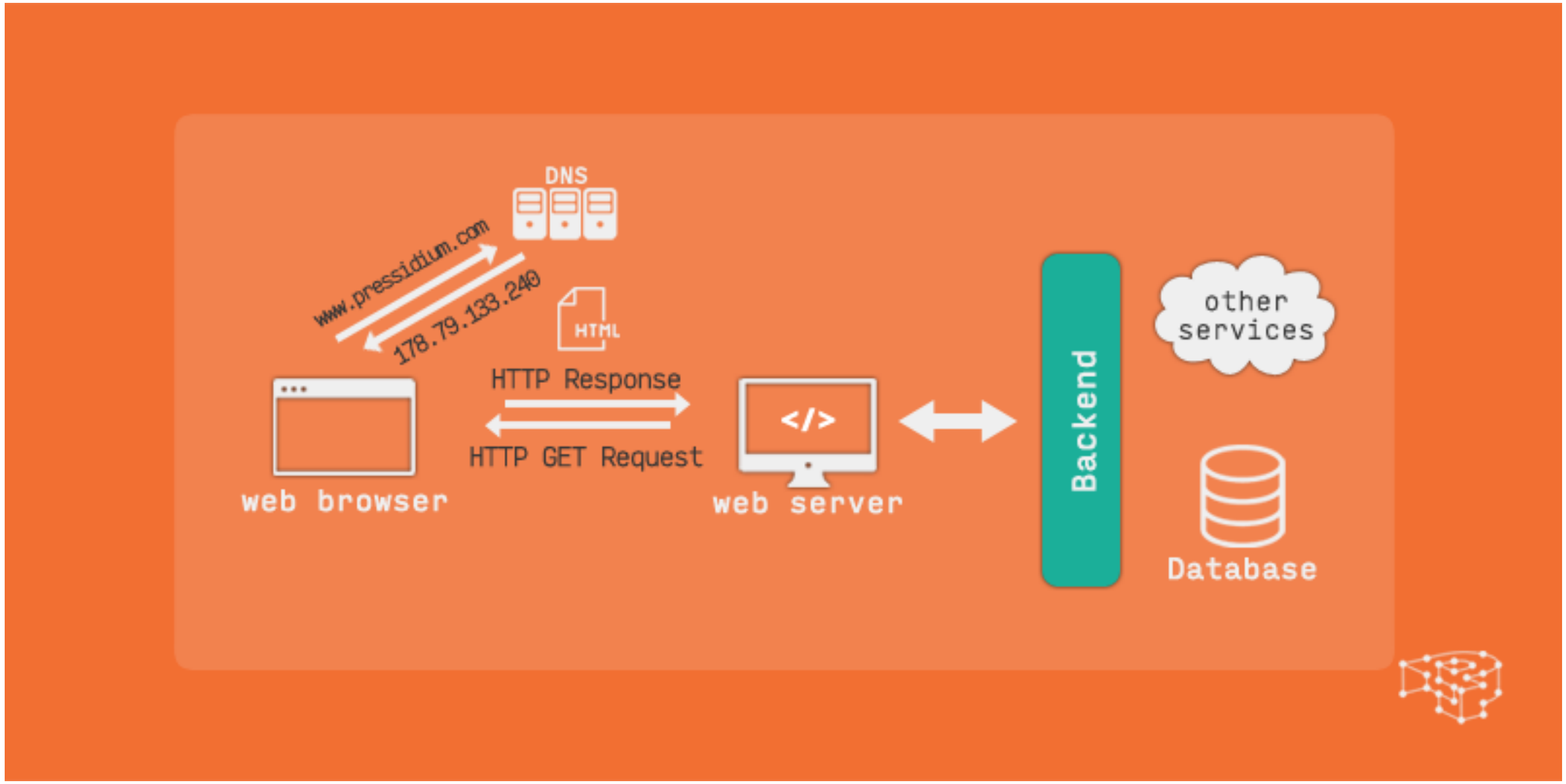
当你打开网页时，到底发生了什么？

网站运行原理讲座

当你访问网站时，到底在访问什么？



域名 → IP的别称

网页 → 服务器上的文件

1. 通过DNS查找获得域名对应IP（也就是服务器的IP）
2. 向这个IP地址发起请求，获得网页文件
3. 浏览器渲染网页文件

当你打开一个网页时

浏览器

前端

Url解析

Uri的标准与含义

DNS Lookup

找到域名对应的IP地址

发送请求

一个HTTP请求该有的样子

服务器

后端

解析&处理请求

把请求“路由”到该去的地方

后端逻辑

处理你的请求，访问数据库，生成响应

发送响应

一个HTTP响应都说了些什么

渲染网页

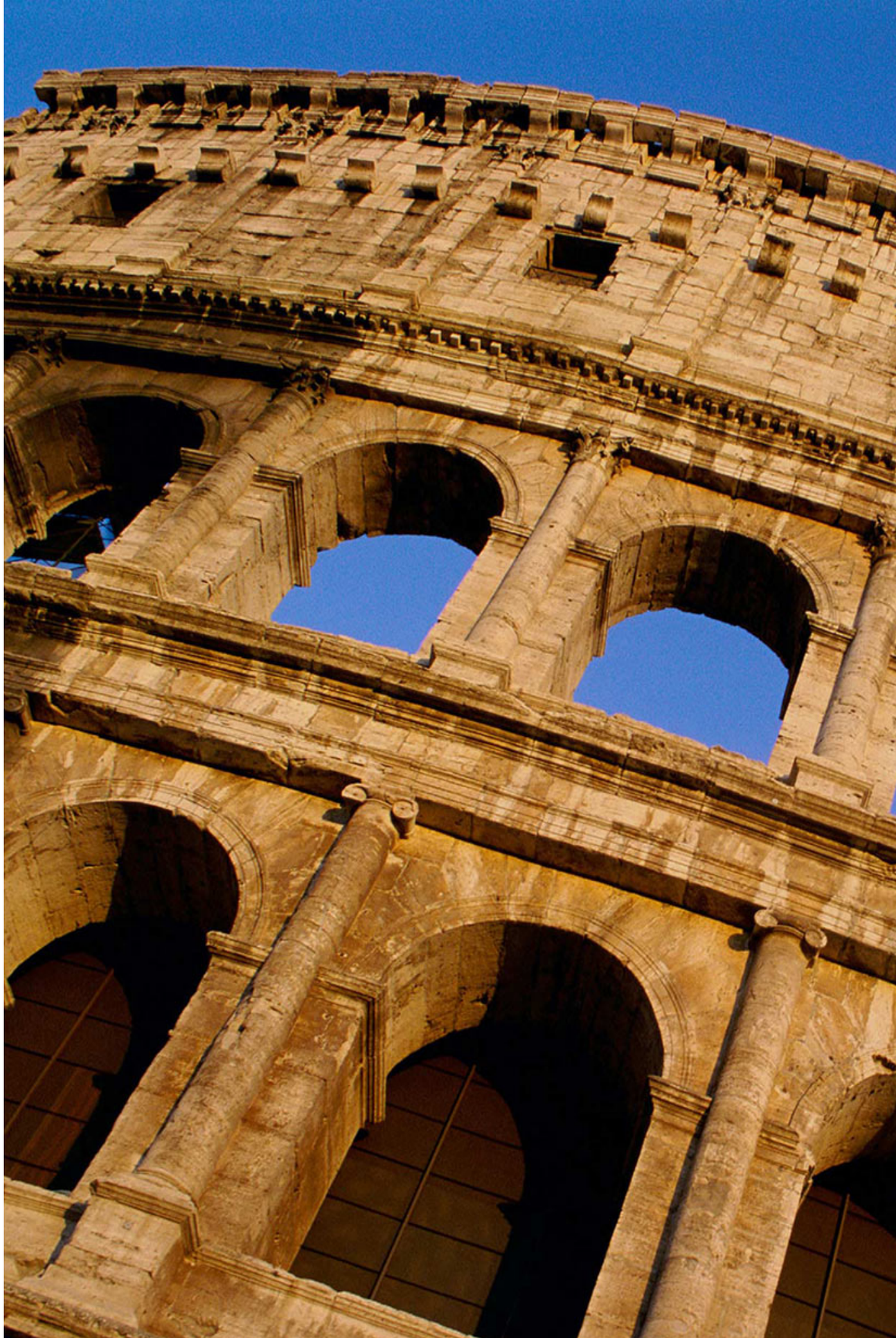
把网页源码“画”出来

不止于此

网页逻辑/更多的请求…

Step 1:
解析URL

1. URL是什么?
2. URL长什么样?
3. URL包含了哪些信息?



URL是什么？

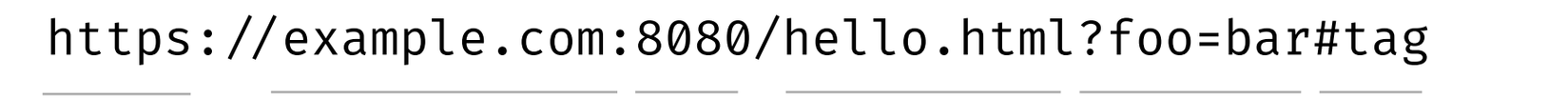
URL的全称是Uniform Resource Locator，也就是我们常说的“网址”。

1. 指向了一个网络资源的位置
2. 说明了怎样获取这个资源

```
https://www.baidu.com
```

URL的组成

一个典型的URL：



`scheme` 协议

获取这个资源的“方式”，也就是传输协议(protocol)。一般来说，在浏览器中访问的网页都会使用http或https协议。

`domain` 域名

网站的“名字”。标示一个网站的标识符，与IP地址对应，能够通过DNS查找得到域名对应的IP地址。

`port` 端口

服务器的端口。若无指定，http协议默认访问80，https协议默认访问443。

URL的组成

一个典型的URL：

```
https://example.com:8080/hello.html?foo=bar#tag
```

scheme　domain　port　path　query　fragment

`path`　路径

希望访问的网页路径。一个网站可能有多个网页组成，通过不同路径标识。

`query`　查询字符串

可以被理解为希望提供给服务器的“额外信息”。
例子：网页语言。`https://example.com/?lang=zh-CN`

URL的组成

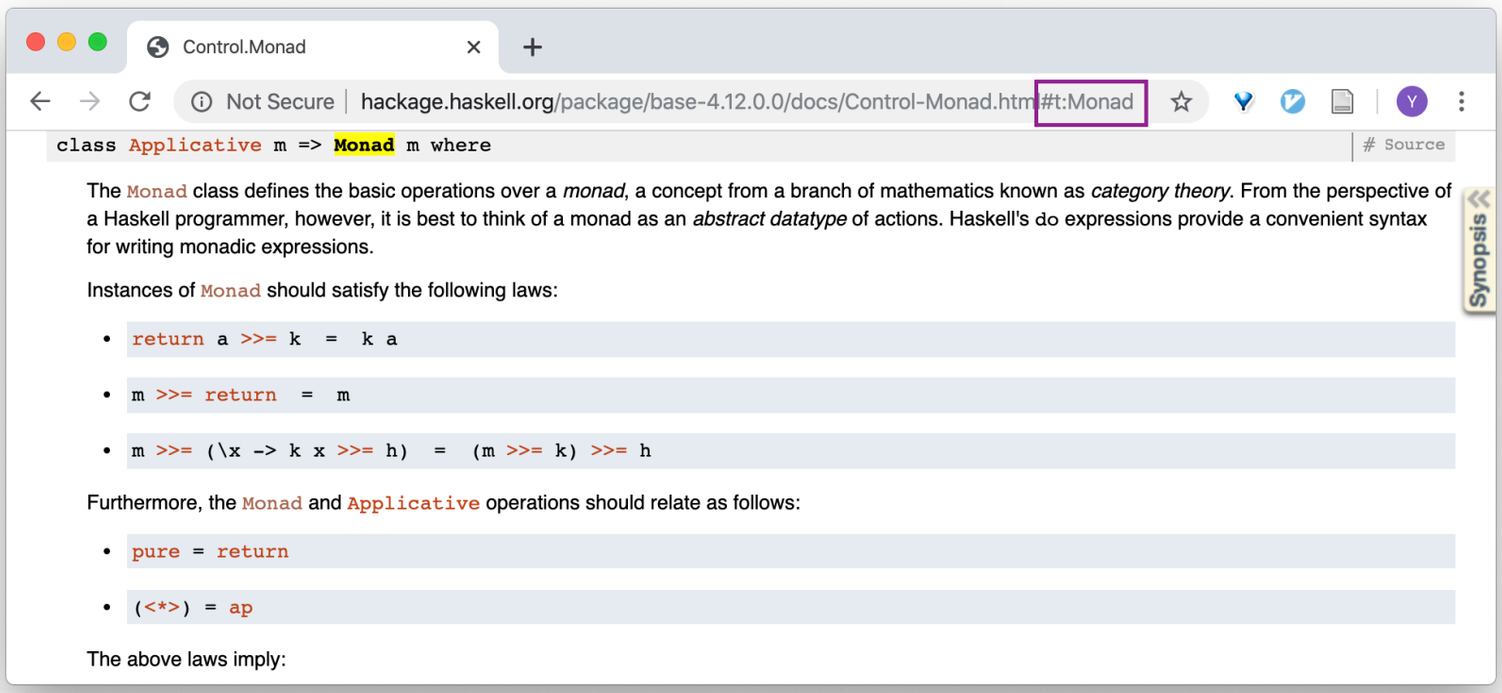
一个典型的URL：

```
https://example.com:8080/hello.html?foo=bar#tag
```

```
scheme      domain    port     path      query   fragment
```

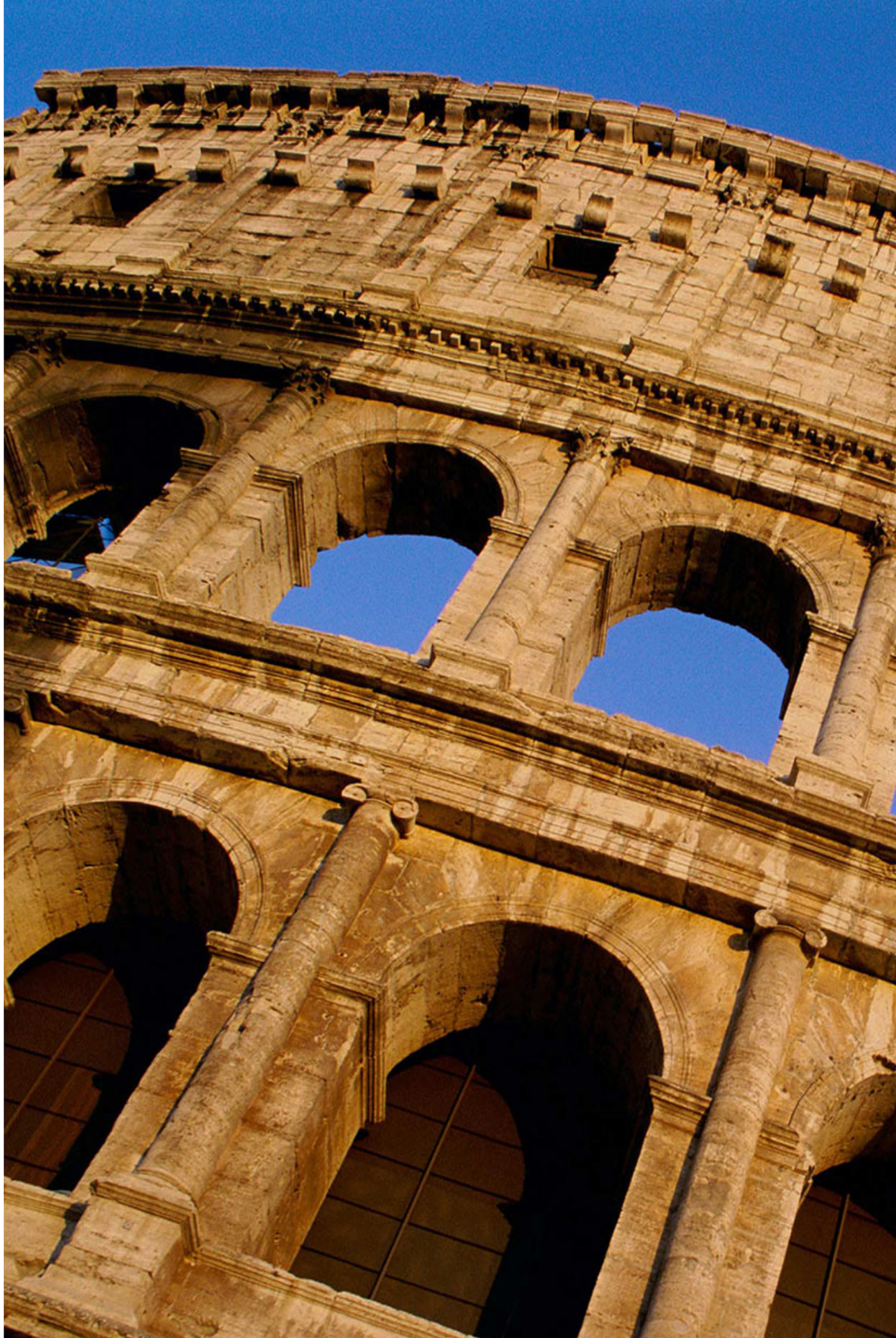
`fragment` 标签

更精细地标识网页中的内容。



Step 2:
DNS Lookup

1. DNS是什么?
2. DNS查询原理
3. IPv4与IPv6



DNS是什么?

全称：Domain Name System

一项服务：
返回查询域名的IP地址。

一个分布式数据库：
保存着全世界域名与IP地址的映射信息。

存在意义：
TCP通信离不开IP地址。
IPv4: 185.199.108.153
IPv6: 2001:da8:215:8f01:4980:62dc:a2aa:fb5e
域名: duckduckgo.com

DNS的工作原理

存储结构：树状

根域名服务器

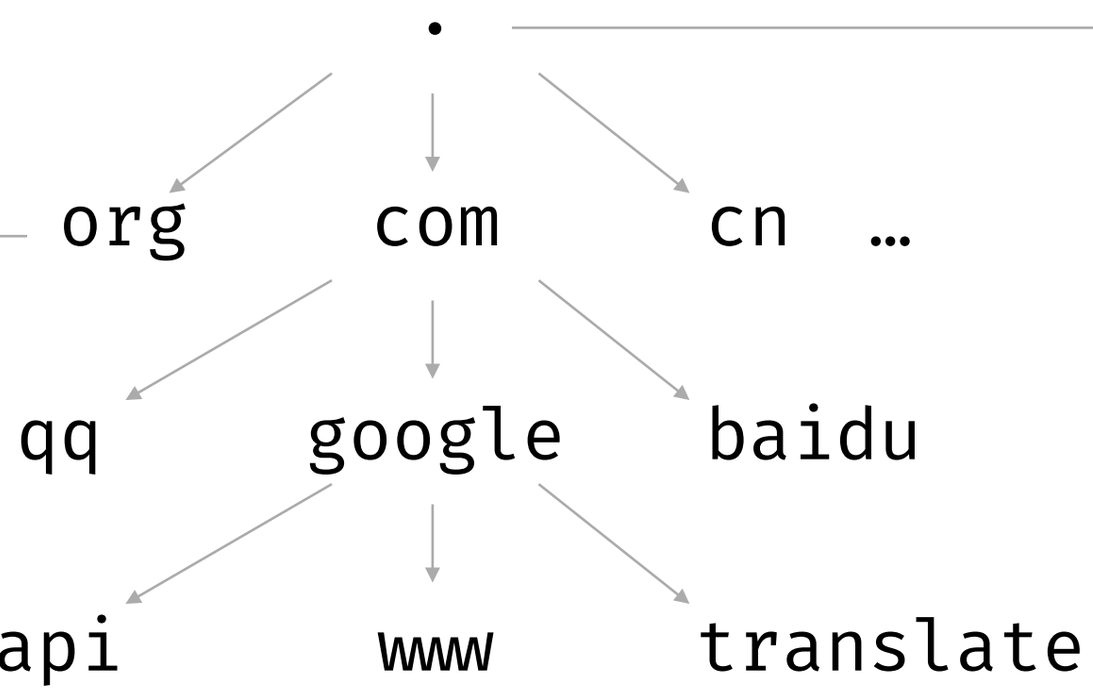
保存了所有顶级域名服务器地址

顶级域名服务器

保存了各个顶级域名的次级域名服务器地址

次级域名服务器

保存了域名地址或相应下一级域名服务器的地址

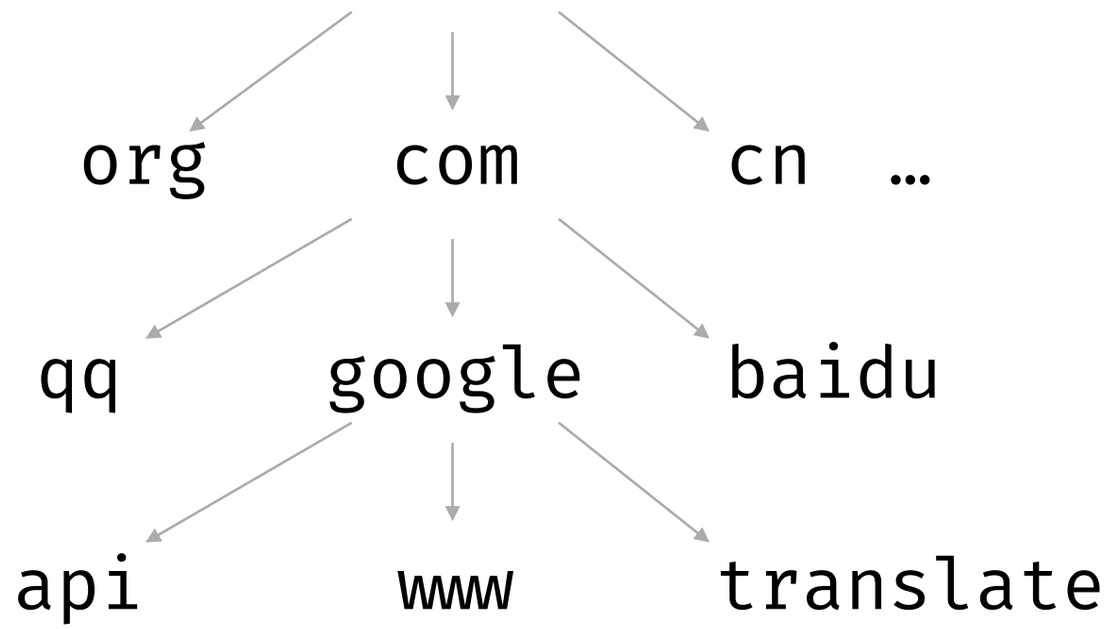


查询域名的过程也就是沿着层级下降直到查询到域名信息记录的过程。

DNS的工作原理

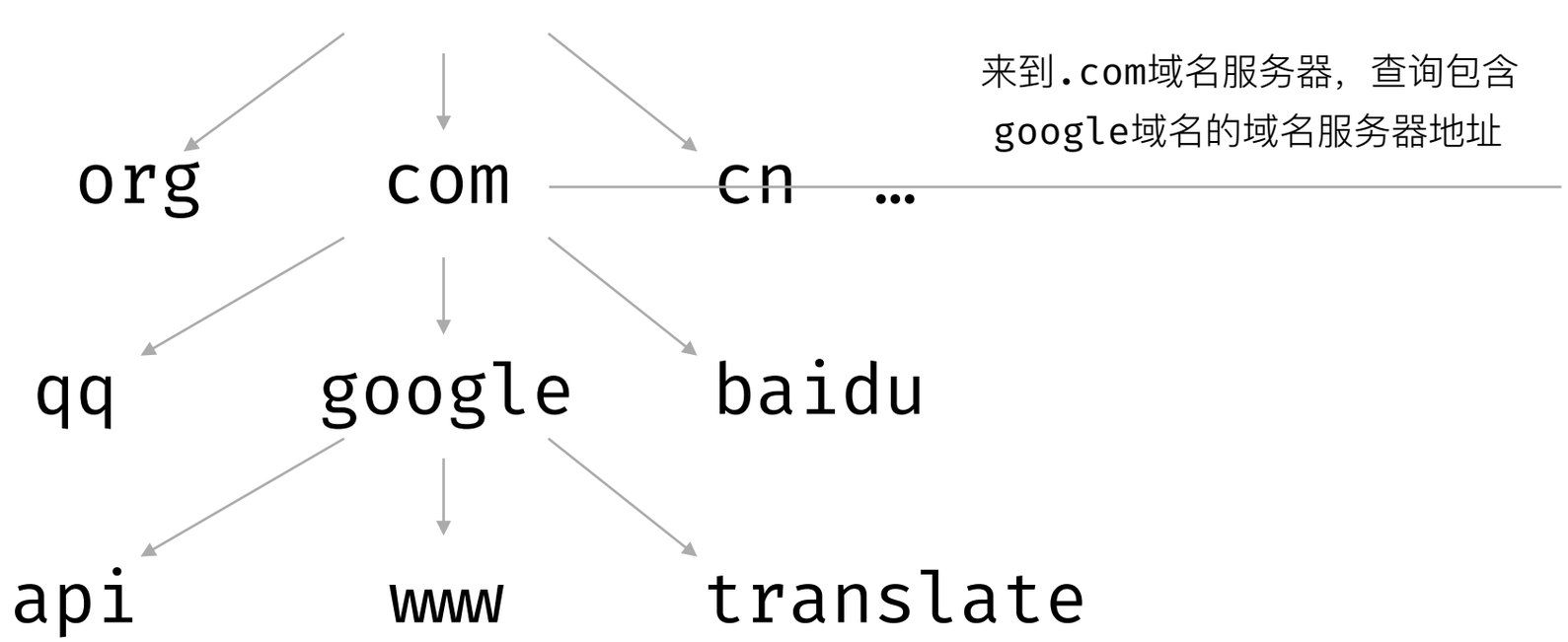
例子：查询www.google.com

查询根域名服务器，得到.com对应顶级域名服务器地址



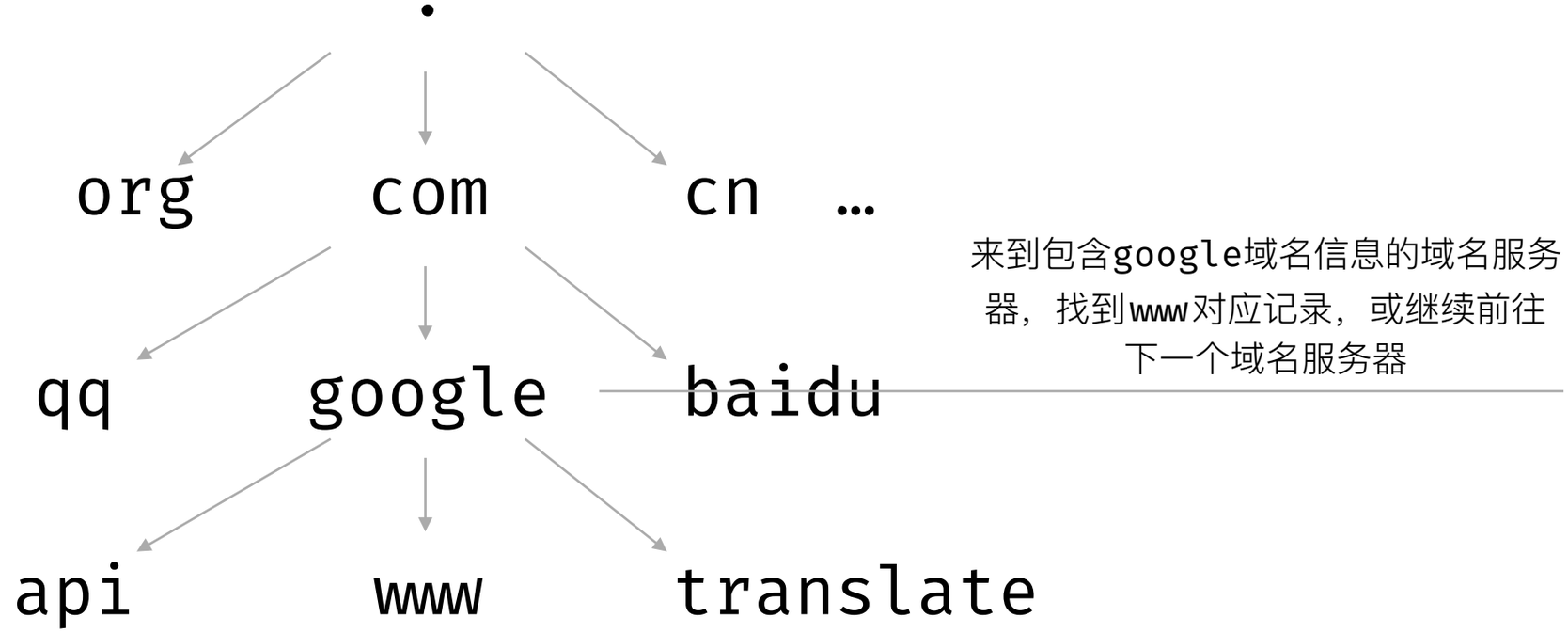
DNS的工作原理

例子：查询www.google.com



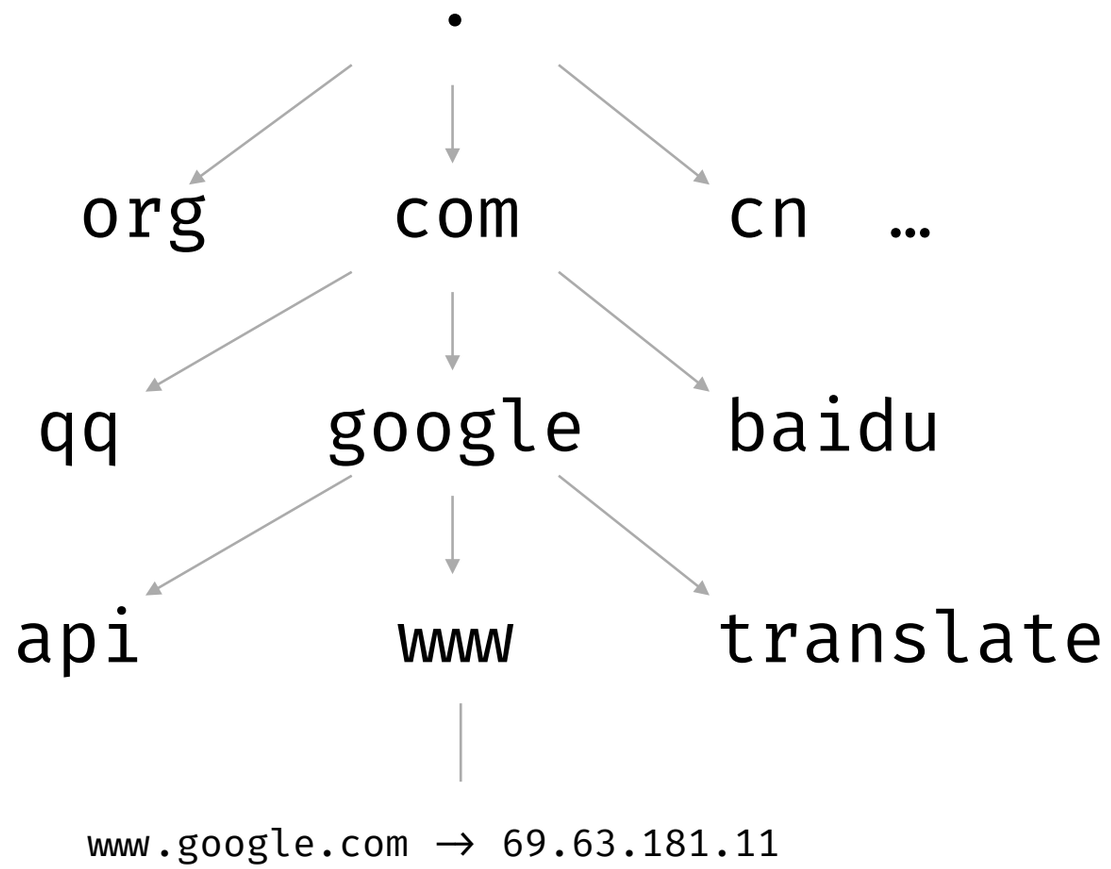
DNS的工作原理

例子：查询www.google.com



DNS的工作原理

例子：查询www.google.com



DNS的工作原理

常见域名记录的种类

A	地址记录 也即域名指向的IP地址
NS	域名服务器记录 包含该域名信息的域名服务器的地址
CNAME	别名记录 值是另一个域名，表示原域名对应的IP地址即为另一个域名对应的IP地址
MX	邮件记录 表示该域名邮件服务器的地址

DNS的工作原理

DNS缓存

为什么要有缓存?
避免每次都要进行追溯到根服务器的DNS查找，加快访问速度

缓存的种类
1. 本地缓存
    1. 浏览器缓存
    2. 系统缓存
2. DNS服务器缓存

域名记录的TTL
每个域名记录都会附加一个TTL(time to live)值，表示最长缓存时间。
超出这个时间后，有新的查询时，应重新查找获取最新的记录信息。

IPv4与IPv6

互联网 IPv4 计算机网络 计算机科学 IP 地址

如何看待 2019 年 11 月 26 日 IPV4 地址耗尽？会产生什么样的影响?

IP(Internet Protocol)

事实上是一种协议（规范），但当我们说起IP时，大多指的都是这个协议中最核心的元素：在网络通信中标识地址的唯一标识符号。可以理解为：邮编 | 门牌号。

IPv4与IPv6

IPv4

一个32位的二进制串，用作地址标识符。

```
192.168.1.1, 8.8.8.8
```

总量：2^32 = 4,294,967,296个

公网地址总量：3,706,452,992个

IPv6

一个128位的二进制串，用作地址标识符。

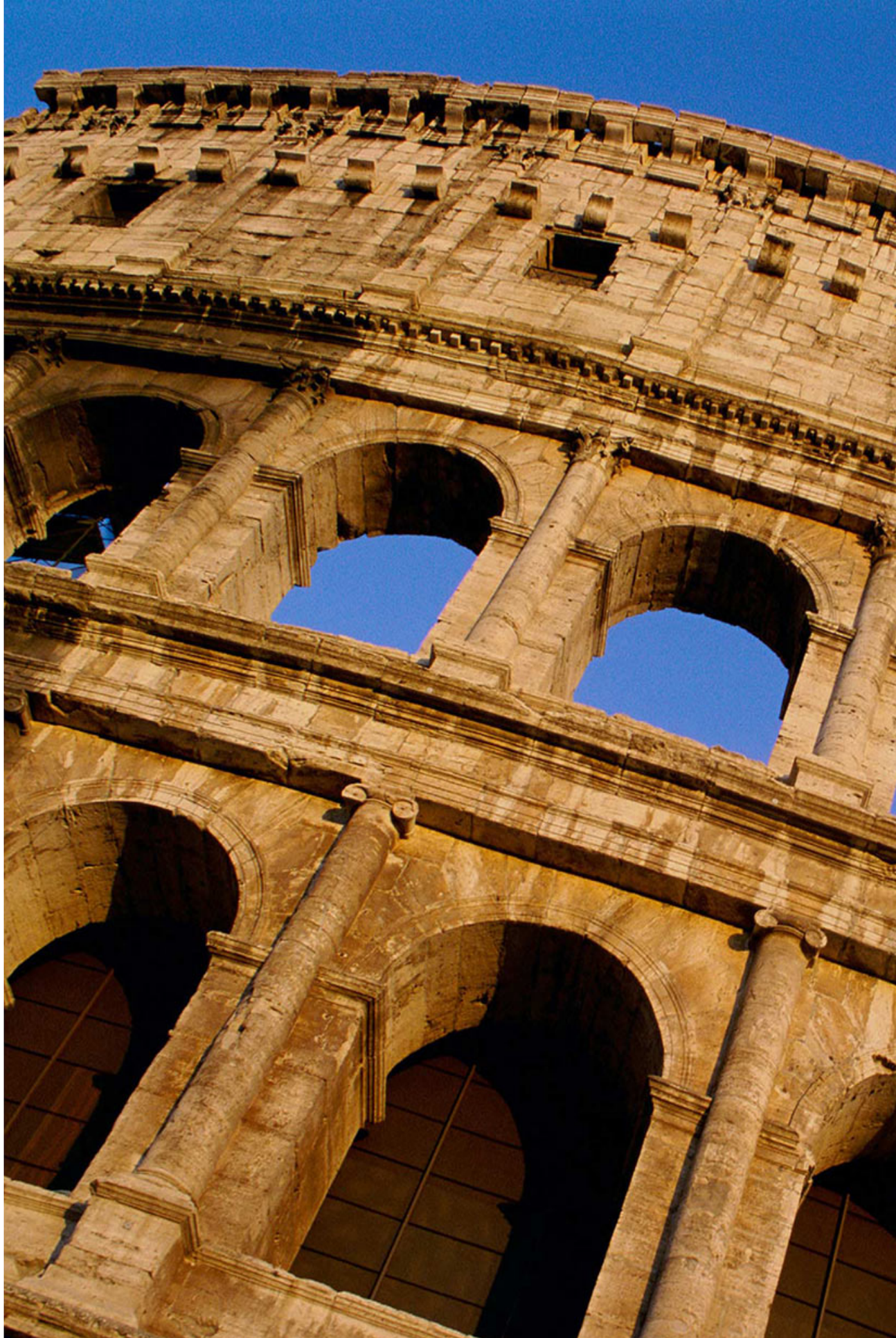
```
2001:0db8:85a3:08d3:1319:8a2e:0370:7344,
a2:b8:12:1, ::1
```

总量：2^32 = 340282366920938463463374607431768211456个

为什么回家了就上不了北邮人BT了?

Step 3:
发送请求

1. HTTP协议
2. HTTP请求的格式与内容



什么是HTTP协议?

HTTP协议，全称HyperText Transfer Protocol，也即超文本传输协议。广泛用于客户端-服务器之间的通信，可以说是互联网的基石之一。

HTTP协议的大概框架：客户端向服务端发送一个约定格式的请求(Request)，在解析处理请求后，服务器将一个约定格式的响应(Response)返回给客户端。这样就完成了一次HTTP通信。

HTTP协议是无状态的。这意味着架构于HTTP的通信双方并不依赖于对方的状态。这极大提高了通信的稳定性与易用性。

HTTPS是引入了TLS(Transfer Layer Security)的HTTP协议。可以被简单理解为HTTPS协议的加密版本。

HTTP请求

一个典型的HTTP请求

起始行

包含了请求方法、请求路径、请求协议版本

```
GET / HTTP/1.1
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108
Safari/537.36
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Cookie: foo=bar;hello=world;

key1=value1&key2=value2
```

HTTP请求

一个典型的HTTP请求

```
GET / HTTP/1.1
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108
Safari/537.36
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Cookie: foo=bar;hello=world;

key1=value1&key2=value2
```

请求头(Request header)

从第二行开始，每一行都以

`Header-Name: Header-Value`

形式给出请求头部。

空行

请求头部结束，留一空行。

HTTP请求

一个典型的HTTP请求

```
GET / HTTP/1.1
Host: www.baidu.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108
Safari/537.36
Accept: text/html,application/xhtml+xml
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
Cookie: foo=bar;hello=world;

key1=value1&key2=value2
```

请求数据

空行后剩下部分，为请求包含的数据，可以为空。

HTTP请求

起始行

```
GET / HTTP/1.1
```

```
method path protocol
```

method 方法

请求的方法。也即要对资源进行的操作。

GET	获取资源。访问网页时，浏览器向服务器发出的一般都是GET请求。
POST	上传资源。在提交表单，上传文件时，都会发出POST请求。
PUT	修改/上传资源。有时上传文件也会使用PUT请求。
DELETE	删除资源。
HEAD	获得资源的响应头。多用于文件下载时，提前知道文件大小。
OPTIONS	获得资源的通信选项。服务器会返回允许的请求方法，请求头部等。

HTTP请求

起始行

```
GET / HTTP/1.1
```

method path protocol

`path` 路径

也即之前提到的URL中包含的资源路径。

`protocol` 协议类型

指明通信所用的HTTP协议版本。

HTTP请求

请求头

请求头存在的意义是向服务器提供请求的主要信息。如连接方式，用户代理，或是接受的数据类型。

常用请求头字段

Host

也即请求URL中的域名。多应用于主机托管中的场景。（一台服务器要处理针对多个域名的请求）

```
Host: www.baidu.com
```

Accept

指明接受的数据类型。

```
Accept: text/html,application/xhtml+xml
```

User-Agent

用户代理类型。也即指明访问服务器的是何种软件。对于用户而言这个子段一般是浏览器，也可能是搜索引擎的爬虫或一些自动化测试工具。

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_1)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108
Safari/537.36
```

HTTP请求

常用请求头字段

`Connection` 与服务器建立连接的类型。常见的是`keep-alive`，是HTTP/1.1后引入的特性。也即复用客户端与服务器之间的TCP连接，来免去每次HTTP通信都要重新建立TCP连接的不便。

```
Connection: keep-alive
```

`Cookie` Cookie是服务器要求客户端保存的一段数据，并在以后的请求中客户端会把这些数据放在请求头中发送给服务器。Cookie充当了无状态的HTTP中的“状态”。能够帮助服务器识别客户端，保留登录信息，个性化设置等等。

```
Cookie: foo=bar;hello=world;
```

`Authorization` 用户凭证信息，帮助服务器鉴权，或是标示用户身份。
常用策略：Basic Authorization，将包含用户凭证的字符串`"username:password"`用Base64编码后放入头部。

```
Authorization: Basic QWxhZGRpbjpPcGVuU2VzYW1l
```

`Content-Type` 标识请求数据的格式。如POST常用的`x-www-form-urlencoded`。

```
Content-Type: application/x-www-form-urlencoded
```

HTTP请求

请求数据

包含在请求中的数据。若为POST请求，可能为表单信息。也可能是上传的文件，JSON数据等。

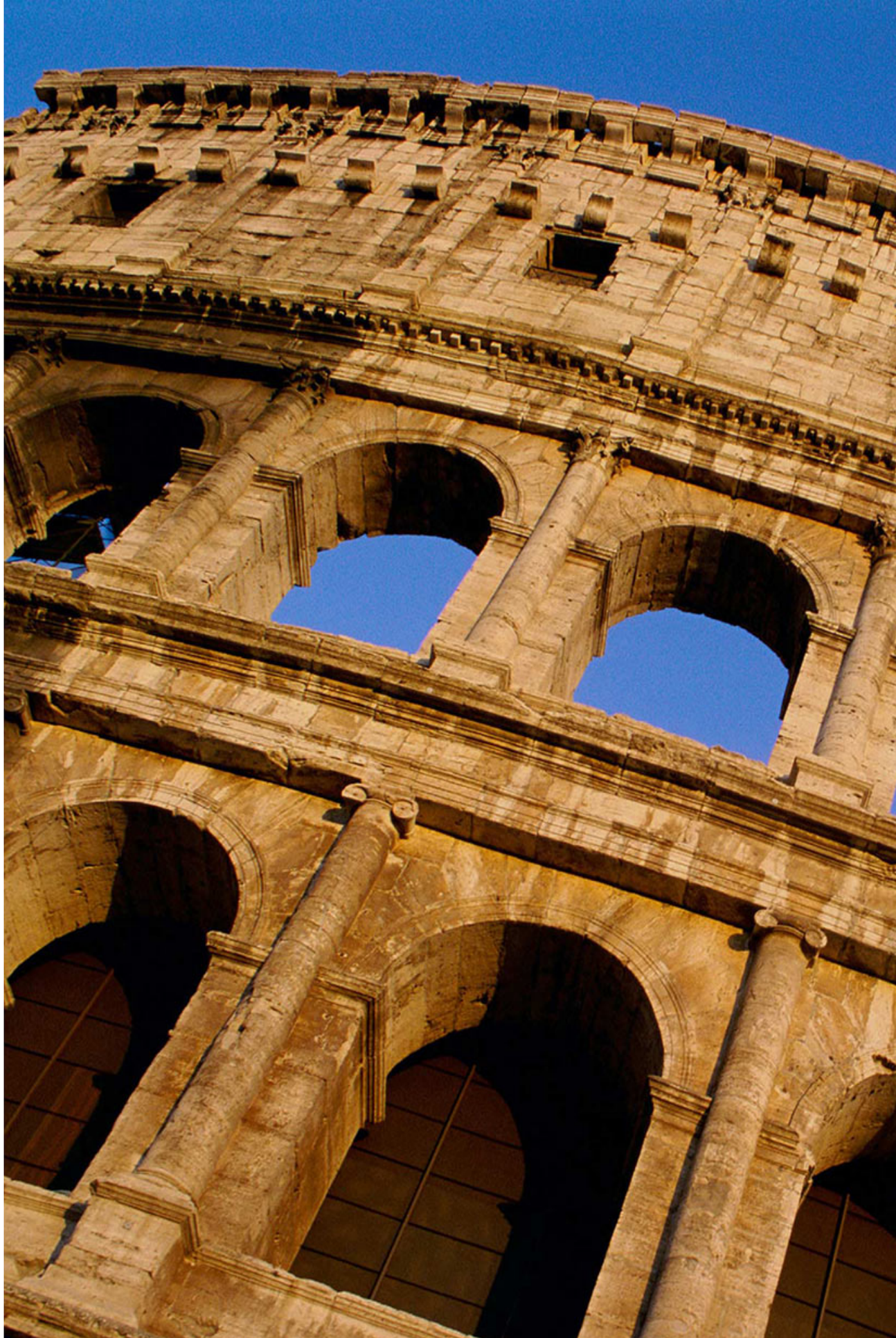
例子：若Content-Type为x-www-form-urlencoded，请求数据可能为

```
key1=value1&key2=value2
```

也即用&连接的键值对。

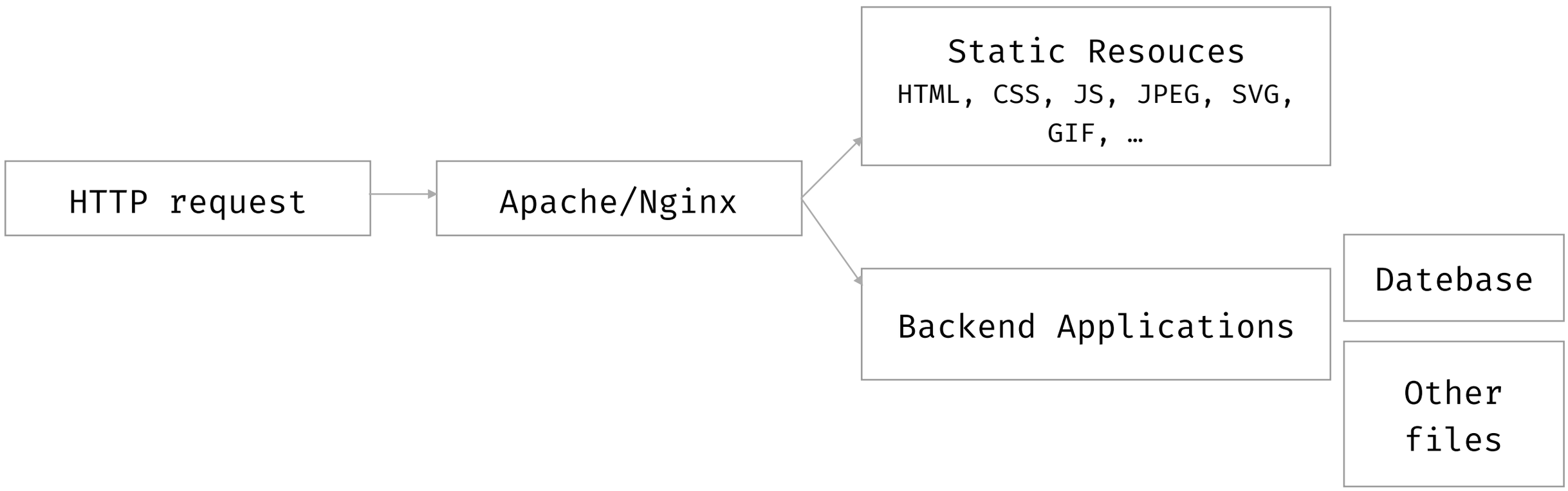
Step 4:
服务器处理请求

1. 路由
2. 负载均衡



后端概览

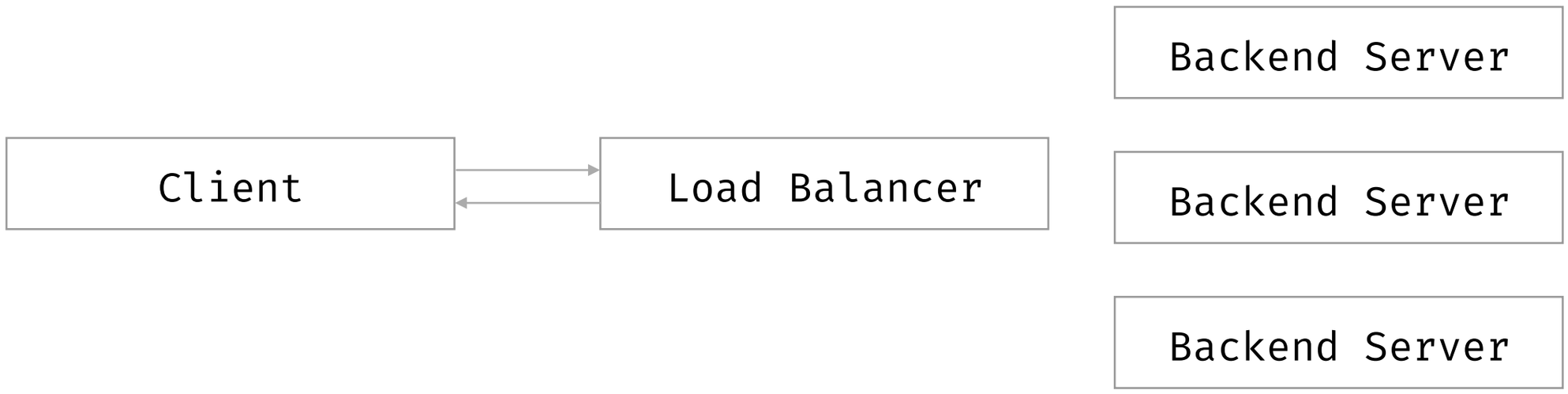
服务器往往不仅存储着静态的文件，还有拥有许多功能（API）。
因而服务器中往往运行着Apache、Nginx这样的软件，负责处理一些静态文件的简单请求，并能把请求路由到其他后端程序中，并返回他们的执行响应。
而由如PHP、C++、Go实现的后端程序，会根据用户请求，访问数据库，执行必要的操作，将数据返回给用户，以JSON形式，或者渲染为HTML网页。



负载均衡

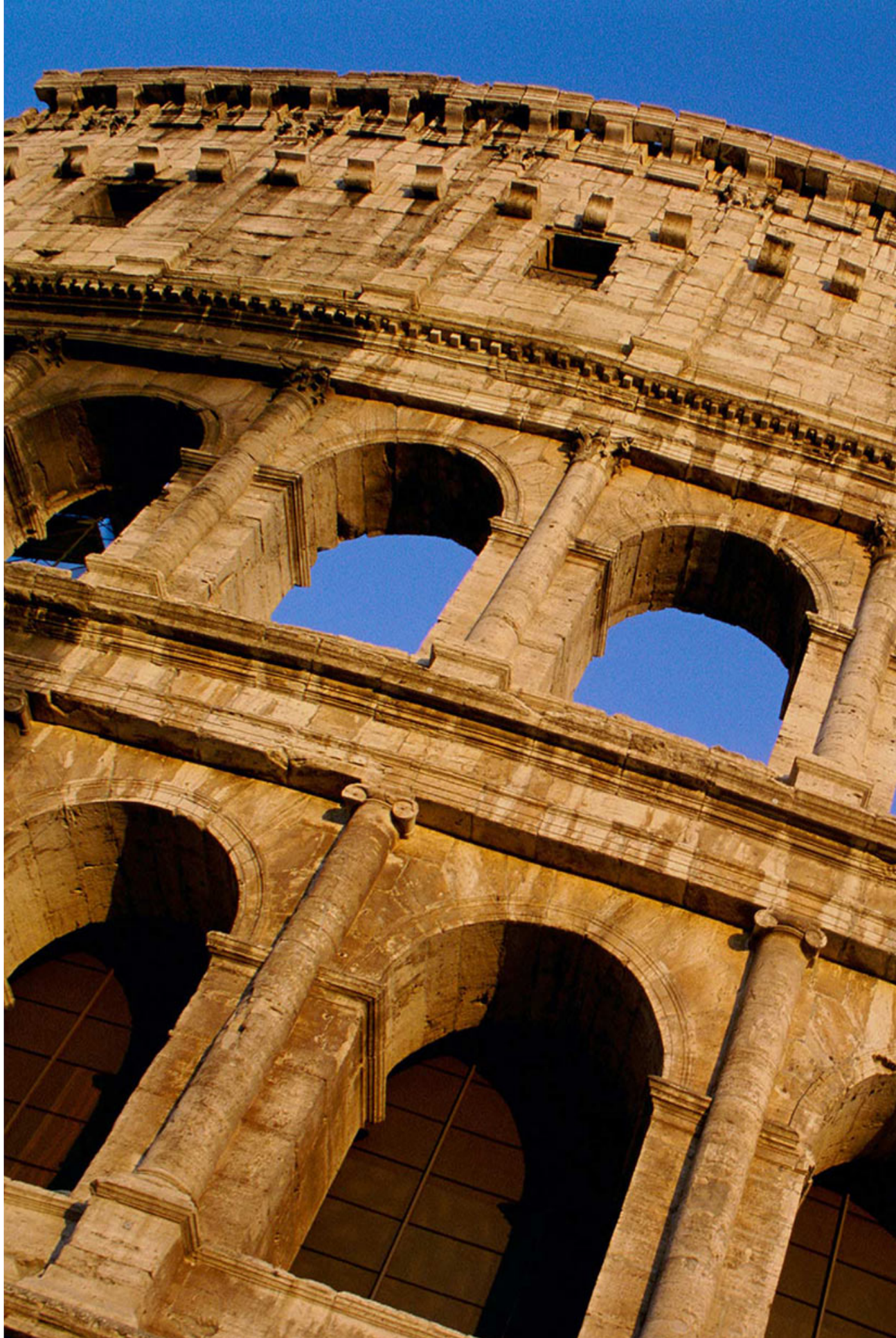
大型网站中，为了提高网站可用性，常常会部署负载均衡(load balancer)。

简单来说，负载均衡就是将网络流量分配给多个服务器，以平衡各个服务器的负载。



Step 5:
执行后端程序，
返回响应

1. 后端程序会做什么
2. HTTP响应



后端程序

通过一个例子来看一看后端程序会做哪些事情

例：访问Wordpress中的一篇文章

后端程序

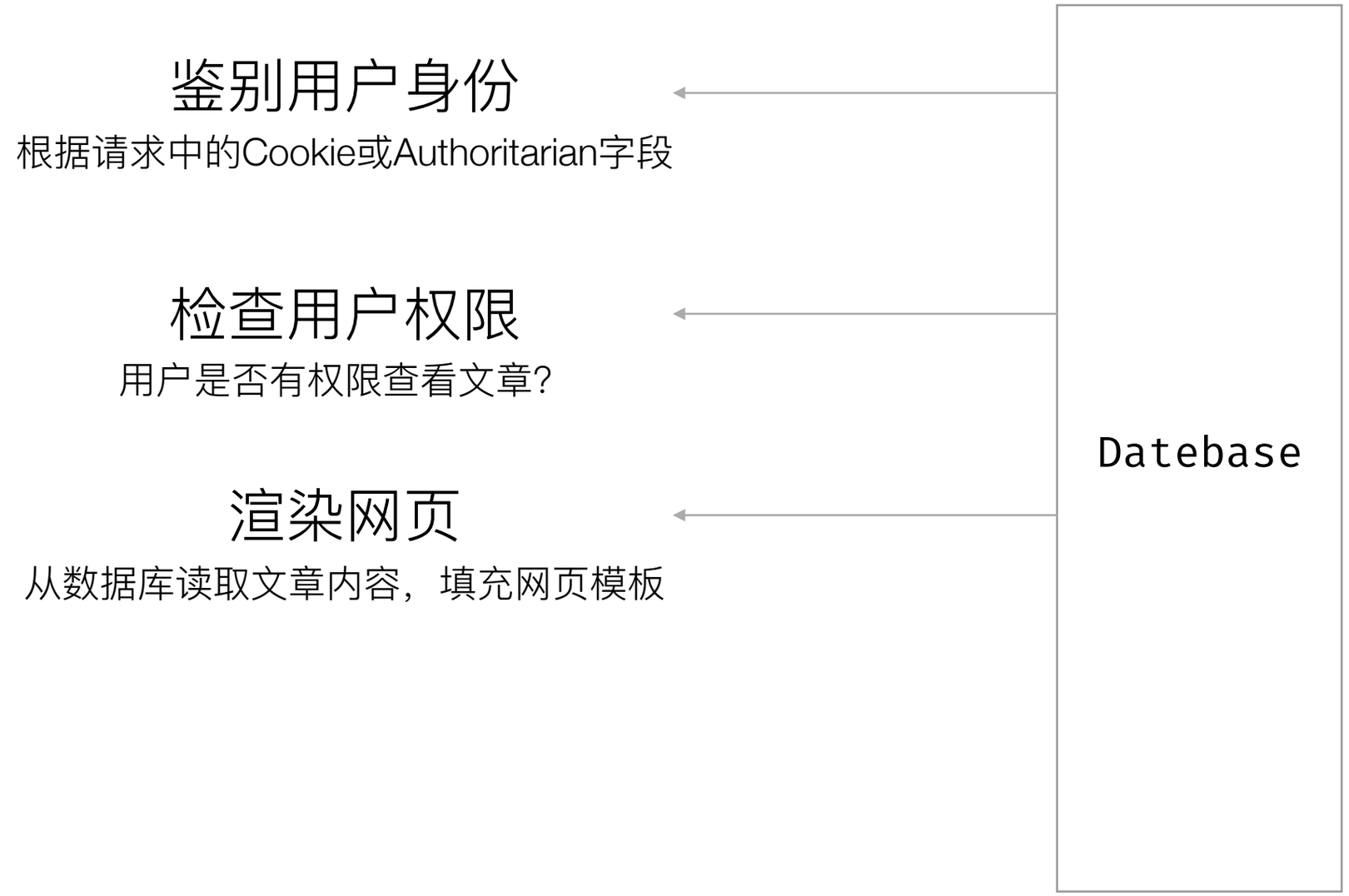
通过一个例子来看一看后端程序会做哪些事情

例：访问Wordpress中的一篇文章

后端程序

通过一个例子来看一看后端程序会做哪些事情

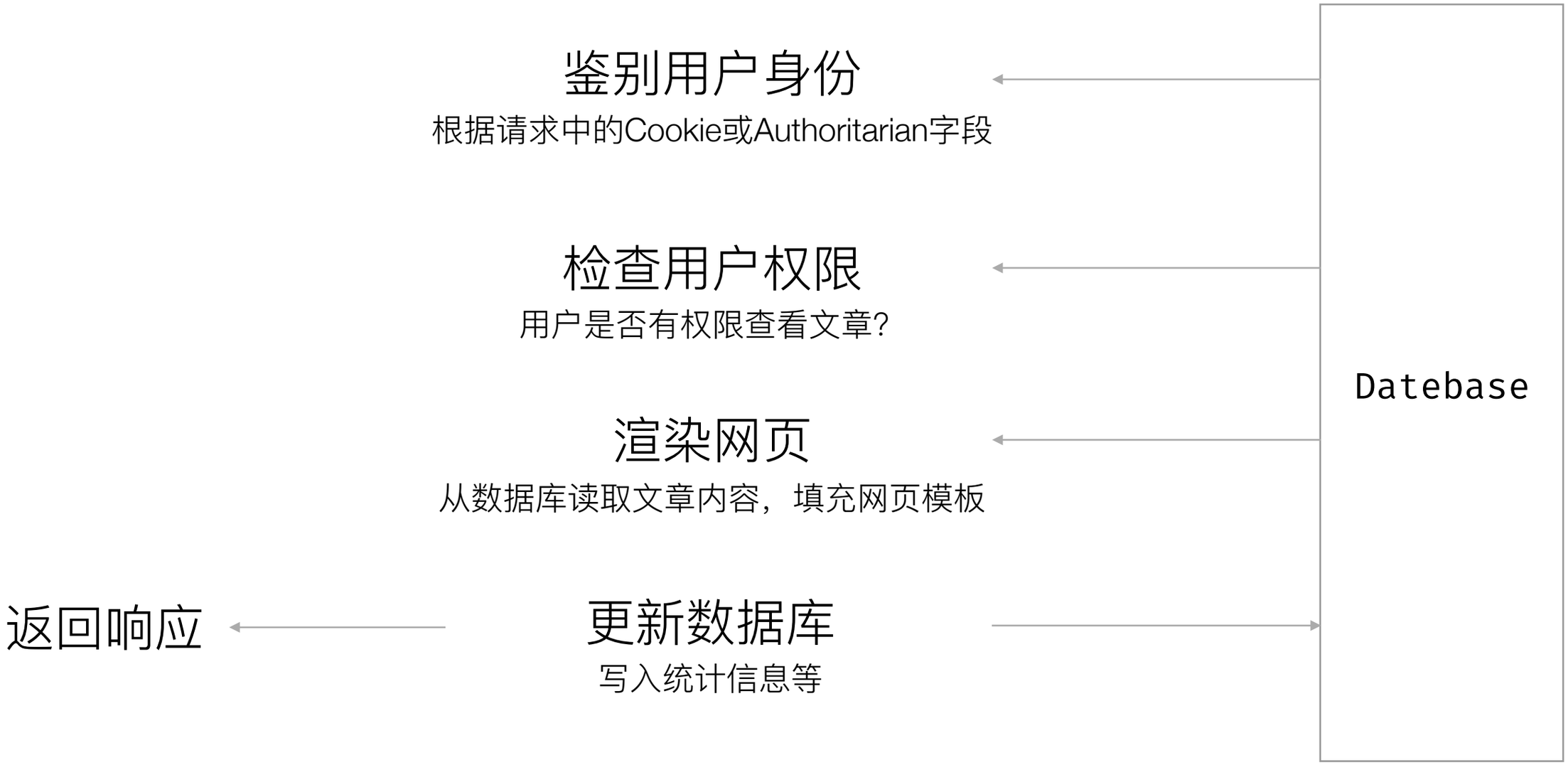
例：访问Wordpress中的一篇文章



后端程序

通过一个例子来看一看后端程序会做哪些事情

例：访问Wordpress中的一篇文章



HTTP响应

起始行

HTTP响应的起始行包含了HTTP协议类型、响应状态

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html;charset=utf-8
Date: Thu, 28 Nov 2019 13:53:58 GMT
Expires: Thu, 28 Nov 2019 13:53:58 GMT
Server: Apache Set-Cookie: foo=bar; hello=world;
X-Ua-Compatible: IE=Edge,chrome=1
Transfer-Encoding: chunked

// body
```

HTTP响应

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html;charset=utf-8
Date: Thu, 28 Nov 2019 13:53:58 GMT
Expires: Thu, 28 Nov 2019 13:53:58 GMT
Server: Apache Set-Cookie: foo=bar; hello=world;
X-Ua-Compatible: IE=Edge,chrome=1
Transfer-Encoding: chunked

// body
```

响应头(Response header)

与请求头部格式一致，除字段类型有所不同

HTTP响应

常见状态代码与含义

200 OK

请求被成功地完成，可能是成功返回了资源或数据，也可能是表单提交成功。

201 Created

资源被成功创建

301 Moved Permanently

表示所请求的资源被永久转移到了新的位置。可用于重定向。

HTTP响应

常见状态代码与含义

`400 Bad Request`

服务器无法理解请求，或请求有语义错误。

`401 Unauthorized`

表示请求为提供凭证。

`403 Forbidden`

标示客户端无权访问资源。

`404 Not Found`

服务器找不到请求的资源。

HTTP响应

常见状态代码与含义

500 Internal Server Error

服务器发生错误，很多时候是后端代码抛出异常。

502 Bad Gateway

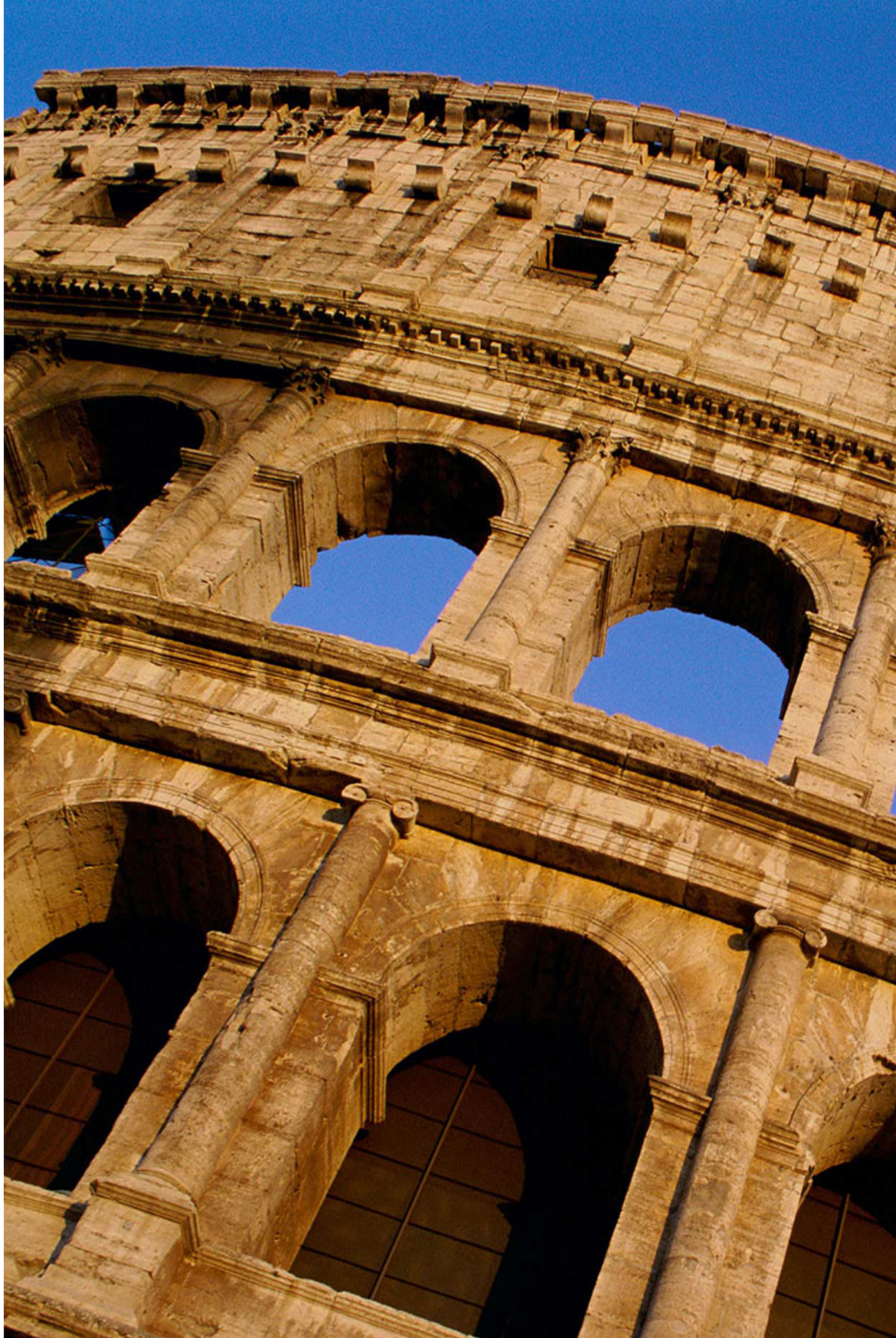
这意味着服务器作为网关时，无法获取有效的响应。如Nginx配置路由时写入了无效的路由。

503 Service Unreachable

表示服务器暂时无法处理请求。如服务器在维护时可能会返回此状态码。

Step 6:
浏览器渲染网页

1. 网页文件概览
2. 网页渲染框架



网页文件

是什么构成了我们看到的网页?

HTML

描述了网页的大概外观，各个元件的位置与嵌套关系，构成了网页的骨架。

CSS

描述了网页各个元件更为精细的外观。

JavaScript

实现了网页逻辑。（在按钮被按下时要做些什么？）

网页文件

HTML

全称为HyperText Markup Language，即超文本标记语言。

HTML是一种标识语言，用有层次嵌套关系的元素节点来表示各类信息。

HTML包含的主要信息：

1. 网页的各类元信息(meta data)
2. 网页需要引入的外部文件，如CSS与JavaScript
3. 网页中各组件的基本架构

网页文件

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta name="viewport" content="width=device-width,initial-width=1.0,min-width=1.0,max-width=1.0">
  <meta charset="UTF-8">
  <title>Hello world</title>
</head>
<body>
  <header>
    <div class="nav">
      <img src="images/navbar.png" alt="Nav" class="nav-img" id="nav">
    </div>
  </header>
  <div class="main">
    <p>Hello, world!</p>
  </div>
</body>
</html>
```

HTML的头部

包含各类元信息，也可引入外部CSS样式表

HTML元素

对应DOM树中的节点，可以被渲染成网页元素

网页文件

CSS

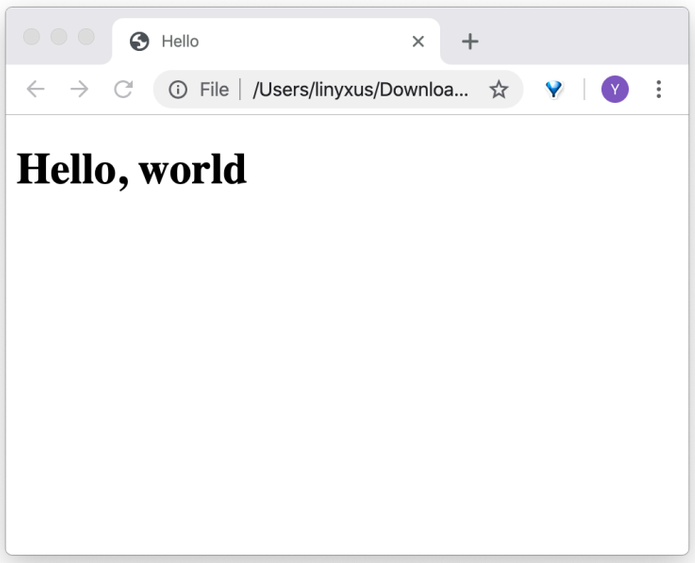
全称为Cascading Style Sheets，也即样式表。
能够定义样式，应用到HTML元素上，精细调整元素外观。

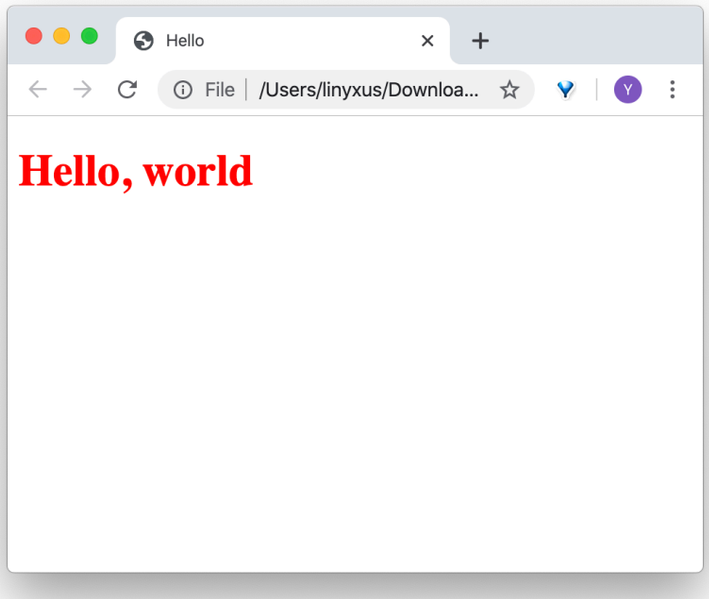
定义样式

```
.red-heading {
    color: red;
}
```

应用至元素

```
<h1 class=“red-heading">
  Hello, world
</h1>
```





网页文件

JavaScript

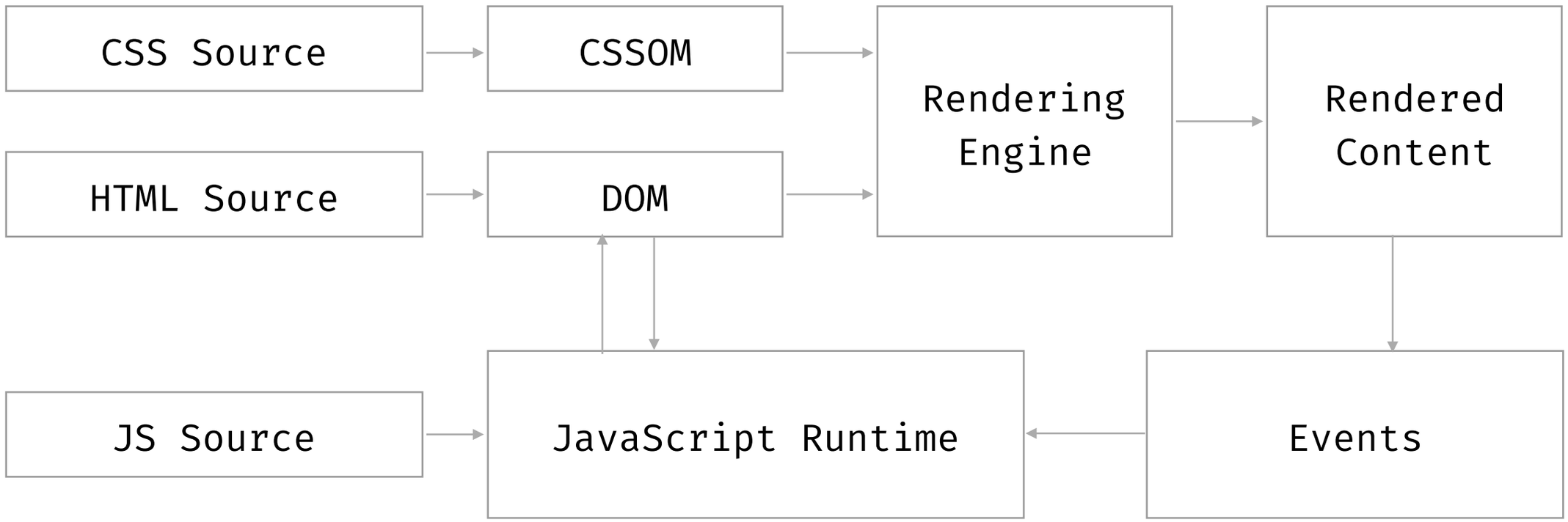
运行在浏览器中的脚本语言，能通过它实现网页逻辑。

JS能够...
1. 操作网页元素（与DOM树交互）
2. 读取本地信息（LocalStorage，Cookie）
3. 发起异步请求（Ajax）
4. ...

不同浏览器有不同的JavaScript实现。
Chrome: V8 Engine
Firefox: SpiderMonkey
Edge (old version): Chakra

而事实上，JavaScript是当今世界上最流行的语言之一，大量语言都能被编译为JavaScript而运行在浏览器中（ClojureScript是一个很流行的例子），而JavaScript也不仅能运行在浏览器中，而能借助Node.js运行在几乎所有桌面平台上，成为实现后端程序的主要技术之一。

网页渲染



至此，从点击链接到完成渲染，我们浏览的网页便呈现我们眼前了

谢谢大家！

徐逸辰

yichen.x@outlook.com

感谢王泽坤同学在讲座准备过程中给予的帮助