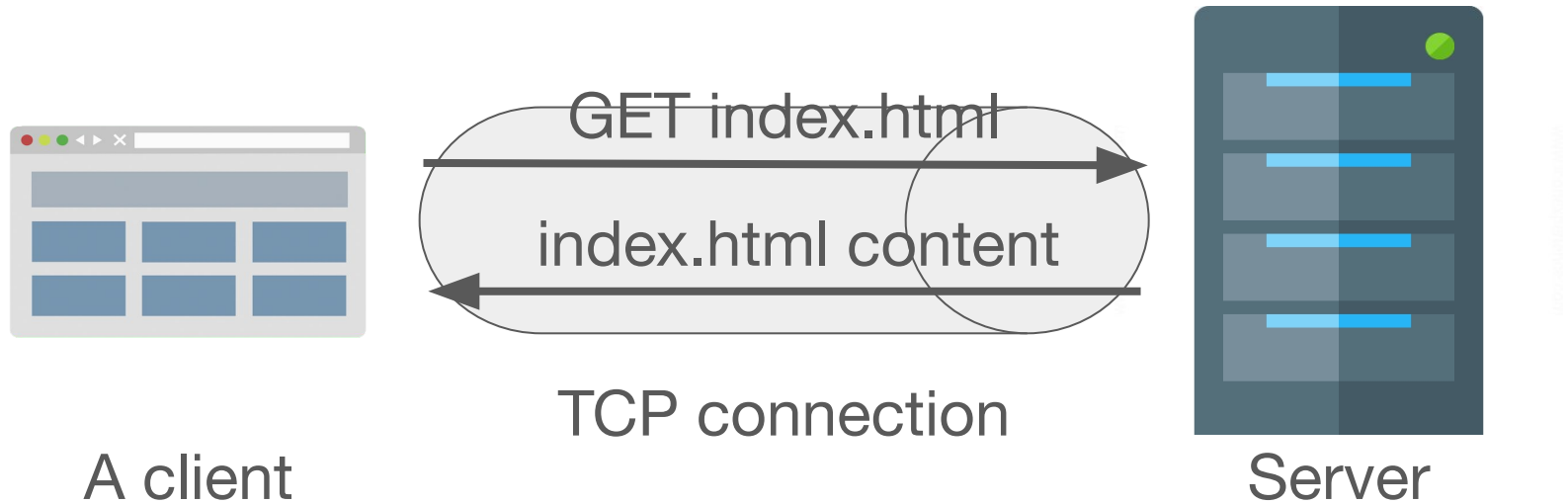


Pseudo class selection

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



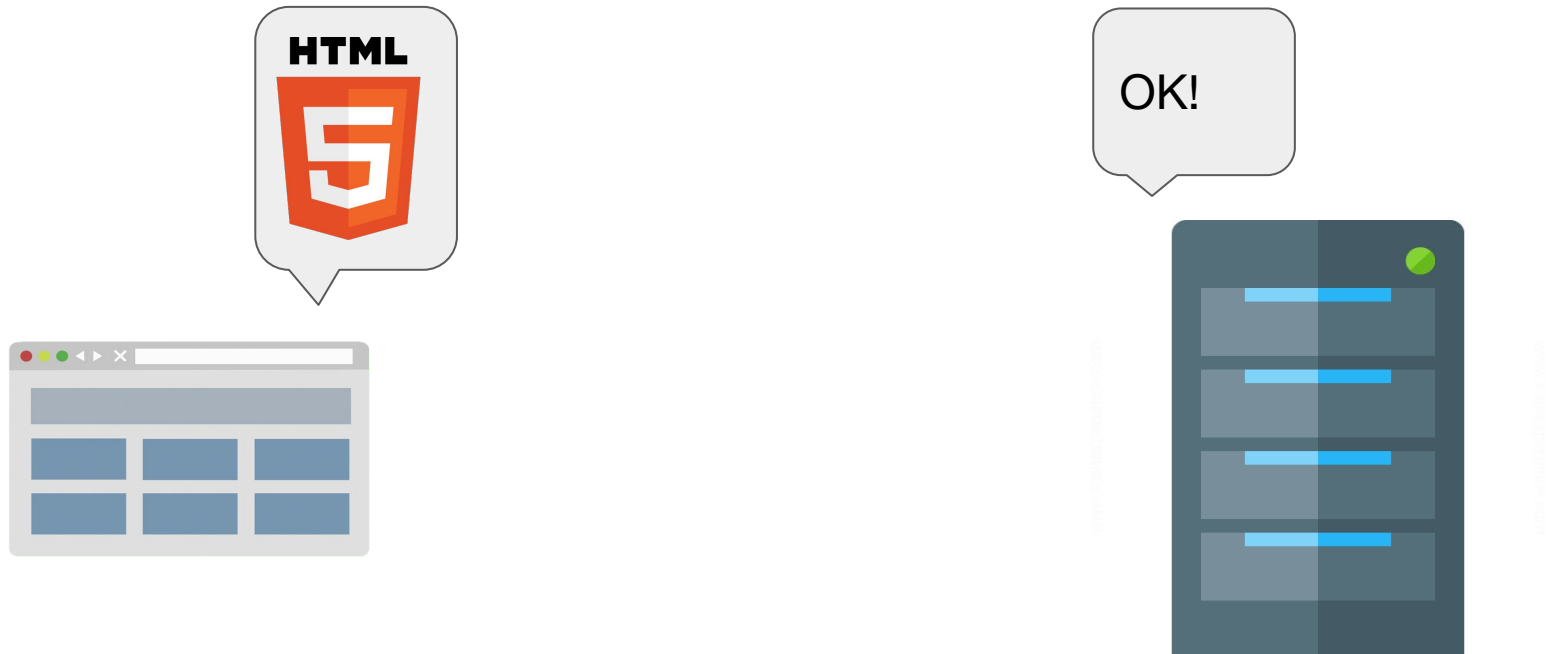
Neat! But how I give CSS to a HTML resource?



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

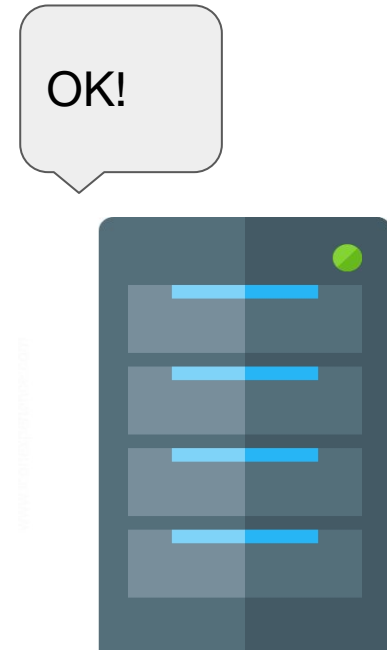
In fact:



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>
```

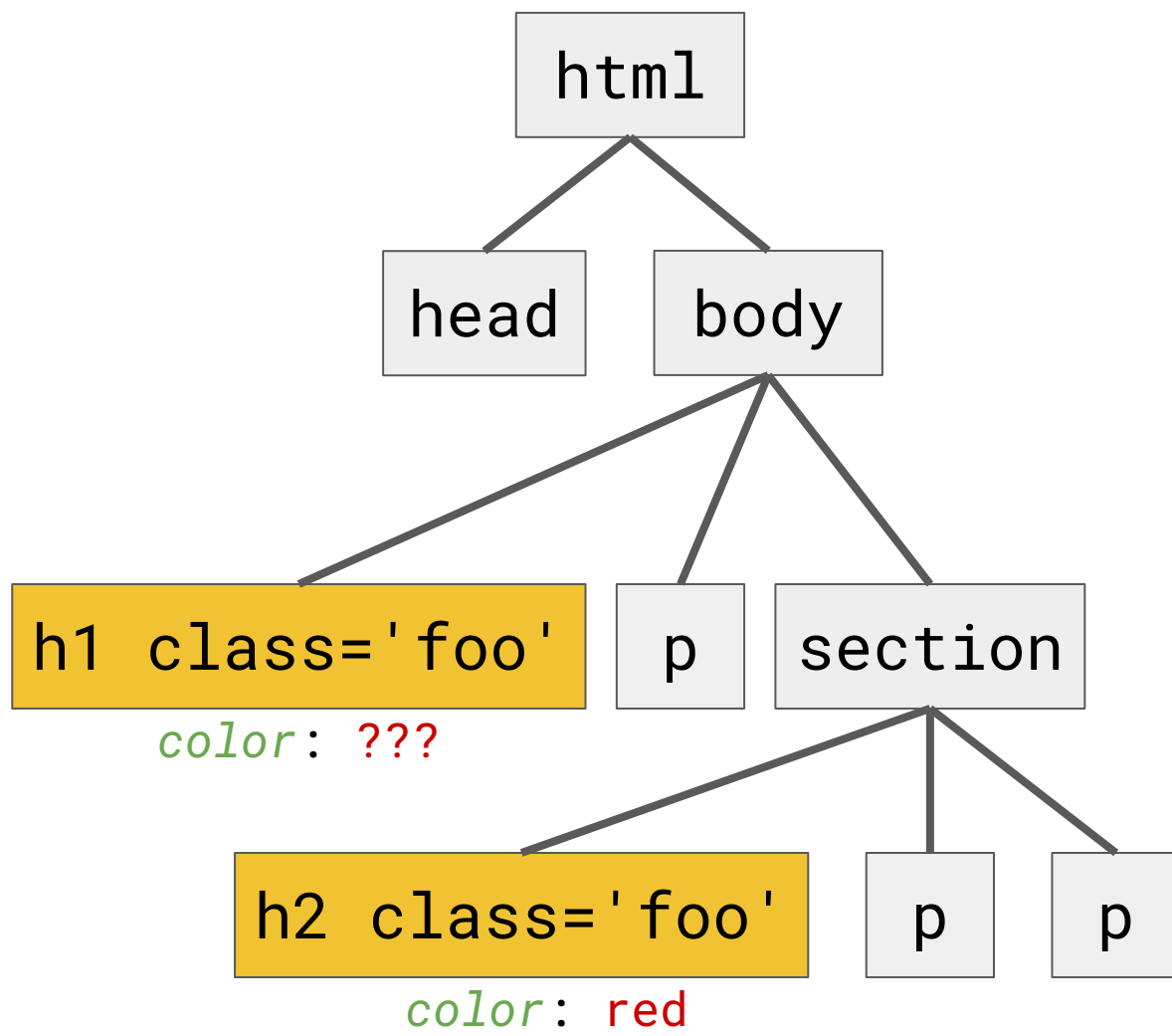
Back to CSS inclusion, the dirty way

```
<!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 (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 equality, 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?**

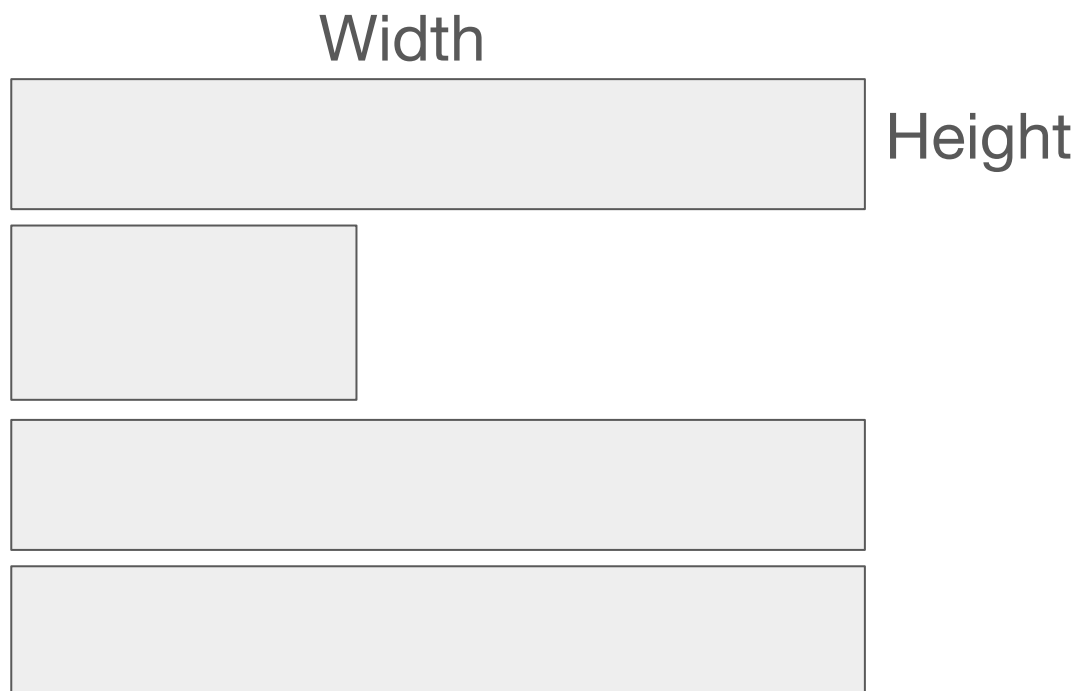
Block and inlines

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

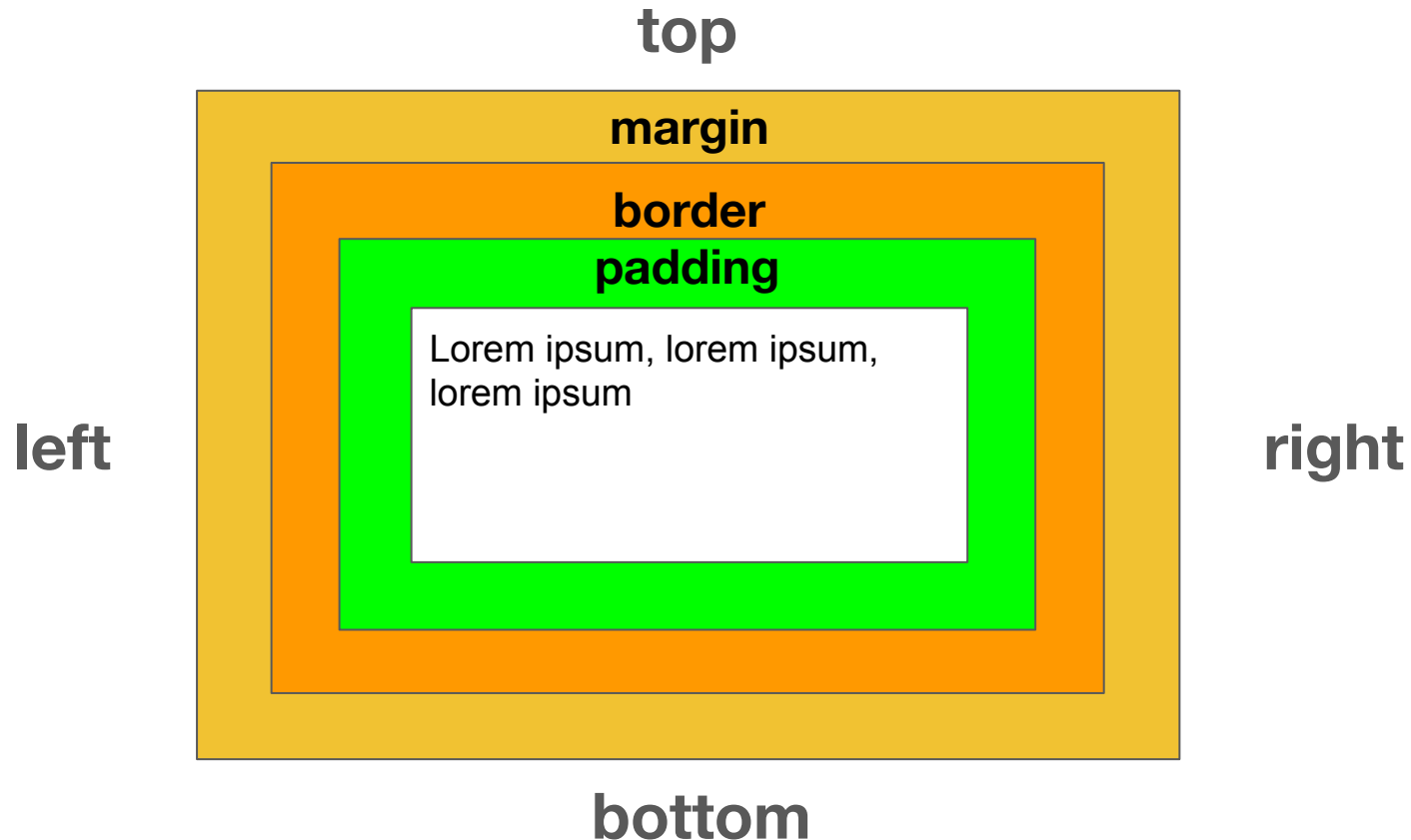


Inline elements

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



Tweaking the size of block elements



Inline elements have only left and right margin/padding

Margin, padding and border properties

https://developer.mozilla.org/en-US/docs/Web/CSS/Guides/Box_model/Margin_collapsing

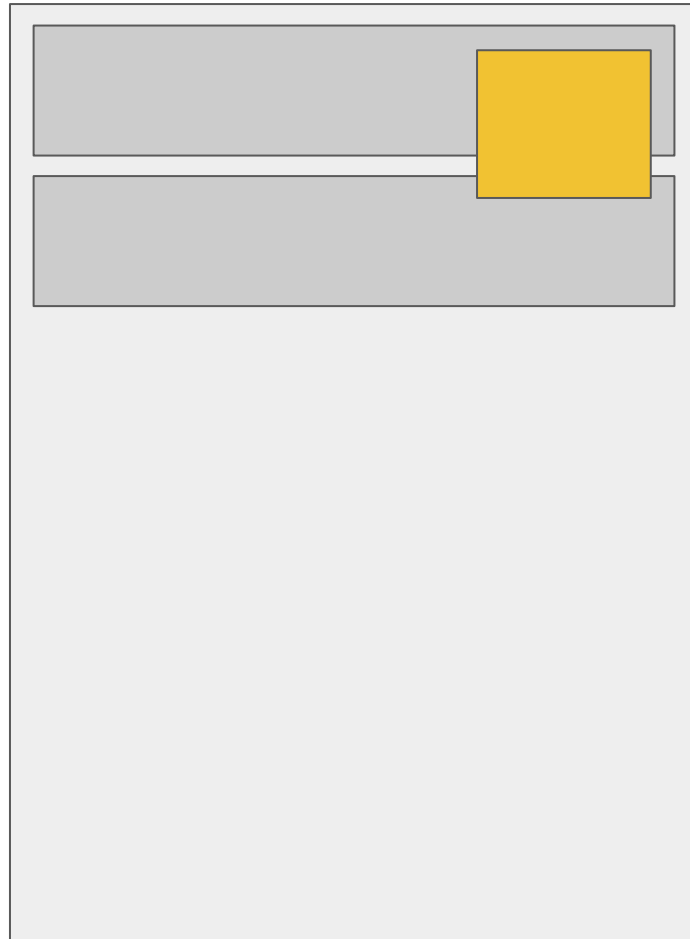
Margin (same for padding):

- `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?

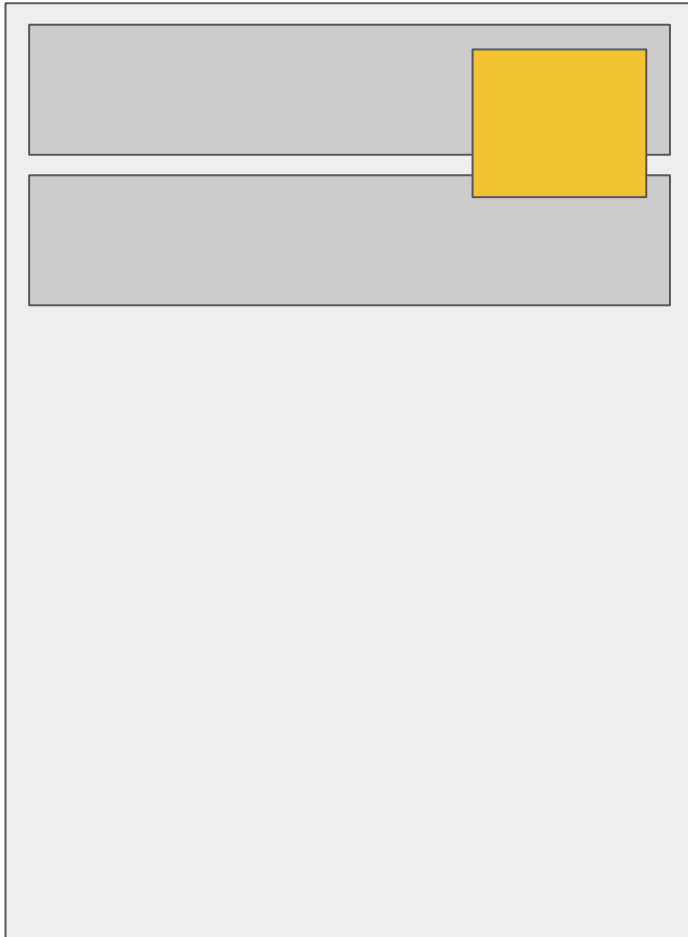


Positioned block

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 🤔



Historical 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**

CSS Grid Layout

- Two-dimensional layout system (rows and columns)
- Ideal for page layouts, dashboards, image galleries
- Works alongside Flexbox (not a replacement)

Flexbox vs Grid

- Flexbox → one dimension (row or column)
- Grid → two dimensions (rows and columns)
- Flexbox → content-driven
- Grid → layout-driven

But layout is to present the content! Leave some flexibility