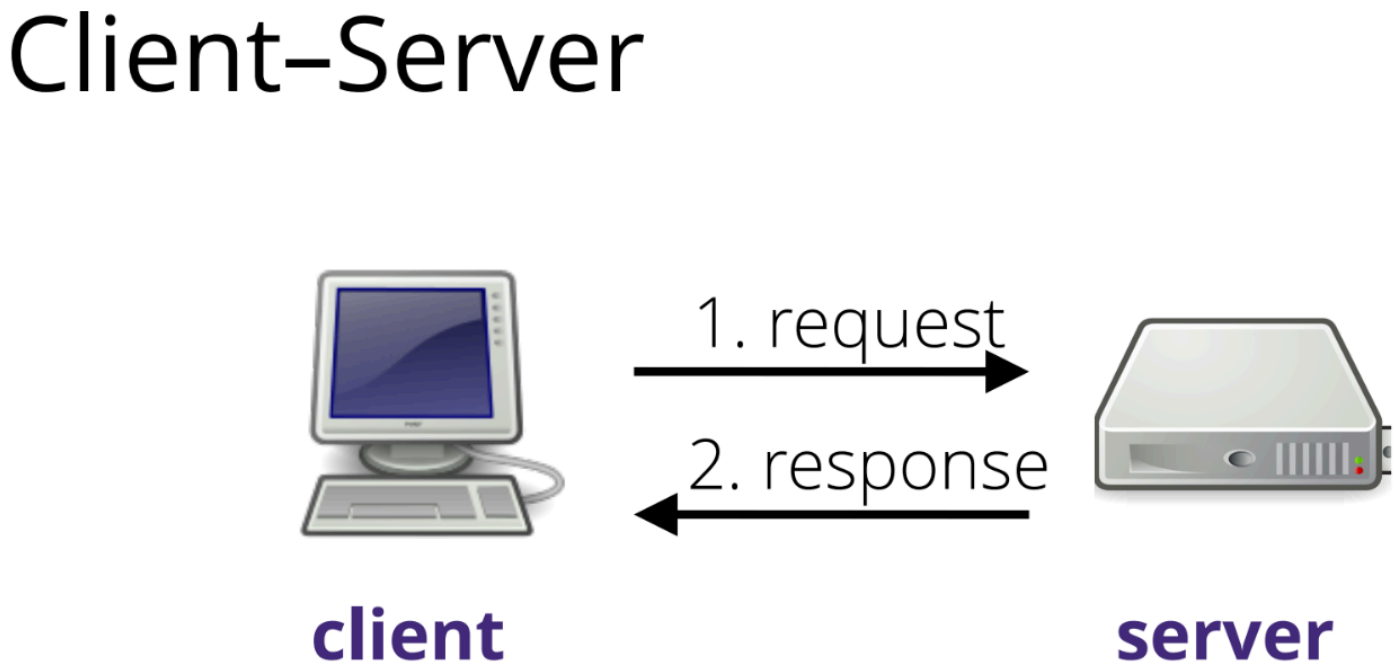
Week6: The Web

6.1 HTTP

HTTP

1. HTTP中的P就是protocal，协议，主要是由client和server之间的互动

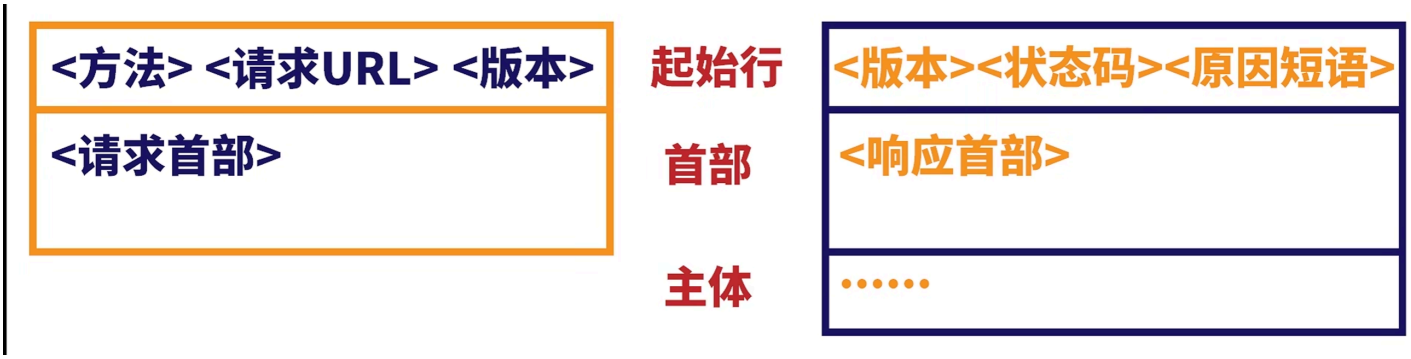


请求报文就是：Header
Client Header

```
GET /index.html HTTP/1.1
Host: www.bristol.ac.uk
Connection: close
```

Server Header

```
HTTP/1.1 200 OK
Content-Type: text/html;
charset=UTF-8
Content-Length: 1009

<!DOCTYPE html>
<html lang="en">
...
```



- Methods有 GET (提取信息) HEAD POST (传输信息) PUT DELETE
- Response codes:

1xx: information

2xx: success (200 OK, 201 Created, ...)

3xx: redirect (301 Moved Permanently, ...)

4xx: client error (400 Bad Request,
403 Forbidden, 404 Not Found, ...)

5xx: server error (500 Internal Server Error ...)

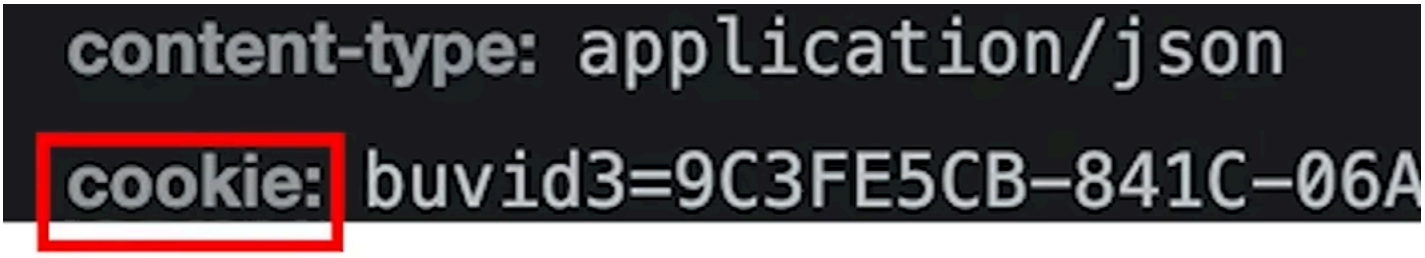
- Content Type:

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


text/plain, text/html, ...
image/jpeg, application/pdf, video/mp4 ...
```

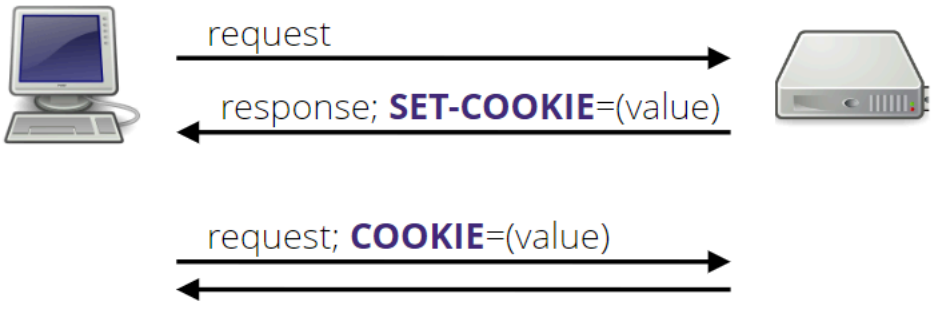
First priority for the browser!

2. Cookies

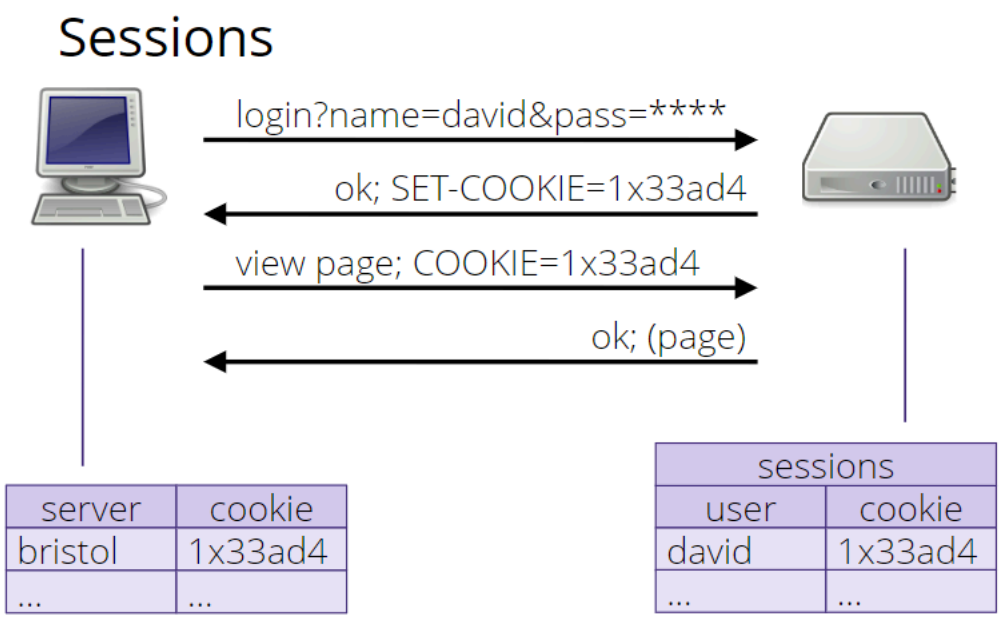
一开始http的设计是无状态的，但是后来有了用户登录，为了保持登陆状态才有了cookies



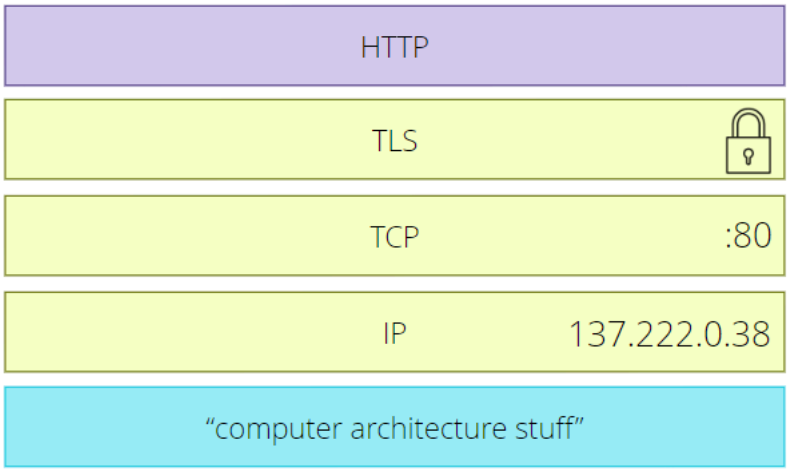
Cookie protocol



(RFC 6265)



Internet



从上往下依次是 应用层、传输层、网络层、硬件接口

一个常见的网络栈结构是基于OSI模型或TCP/IP模型。以下是TCP/IP模型的四个层次：

1. **网络接口层（Network Interface Layer）：** 处理物理网络介质和硬件设备之间的通信，负责封装和解封装数据帧，并提供设备驱动程序。
2. **网络层（Internet Layer）：** 主要负责在网络上定位和路由数据包。IP协议是网络层的核心协议，负责为数据包分配和处理IP地址。
3. **传输层（Transport Layer）：** 处理端到端的通信，确保数据的可靠传输。TCP（Transmission Control Protocol）和UDP（User Datagram Protocol）是传输层的两个常见协议。
4. **应用层（Application Layer）：** 提供用户和应用程序网络服务的接口。常见的应用层协议包括HTTP（Hypertext Transfer Protocol）、FTP（File Transfer Protocol）等。

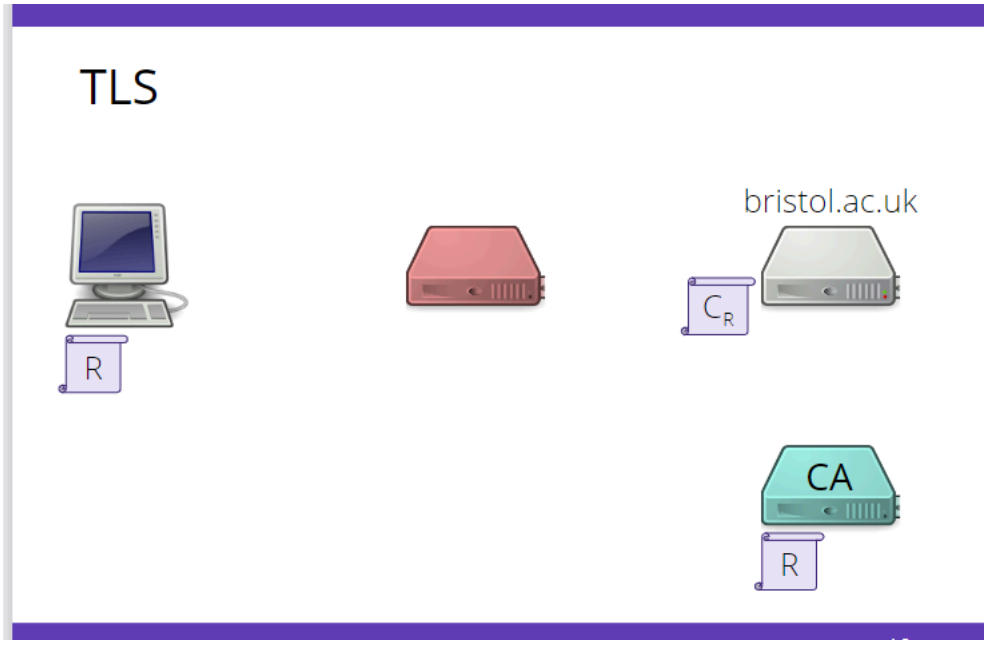
问了chatgpt这些都是计算机网络中不同的协议：

1. HTTP（Hypertext Transfer Protocol）-应用层：

- HTTP是一种**应用层**协议，用于在客户端和服务器之间传输超文本文档，通常用于Web浏览器与Web服务器之间的通信。
  HTTP使用TCP协议作为其传输层协议，使用TCP保证可靠传输

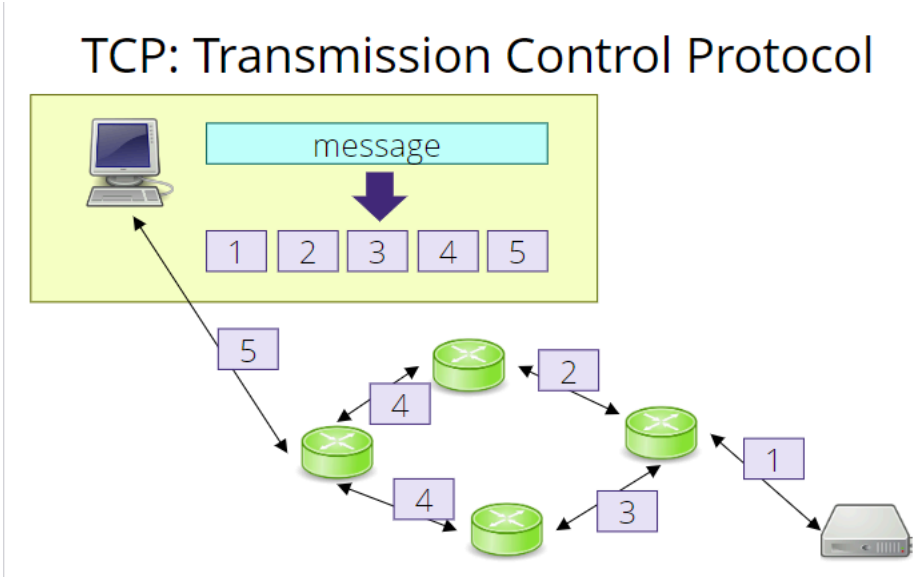
2. TLS（Transport *Layer* Security）-传输层：

- TLS是一种**安全协议**，用于在通信过程中加密数据，确保数据传输的机密性和完整性。TLS通常在传输层使用，提供对数据的加密和认证功能。
  就是有一个证书（certificate agency）



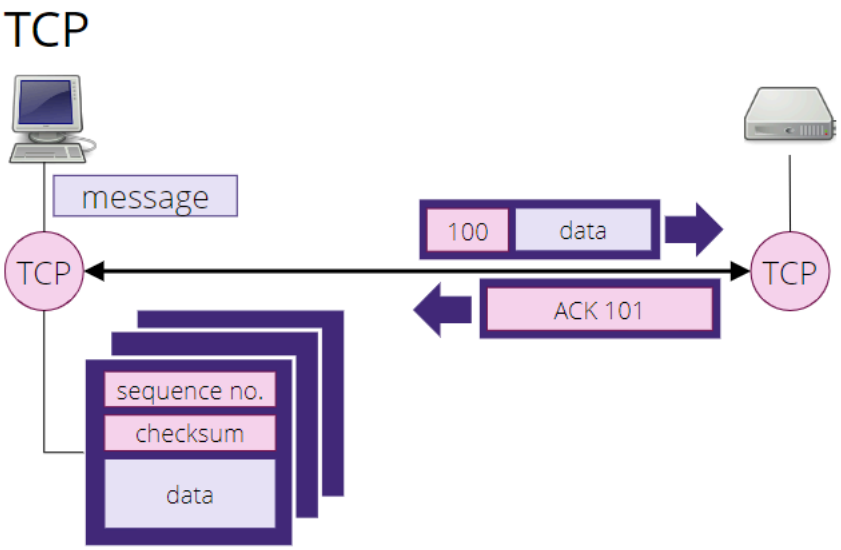
3. TCP（Transmission *Control* Protocol）-传输层：

- TCP是一种**传输层**协议，负责在网络上可靠地传输数据。它提供错误检测、流量控制和顺序传输等功能。HTTP和TLS通常运行在TCP之上。
- 建立连接：三次握手
- 终端连接：四次握手
- 保证传输：
  这张图的意思是，message过大给他五马分尸，然后都通过IP协议发送



- 错误检测：
  比如说上图的4重复了，那么header和server就会通过TCP协议进行检测
  - Sequence Number：header中的，用于标识每个数据包的唯一序号。接收方通过序列号来确定数据包的顺序，并且检测和丢弃重复的数据包。如果接收方收到一个已经接收过的数据包，它会根据序列号进行比较，如果发现重复，就会丢弃这个数据包，以确保数据的一致性。

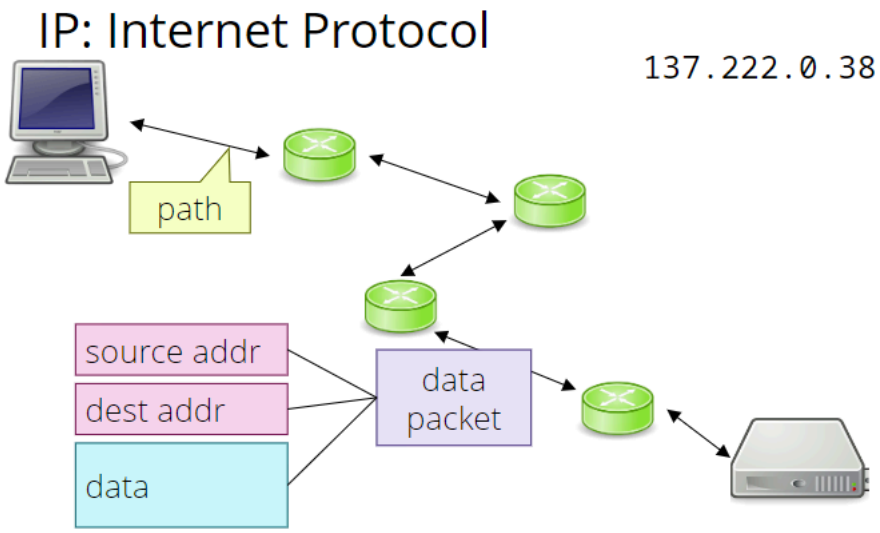
- Checksum：校验和（Checksum）字段，校验和用于检测数据包在传输过程中是否发生了错误或损坏。通过对数据包中的数据进行计算得出一个校验值，并将这个校验值添加到数据包中。发送方在发送数据包时会计算校验和，并将其添加到数据包的头部；接收方在接收到数据包后也会重新计算校验和，并将其与数据包中的校验和进行比较。如果两者不匹配，则表示数据包在传输过程中发生了错误或损坏
- `ACK 101  100 data` 是一对搭配，表示数据包的确认和数据的传输
  接收方已经成功接收了序列号为 100 的数据包，并且期望接收下一个序列号为 101 的数据包
  100就是上面说的sequence number



- 约定俗称的ports（一般0-1023是众所周知的），比如 HTTP 服务（端口号 80）、FTP 服务（端口号 21）、SSH 服务（端口号 22）

4. IP（Internet Protocol）-网络层：

- IP是一种**网络层**协议，负责在网络上标识和定位设备。它通过IP地址标识网络上的主机，并确保数据正确地从源主机传输到目标主机。
  是互联网核心协议，为TCP



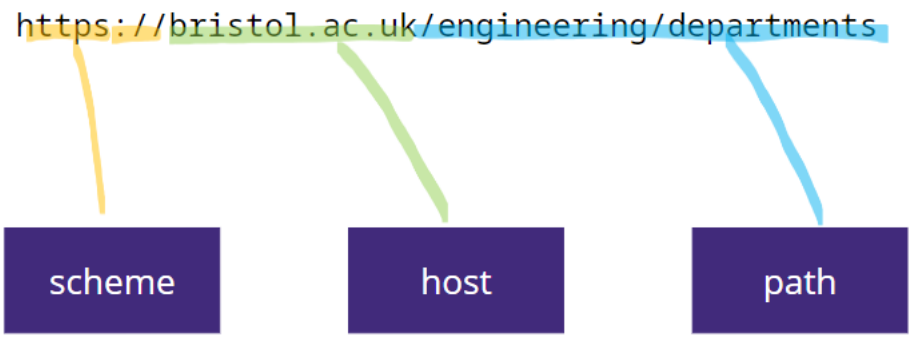
- Path：一般从网络到主机需要经过很多路由器、交换器、网络等
- Router（路由器）：用于在不同网络中转发数据包（一个数据包如果所示，包含着data、source address、destination address）路由器通常用于连接不同的网络段，它们具有将数据包从一个网络传输到另一个网络的能力
- 其中所谓的 `137.222.0.38`：
  i. `137` Network ID，网络号指示了数据包应该通过哪个网络传输
  ii. `222` Subnet ID，子网号用于将一个网络划分成多个子网，以实现更有效的网络管理和资源分配
  iii. `0` 通常用于标识特定的子网内的主机或设备。它可能表示子网内的某个特定主机，也可能是网络中的其他设备
  iv. `38` Host ID,标识了特定网络中的主机或设备

总结：

- HTTP 是应用层协议，用于传输超文本；
- TLS 是传输层安全协议，用于保护通信的安全性；

- TCP 是传输层协议，负责建立可靠的数据传输通道；
- IP 是网络层协议，负责在网络中传输数据包

URLs



- 总体组成部分：
  i. 协议（Protocol）： 指定访问资源所使用的协议或规范，例如 HTTP、HTTPS、FTP 等。
  ii. 主机名（Host）： 标识托管资源的服务器的域名或IP地址。
  iii. 端口号（Port）： 指定用于访问资源的端口号。如果未指定，默认使用协议的默认端口（如HTTP的默认端口是80）。
  iv. 路径（Path）： 标识服务器上资源的具体位置或路径。
  v. 查询参数（Query Parameters）： 传递给服务器的附加参数，通常以键值对的形式出现，用于定制请求。
  vi. 片段标识符（Fragment Identifier）： 标识资源中的特定部分，通常用于指定文档中的锚点或特定位置。
  `协议://主机名[:端口号]/路径?查询参数#片段标识符`
  `https://www.example.com:443/path/to/resource?param1=value1&param2=value2#section1`
  Protocol:HTTPS
  Host:www.example.com
  Port: 443
  Path:/path/to/resource
  查询参数:param1=value1和param2=value2,就是图下面说的query吧，有 ？ 标识符

- URI Schemes
  i. 定义：URI schemes（统一资源标识符方案）是用于标识和定位资源的命名方案或协议的一部分。在一个 URI 中，scheme 通常是 URI 的第一部分，后面跟着一个冒号（:）。URI schemes 指定了用于访问或定位资源的协议或协议族
  ii. 常用的有：
    a. HTTP（用于在 Web 上传输超文本的协议，通常用于访问网页）
    `http://www.example.com/`
    b. HTTPS（在 HTTP 的基础上通过加密（TLS/SSL）提供安全传输的协议）
    `http://www.example.com/`
    c. FTP（File Transfer Protocol）：用于在网络上传输文件的协议
    `ftp://username:password@ftp.example.com/path/to/directory/file.txt`
    d. File：用于访问本地文件系统中的文件
    `http://www.example.com/`

e. mailto：用于发送电子邮件的协议。

```
mailto:info@example.com
```

f. tel：用于拨打电话的协议

```
tel:+123456789
```

g. data：用于内嵌数据，通常用于图像或其他小型数据。

```
http://www.example.com/
```

h. telnet：用于在远程主机上执行命令的协议。

```
http://www.example com/
```

iii. PPT的详细解释

a. ftp://ftp.is.co.za/rfc/rfc1808.txt：
使用 FTP 协议访问资源，指向了一个文本文件 rfc1808.txt。

b. http://www.ietf.org/rfc/rfc2396.txt：使用 HTTP 协议访问资源，指向了一个文本文件 rfc2396.txt

c. ldap://[2001:db8::7]/c=GB?objectClass?one：使用 LDAP（Lightweight Directory Access Protocol）协议，访问了一个位于 IPv6 地址 [2001:db8::7] 的 LDAP 目录，并使用查询参数 c=GB?objectClass?one。(Lightweight Directory Access Protocol（轻量级目录访问协议）。它是一种用于在网络上访问和维护分层目录信息的协议。LDAP通常用于组织和存储有关用户、计算机、网络资源等信息的目录服务)

d. mailto:John.Doe@example.com：使用 mailto 协议，表示一个电子邮件地址。

e. news:comp.infosystems.www.servers.unix：使用新闻协议，指向了一个新闻组 comp.infosystems.www.servers.unix。

f. tel:+1-816-555-1212：使用 tel 协议，表示一个电话号码 +1-816-555-1212。

g. telnet://192.0.2.16:80/：使用 Telnet 协议，连接到了 IP 地址 192.0.2.16 的端口 80。

h. urn:oasis:names:specification:docbook:dtd:xml:4.1.2：使用 URN（Uniform Resource Name）命名空间，表示一个 DocBook XML DTD 版本 4.1.2 的规范。(URN 代表 Uniform Resource Name（统一资源名称），是一种用于在互联网上唯一标识资源的命名方式。URN的目的是提供一个持久性的、位置无关的标识符，使得资源可以在不同的上下文中被唯一地识别)

iv. REST(Representational State Transfer)

是一种设计原则和架构风格，用于构建分布式系统和网络应用程序。REST并非一种标准或协议，而是一组设计原则，旨在使网络通信更简单、可扩展和灵活。

a. State goes in the request:
表达了 REST 的无状态性。在 RESTful 系统中，每个来自客户端到服务器的请求应包含处理请求所需的所有信息。服务器不在请求之间存储客户端的状态，而是依赖于客户端在每个请求中提供必要的状态信息，例如身份验证信息或其他必要的数据。

b. Resources have names (URLs):
在 REST 中，资源通过 URI（Uniform Resource Identifier）标识，通常表示为 URL。每个资源，无论是物理实体还是抽象概念，都应该有一个唯一的名称或标识符，用于寻址和与之交互。URL 提供了一种命名和访问资源的方式

c. Use HTTP verbs as intended:
RESTful 服务应该充分利用标准的 HTTP 方法（GET、POST、PUT、DELETE 等），并按照这些方法的原始设计意图使用它们。每个 HTTP 方法在 REST 中都有特定的语义含义：
GET： 获取资源的表示。
POST： 创建新资源或提交要处理的数据。
PUT： 更新资源或在不存在时创建资源。
DELETE： 删除资源。

```
GET    /files/README.txt
DELETE /files/README.txt


POST   /files/README.txt?action=delete
GET    /files?name=README.txt
```

12 符合REST原则，34不符合
3是因为如果是删除的话，应该是DELETE xxx
4是因为通常不直接将资源名作为查询参数来传递。相反，资源通常通过 URL 的路径部分来指定

1. **GET /files/README.txt:**
   - 这是一个GET请求，用于获取位于 "/files/README.txt" 路径的资源。这可能表示客户端想要获取 README.txt 文件的内容。
2. **DELETE /files/README.txt:**
   - 这是一个DELETE请求，用于删除 "/files/README.txt" 路径下的资源，即删除 README.txt 文件。
3. **POST /files/README.txt?action=delete:**
   - 这是一个POST请求，用于在 "/files/README.txt" 路径下执行特定的操作。通过在请求中添加参数 "?action=delete"，可能表示客户端希望执行删除操作。在 RESTful 设计中，通常使用 HTTP的 DELETE 方法来表示删除操作，而不是在 POST 请求中添加特定操作的参数。因此，这个例子中的设计可能略显不符合 RESTful 的最佳实践。
4. **GET /files?name=README.txt:**
   - 这是一个GET请求，用于获取 "/files" 路径下的资源，通过查询参数 "name=README.txt" 过滤结果。这可能表示客户端想要获取具有特定名称（例如 README.txt）的文件列表。

Exercise

Exploring HTTP

1. `wget localhost:8000` :工具从本地主机的端口 8000 获取内容。wget 是一个用于从网络下载文件的命令行工具，您提供的命令表示要从本地主机上运行的 Web 服务器（通常在端口 8000 上）下载内容

   `netstat -tan` :这个命令用于显示系统的网络连接情况。netstat 是一个用于显示网络连接、路由表和网络接口信息的工具，而 -tan 选项是一组参数，表示：

   -t：仅显示 TCP 连接。

   -a：显示所有的连接和监听端口。

   -n：显示 IP 地址和端口号，而不进行 DNS 解析。

   因此，netstat -tan 命令将显示系统上所有的 TCP 连接，包括其 IP 地址和端口号，以及连接状态等信息
2. 基础server和client

   `nc -l -p 8000 < http-response` :network cat and tells it to listen on TCP port 8000 (-l -p 8000), e.g. to run a server there启动一个network cat让其在端口8000 run一个server

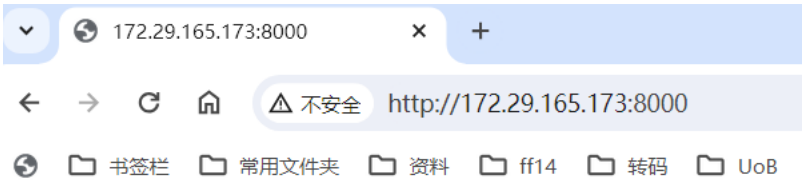
```
ada@LAPTOP-CQVA1E4P:~/HTTP-exer$ nc -l -p 8000 < http-response
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Wget/1.21.2
Accept: */*
Accept-Encoding: identity
Connection: Keep-Alive
```

```
ada@LAPTOP-CQVA1E4P:~/HTTP-exer$ wget -q -S -O - localhost:8000
  HTTP/1.1 200 OK
  Content-type: text/plain
  Content-length: 16
  Connection: close
```

- -q: 这是 wget 命令的选项之一，它用于关闭 wget 的输出，使其在运行时不显示任何消息或进度信息。-q 是 "quiet" 的缩写。
- -S: 这也是 wget 命令的选项之一，它用于在输出中显示服务器的响应头信息。-S 是 "server response" 的缩写。

- -O -: 这也是 wget 命令的选项之一，它用于指定要将下载的内容输出到哪里。在这种情况下，-O - 表示将下载的内容输出到标准输出流（stdout），而不是保存到文件中。

3. Connect to a web browser




```
Hello over HTTP!
```

4. A web server in C
   这台电脑没有C= = 先不做了大概

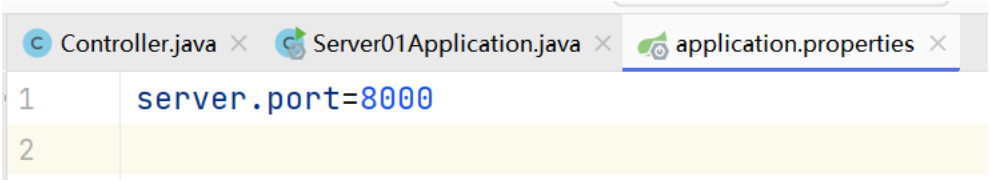
HTTP/URL research exercises

- The fragment part of a URL
  就是#之后的部分啦！用于标识文档中的特定部分，通常是文档的特定节或位置，页面加载时自动滚动到指定的部分
  比如说 `https://example.com/page.html#section1` ，当用户访问这个 URL 时，浏览器会请求 page.html 页面，然后自动滚动到文档中 section1 部分的位置。
- The Accept header sent by the HTTP client
  用于指示客户端可以接受的内容类型。它告诉服务器客户端可以接受哪些媒体类型的响应。
  `Accept: text/html, application/xhtml+xml, application/xml;q=0.9, */*;q=0.8` 客户端表示它可以接受 HTML、XHTML 和 XML 格式的内容，并且给出了每种类型的优先级权重。通配符 / 表示客户端接受任意类型的内容
- The User-agent header sent by the HTTP client
  通过检查 "User-Agent" 头，服务器可以根据客户端的类型和版本来适应性地提供服务，例如，可以根据不同的客户端类型提供不同的页面布局或功能
  `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99 Safari/537.36` User-Agent 字段告诉服务器，发送请求的客户端是运行在 Windows 10 操作系统上的 Chrome 浏览器的版本号为 97.0.4692.99
- How do you encode a path that contains a space in an URL? Which other characters are "special" and need to be treated this way in paths?
- Which web server is the University of Bristol using for www.bristol.ac.uk, according to the headers? Read up online on this server and the organisation of the same name that maintains it.

A server in Java

用到了Spring：

application.properties
这是一个configuration，让server在8000端口上跑x
里面就一个这个东西：



源代码：

- Model-View-Controller（MVC）是一个设计模式，constroller是负责用户输入和请求的，大概是一个中间人
- Spring是用@来作为class分类的：
  - `@SpringBootApplication` -在main class里面x表明这个是main函数
  - `@RestController` 告诉Spring 这个主要是和HTTP请求有关的，this class contains methods to deal with HTTP requests. 用REST命名主要because spring has libraries to make implementing the REST principles particularly easy.（REST是那个principle）注解告诉 Spring 框架，这个类处理的请求的返回值是直接返回给客户端的，而不是视图（View）。这意味着它的方法将直接返回 JSON、XML 或其他格式的数据给客户端。
  - `@AutoWired` 自动装配（注入）依赖关系，将需要的 Bean 注入到目标类中。on a field tells spring that this field is spring's responsibility to set up when the application starts这是一个 Spring 的依赖注入注解，它告诉 Spring 容器，在运行时，自动将依赖项注入到目标类中。通常，@Autowired 注解可以用在字段、构造函数、Setter 方法上，用来标记需要被自动注入的依赖
  - `@GetMapping(PATH)` this method should be called to reply to a HTTP GET request for the provided path (there is of course also a @PostMapping and so on).用于将特定的 HTTP GET 请求映射到控制器的特定方法上，@GetMapping 注解的 PATH 参数指定了该方法应该响应的请求路径

```java
    @GetMapping("/")
    public ResponseEntity<String> mainPage() {
        HttpHeaders headers = new HttpHeaders();
        headers.set(HttpHeaders.CONTENT_TYPE, "text/plain");

        return new ResponseEntity<String>( body: "Hello from Spring", headers, rawStatus: 2(
    }

    Alex Kavvos
    @GetMapping("/html")
    public ResponseEntity<Resource> htmlPage() {
        Resource htmlfile = loader.getResource( location: "classpath:web/page.html");
        return ResponseEntity
            .status(200)
            .header(HttpHeaders.CONTENT_TYPE, ...headerValues: "text/html")
            .body(htmlfile);
    }
}
```

- `mainPage`：load `localhost: 8000` 的时候就被调用，HTTP请求的是最基本的回复：
  - set up header，returns ResponseEntity
  - Response 包含三个部分：the HTTP body of the response to return (this will be shown in your browser), the HTTP headers to set, and the response code which is 200 (OK) here.

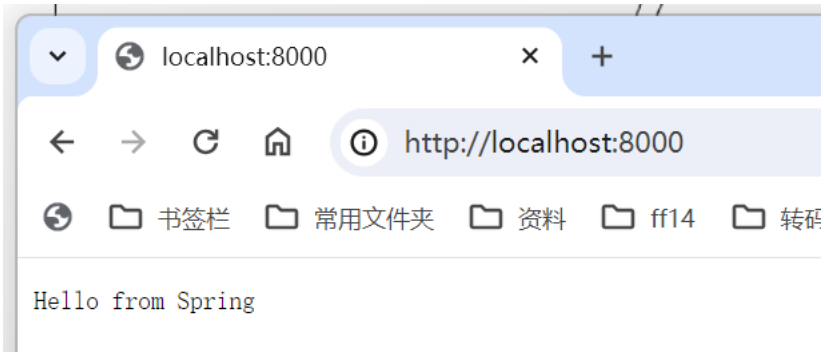
```java
    @GetMapping("/")
    public ResponseEntity<String> mainPage() {
        HttpHeaders headers = new HttpHeaders();
        headers.set(HttpHeaders.CONTENT_TYPE, "text/plain");

        return new ResponseEntity<String>( body: "Hello from Spring", headers, rawStatus: 200);
    }
```

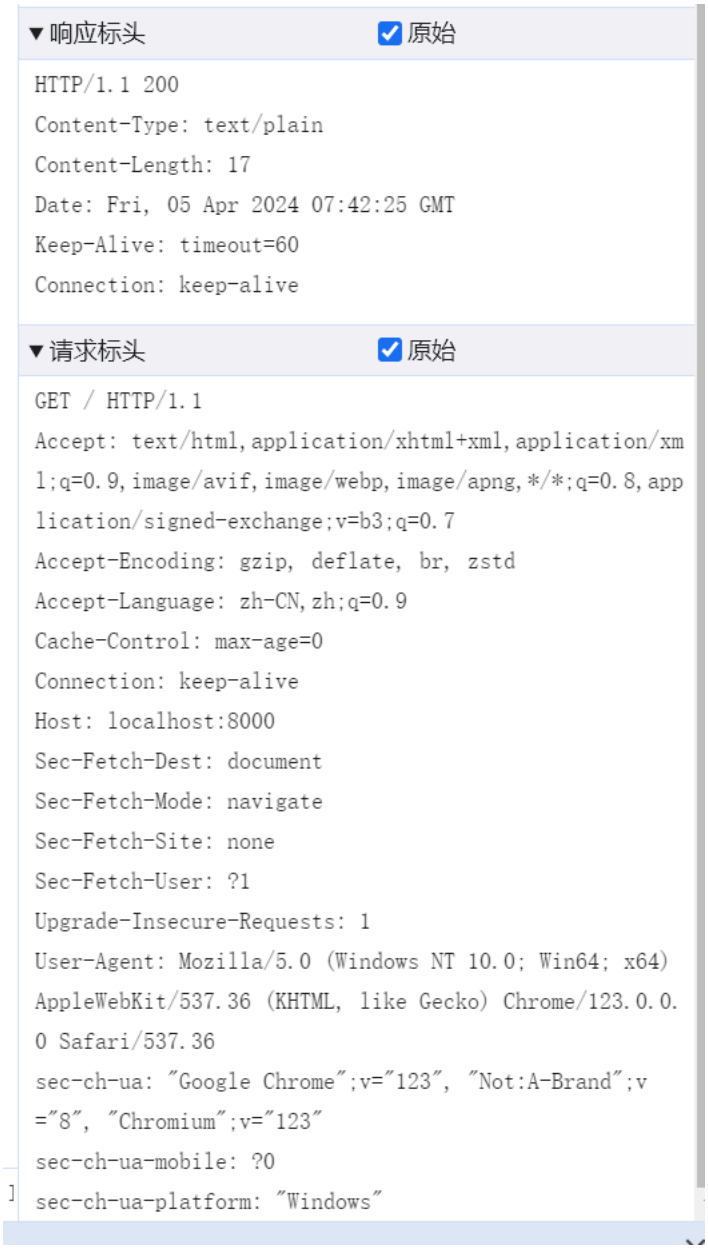
- `htmlPage`： 对于 `localhost: 8000/html` 的回复，要serve a file锁用了resource loader
  - 第二种方法builder来return ResponseEntity

```java
@GetMapping("/html")
public ResponseEntity<Resource> htmlPage() {
    Resource htmlfile = loader.getResource( location: "classpath:web/page.html");
    return ResponseEntity
        .status(200)
        .header(HttpHeaders.CONTENT_TYPE, ...headerValues: "text/html")
        .body(htmlfile);
}
```

结果：

- 第一种：


localhost:8000
http://localhost:8000
书签栏
常用文件夹
资料
ff14
转码
Hello from Spring


▼响应标头
原始
HTTP/1.1 200
Content-Type: text/plain
Content-Length: 17
Date: Fri, 05 Apr 2024 07:42:25 GMT
Keep-Alive: timeout=60
Connection: keep-alive
▼请求标头
原始
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: zh-CN,zh;q=0.9
Cache-Control: max-age=0
Connection: keep-alive
Host: localhost:8000
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36
sec-ch-ua: "Google Chrome";v="123", "Not:A-Brand";v="8", "Chromium";v="123"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"

- 第二种:

401.33px × 1111.33px

Sample HTML5 page

Hello from HTML!

▼响应标头 原始

```
HTTP/1.1 200
Accept-Ranges: bytes
Content-Type: text/html
Content-Length: 239
Date: Fri, 05 Apr 2024 07:45:46 GMT
Keep-Alive: timeout=60
Connection: keep-alive
```

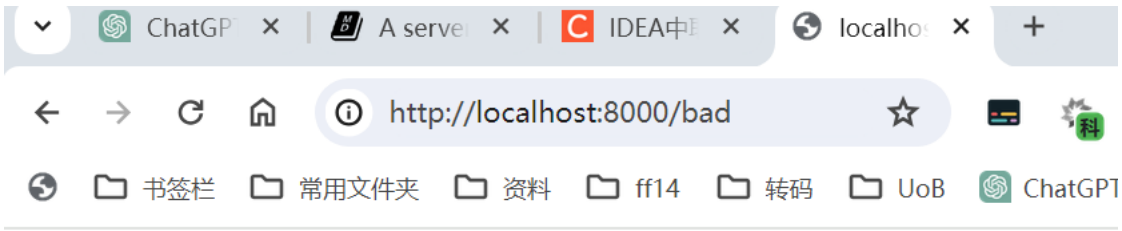
▼请求标头 原始

```
GET /html HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
mage/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br, zstd
Accept-Language: zh-CN,zh;q=0.9
Connection: keep-alive
Host: localhost:8000
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36
sec-ch-ua: "Google Chrome";v="123", "Not:A-Brand";v="8", "Chromium";v="12
3"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
```

```
2024-04-05 08:39:59.019  INFO 4056 --- [           main] s.server01.Server01Application
2024-04-05 08:39:59.024  INFO 4056 --- [           main] s.server01.Server01Application
2024-04-05 08:39:59.836  INFO 4056 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServ
2024-04-05 08:39:59.848  INFO 4056 --- [           main] o.apache.catalina.core.StandardServic
2024-04-05 08:39:59.849  INFO 4056 --- [           main] org.apache.catalina.core.StandardEngi
2024-04-05 08:39:59.917  INFO 4056 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]
2024-04-05 08:39:59.918  INFO 4056 --- [           main] w.s.c.ServletWebServerApplicationCont
2024-04-05 08:40:00.156  WARN 4056 --- [           main] ion$DefaultTemplateResolverConfigurat
2024-04-05 08:40:00.232  INFO 4056 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServ
2024-04-05 08:40:00.239  INFO 4056 --- [           main] s.server01.Server01Application
2024-04-05 08:40:51.889  INFO 4056 --- [nio-8000-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2024-04-05 08:40:51.889  INFO 4056 --- [nio-8000-exec-1] o.s.web.servlet.DispatcherServlet
2024-04-05 08:40:51.890  INFO 4056 --- [nio-8000-exec-1] o.s.web.servlet.DispatcherServlet
```

- 写一个 http://localhost:8000/bad

```
    @GetMapping("/bad")
    public ResponseEntity<String> notFoundPage() {
        String errorMessage = "404 Not Found: The requested page was not found.";
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.TEXT_PLAIN);
        return ResponseEntity
                .status(HttpStatus.NOT_FOUND)
                .headers(headers)
                .body(errorMessage);
    }
}
```



404 Not Found: The requested page was not found.

6.2 HTML5

Basics:

- `<p>my cat</p>`
- attribute: `attributeName="attribute Value"`

```
<p class="editor-note">my cat</p>
```

这里的这个就是所谓的属性，class大概就是给一个identifier很快的target过来（类似于索引？）

- nested:

```
<p>my <stong>cat</strong></p>
```

- Void elements:

```
<img src="images/firefox-icon.png" alt="My test image" />
```

这个没有也没有内部信息，This is because an image element doesn't wrap content to affect it. Its purpose is to embed an image in the HTML page in the place it appears.
src：source； alt：无法加载时的信息

- Anatomy of an HTML document

```
<!-- 有点类似于latex，有一个开头和结尾<HEAD> </HEAD>    -->
<!DOCTYPE html>
<html lang="en"><!-- language的意思   -->
<head><!-- 包含所有不想显示的信息   -->
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!--ensures the page renders at the width of viewport -->
    <title>Document</title>
</head>
<body>
    <!-- 这里才是主要显示内容得地方 -->
    <!-- h1 是标签，而且h1是最大的标签 -->
    <h1>HEAD LINE</h1>
    <h2>HEAD LINE</h2>
    <!-- 段落 -->
    <p>
        BALABLALABLALAALA
        <!-- 超链接 -->
        <a href="https://baidu.com"> getopen</a>
        <!-- ctrl+B，粗体  -->
        <strong> </strong>
        <!-- 斜体，emphasis  -->
        <em> </em>


    </p>

</body>
</html>
```

- lists:
  有两种类型：
  unordered list，
  ordered lists(1,2,3这种吧)

```
<ul> <!-- changed to list in the tutorial -->
  <li>technologists</li>
  <li>thinkers</li>
  <li>builders</li>
</ul>
```

- technologists
- thinkers
- builders

HTML text fundamentals

倒不如说很多东西前面讲过啦
就是讲讲分段、heading之类的东西

- span：

```
<span style="font-size: 32px; margin: 21px 0; display: block;">
  Is this a top level heading?
</span>
```

改改大小字号什么的，和CSS和JavaScript一起用
margin: 21px 0; 设置了文本的上下边距为 21 像素，左右边距为 0
display: block; 将 元素的显示属性设置为块级元素，使得它会在页面上单独占据一行，并且上下都有一定的空间

- 上面两个list相关的
- Emphasis和Strong

```
<!-- ctrl+B, 粗体  -->
<strong> </strong>
<!-- 斜体, emphasis  -->
<em> </em>
```

Creating hyperlinks

- 就是前面的那个href
  主要就是 `<a href="https://www.mozilla.org/en-US/">the Mozilla homepage</a>.`
- Document fragments
  可以先这样（id用于唯一标识一个元素）
  `<h2 id="Mailing_address">Mailing address</h2>`
  然后再

```
Want to write us a letter? Use our
<a href="contacts.html#Mailing_address">mailing address</a>.
</p>
```

  - Absolute versus relative URLs
    相对和绝对地址嘛

Links

```
<a href="/courses">Our Courses</a>
```

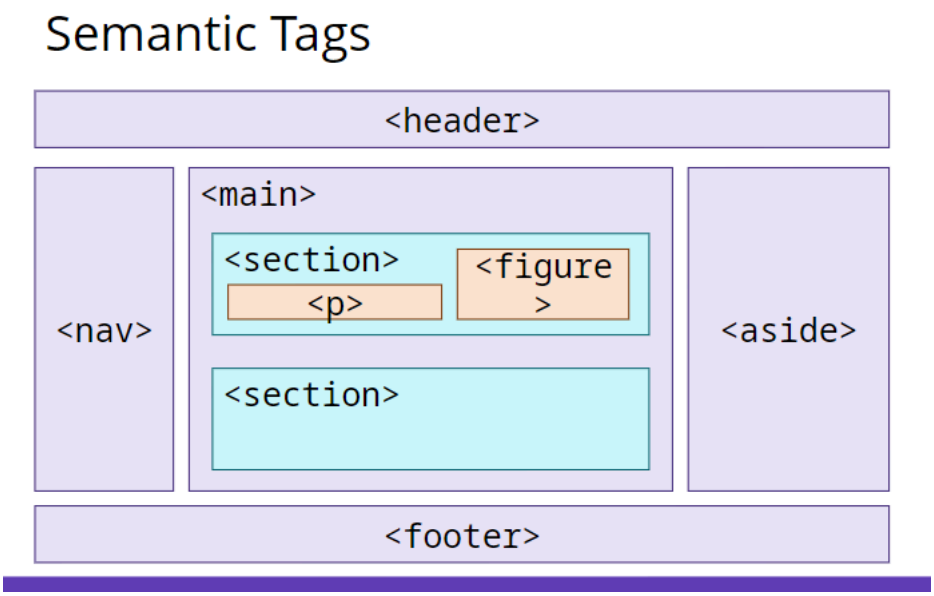
bristol.ac.uk/students/info.html :

```
"/courses"    => bristol.ac.uk/courses
"courses"     => bristol.ac.uk/students/courses
"../courses" => bristol.ac.uk/courses
```

"/courses"：相对于网站根目录的路径
"courses"：该文件夹下的另一个文件
"../courses"相对于当前文件所在目录的上一级目录中的 "courses" 文件

Document and website structure

- HTML for structuring content



- HTML layout elements in more detail
  - `<main>` content unique to this page一页只能用一次，一般放在 `<body>` 里面也没有nested的内容
  - `<article>` 元素用于表示页面中独立、完整、可独立分配的内容块，通常是一篇新闻、一篇博客文章、一段论文
  - `<section>` 元素用于将页面内容组织成主题性的节(section)，通常用于将相关的内容块进行分组。
  - `<aside>` provide additional information
  - `<header>`
  - `<nav>`
  - `<footer>`
- Non-semantic wrappers
  - `<span>` 反正是和CSS JAVAscript一起搞文字样式的

HTML Demo:

RESET

HTML | CSS

```
1 <p>
2   Add the <span
  class="ingredient">basil</span>, <span
  class="ingredient">pine nuts</span> and
3   <span class="ingredient">garlic</span>
  to a blender and blend into a paste.
4 </p>
5
6 <p>Gradually add the <span
  class="ingredient">olive oil</span>
  while running the blender slowly.</p>
7
```

OUTPUT

Add the basil, pine nuts and garlic to a blender and blend into a paste.

Gradually add the olive oil while running the blender slowly.

- `<div>` 用作容器，分别用来包裹页面的整体内容和页面的主要内容区域。然后通过 CSS 样式对这些区块进行了布局和样式化

HTML Demo:

RESET

HTML | CSS

```
1 <div class="warning">
2   <img src="/media/examples/leopard.jpg"
  alt="An intimidating leopard." />
3   <p>Beware of the leopard</p>
4 </div>
5
```

OUTPUT

- Line breaks and horizontal rules
  - `<br>` : the line break element
    换行

HTML Play

```
<p>
  There once was a man named O'Dell<br />
  Who loved to write HTML<br />
  But his structure was bad, his semantics were sad<br />
  and his markup didn't read very well.
</p>
```

Without the `<br>` elements, the paragraph would just be rendered in one long line (as we said earlier in the course, HTML ignores most whitespace); with `<br>` elements in the code, the markup renders like this:

Play

There once was a man named O'Dell
Who loved to write HTML
But his structure was bad, his semantics were sad
and his markup didn't read very well.

- `<hr>` : the thematic break element
  分行符

HTML

Play

```
<p>
  Ron was backed into a corner by the marauding netherbeasts. Scared, but
  determined to protect his friends, he raised his wand and prepared to do
  battle, hoping that his distress call had made it through.
</p>
<hr />
<p>
  Meanwhile, Harry was sitting at home, staring at his royalty statement
and
  pondering when the next spin off series would come out, when an enchanted
  distress letter flew through his window and landed in his lap. He read it
  hazily and sighed; "better get back to work then", he mused.
</p>
```

Would render like this:

Play

Ron was backed into a corner by the marauding netherbeasts. Scared, but determined to protect his friends, he raised his wand and prepared to do battle, hoping that his distress call had made it through.

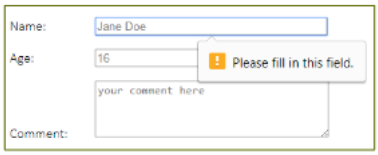
Meanwhile, Harry was sitting at home, staring at his royalty statement and pondering when the next spin off series would come out, when an enchanted distress letter flew through his window and landed in his lap. He read it hazily and sighed; "better get back to work then", he mused.

The HTML5 input types

- Forms

forms就是给服务器提交东西的

```
<form method="post" action="/comment">
<!--提交为post方法，数据作为HTTP request的一部分发送到server-->
<!--指定了数据提交的目标 URL，即表单数据将发送到 /comment 路径对应的服务器端处理程序-->
<p>
  <label for="name">Name:</label>
  <input id="name" name="name"/>
</p>
<p>
  <button type="submit">OK</button>
</p>
```

```

- Validation

```
<input required type="number">
```

规定必须是数字




- Autocomplete

```
  <input type="text"
autocomplete="name">
```

- input Types
- Select

```
<select name="animal">
  <option value="dog">Dog</option>
  <option value="cat">Cat</option>
</select>
```

- Tables
  - `<table>`：用于定义表格。
  - `<tr>`：表格中的行，即 table row。
  - `<th>`：表格中的表头单元格，通常用于表示列的标题。
  - `<td>`：表格中的数据单元格，即 table data。

```
<table>
  <thead>
    <tr><th>Name</th><th>ID</th></tr>
  </thead>
  <tbody>
    <tr><td>Sarah</td><td>100</td></tr>
    <tr><td>Jon</td><td>101</td></tr>
  </tbody>
</table>
```

或者有：
```

```
<table border="1">
  <tr>
    <th>列1</th>
    <th>列2</th>
    <th>列3</th>
  </tr>
  <tr>
    <td>数据1</td>
    <td>数据2</td>
    <td>数据3</td>
  </tr>
</table>
```

结果就是

```
+------+------+------+
| 列1  | 列2  | 列3  |
+------+------+------+
| 数据1 | 数据2 | 数据3 |
+------+------+------+
```

Exercises

Basic HTML5

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="utf-8" />
   <title>A web page</title>
 </head>
 <body>
    <h1>COMS10012 Software Tools</h1>
    <p><strong>This unit teaches you the basics of web development.</strong></p>

    <h2>Content</h2>
    <p>Content is a combination of video lectures and reading assignments from the <em>Mozilla Developer Network.</em></p>
    <p>You can find the unit page on <a href='https://cs-uob.github.io/COMSM0085/'>on github</a>.</p>

    <h2>Learning Outcomes</h2>
    <p>After completing this unit, you will be able to use the following:</p>
    <ul>
      <li>HTML5</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
 </body>
</html>
```

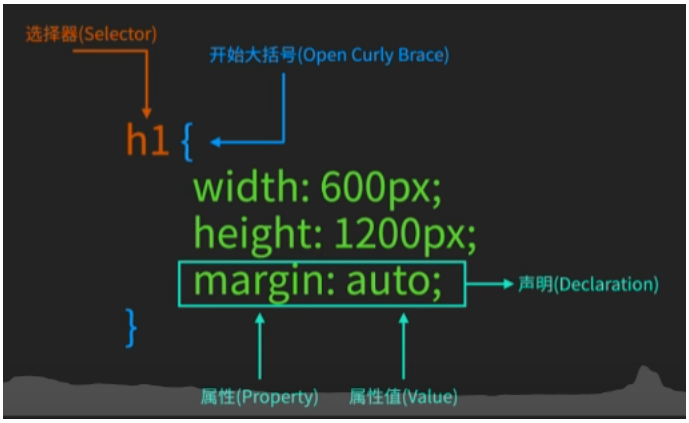
Templating

先不做了!

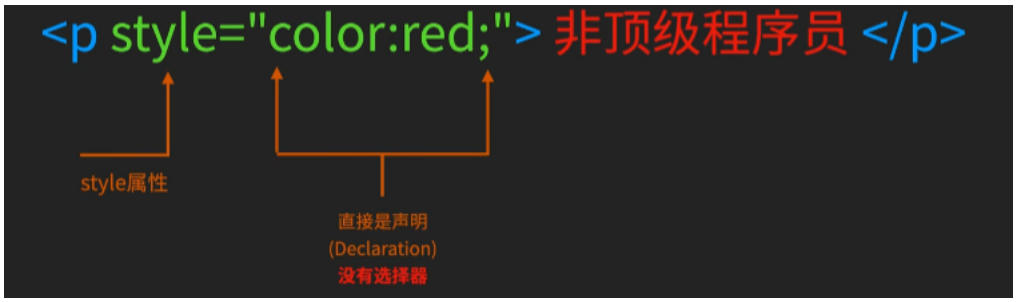
Week 7: CSS

CSS

基本语法

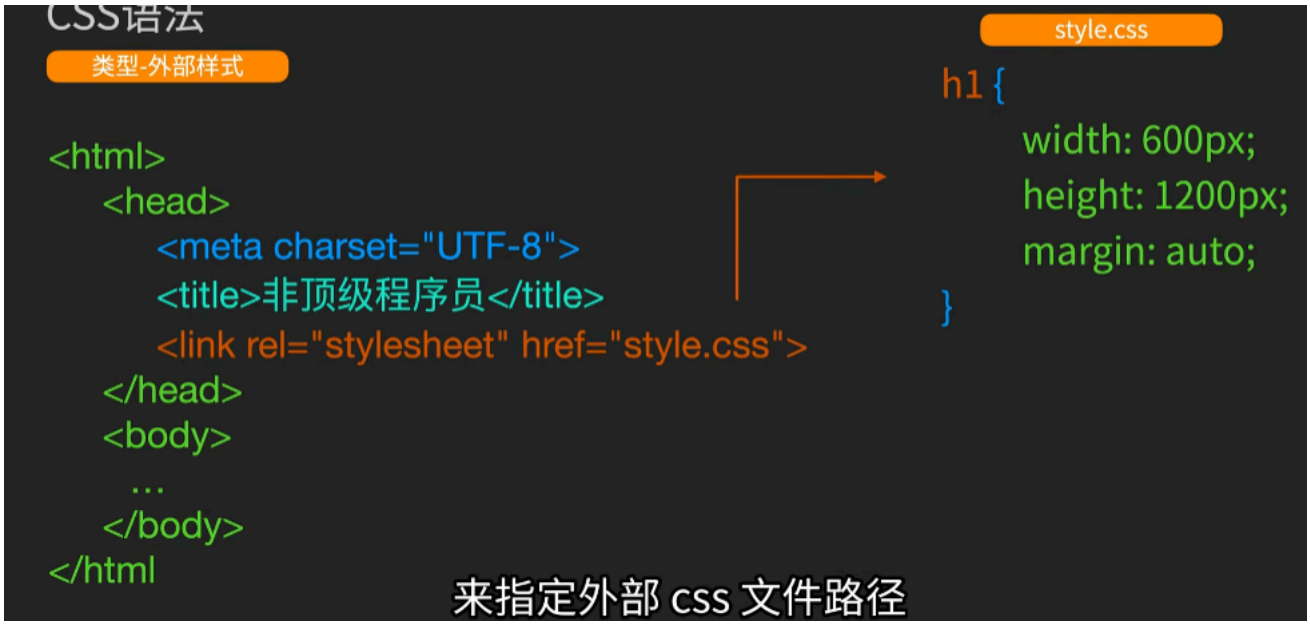


- inline Style
  没有选择器



- Internal Style：写到head里面

- External Style：指定外部文件

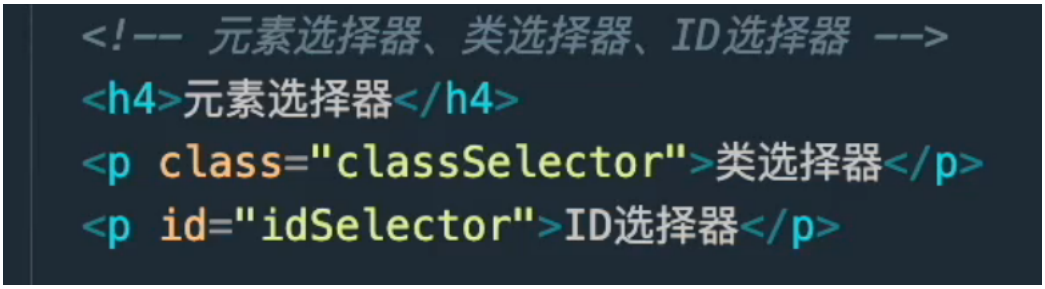



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        h1{
            color:blue;
        }
    </style>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <h1 style="color: red;">First</h1>
    <h2>external</h2>
    <h1>Second</h1>
    <h1>Third</h1>
</body>
</html>
```

选择器Selector

除了常用的那个

元素选择器



对应就是

```
h4{
  color:blue;
}
.classSelector{
  color:green;
}
#idSelector{
  color:blue;
}
```

但是常用的还是类选择器，ID选择器一般只能用于一个元素，并且ID选择器的优先级是最高的

类的层级关系

```
<div class="child">外部关系</div>
<div class="parent">
  <div class="child">内部关系</div>
</div>
```

```
.child{
  color:red;
}
.parent{
  .child{
    color:blue;
  }
}
```

最后外部关系应该是：红色
内部关系应该是：蓝色
parent-child虽然同名，但是有层级关系的可以直接指定（

分配不同的样式

大概这样就可以分配很多类了）

```
<p class="text-color font-size">选择器来</p>
```

CSS	HTML
p	`<p>`
.important	`<div class="important">`
#title	`<div id="title">`
p.important	`<p class="important">`
h1#title	`<h1 id="title">`

选择范围

`.container p` 选择所有 container **内部的p元素**
`.container > p` 选择所有p元素的直接子后代?

```
<div class="container">
    <p>This is a direct child paragraph.</p>
    <div>
        <p>This is a nested paragraph.</p>
    </div>
</div>
```

- 比如这里， `.container p` 两个p元素都会选，而 `.container > p` 只有direct child

`.container ~ p` 紧跟在 .container 元素**之后**的所有 `<p>` 元素
`.container + p` 紧跟在 .container 元素**之后**的第一个 `<p>` 元素

```
<div class="container"></div>
<p>Paragraph 1</p>
<p>Paragraph 2</p>
```

- `.container ~ p` 选择两个， `.container + p` 只会选择第一个

```
<div class="container"></div>
<section>
    <p>Paragraph 1</p>
    <p>Paragraph 2</p>
</section>
```

- `.container ~ p` 会选择两个，而 `.container p` 一个都不会选，因为不在内部

Colour

要不就上面说的color name（red，green etc.）

- HEX

```
.hex{
  /*rrggbb*/
  /*#FF0000 表示红色，#00FF00 表示绿色，#0000FF 表示蓝色*/
  color:#eb45e0;
}
```

- RGB

```
.rgb{
  /*RED GREEN BLUE:0-255*/
  color:rgb(0,255,255);
}
```

任意赋值，比如

```
.background-color{
  background-color:rgb(0,0,0);
}
.border-color{
  border:3px solid #FF00FF;
}
```

还可以调整透明度比如：

```
.background-transparent{
  color:rgba(255,255,255,0.5);
}
/*或者这样也是透明度*/
.opacity{
  opacity:0.5
}
```

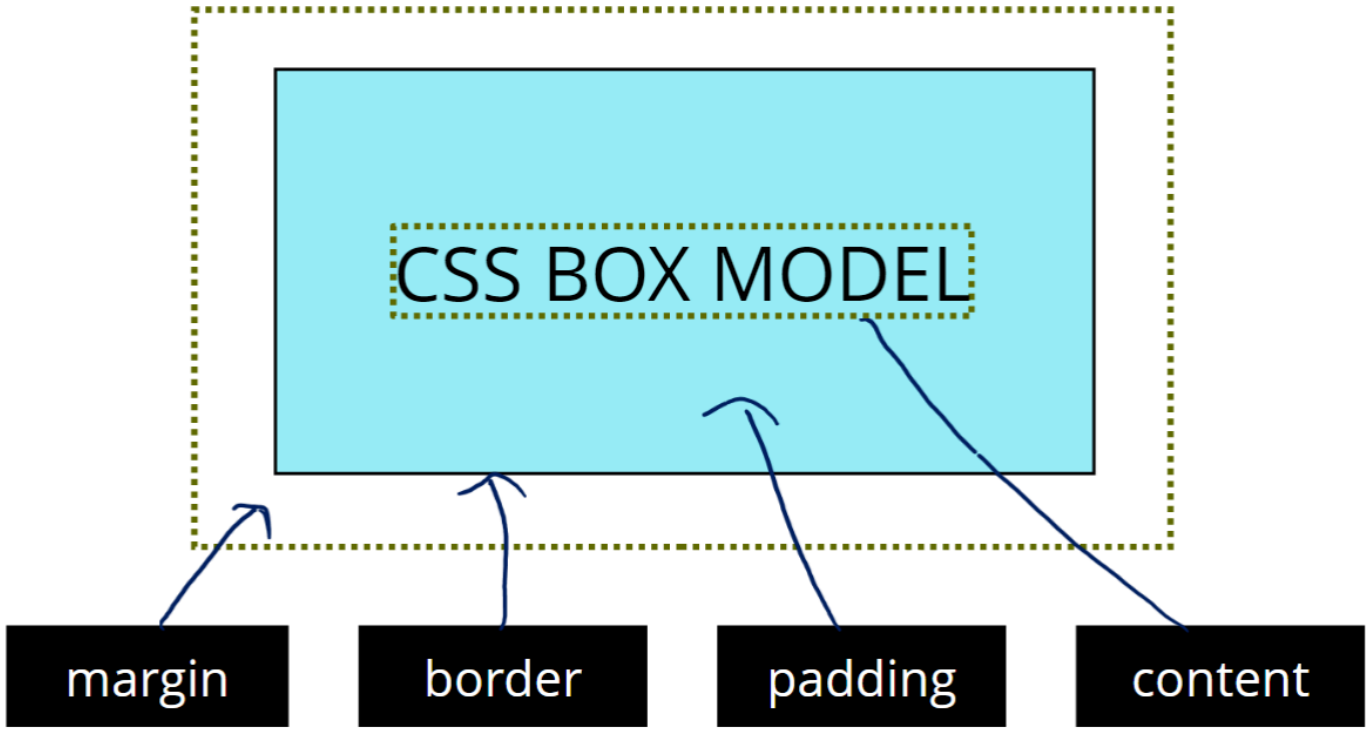
text

```
.selector {
    font-family: family-name | generic-family;
}
```

family-name：指定一个具体的字体族名称，可以是字体的名称、字体系列的名称，或者是一个通用字体族名称。
generic-family：指定一个通用的字体族名称作为后备选择，如果用户系统上没有指定的字体族，则会使用通用字体族。

```css
@font-face {
  font-family: "MaShanZheng";
  src: url("./MaShanZheng-Regular.ttf");
}

.text-1 {
  font-family: MaShanZheng, 'sans-serif';
}

.text-2 {
  font-size: 24px;
  line-height: 32px;
}

.text-3 {
  font-weight: 200;
  font-style: italic;
  text-decoration: underline;
}
```

盒子模型（box model）



```
.box-model{
  margin-top: 10px;
  padding-left: 20px;

  margin: 10px 5px 15px 20px;
  /*top right bottom left*/
  margin: 25px 50px 75px;
  /*
  top margin is 25px
  right and left margins are 50px
  bottom margin is 75px
  */
  margin: 25px 50px;
  /*
  top and bottom margins are 25px
  right and left margins are 50px
  */


  border: 1px solid black;
}

div {
  width: 300px;
  height:120px;
  /*本体可以直接设置宽度和高度*/

  margin: auto;
  /*auto set the margin: 将使浏览器自动计算并分配元素的左右外边距，以使其在父容器中水平居中。这意味着元素的左右外边距将自动设置为相等的值，从而使元素水平
  border: 1px solid red;

}
```

Unites of Measurement

dpi(ppi)：dots per inch
PC->72
Units：

- px->pixel
- pt->point
- cm,in
- em->width of letter m
- ex->height of letter x
- lh->line height(+space)
- vm,vh 1% of viewport width/height
- rem root em(html)
- % parent width/height

Exercise

i. 要在HTML里面的里加一个 `<link rel="stylesheet" href="sometext.css">`

```
body {
    margin: 0 auto;
    max-width: 40em;
    line-height: 150%;
    font-size: 20px;
}
```

- 提到了现在一般不用double-spacing(line-height: 200%)

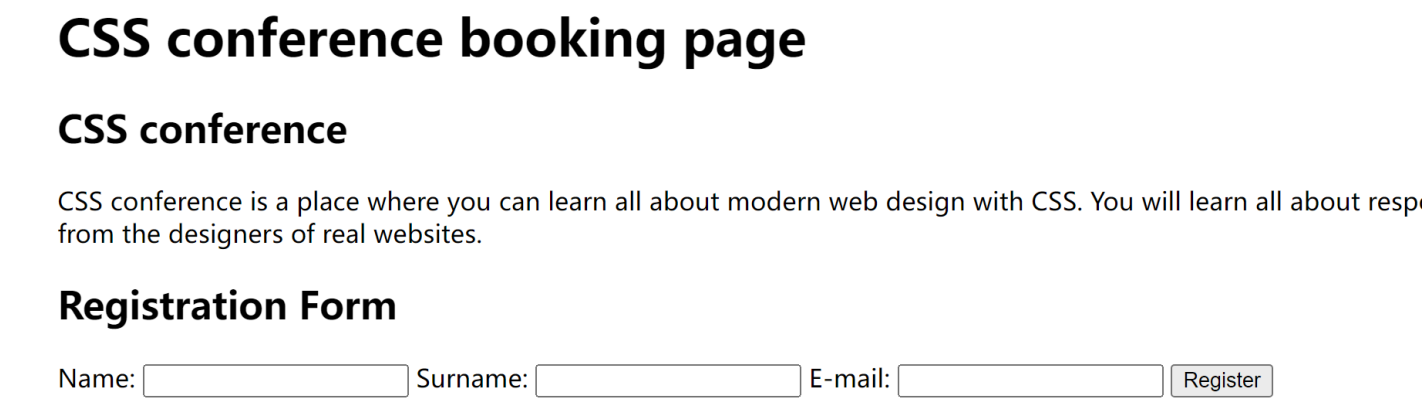
ii. Starting from Scratch
重头写一遍所有的Styling

- h1 h2的font-size可以写成 `font-size:150%;` ,也可以写成 `1.5em` 这种

iii. Frameworks:

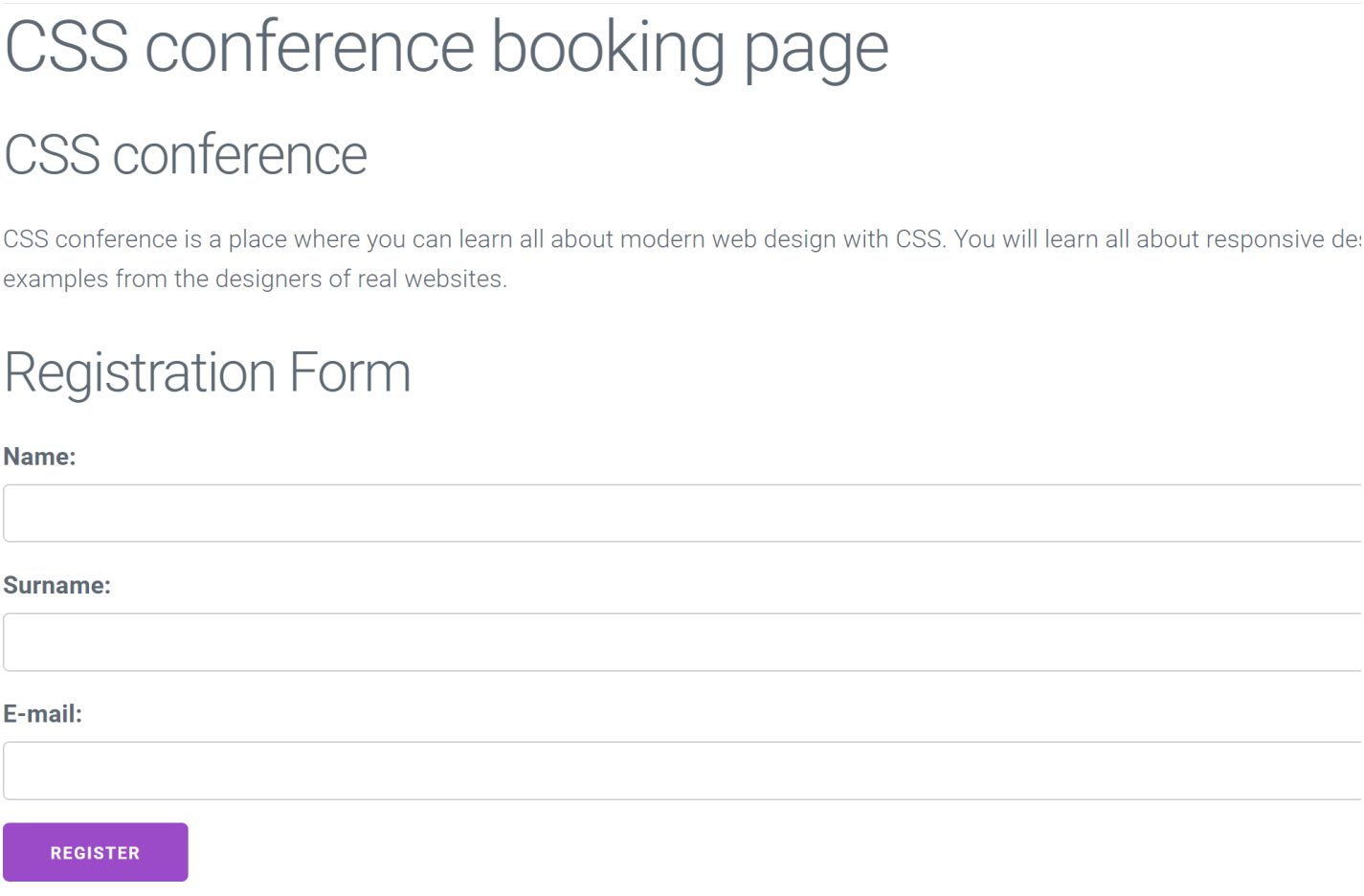
a. **Milligram**

Milligram chooses to style the whole page by default, but you can customise this further



- 下载下来是这样的
- 然后加了

```
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,300italic,700,700italic">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/milligram/1.4.1/milligram.css">
```



突然变好看了（

第一个链接是字体，第二个链接指向的 `normalize.css`

- a. Correct the line height in all browsers.

- b. Prevent adjustments of font size after orientation changes in iOS.
  第三个才是加载*Milligram*
  milligram能自主设置，比如main中加一个container

CSS conference booking page

CSS conference

CSS conference is a place where you can learn all about modern web design with CSS. Y
tricks and practical examples from the designers of real websites.

Registration Form

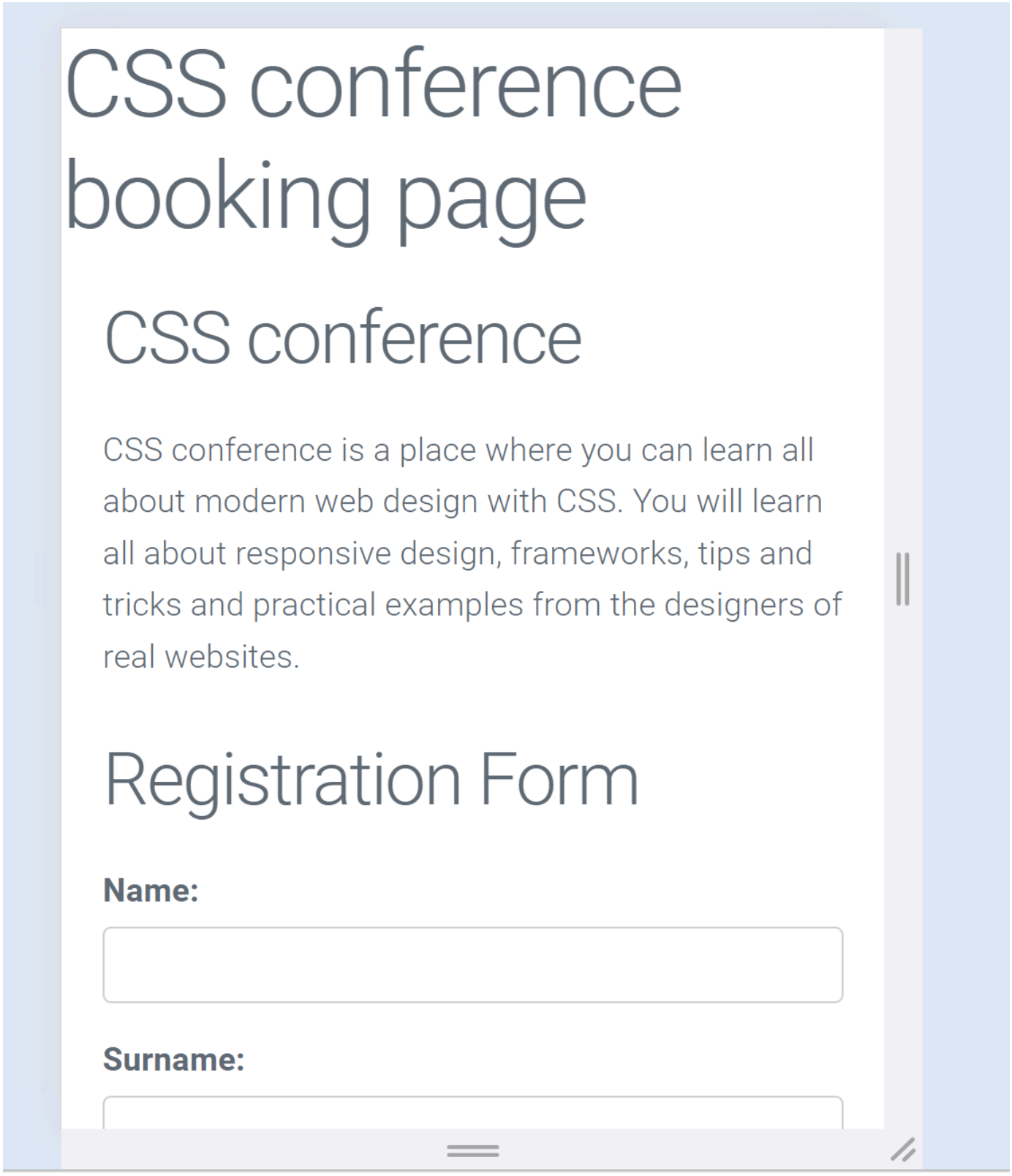
Name:

Surname:

E-mail:

REGISTER

- 加一个移动端：



这是因为有一个这个 `<meta name="viewport" content="width=device-width, initial-scale=1">`

其余的就是：Milligram是怎么设置以下内容的?

- Size of heading fonts

```
h1,
h2,
h3,
h4,
h5,
h6 {
  font-weight: 300;
  letter-spacing: -.1rem;
  margin-bottom: 2.0rem;
  margin-top: 0;
}
h1 {
  font-size: 4.6rem;
  line-height: 1.2;
}
h2 {
  font-size: 3.6rem;
  line-height: 1.25;
}
```

- Form fields take up the full width of the container

```
dl,
ol,
ul {
  list-style: none;
  margin-top: 0;
  padding-left: 0;
}
```

- Form labels and fields appear below each other

```
label,
legend {
  display: block;
  font-size: 1.6rem;
  font-weight: 700;
  margin-bottom: .5rem;
}
```

label一般用的block

- Labels are closer to their own field than the field above

  `Milligram may use CSS margins or padding to control the spacing between form labels and fields.`

- Size and centering of everything in a container on wide enough screens

```
@media (min-width: 40rem) {
  .row {
    flex-direction: row;
    margin-left: -1.0rem;
    width: calc(100% + 2.0rem);
  }
  .row .column {
    margin-bottom: inherit;
    padding: 0 1.0rem;
  }
}
```

所以是用的media queries

b. **Bulma**

it only styles parts of the page you tell it to (but it sets a default font), and it has some more components to build things like menus or panels.

加了 `<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.3/css/bulma.css">` 之后就是这样的：

CSS conference booking page

CSS conference

CSS conference is a place where you can learn all about modern web design with CSS. You will learn all about responsive design, framewc practical examples from the designers of real websites.

Registration Form

Name: Surname: E-mail: Register

- 提到了 `bulma.css` 和 `bulma.min.css` 其实意思就是，前面的是给人读的，后面的就直接把各种换行给去掉了）
- 搞点自定义的bulma试试：
  a. HERO title：
  Add class hero to the header tag.
  Add a child div hero-body inside it that wraps the h1 tag.
  Set the classes title is-1 on the h1 tag itself.

Google Translate

CSS conference booking page

CSS conference

CSS conference is a place where you can learn all about modern web design with CSS. You will learn all about responsive design, frameworks, tips and tr practical examples from the designers of real websites.

Registration Form

Name: Surname: E-mail: Register

```
    <header class="hero">
    <!-- Child div with class "hero-body" -->
    <div class="hero-body">
        <!-- H1 tag with classes "title is-1" -->
        <h1 class="title is-1">CSS conference booking page</h1>
    </div>
</header>
```

ii. 内容方面的
加一个

```
<main class="content">
```

CSS conference booking page

CSS conference

CSS conference is a place where you can learn all about modern web design with CSS. You will learn all about responsive design, frameworks, tips and tricks a from the designers of real websites.

Registration Form

Name: Surname: E-mail: Register

CSS grids

Display 有很多种，比如block, inline, inline-block, flex 等，课程主要讲的是grids（网格）

```css
.block {
  display: block;
}

.inline {
  display: inline;
}

.inline-block {
  display: inline-block;
  width: 100px;
  height: 120px;
}
```

其实就是：

block->独占一行
inline->不会独占一行
block->inline->能设置宽高

grid

需要设置： `display:grid;`

```
<!DOCTYPE html>
<html>
<head>
<style>
.item1 { grid-area: header; }
.item2 { grid-area: menu; }
.item3 { grid-area: main; }
.item4 { grid-area: right; }
.item5 { grid-area: footer; }

.grid-container {
  display: grid;
  grid-template-areas:
    'header header header header header header'
    'menu main main main right right'
    'menu footer footer footer footer footer';
  gap: 10px;
  background-color: #2196F3;
  padding: 10px;
}

.grid-container > div {
  background-color: rgba(255, 255, 255, 0.8);
  text-align: center;
  padding: 20px 0;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>Grid Layout</h1>

<p>The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having t

<div class="grid-container">
  <div class="item1">Header</div>
  <div class="item2">Menu</div>
  <div class="item3">Main</div>
  <div class="item4">Right</div>
  <div class="item5">Footer</div>
</div>

</body>
</html>
```

- Grid Layout
  - 在HTML中应该有

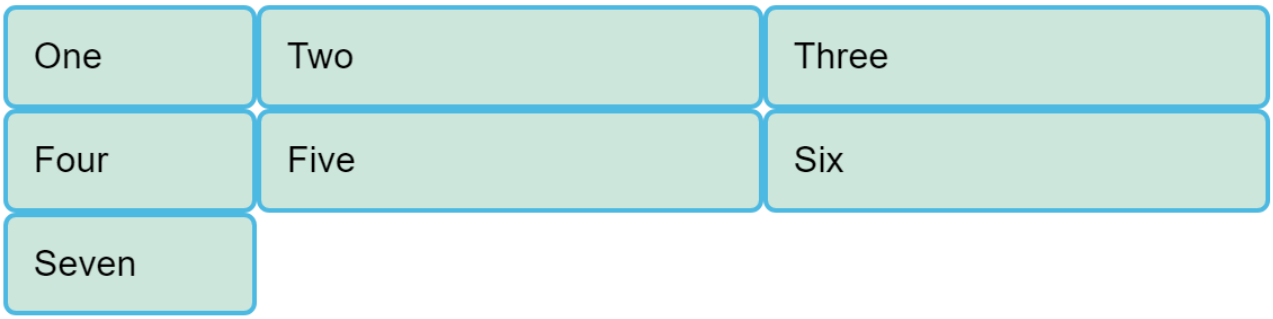
```
<div class="container">
  <p>item1</p>
  <p>item2</p>
</div>
```

然后CSS有

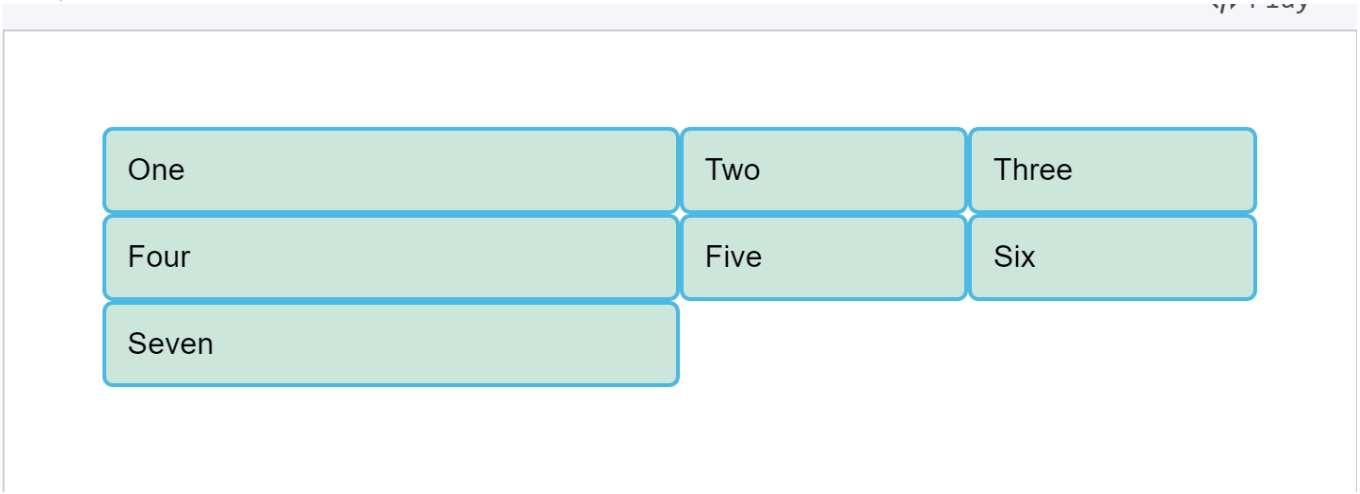
```
.container{
  display:grid;
}
```

  - 调整宽高

```
.container{
  display:grid;

  grid-template-rows: 200px 100px 100px;
  grid-template-columns: 2fr 1fr 1fr;
  grid-template-columns: repeat(3, 1fr);
}
```



grid-template-rows的效果



grid-template-columns的效果

One

Two

Three

Four

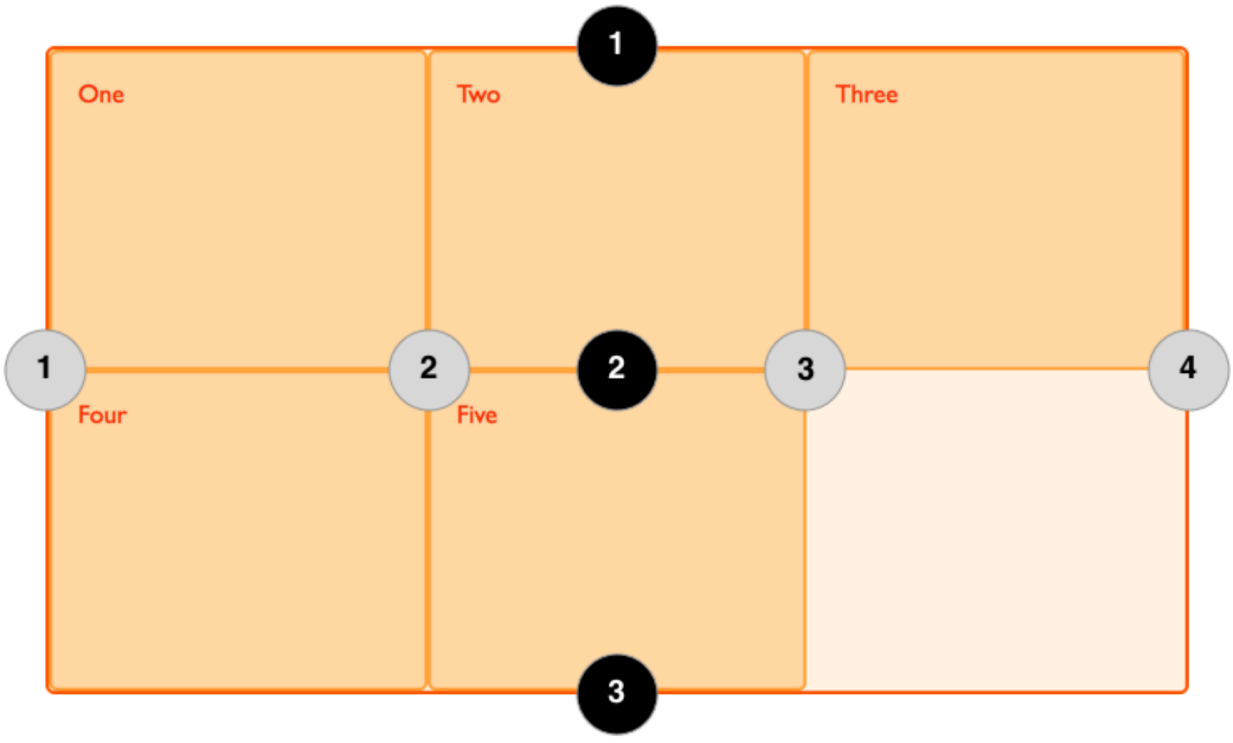
Five

Six

Seven

gap.repeat的效果

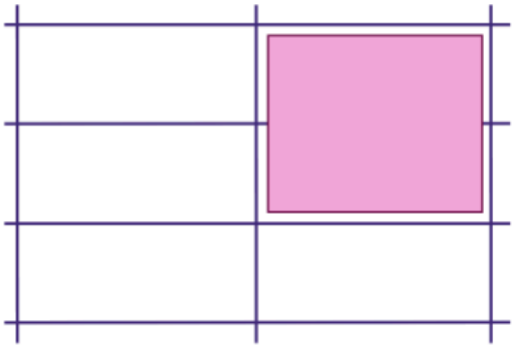
- 获取区域



网格元素设置时可以参考这些行号

人家是从1开始的

```
grid-row-start: 1;
grid-row-end: 3;
grid-column-start: 2;
grid-column-end: 3;

grid-row: 1 / 3;
grid-column: 2 / 3;

grid-area: 1 / 2 / 3 / 3;
```



```
grid-area: row-start/column-start/row-end/column-end
```

学了这个就可以类似的:

```
header {
  grid-column: 1 / 3;
  grid-row: 1;
}
article {
  grid-column: 2;
  grid-row: 2;
}
aside {
  grid-column: 1;
  grid-row: 2;
}
footer {
  grid-column: 1 / 3;
  grid-row: 3;
}
```

Design

- 有个什么principle 1：test with your intended audience!
- Gestalt:整体性，然后Whitespace，text width
- principle 2：on the web, use responsive design（就是屏幕适应）
  比如说：这个auto就是自适应

```
<!--General-->
Bad:  main { width: 800px; }
OK:  main { max-width: 800px;
       margin: 0 auto; }
```

还有用media queries：

```
@media screen and (max-width: 600px) {
    body {
        font-size: 14px;
    }
}
```

Exercise

display:grid就是默认将child都呈网格状排布了，有 `grid-templates-colmuns` 和 `grid-gap`
对于每一个child，有：

- By default, each child takes up the next free 1x1 space.
- `grid-row` 和 `grid-column`：
  - `span N` makes a child element N cells wide/tall
  - `M / N` positions the child absolutely from dividing line M to dividing line N (you can overlap or stack elements on top of each other this way if you want to).

大概练习就是：做grid的那道题（
主要是有一个特殊的要去自定义一下row和column，其他的基本就是跟着描述span就完事了

```
.y2-tb4 {
    grid-row: 3/5; /* SPE unit starts from row 7 */
    grid-column: 11/13;
    grid-row-end: span 2; /* SPE unit spans 2 rows */
}
```

自适应主要是用@media:

```
@media (min-width: 400px) and (max-width: 600px) {
    .container {
        grid-template-columns: repeat(2, 1fr); /* Two equally wide columns */
    }

    .featured {
        grid-column-end: span 2; /* Featured trees take up 2x2 space on the grid */
        grid-row-end: span 2;
    }
}
```

Week 8: Javascript

JavaScript: Basics

可以放到 `<body></body>` 的一堆元素后面：　　`<script></script>`，但是一般是分离开写的：

```
<script src="index.js"></script>
```

- **变量**：
  var：全局作用，一般不用了
  let：普通
  const：常量，一旦赋值无法改变（除非数组或者对象）声明时就初始化数值
- **原生数据类型**
  String，Number，Boolean，null,underfined
  `const username = "John"`
  `const age = 30;`
  `const isCool =true;`
  `const rate = 4.5;` （统一为number类型）
  `const x = null ;` (但是console打印出来是Object)
- **模版字符串**
  字符串连接：
  本来是 `console.log("my name is"+username);`
  但是可以写成这样
  `console.log('my name is ${username}`);`
- **字符串内置**
  `s.length`
  `s.toUpperCase()`
  `s.substring(0,5)` (分割字符串)
  `s.split(",")`

- **数组**

  构造函数

  `const numbers = new Array(1,2,3,4,5);`

  `const fruits = ["apples", "banna", 10];`

  可以有各种类型的

  `fruits.push("mango");` 数组末尾添加元素

  `fruits.unshift("C");` 数组头部添加元素

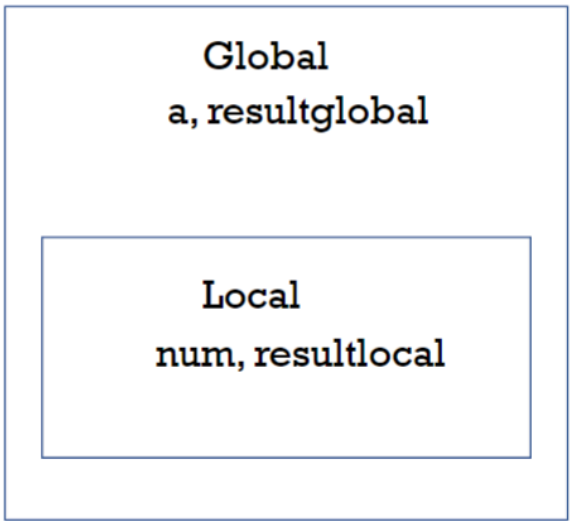
  `fruits.pop()` 删除末尾的元素

  `Array.isArray("hello"))` 检测是否是数组

  `fuits.indexOf("oranges");` 给index

- **Execution Context(执行上下文)**

```
var a =2;
function add (num){
  var resultlocal=num+10;
}
var resultglobal=add (a);
```




```
var a = 2;

function add (num) {

  var resultlocal = num + 10;

}

var resultglobal = add(a);
```

这个讲的就是作用域的东西

- 浏览器看js code->create a box to run code(Execution context)->Global&Local execution context
- 每当执行一段代码时，都会创建一个新的执行上下文。常见的执行上下文包括全局执行上下文（global execution context）和函数执行上下文（function execution context）
- 全局执行上下文是在执行 JavaScript 代码时创建的默认上下文，它代表了整个 JavaScript 程序的运行环境。而函数执行上下文则是在调用函数时创建的，每次函数调用都会产生一个新的函数执行上下文
- **Hoisting（提升）**

Hoisting（变量提升）是 JavaScript 中的一种行为，指的是在代码执行阶段，JavaScript 引擎会将**变量和函数的声明**提升到它们所在作用域的顶部，但是只提升声明，**不提升赋值**

比如说

```
console.log(x); // undefined
var x = 5;
```

```
1   console.log(x);
2
3   var x = 10;
```

```
PROBLEMS    OUTPUT    DE

ja21121@C02DWCVPML7H Te
Debugger attached.
undefined
```

```
1   console.log(x);
2
3   //var x = 10;
```

```
PROBLEMS    OUTPUT    DEBUG CONSOL

    at Function.executeUserEntryP
ja21121@C02DWCVPML7H Teaching 202
Debugger attached.
Waiting for the debugger to disco
/Users/ja21121/Documents/Teaching
console.log(x);
            ^

ReferenceError: x is not defined
```

左边是undefined->所以的意思是提升声明，而不是赋值
但是能不报错因为有hoisting这个行为
如果是两次赋值的话，只hoist第一次的

- var就是declaration hoisting
- functions就是value hoisting
- **Scope**
  这个就是作用域

Users > ja21121 > Documents > Teach

```
1   function first(){
2       console.log(a);
3       function second(){
4
5       }
6   }
7   var a = 10;
8   first();
```

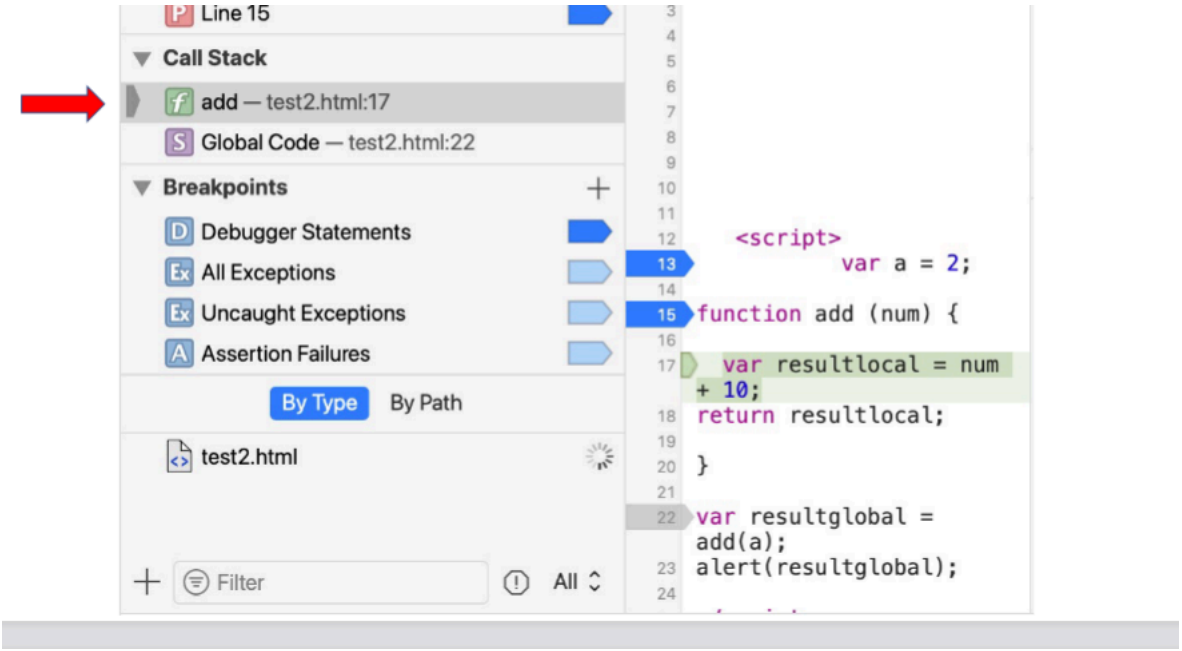
```
PROBLEMS    OUTPUT    DEBUG CONS

ja21121@C02DWCVPML7H Teaching 2
10
```

Users > ja21121 > Documents > Teaching

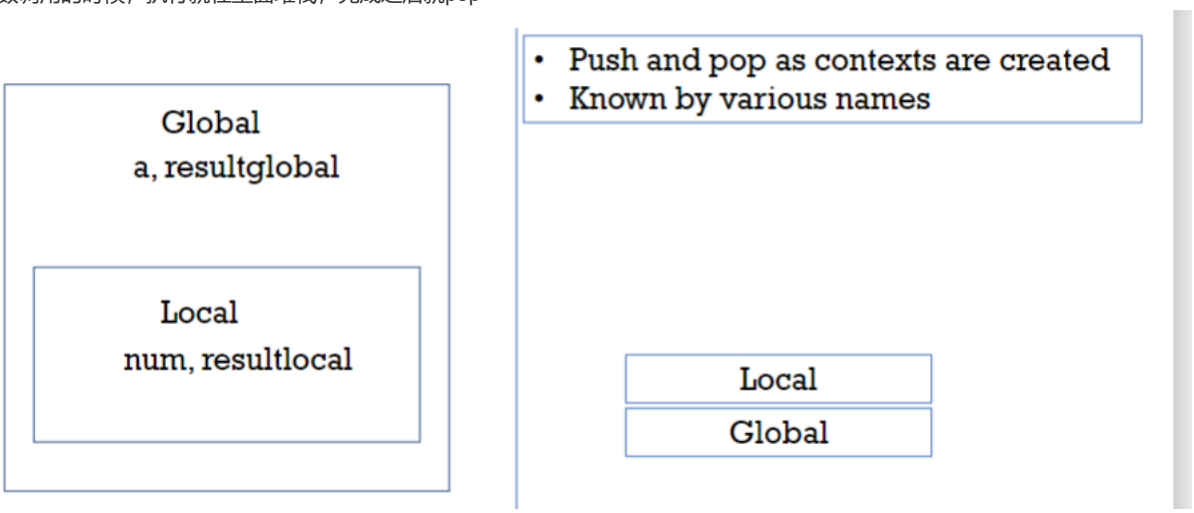
```
1   function first(){
2       console.log(a);
3       function second(){
4           var a = 10;
5       }
6   }
7
8   first();
```

```
PROBLEMS    OUTPUT    DEBUG CONSOL

    console.log(a);
                ^

ReferenceError: a is not defined
```

- **Call Stack**



函数调用的时候，执行就往里面堆栈，完成之后就pop



- **Syntax**

- break
- If…else
- for….
- function
- do….while
- var, let & const
- return
- switch
- throw
- try…catch
- ……
- arrays (You can have mixed arrays)
  - var example = [1, joe, 5.555]

And these too....

- Comparison operators : <,<=,>,>=, ==, ===
- Logical Operators: &&, ||, !
- ("3.0"=== 3) ("3.0"== 3)

We don't need to declare types

- Language makes the best guess
- It mostly gets it right, when it doesn't use parseInt for Integers.
- Many operators automatically convert types so happens in many operations.
- var name = prompt("What's your name?");
- var age = 20;

=== 严格相等运算符（Strict Equality Operator），用于比较两个值是否具有相同的值和相同的数据类型

- **Arrays,For loop**

```
var name = `` johndoe";

console.log(name.length);  = 7


console.log(name[0]); = 'j'
console.log(name[name.length]); = undefined
console.log(name[name.length-1]); = 'e'
```

他这里给了例子但是String实际上并不是Array，只是可以用这个形式访问

```
var example = [1, joe, 5.555]          A mixed array

for var element in
example {                          ➡   index
    console.log(element);
    }
for var element of
example {                          ➡   value
    console.log(element);
    }
```

for有两种， `in` 是返回的index？ `of` 返回的是value?

- **Anonymous Functions**

functions不一定要名字

`map()` 可以让某个函数apply（应用吧？）到array里面的每一个value

所以例子可以是：

```
var age = [20,25,30,35];
ages = ages.map(function(age) {
  return age + 10;
});
```

[30, 35, 40, 45]

或者有

```
var add = function(x, y) {
    return x + y;
};
```

- **Arrow Functions**

```
<script>
let hello = "";

hello = () => {
  return "Hello World!";
}

document.getElementById("demo").innerHTML = hello();
</script>
```

```
var add = (x, y) => x + y;
```

- **Events in JavaScript**
  - Anything we do on a web page is an event
    - Click on buttons, enter a text, hover our mouse
    - HTML pages can capture the event which can be passed to an event handler in JavaScript.

JS Objects (incl. JSON)

- 对象：属性加方法

```
const person = {
  firstName: "John",
  lastName: "Done",
  hobbies:["music","sports"],
  address:{
    street:"50",
    city:"boman"
  },
};
```

里面内容都用，隔开的
还可以这样，比如说

```
const { firstName, lastName }= person;
console.log(firstName);
-John
```

```
const people = [
...   { name: 'Alice', age: 30 },
...   { name: 'Bob', age: 25 },
...   { name: 'Charlie', age: 35 }
... ];

const stringtest = JSON.stringify(people);

console.log(stringtest);
[{"name":"Alice","age":30},{"name":"Bob","age":25},{"name":"Charlie","age":35}]

> const testBack = JSON.parse(stringtest);
> console.log(testBack);
[
  { name: 'Alice', age: 30 },
  { name: 'Bob', age: 25 },
  { name: 'Charlie', age: 35 }
]
```

- 对象数组：

```
const people = [
  { name: 'Alice', age: 30 },
  { name: 'Bob', age: 25 },
  { name: 'Charlie', age: 35 }
];
```

- JSON
  JSON: 轻量级的数据交换格式,基于JavaScript对象的语法，但是独立于编程语言,JSON由键值对构成，键值对之间使用逗号分隔，键与值之间使用冒号分隔，整个数据结构被包含在大括号 {} 中
  JSON 中的字符串需要使用双引号括起来,数字不用，JSON 不支持包含函数，而 JavaScript 对象可以包含函数
  转换到JSON：

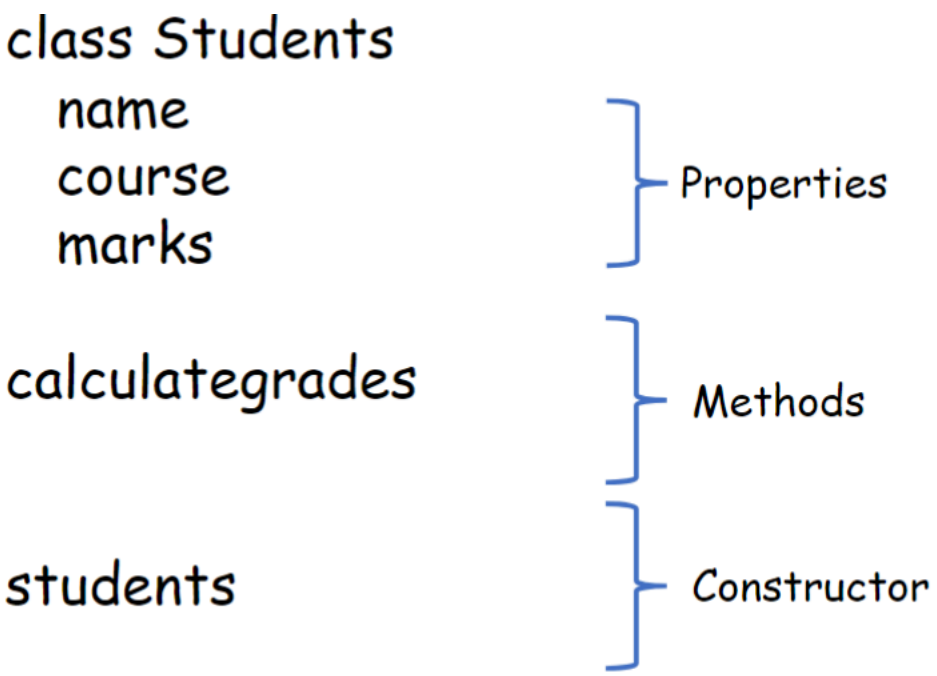
```
const peopleJSON=JSON.stringify(people);
```

JSON转换到JavaScript:

```
const peopleJSON=JSON.parse(people);
```

- JSON vs. XML
  JSON更好parse
  JSON可以用arrays

Object Oriented Programming



- Classes do not exist in the memory
- Instances of class known as objects do
- Classes have a specific method known as Constructors
- The constructors initialize values into objects

在，JavaScript里面，Objects轻量：

- JavaScript objects are lightweights
- Objects can be created without using a class using literals
- Objects can be created using functions with a constructor

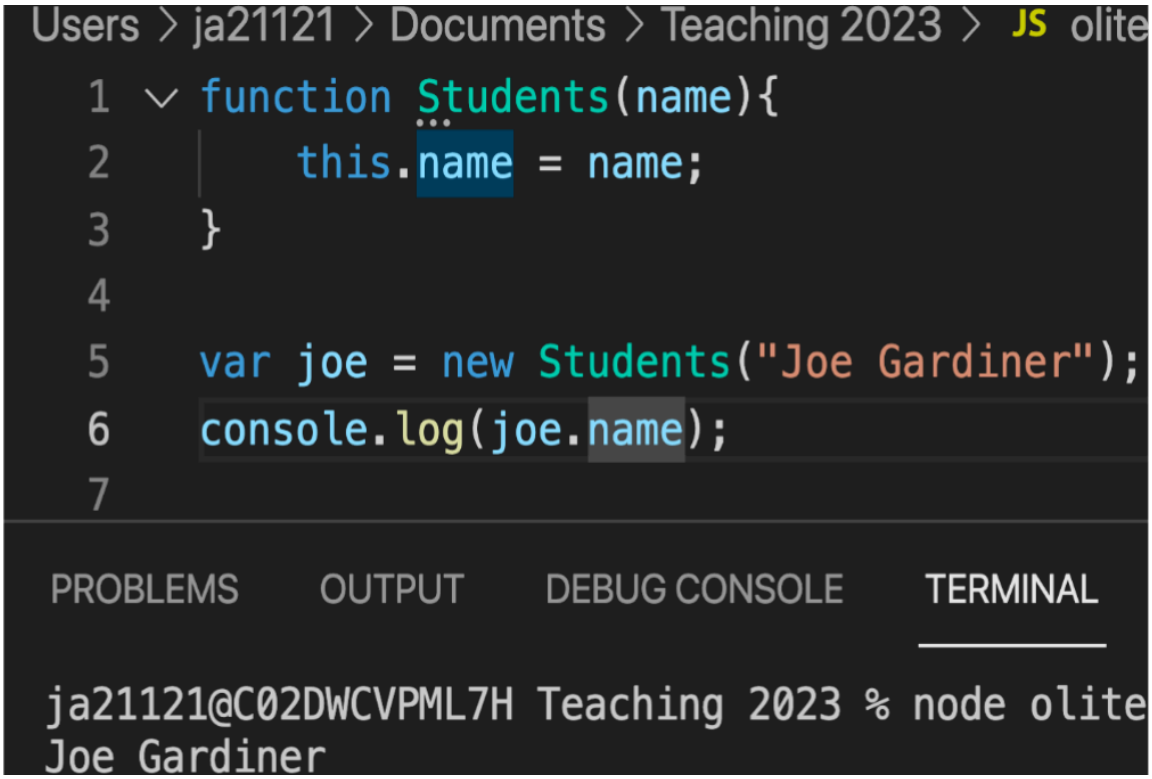
Literals

Users > ja21121 > Documents > Teaching 2023 >

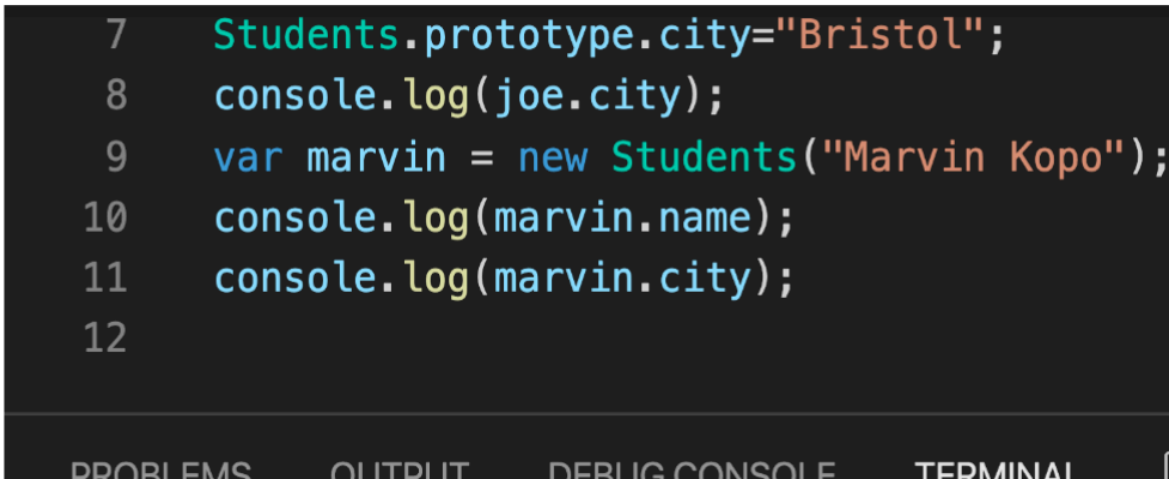
```
1  const Students = {
2      name: "Joe Gardiner",
3      course: "Software Tools",
4
5  }
6
7  console.log(Students.name);
8  console.log(Students.course);
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TER

ja21121@C02DWCVPML7H Teaching 2023 % nod
Joe Gardiner
Software Tools
ja21121@C02DWCVPML7H Teaching 2023 %
```

Functions



Inheritance



Asynchronous JavaScript（异步）

JavaScript是同步执行的，但是有时候要异步功能：callback&promises

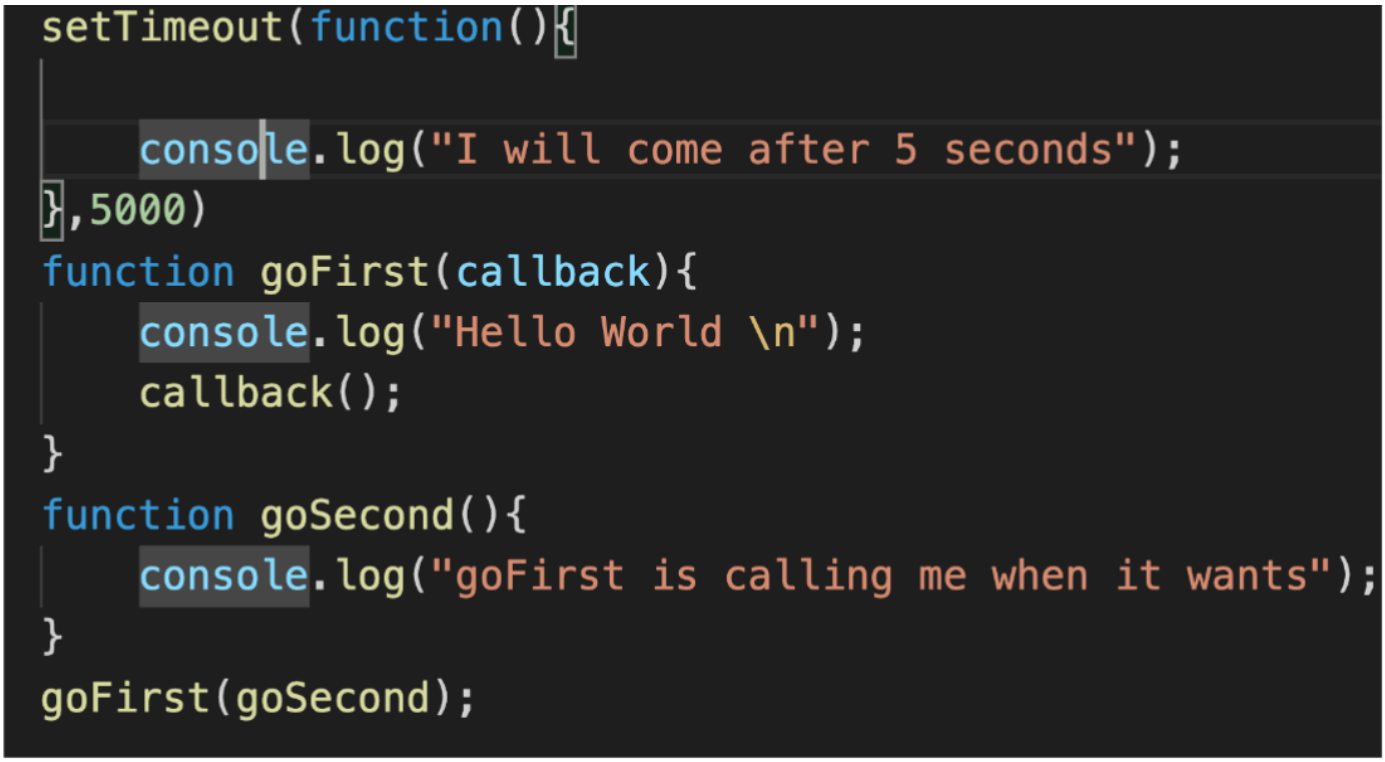
- **callback**:
  A callback is a function passed as an argument to another function
  比如原来有：

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}
function myCalculator(num1, num2) {
  let sum = num1 + num2;
  return sum;
}
let result = myCalculator(5, 5);
myDisplayer(result);
```

但是要调用两次

```
function myDisplayer(some) {
  document.getElementById("demo").innerHTML = some;
}

function myCalculator(num1, num2, myCallback) {
  let sum = num1 + num2;
  myCallback(sum);
}

myCalculator(5, 5, myDisplayer);
```

In the example above, myDisplayer is a called a **callback function**.
It is passed to myCalculator() **as an argument**.



但是可能出现call back hell:
"callback hell"（回调地狱）是指在异步编程中出现的一种情况，其中多个回调函数嵌套在一起，形成了深层次的回调嵌套结构，使得代码难以理解、难以维护、难以扩展的情况。
由此引入promise

- **promise**:
  Promise 对象有三种状态：pending（进行中）、fulfilled（已成功）和rejected（已失败），可以通过 then() 方法处理成功状态和 catch() 方法处理失败状态

```
// 定义一个返回 Promise 对象的函数
function wait(ms) {
    return new Promise((resolve, reject) => {
        setTimeout(resolve, ms);
    });
}

// 调用 wait 函数，等待 2 秒钟后输出消息
wait(2000)
    .then(() => {
        console.log('Two seconds have passed.');
    })
    .catch((error) => {
        console.error('An error occurred:', error);
    });
```

在这个例子中，wait() 函数返回一个 Promise 对象，它等待指定的时间（以毫秒为单位）后调用 resolve() 方法。然后我们使用 then() 方法来处理成功状态，即等待 2 秒钟后输出消息。如果发生错误，我们可以使用 catch() 方法来处理失败状态

- **forEach**:
  forEach 是 JavaScript 中数组的一个方法，用于遍历数组的每个元素并对其执行指定的操作

```
const numbers = [1, 2, 3, 4, 5];
numbers.forEach((number, index, array) => {
    console.log('Index: ' + index + ' Value: ' + number);
});
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

ja21121@C02DWCVPML7H Teaching 2023 % node foreach.js
Debugger attached.
Index: 0 Value: 1
Index: 1 Value: 2
Index: 2 Value: 3
Index: 3 Value: 4
Index: 4 Value: 5
```

Exercise

这个作业的目的是API to fetch DATA+add it to DOM of HTML

i. API：

- 接口？用于access and manipulate information，比如you send GET, PUSH and POST requests as normal and get data sent back in a suitable format (usually JSON or XML).
- standardized，security（就是标准化，适用于各种应用，然后只有授权的才给access）
- TMDB：
  key： `d7f89781d01251450fa7113ad85c5e1a`
  Token：

```
eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkN2Y4OTc4MWQwMTI1MTQ1MGZhNzExM2FkODVjNWUxYSIsInN1YiI6IjY2MjAyZmY4MDgxNmM3MDE0OWVkZWM5MiIsInNjb3BlcyI6WyJhcGlf
```

- Python：

```
PS C:\Users\Ada> python -m http.server
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
```

ii. JavaScript

1.设置开头，将HTML和JS连接起来

```
const API_URL =d7f89781d01251450fa7113ad85c5e1a
const IMG_PATH = 'https://image.tmdb.org/t/p/w500'
const SEARCH_API = 'https://api.themoviedb.org/3/search/movie?api_key=XXX&query="'

const IMG_PATH = 'https://image.tmdb.org/t/p/w1280'
const SEARCH_API = API_URL+'&query="'

const main =  document.getElementById("main");
const form = document.getElementById("form");
const search = document.getElementById("search");
```

这里的API_URL就是上面的key

然后下面的main，form，search用这个方法实现和HTML的互动，是 JavaScript 中的一个内置对象，代表当前 HTML 文档。它提供了访问和操作 HTML 文档内容的方法和属性

2. 修改getMovies function

这个是用用API来fetch movies的

```
function getMovies(url) {
    fetch(url)
        .then(response => response.json())
        .then(data => {
            showMovies(data.results);
        })
        .catch(error => {
            console.error('Error fetching movies:', error);
        });
}
```

async/await :async：用于声明一个函数是异步的。当函数声明为 async 时，它将隐式地返回一个 Promise 对象。在函数内部，可以使用 await 关键字来等待 Promise 对象的解析，并暂停函数的执行，直到 Promise 对象状态变为 resolved（已解决）。

await：只能在 async 函数内部使用。它用于等待一个 Promise 对象的解析结果。当遇到 await 关键字时，函数的执行将暂停，直到 Promise 对象状态变为 resolved（已解决）或 rejected（已拒绝）

则上面的修改为：

```
async function getMovies(url) {
    try {
        const response = await fetch(url); // Fetch data from the URL
        if (!response.ok) {
            throw new Error('Failed to fetch movies');
        }
        const data = await response.json(); // Parse response as JSON
        showMovies(data.results); // Call showMovies function with the results
    } catch (error) {
        console.error('Error fetching movies:', error); // Handle any errors
    }
}
```

然后这里加上

```
form.addEventListener('submit', (e) => {
    e.preventDefault();
```

然后debug是上面有两个

iii. API security

how? *using Environment Variables in Server-Side Code Only*

使用环境变量仅在服务器端代码中" 意味着将敏感信息（如 API 密钥、数据库密码等）存储在服务器的环境变量中，而不是硬编码到代码中。这样

做的好处是可以保护敏感信息，因为环境变量在大多数情况下不会被公开。当服务器端代码需要使用这些敏感信息时，它可以通过读取环境变量来访问它们，而不是直接在代码中使用它们。
->save our API key in a .env file and keep it secure and separate for the rest of the code on the server
我们刚刚是直接使用了API key，但是可以放在另一个.env文件中

Week 9: Web Scraping

Crawling the Web

- `wget <url>` 下载什么什么东西
- `wget -p <url>` -Download webpage with requisites, 下载网页及其相关的所有必需元素,包括stylesheets，images，包括**ROBOTS.TXT**
- **ROBOTS.TXT**通常位于网站的顶级目录下，用于指定搜索引擎爬虫访问网站时的行为规则。它是网站管理员用来控制搜索引擎爬虫对网站内容进行抓取和索引的一种方式。告诉搜索引擎爬虫哪些页面可以访问，哪些页面不应该被访问，以及爬虫在抓取网站时应该遵守的其他规则
- `wget -r -l N <url>` -Download webpage and linked pages下载网页本身+recursion
  - **-l N**: The level of recursion to permit. (Default if omitted is 5).
- `wget -m <url>` creating a local copy of a full website
  - `-w 1` : Wait 1 second between each request (to avoid annoying a server)
- Ethics
  - Does the site permit you to crawl this resource? (robots.txt)检查robot.txt查看是否允许爬虫
  - Is there a better way to get a copy 除了copy还有其他的更好地方式吗
  - Are you allowed to republish downloaded content? (e.g., copyright) 是否允许
  - Are you going to make something more public than it should be?

Exercise

i. 设置服务器
ii. 单页下载
iii. 抓取网站

其他：

- `-i` 选项用于指定包含要下载的URL列表的文件。
  `--force-html` 选项用于将非HTML文件（如PDF或文本文件）视为HTML文件。
  `--spider` 选项用于在不下载文件的情况下测试URL的可用性。
  `wget --spider -r -l inf -o wget_log.txt <当前正在阅读的网页URL>` 这将在wget_log.txt日志文件中记录所有链接的测试结果。通过查看日志文件，您可以检查是否有任何损坏的链接
- 可以使用--user-agent选项来指定wget发送的用户代理字符串
  `wget --user-agent="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36" <UR`
- `wget -r -l 1 http://example.com` 递归地下载 `http://example.com` 页面及其链接，并**限制递归深度为1**,这意味着只会下载 `http://example.com` **页面及其直接链接**，而不会继续下载链接中的其他页面
  `wget -p http://example.com` 将下载 `http://example.com` 页面，并**下载该页面所链接的所有外部资源**（如图像、CSS文件、JavaScript文件等），但不会递归下载其他页面```
- `-nc` 或 `--no-clobber` 选项用于防止wget覆盖已经存在的文件。如果文件已经存在，则wget不会重新下载该文件

BeautifulSoupw

BeautifulSoup 是一个功能强大且易于使用的工具，用于解析和操作 HTML 或 XML 文档，常用于网络爬虫、数据抓取和数据分析等任务中

Python requirements:

```
$ sudo apk add python3 py3-pip
```

BeautifulSoup itself:

```
$ pip install bs4
```

Test using interpreter.

一般是用file pointer来读取文件：

```
file = "cattax/index.html"
soup = BeautifulSoup(open(file, 'r')
```

Common desire in scraping: get the visible text on a page:

```
text = soup.get_text()
print(text)
```

Navigating page elements:

```
Select a page element by tag name:
>>> soup.title
Navigate the element heirarchy
>>> soup.body.main
Parent and child (+ ‘sibling’) relationships.
```

Finding page elements：

```
Avoid navigation, have BeautifulSoup find elements by a
specification.
>>> soup.find(‘strong’)
Find more than the first match:
>>> soup.find_all(‘strong’)
Can also refine search by ‘attrs=’ – see documentation!
```

Week 10: Practical Encryption

Introduction to OpenSSL: A Hands-On Lab using OpenSSL and encryption

- Understand the purpose and role of SSL Certificate Authorities in web security.
- Create a root Certificate Authority (CA) for local development.
- Generate SSL certificates for localhost domains.
- Configure web servers to use the self-signed SSL certificates for HTTPS.
- Test the secure communication with HTTPS-enabled web servers.

i. First we install the OpenSSL tool

ii. Use OpenSSL to encrypt files with symmetric encryption

创建了一个txt文件，然后使用openssl加密： `openssl aes256 -in mytext.txt -out mytext.enc`

但是这里会让你写密码，不想输入密码就是查询了chatGPT是 `openssl aes256 -in mytext.txt -out mytext.enc -pass pass:`

加密文件和非加密文件有什么区别呢?

加密之后打开是: `Salted__D`  `^H*Q}x`  `;``j`  `^B^R`  `A*`  `C^A` 

由于加密之后实际上是二进制文件，包含有些不可见内容，所以要全部变成txt课件形式可以

`openssl enc -base64 -in encrypted_file.enc -out encrypted_file_base64.txt` 用所谓的base64来进行加密

Is the file size larger than before? If so, why?

在大多数情况下，加密后的文件大小会略微增加。这是由于加密算法的性质以及可能添加到文件中的任何元数据或填充导致的。

How to decrypt using OpenSSL

加个 `-d`

```
openssl aes256 -d -in encrypted_file.enc -out decrypted_file.txt -pass pass:yourpassword
```

3. Use OpenSSL to encrypt files with asymmetric(非对称加密) encryption(就是公钥和私钥那个东西)

i. Generate Key Pairs: Use genrsa to generate a 1024-bit public-private key pair.

```
openssl genrsa -out private_key.pem 1024
```

ii. Distribute the Public Key: Extract the public key from the private key and save it to a .pem file.

```
openssl rsa -in private_key.pem -out public_key.pem -pubout
```

iii. Exchange Public Keys: Share your public key (public_key.pem) with the person you want to communicate with and obtain their public key as well.

iv. Create Encrypted Message: Encrypt a message using the recipient's public key.

```
openssl rsautl -encrypt -pubin -inkey recipient_public_key.pem -in plaintext.txt -out encrypted_message.bin
```

v. Decrypt the Message: The recipient can decrypt the message using their private key.

```
openssl rsautl -decrypt -inkey private_key.pem -in encrypted_message.bin -out decrypted_message.txt
```

why we use HTTPS?

使用 HTTPS 的网站可以向客户保证其连接是安全的。 HTTPS（即安全超文本传输协议）是 HTTP 的受保护版本，HTTP 是用于在浏览器和您正在访问的网站之间传输数据的协议。优先保护用户的隐私和数据安全对于向使用我们在线开发的网络服务的人们灌输信任至关重要

why we use HTTPS localy

我们需要创建一个与生产非常相似的开发环境。这种做法有助于避免将来在线部署 Web 应用程序或网站时出现问题。

4. Configure and utilize Nginx with HTTPS for creating a localhost web app and sign a certificate request with OpenSSL.

配置并使用带有HTTPS的Nginx创建本地主机的Web应用程序

- CSR）certificate signing request 包含三个东西
  the public key
  the system that triggers the request
  the signature of the request.

i. Generate RSA Private Key
跟上面一样，先创造一组

```
openssl genrsa -out cert.key 2048
```

ii. Create CSR
创造一整个CSR环境

```
openssl req -new -key cert.key -out cert.csr
```

- Create the Configuration File
  创建一个名字叫 `openssl.cnf`

```
[#]Extensions to add to a certificate request

basicConstraints       = CA:FALSE

authorityKeyIdentifier = keyid:always, issuer:always

keyUsage               = nonRepudiation, digitalSignature, keyEncipherment, dataEncipherment

subjectAltName         = @alt_names

[ alt_names ]

DNS.1 = your hostname
```

然后签名+输入

```
$ ~/store-csr$ openssl verify -CAfile rootCert.pem -verify_hostname your hostname cert.crt
```

Nginx Configuration

iii. How to use OpenSSL to test an SSL connections

- s_client：OpenSSL的s_client专门用于测试SSL连接。它提供了关于SSL握手过程、证书细节和SSL/TLS协商的详细信息。它特别适用于调试SSL相关的问题和验证服务器配置。
- netcat（nc）：Netcat是一个通用的网络工具，可用于各种用途，包括与服务器建立TCP或UDP连接。虽然netcat可以用于连接到启用SSL的服务器，**但它不像OpenSSL的s_client那样提供详细的SSL/TLS信息**。

- telnet：Telnet是一种网络协议和工具，允许用户通过网络与远程计算机通信。s_client：OpenSSL的s_client专门用于测试SSL连接。它提供了关于SSL握手过程、证书细节和SSL/TLS协商的详细信息。它特别适用于调试SSL相关的问题和验证服务器配置。
  netcat（nc）：Netcat是一个通用的网络工具，可用于各种用途，包括与服务器建立TCP或UDP连接。虽然netcat可以用于连接到启用SSL的服务器，但它不像OpenSSL的s_client那样提供详细的SSL/TLS信息。
  telnet：Telnet是一种网络协议和工具，允许用户通过网络与远程计算机通信。Telnet可以用于与服务器建立TCP连接，但它不支持SSL/TLS加密。因此，它不能用于测试SSL连接。。因此，它不能用于测试SSL连接。

Introduction to PGP encryption

PGP（Pretty Good Privacy）
创建密钥对： `gpg --generate-key`
发送 `$ gpg --output ~/mypub.key --armor --export youremail@mail.com`
导入公钥： `gpg --import external_public_key.asc`
签名公钥:可选地，您可以对导入的公钥进行签名，以表示对该密钥的信任。您可以使用--sign-key参数对公钥进行签名。
`gpg --sign-key user@example.com`
导出签名后的公钥: 如果您签署了外部用户的公钥，可以将签名后的公钥导出并与他们共享，以加强对彼此的信任 `gpg --output signed_external_public_key.asc --export user@example.com`
解密消息 `gpg --decrypt received_message.asc`
导出公钥 `gpg --armor --export email > my_public_key.asc`
发送到公钥服务器 `gpg --send-key email`
搜索用户公钥 `gpg --search user@example.com`
导入用户的公钥 `gpg --recv-keys key_id`
验证公钥的真实性 `gpg --fingerprint user@example.com`

Past Paper

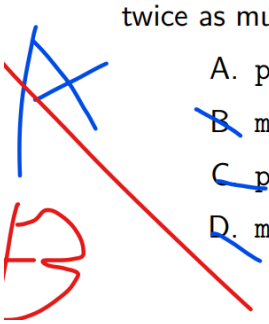
Paper 1

Q29

Q29. You have been tweaking the style of an `<img>` element so that it has a visible border. You want to create some visual space between the border and other elements on the page, with twice as much space below as there is above. Which of these rules should you use?

A. `padding: 10px 5px 20px 5px;`
B. `margin: 8px 8px 16px 8px;`
C. `padding: 16px 8px 16px 8px;`
D. `margin: 5px 5px 5px 10px;`

[1 mark]



四个元素是上左下右，顺时针的顺序，这个倒是答对了
margin的关键词可能是因为是border(
margin-border-padding-content

Q30

Q30. Consider the following stylesheet:

```
html { font-size: 14px; }
p {font-size: 0.8em; }
div {font-size: 1.2em; }
```

Imagine this stylesheet is applied to a page which contains a <p> element inside a element. What would be the size of the text in the <p> element, as measured in px?

A. 14

B. 12

C. 16.80

D. 13.44

[1 mark]

the <p> size is defined relative to the that contains it, which modifies the root font-size
实际上还是考了那个p是在div里面的（?
所以答案应该是 `14*1.2*0.8`

Q32

Q32. Karen is using `wget` to download a website `example.com`. The index page of `example.com` contains the following HTML:

```
<!DOCTYPE html>
<html>
<head>
      <meta charset="UTF-8">
      <link type="text/css" href="style.css" rel="stylesheet">
      <title>Example Page</title>
</head>
<body>
<h1>Example Page</h1>
<p>See the <a href="./about.html">about</a> page for more about this
site, or check out the <a href="https://www.iana.org/domains/example">
IANA resources</a> for more about example pages.</p>

</body>
</html>
```

If she downloads the page by invoking

```
wget -r -l 1 http://example.com
```

How many files will be retrieved in total?

A. 1

B. 2

C. 3

D. 4

首先的首先，*retrieved*是取回的意思，貌似和web scraping相关还没写

The index page, the about.html page and the style.css from the header will be retrieved.

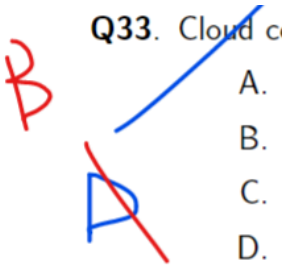
The main HTML file (index.html or whatever the default name is)

The linked stylesheet (style.css)

The linked about page (about.html)

***-l 1** means no further recursion will proceed on linked pages, and wget will not follow the link to the external domain. Controlling wget recursion is part of the first set of web scraping exercises*

Q33



Q33. Cloud computing involves:

A. Moving data to near-edge devices such as routers.

B. The on-demand availability of resources such as storage and computational power.

C. A distributed architecture making use of a shared hash table to achieve consensus.

D. All of the above.

[1 mark]

Q34

Q34. Which of the following statements are true?

1. JavaScript is a interpreted language
2. Javascript is a compiled language

A. Both are correct

B. Only 2 is correct

C. Both are wrong

D. Only 1 is correct

PPT上说的是script language，问了chatgpt是都对= =

然后stack overflow是 interpreted ->Parsing -> Compiling -> Executing

所以是JavaScript code is initially interpreted before any execution begins，interpreted

Q37

Q37. What will be the output of `console.log(a);` in each of the cases?

1
```
function hello(b){
  console.log(a);
}
var a = "John";
hello();
```

2
```
function hello(b){
 console.log(a);
    function Name(c){
    var a = "John";
    }

}
hello();
```

A. John & John

B. John & Undefined

C. Undefined & John

D. John & Reference Error

i. 在JavaScript中，函数的参数是可选的。即使在函数定义时声明了参数，但在调用函数时没有提供参数，函数仍然可以正常执行

ii. 跟作用域有关，所以是reference error

Q38

Q38. What will be the output of `console.log();` below?

```
const obj1 = {Name: "John", Age: 20};
const obj2 = {Name: "John", Age: 20};

console.log(obj1===obj2);
```

A. True

B. Undefined

C. False

D. None of the above

昨天看的以为是 === 是函数类型和值完全一样就可以了，但实际上还可以是指引用是否完全一致

并且对于对象而言， === 和 == 都可以指对象的引用

Paper 2

Q23

Q23. Consider the following HTTP request as received by a web server:

```
GET /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi
```

Assuming the request headers can be trusted, which of the following best describes what has happened to cause this request?

A. An internal server error has occurred.
B. Someone accessed `http://www.example.com/hello.txt` with curl
C. Someone accessed `www.example.com` in Safari.
D. The client has made an error in forming the request.

- **请求行（Request Line）**：`GET /hello.txt HTTP/1.1`
  - 这表示客户端（可能是浏览器或者其他HTTP客户端，比如curl命令行工具）正在请求服务器上的`/hello.txt`文件。
- **User-Agent头部**：`curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3`
  - 这个头部提供了发送请求的客户端的相关信息。在这种情况下，它显示了请求是由curl命令行工具发送的，版本是7.16.3，还包含了一些其他库和组件的信息。
- **Host头部**：`Host: www.example.com`
  - 这个头部指定了服务器的主机名（或IP地址），这是请求的目标主机。
- **Accept-Language头部**：`Accept-Language: en, mi`
  - 这个头部指示客户端所接受的语言偏好。在这种情况下，它表明客户端希望接受英语（en）和一种语言标识符（mi）。

因为请求行中的路径指定了访问的资源（/hello.txt），而User-Agent头部显示了请求是由curl发送的。所以，最合理的解释是有人使用curl工具来请求 http://www.example.com/hello.txt

Q25

Q25. Consider this snippet of HTML:

```
<p>
  I'm creating a link to
  <a
    href="https://www.mozilla.org/en-US/"
    title="The best place to find more information about Mozilla's
          mission and how to contribute">the Mozilla homepage</a>.
</p>
```

Which of the below reflects the text you would click on in the browser to follow the link?

A. The whole paragraph.
B. The contents of the anchor tag.
C. The href attribute.
D. The title attribute.

[1 mark]

是 `The Mozillar homepage` 但是不知道这个是什么（
原来是contents of the anchor tag啊
Anchor tag（锚点标签）是HTML中的一个标签，用于创建超链接。在HTML中，锚点标签用元素表示

Q26

Q26. Client-side validation of HTML form fields...

A. Protects the server from malicious requests.
B. Provides a helpful aid for users to fill in a form correctly.
C. Cannot be disabled in most browsers.
D. All of the above.

Client-side validation was part of the required reading in the second session. It provides no protection to the server, and is easily disabled in most browsers.
???
required reading?
这句话是在讨论客户端验证（Client-side validation）。客户端验证是在用户的设备（通常是浏览器）上执行的验证，用于验证用户输入的数据是否符合要求，通常在提交表单之前进行。这可以提供即时的反馈给用户，帮助他们更快地发现并纠正输入错误。
就是输入表单的时候判断是不是数字哪些东西吧

Q28

Q28. A Thymeleaf application is acting as a frontend to a database of student grades. You see that one method in the `Controller.java` file has an annotation:

```
@GetMapping("/student/ids")
```

What do you know about the method this annotation is attached to?

A. It accepts a student ID as a parameter.
B. It returns a student ID.
C. The method will not be called because the mapping is malformed.
D. None of the above.

[1 mark]

这个原文说的是

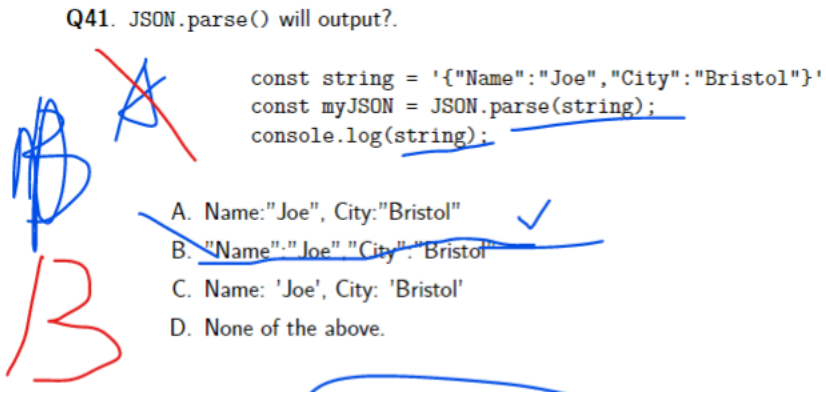
this method should be called to **reply** to a HTTP GET request for the provided path (there is of course also a @PostMapping and so on).

所以选了B，但其实好像不会return东西的

```
@GetMapping("/html")
public ResponseEntity<Resource> htmlPage() {
    Resource htmlfile = loader.getResource( location: "classpath:web/page.html");
    return ResponseEntity
        .status(200)
        .header(HttpHeaders.CONTENT_TYPE, ...headerValues: "text/html")
        .body(htmlfile);
}
```

SpringMVC相关：*@GetMapping("/student/ids")中的"/student/ids"部分指示了控制器方法应该处理的URL路径，但是没有指定方法的参数。在Spring MVC中，使用花括号（curly braces）{}表示路径变量，例如@GetMapping("/student/{id}")，其中{id}是一个占位符，表示接受一个名为"id"的路径变量*
由于没有花括号所以其实没有传入任何东西

Q41



就是想多了，本身是JSON形式，转变成JavaScript不会自动删掉的x虽然
确实有

```
const obj = {Name: "Joe", City: "Bristol"};
const jsonString = JSON.stringify(obj);
console.log(jsonString);
```

结果是 `{"Name":"Joe","City":"Bristol"}`
但是如果本身是String->JSON.parse反而结果是

```
{
  "Name": "Joe",
  "City": "Bristol"
}
```

Q44

Q44. What happens when you put this in a render() method?

```
let names = ["John","Matt","Joseph"];
function App() {
  return (
<div className="App">
{names.map(it => <p>{names[1]}</p>)}
</div>)
}
```

A. Matt

B. Matt
Matt
Matt

C. Joseph

D. None of the above.

是React的渲染方法，render()实际上是
在这个方法中，有一个数组names，包含了三个字符串元素："John"，"Matt"和"Joseph"。然后，在return语句中，通过map方法遍历names数组的每个元素，并返回一个

元素，其中的文本内容是names[1]，也就是数组中索引为1的元素
但是我觉得更多是和map相关，map会映射所有的元素，所以会返回三次

Mock Test

Q1&2

QUESTION 1

Type the name of the HTML tag used to create a line break without ending the paragraph. (Type no spaces, nor angled brackets.)

br

QUESTION 2

Complete the following HTML tag with one word (no spaces or other punctuation)

dogs

so that it creates a link that, when clicked, loads the page `/dogs`.

这个可以直接找到
一直在考的超链接

Q3

QUESTION 3

Which of the following HTML fragments will make the words "Piano", "Guitar", and "Trumpet" appear as bullet points?

- (●)
```
<ul>
  <li>Piano</li>
  <li>Guitar</li>
  <li>Trumpet</li>
</ul>
```
- ( )
```
<ul>
  <td>Piano</td>
  <td>Guitar</td>
  <td>Trumpet</td>
</ul>
```
- ( )
```
<ol type="A">
  <li>Piano</li>
  <li>Guitar</li>
  <li>Trumpet</li>
</ol>
```
- ( )
```
<table>
  <li>Piano</li>
  <li>Guitar</li>
  <li>Trumpet</li>
</table>
```

QUESTION 4

他说bullet point就是unordered
然后里面都是li

Q4

QUESTION 4

Consider the following sentence:
"Click here for more information."
Which of the following HTML snippets would make the word 'here' link to the page 'www.example.com'?

- ( )
```
<p>
  Click
  <href="www.example.com">here</href>
  for more information.
</p>
```
- ( )
```
<p>
  Click
  <a href="www.example.com">here<a />
  for more information.
<p>
```
- (●)
```
<p>
  Click
  <a href="www.example.com">here</a>
  for more information.
</p>
```
- ( )
```
<p>
  Click
  <a url="www.example.com">here</a>
  for more information.
</p>
```

也是考的超链接

Q5

QUESTION 5

Which of the following is not a property of the CSS Box Model?

- ( ) margin
- ( ) border
- ( ) padding
- (●) column

这个也很好考，就是box model margin-border-padding-content

Q6

QUESTION 6

Which of the following CSS rules is *not* correct?

- ○ #id { property: value }
- ◉ $div { property: value }
- ○ .class { property: value }
- ○ tag { property: value }

但在标准的 CSS 中，并没有使用 $ 符号作为选择器的一部分
所以应该是#div

Q7

QUESTION 7

Are the following statements about JavaScript variables true or false?

- true Variables with block scope are declared using the let keyword.
- true The keyword const is used to to declare variables that you should not change after they are assigned.
- false A variable declared outside a function definition using the let keyword is a global variable.
- true A variable declared outside a function definition using the var keyword is a global variable.

- var是什么function scope，如果是在函数外部就是全局
- var室友hoisting的，而且可以被重新赋值和声明
- let是block scope的生命变量，block是任何一对被｛｝包围的区域，但是不会被hoisting，只能在代码块内部访问
- let 变量可以在同一作用域内重新赋值，但不能被重新声明
- 确实const是常量
- 就是上面说的，之能是var变成全局

Q8

QUESTION 8

Type the output logged to the console when the following code is run. (Do not type any spaces.)

```
var x;
console.log(typeof(x));
```

Undefined

而且是大写？？小写还不认 但其实node.js打印出来时undefined

Q9

QUESTION 9

Given the following JavaScript object, which of the following expressions *cannot* be used to determine whether the object has the property description?

```
let flowers = {
  type: "Spring flowers",
  region: "Europe",
  examples: [
    {
      name: "Monkey Orchid",
      appears: "April",
      colour: "lilac",
      habitat: "Dry, sunny grasslands on hills"
    },
    {
      name: "Common Rock-Rose",
      appears: "April",
      colour: "Yellow",
      habitat: "Forests, dry grasslands, waysides"
    },
    {
      name: "Star of Bethlehem",
      appears: "May",
      colour: "White",
      habitat: "Cultivated land, vines, grass lands"
    },
    {
      name: "Field Poppy",
      appears: "May",
      colour: "Red",
      habitat: "Along walls, hedges, paths"
    }
  ]
}
```

- ○ flowers.hasOwnProperty("description");
- ◉ flowers.contains(description);
- ○ if ('description' in flowers) { return true; }
- ○ if (flowers.description != undefined) {

在 JavaScript 中，contains 方法通常用于检查一个字符串是否包含另一个字符串，但它不是字符串原型链上的方法

Q10

QUESTION 10

The following is *not* a valid JSON object.

```
{
  name = David,
  prizes_won = 3
}
```

What needs to change to make it valid? Mark the following as true or false.

- true The keys (name, prizes_won) need to be double-quoted.
- true The string value David needs to be double-quoted.
- false The numeric value 3 needs to be double-quoted.
- true The equals sign needs to be replaced by a different character.

这个考的是JSON和JavaScript互相转化

- 所有的String都会被“”，但是int不会
- 只有： 而不是等号