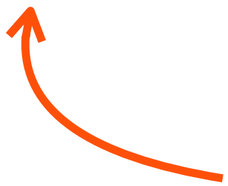
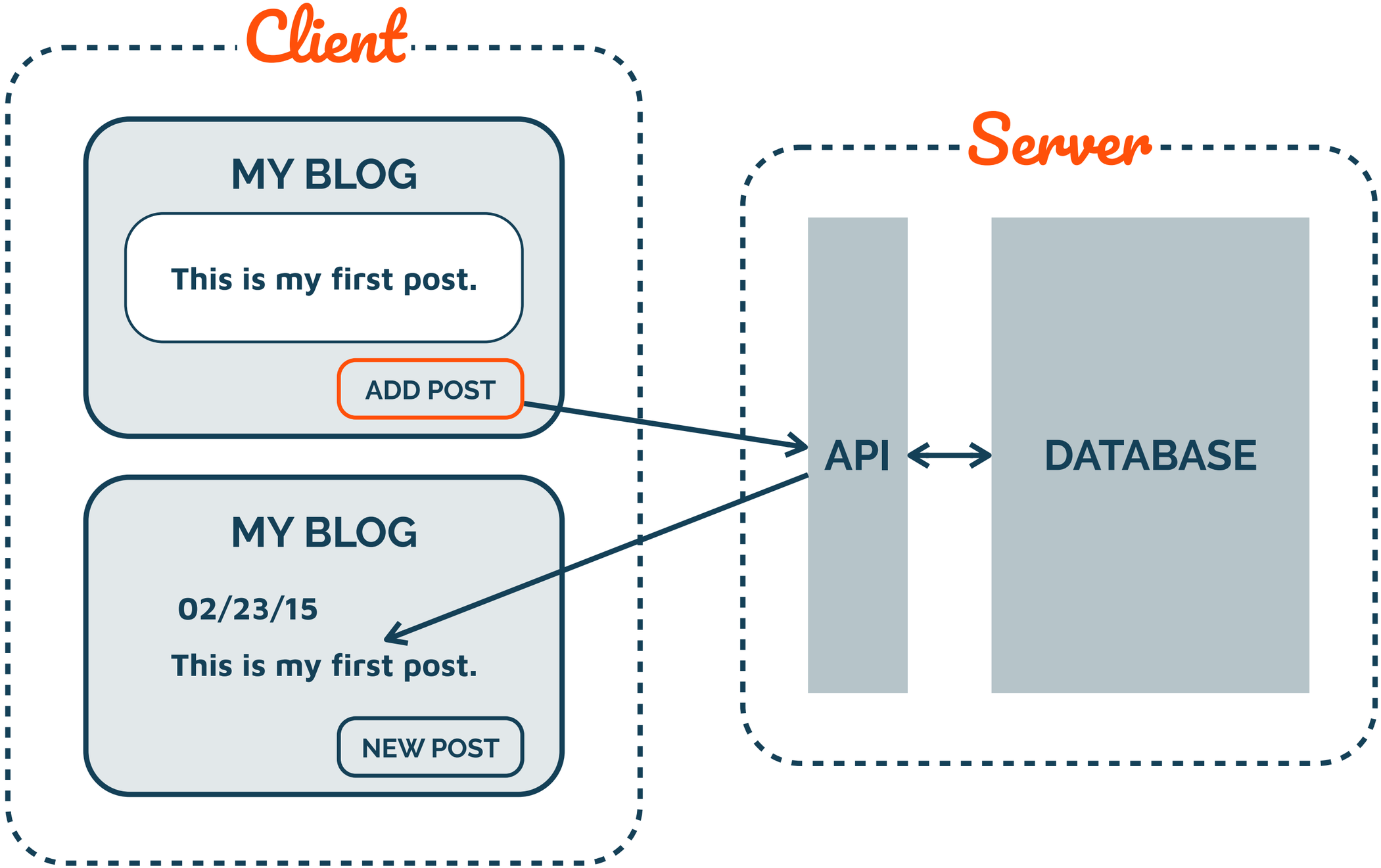
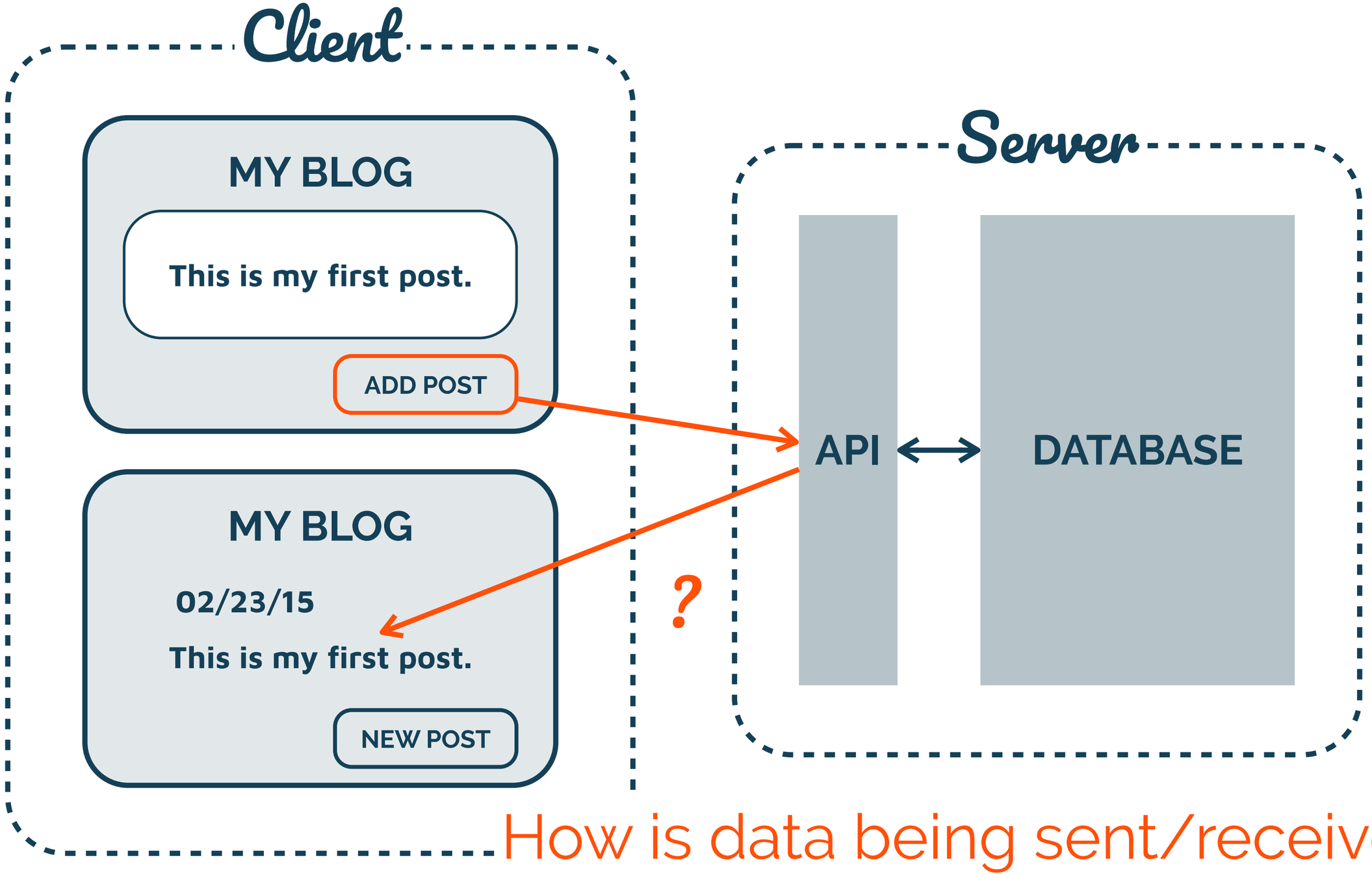
DATA BINDING



Client-side View of Data


Client
MY BLOG
This is my first post.
ADD POST
MY BLOG
02/23/15
This is my first post.
NEW POST
Server
API
DATABASE


Client
MY BLOG
This is my first post.
ADD POST
MY BLOG
02/23/15
This is my first post.
NEW POST
?
Server
API
DATABASE
How is data being sent/received?


Http

HTTP

HyperText Transfer Protocol

Request/Response protocol used by browsers to communicate with servers

All about applying **verbs** to **nouns**

Verbs: GET, POST, PUT, DELETE

Nouns: resources (i.e., concepts)

URL

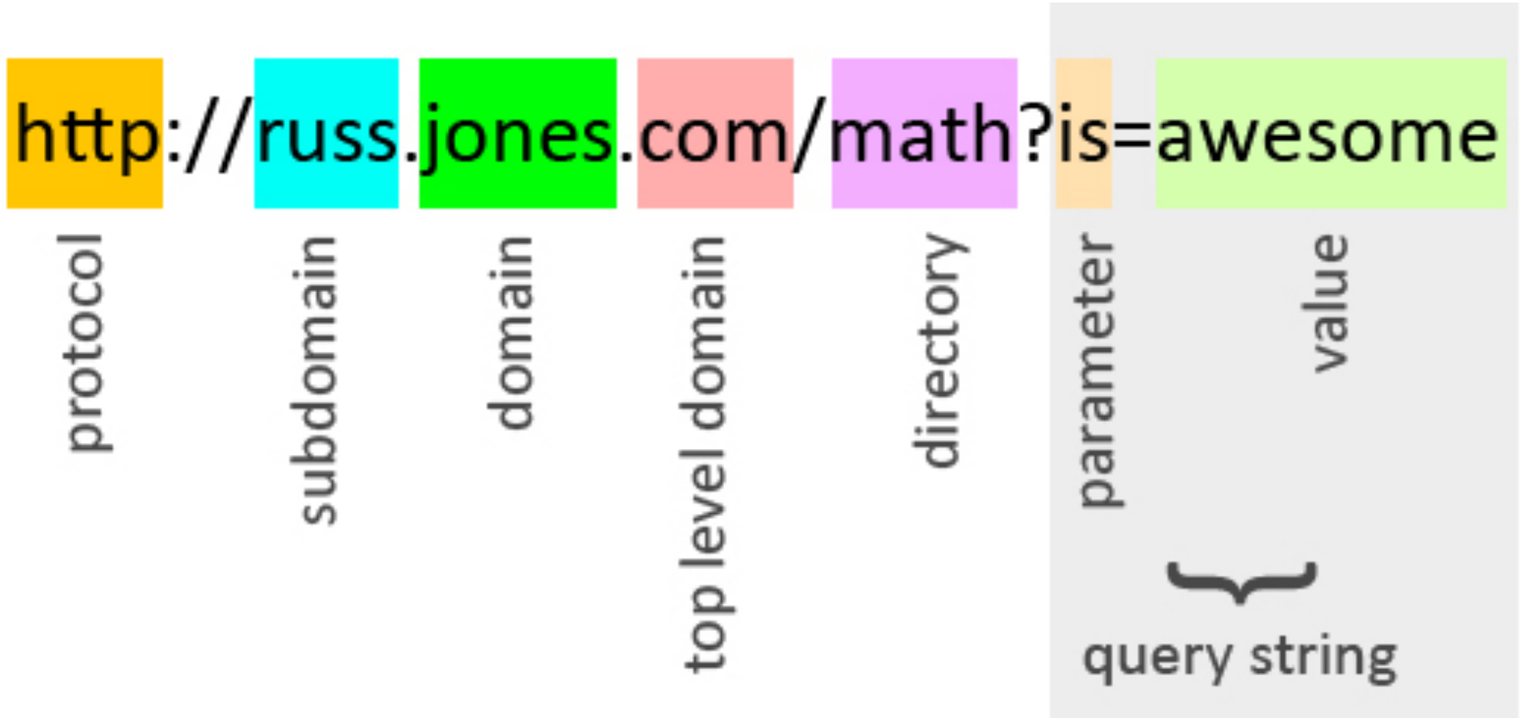
Uniform Resource Locator

Type of **URI** (`Identifier`)

Specifies the **location of a resource** on a network

Server responds with **representations** of resources and not the resources themselves

ANATOMY OF A URL



LOADING A PAGE IN A BROWSER

representations of resources

Browser

HTML

Other Resources

http://creativecommons.org

HTTP GET

```
http://creativecommons.org
<a><span id="home-button">
</span></a>
<div id="logo">
 <span>
  Creative Commons
 </span>
```

HTTP GET

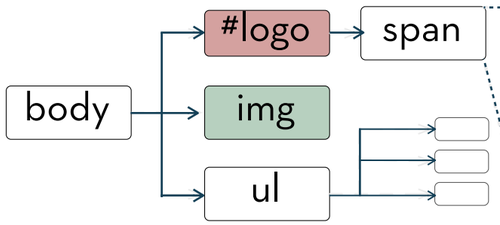
```
cforms.js
//Collap
String.p
function
  return
this.rep
```

```
creativecommons.css
topbar #home-button{
 position: relative;
 float: lef
 display: b
 height: 40
 width: 150
```

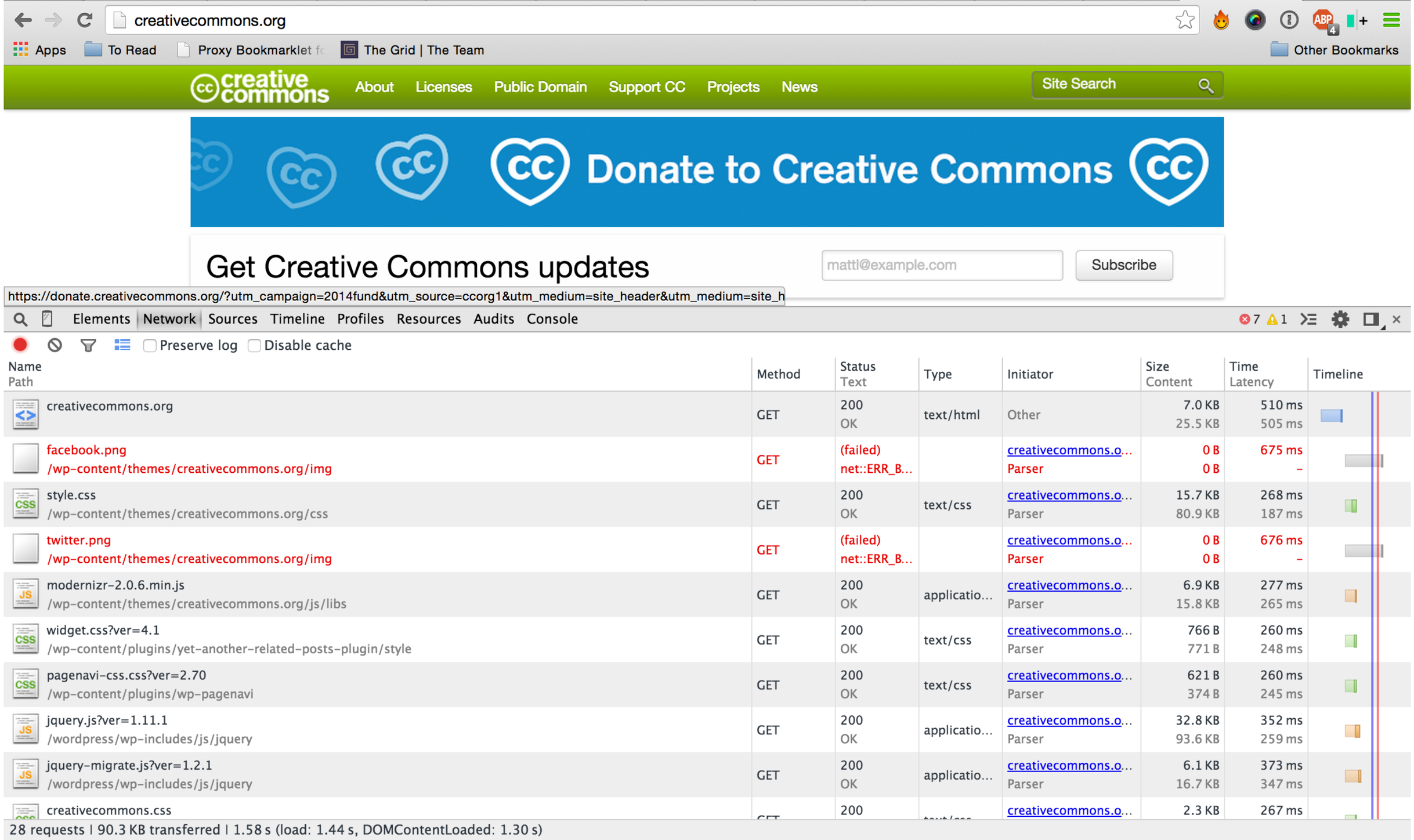
cc-logo.png

Rendered Page

Document Object Model (DOM)




```
topbar span {
  float: left;
  display: block;
  height: 40px;
  width: 150px;
  cursor:
pointer;
  z-index: 1;
  top: 0;
```


creativecommons.org
Apps
To Read
Proxy Bookmarklet f
The Grid | The Team
Other Bookmarks
creative commons
About
Licenses
Public Domain
Support CC
Projects
News
Site Search
Donate to Creative Commons
Get Creative Commons updates
mattl@example.com
Subscribe
https://donate.creativecommons.org/?utm_campaign=2014fund&utm_source=ccorg1&utm_medium=site_header&utm_medium=site_h
Elements
Network
Sources
Timeline
Profiles
Resources
Audits
Console
7
1
Preserve log
Disable cache
Name
Path
Method
Status
Text
Type
Initiator
Size
Content
Time
Latency
Timeline
creativecommons.org
GET
200
OK
text/html
Other
7.0 KB
25.5 KB
510 ms
505 ms
facebook.png
/wp-content/themes/creativecommons.org/img
GET
(failed)
net::ERR_B...
creativecommons.o...
Parser
0 B
0 B
675 ms
-
style.css
/wp-content/themes/creativecommons.org/css
GET
200
OK
text/css
creativecommons.o...
Parser
15.7 KB
80.9 KB
268 ms
187 ms
twitter.png
/wp-content/themes/creativecommons.org/img
GET
(failed)
net::ERR_B...
creativecommons.o...
Parser
0 B
0 B
676 ms
-
modernizr-2.0.6.min.js
/wp-content/themes/creativecommons.org/js/libs
GET
200
OK
applicatio...
creativecommons.o...
Parser
6.9 KB
15.8 KB
277 ms
265 ms
widget.css?ver=4.1
/wp-content/plugins/yet-another-related-posts-plugin/style
GET
200
OK
text/css
creativecommons.o...
Parser
766 B
771 B
260 ms
248 ms
pagenavi-css.css?ver=2.70
/wp-content/plugins/wp-pagenavi
GET
200
OK
text/css
creativecommons.o...
Parser
621 B
374 B
260 ms
245 ms
jquery.js?ver=1.11.1
/wordpress/wp-includes/js/jquery
GET
200
OK
applicatio...
creativecommons.o...
Parser
32.8 KB
93.6 KB
352 ms
259 ms
jquery-migrate.js?ver=1.2.1
/wordpress/wp-includes/js/jquery
GET
200
OK
applicatio...
creativecommons.o...
Parser
6.1 KB
16.7 KB
373 ms
347 ms
creativecommons.css
200
creativecommons.o...
2.3 KB
267 ms
28 requests | 90.3 KB transferred | 1.58 s (load: 1.44 s, DOMContentLoaded: 1.30 s)

HTTP GET Request

method url version

```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/xml,application/
xml,application/xhtml+xml,text/html*/*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
<blank line>
```

request headers

HTTP GET Response

version

status code

text explanation

```
HTTP/1.1 200 OK
```

response headers

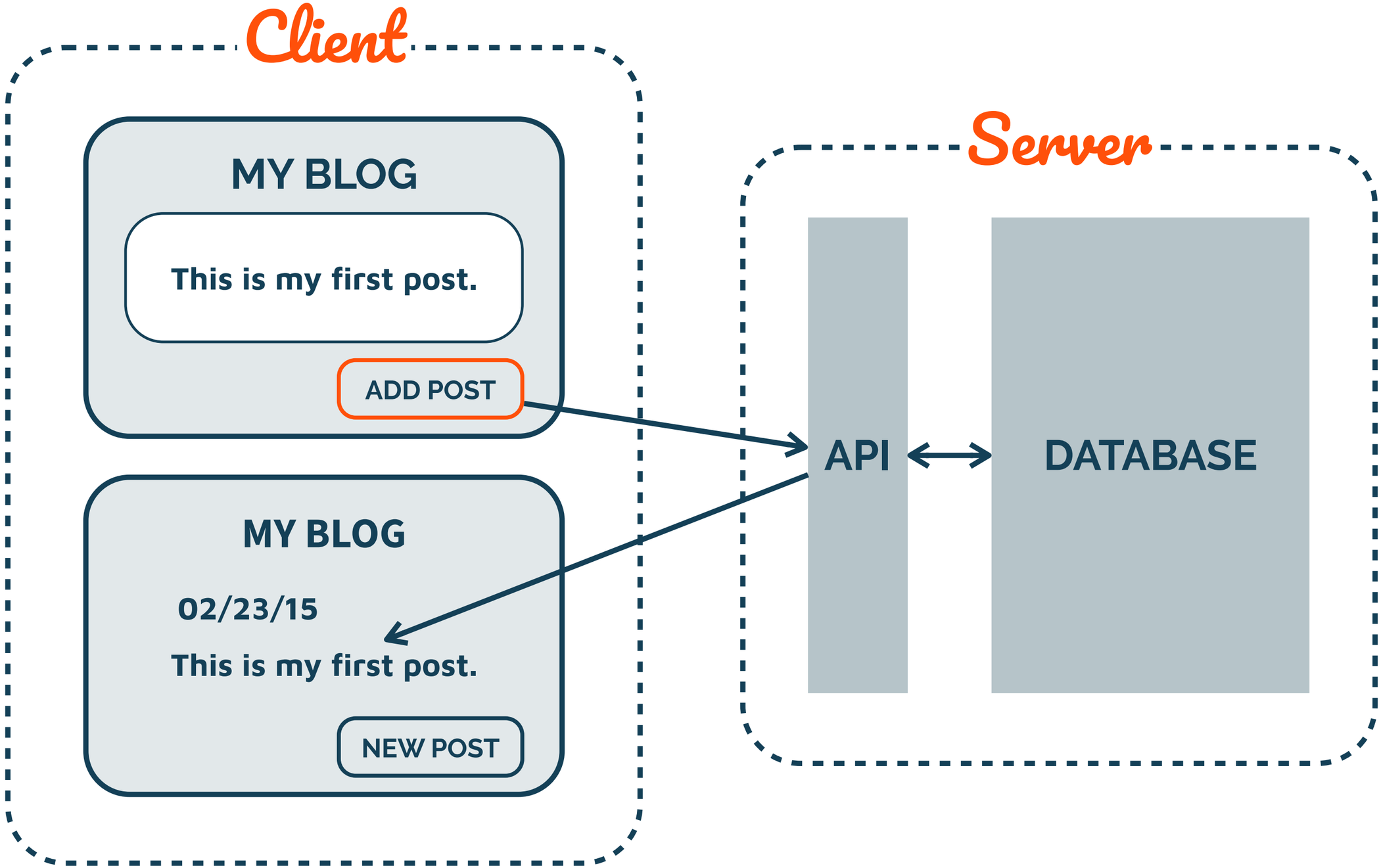
```
Date: Mon, 23 May 2005 22:38:34 GMT

Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)

Content-Type: text/html; charset=UTF-8

Content-Length: 131
```

body

```
<!DOCTYPE html>

<html>

…

</html>
```

HTTP GET Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
Content-Type: text/html; charset=UTF-8
Content-Length: 131

<!DOCTYPE html>
<html>
…
</html>
```


Client
MY BLOG
This is my first post.
ADD POST
MY BLOG
02/23/15
This is my first post.
NEW POST
Server
API
DATABASE

HTTP POST Request

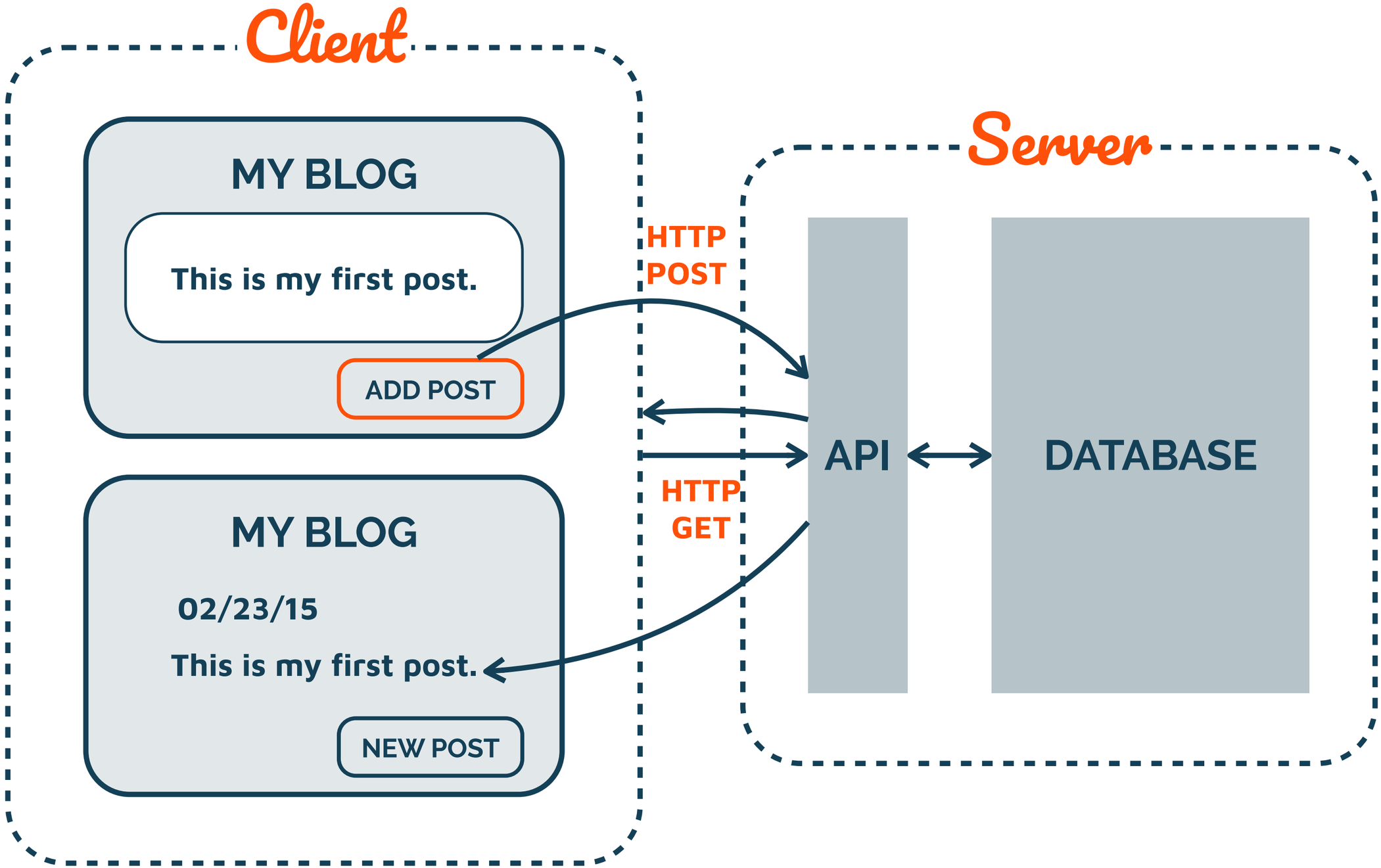
```
POST /messages HTTP/1.1

Host: www.anotherblogpost.com

Content-type: application/x-
www-form-urlencoded

<blank line>

entity-body
```

HTTP POST Response

HTTP/1.1 303 See Other

Content-type: text/html

Location: http://
www.anotherblogpost.com/
messages/3486152


Client
MY BLOG
This is my first post.
ADD POST
MY BLOG
02/23/15
This is my first post.
NEW POST
HTTP
POST
HTTP
GET
Server
API
DATABASE

GET

retrieve representations of resources

no side effects

no data in request body

vs

POST

upload data from the browser to server

returns information from the server

side effects are likely

data contained in request body

HTTP STATUS CODES

1XX	Informational Responses	100 Continue
2XX	Successful Responses	200 OK 201 Created
3XX	Redirects	301 Moved Permanently 304 Not Modified
4XX	Client Errors	400 Bad Request 404 Not Found
5XX	Server Errors	500 Internal Server Error 503 Service Unavailable

HTTPS

Request and response messages are transmitted **securely**

Use SSL (Secure Sockets Layer) certificates to **encrypt** data

more on this in coming weeks

Ajax

AJAX

Asynchronous JavaScript and XML

Before, every user interaction required the complete page to be **reloaded**

Now, we can send and receive data **without reloading** page

Issue HTTP **request** to the server from Javascript

Process **response** with Javascript in the browser

JSON

Javascript Object Notation

AJAX doesn’t require XML

JSON has become the de-facto standard **data interchange** format

Lightweight and simple

Types: Number, String, Boolean, Array, Object, null

Objects are **key/value** pairs

SAMPLE JSON

```
{
  “camelids”: [
    {
      “name”: “llama”,
      “height”: 1.8
    },
    {
      “name”: “alpaca”,
      “height”: 0.9
    }
  ]
}
```

Look familiar?

XHR

XMLHttpRequest

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = xhrHandler;
xhr.open('get', 'llama.json');
xhr.send(null);
```

XHR

XMLHttpRequest

```
function xhrHandler() {
  if (xhr.readyState == 4
      && xhr.status == 200) {
    var data = JSON.parse(xhr.responseText);
    myFunction(data);
  }
};
```

CodePen

AJAX CHALLENGES

Hard to go back to a particular state

Content retrieved by AJAX **not easily indexable**

Same-origin policy prevents some AJAX techniques from being used across domains

Callback-style programming is **hard to maintain/test**

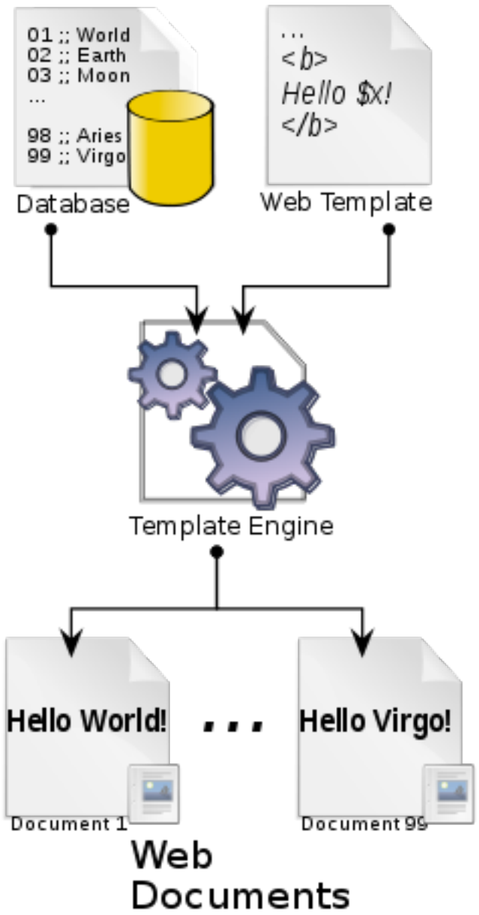
Client-side Templating

TEMPLATES

Common way to generate **dynamic HTML** for **multi-page** web sites and apps

Separation of **markup** and **data** (content)

SERVER-SIDE TEMPLATES



Server puts HTML and data together
and sends it to the browser

Platforms like Rails, PHP, JSP

CLIENT-SIDE TEMPLATES

Browser receives HTML and data and **puts it together**

Server serves templates and data required by the templates

Made popular by AJAX

Resources - Nouns

RESOURCES

Nouns

If your users might
“want to create a hypertext link to it, make or refute assertions about it, retrieve or cache a representation of it, include all or part of it by reference into another representation, annotate it, or perform other operations on it”
then, make it a **resource**

They can be anything: a document, a row in a database, the result of running an algorithm, etc.

REPRESENTATION OF RESOURCES

When a client issues a GET request for a resource, server responds with **representations** of resources and not the resources themselves

Any **machine-readable** document containing any information about a resource

Server may send data from its database as HTML, XML, JSON, etc.

REST

Representational State Transfer

Architectural style, set of **design constraints**

Coined in Roy T. Fielding's dissertation (2000)

The Web is the largest implementation

Three important technologies: HTTP, URL, HTML

REPRESENTATIONAL STATE TRANSFER

Representations are transferred back and forth from client and server

Server sends a representation describing the **state of a resource**

Client sends a representation describing the **state it would like the resource to have**

MULTIPLE REPRESENTATIONS

A resource can have **more than one** representation: different languages, different formats (HTML, XML, JSON)

Client can distinguish between representations based on the value of **Content-Type** (HTTP header)

A resource can have **multiple representations**—one URL for every representation

HTTP Methods - Verbs

Verbs

GET	Get a representation of resource
DELETE	Destroy resource
POST	Create a new resource based on the given representation
PUT	Replace resource state with the one described in the given representation
HEAD	Get the headers that would be sent with a representation, but not the representation itself
OPTIONS	Discover which HTTP methods this resource responds to
PATCH	Modify part of the state of this resource based on the given representation

GET

Retrieve representations of resources

Safe Method: no side effects, not intended to change any resource state

Response codes: 200 (OK), 302 (Moved Permanently), 404 (Not Found)

DELETE

Destroy a resource on the server

Success response codes: 200 (OK), 204 (No Content), 202 (Accepted)

Not safe, but **idempotent**

POST

Upload data from the browser to server

Usually means “create a new resource,” but can be used to convey **any kind of change**: PUT, DELETE, etc.

Data contained in request body

Success response codes: 201 (Created), Location header contains URL for created resource; 202 (Accepted), new resource will be created in the future

Not safe or idempotent

PUT

Request to **modify** resource state

Success response codes: 200 (OK), 204 (No Content)

Can also be used like POST

Idempotent

	Request Body	Response Body	Safe	Idempotent
GET	Optional	Yes	Yes	Yes
DELETE	Optional	Yes	No	Yes
POST	Yes	Yes	No	No
PUT	Yes	Yes	No	Yes
HEAD	Optional	No	Yes	Yes
OPTIONS	Optional	Yes	Yes	Yes
PATCH	Yes	Yes	No	No

demo

https://gitlab.com/uiuc-web-programming/http-demo

NEXT CLASS:
FRONT END FRAMEWORKS

https://uiuc-web-programming.gitlab.io/fa21/