**HTTP**

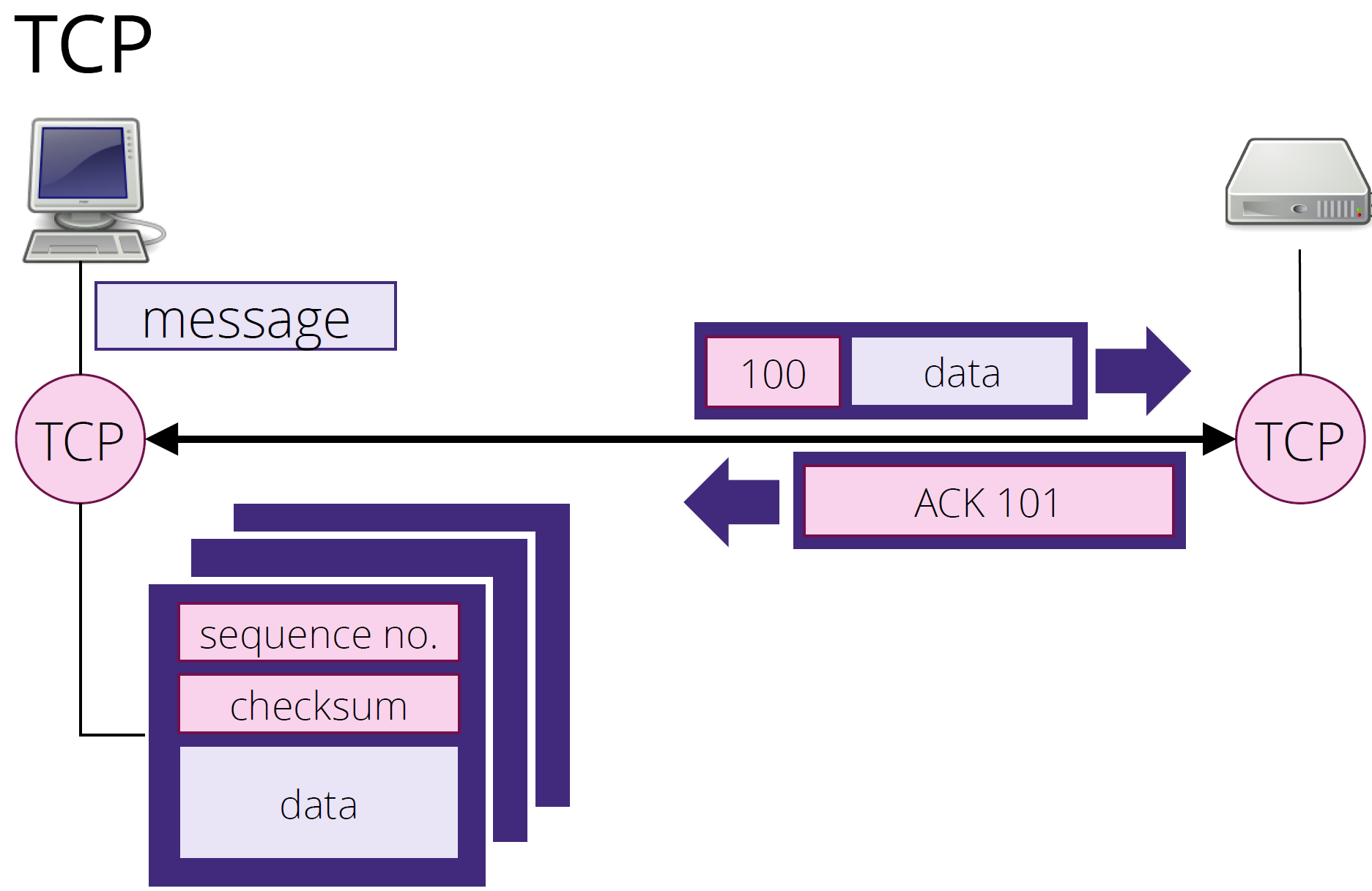
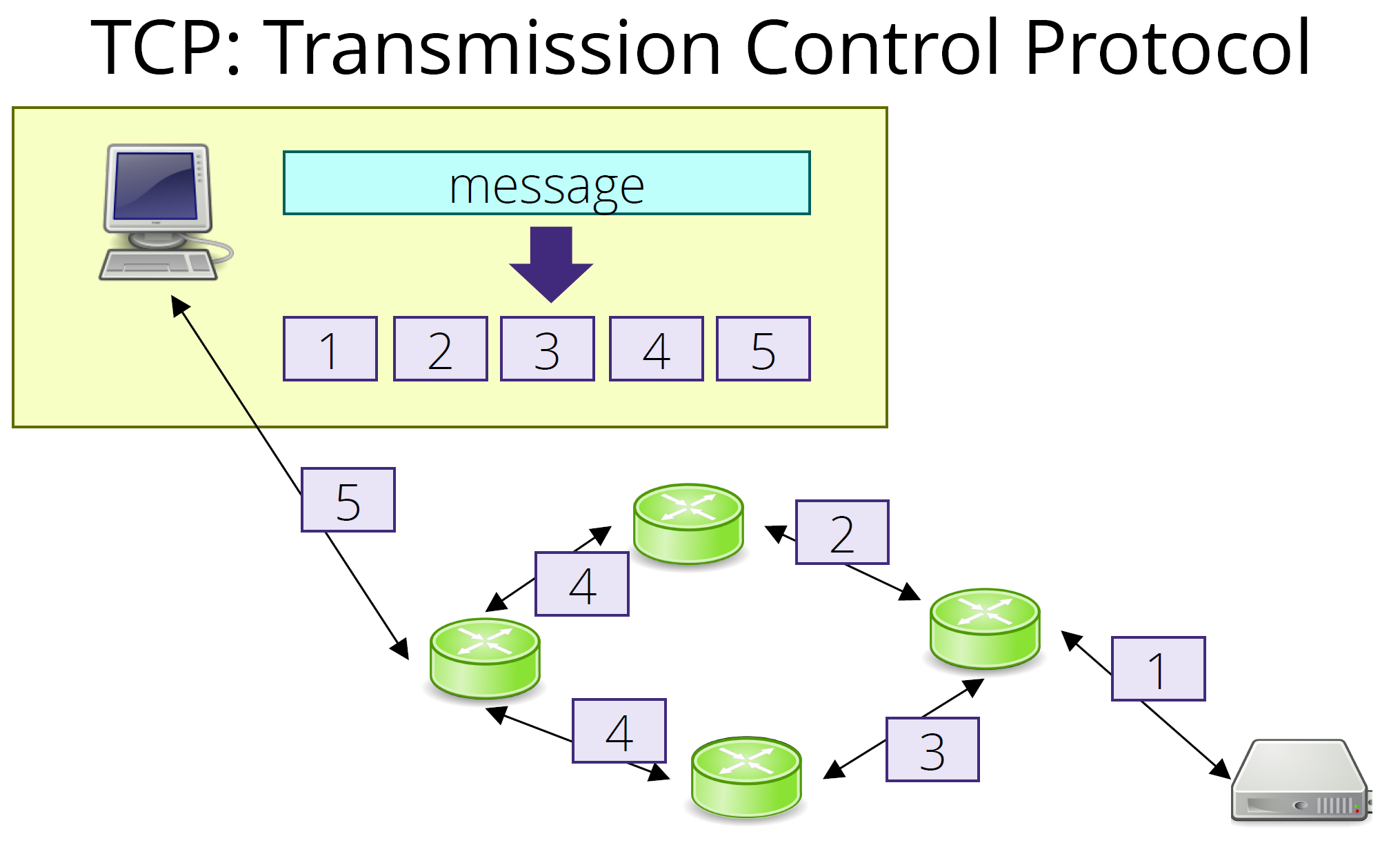
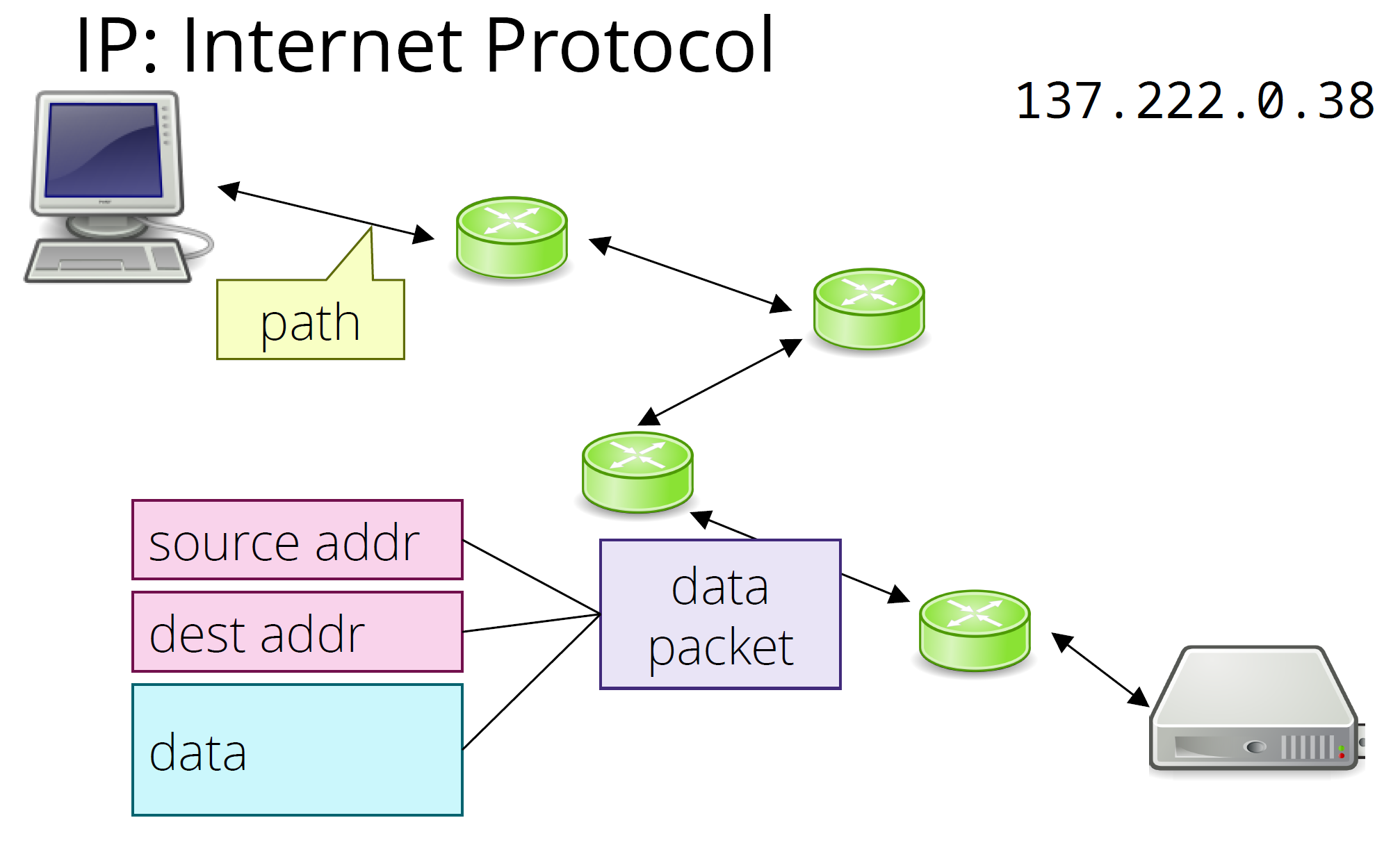
超文本传输​​协议。由欧洲核子研究中心的 Tim Berners-Lee 于 1989 年开发。version 1.1, RFC 7230-7235

协议是系统协作组件如何交互的计划。

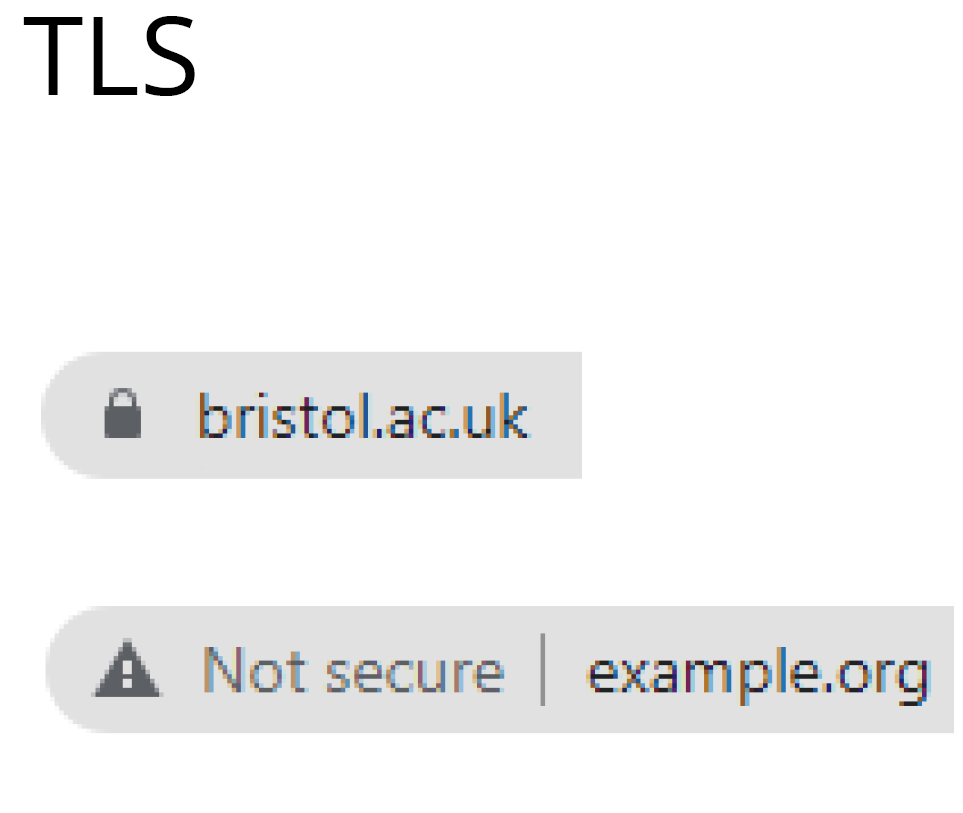
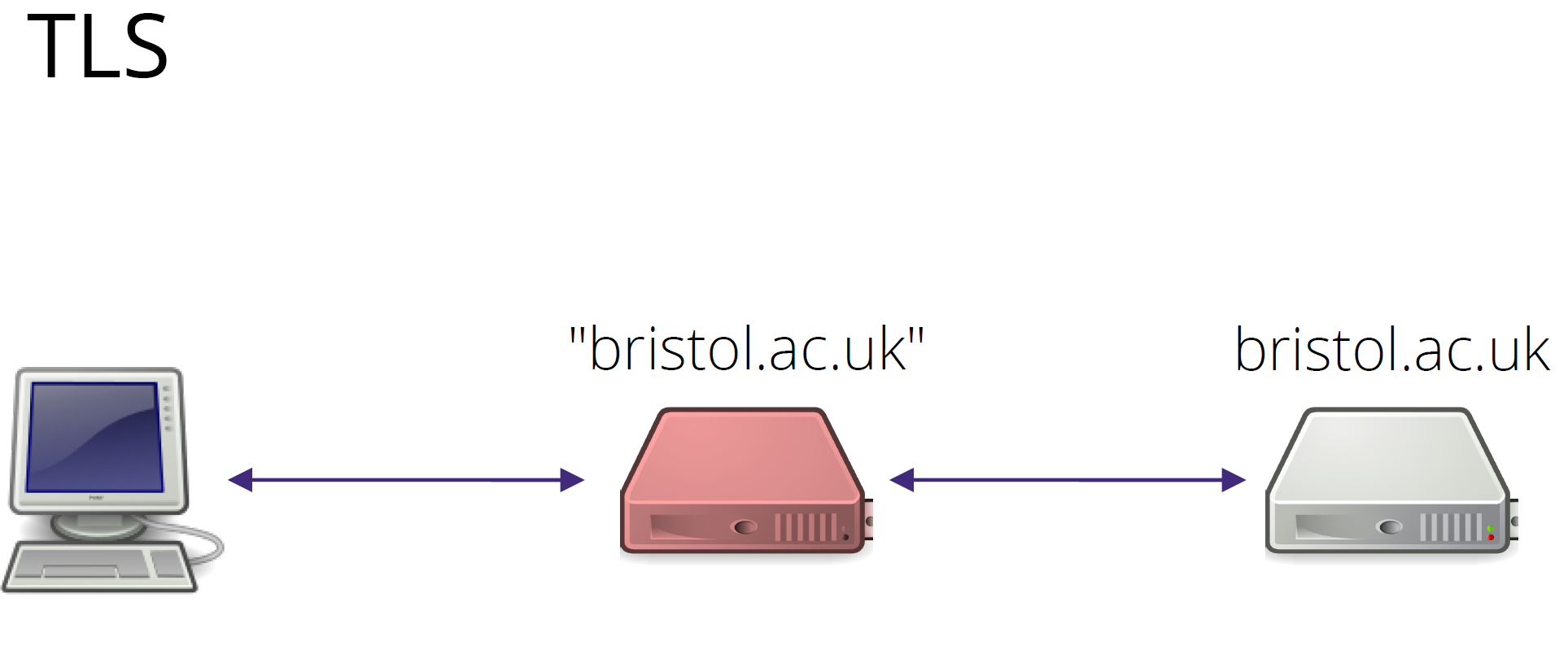
简单示例：**客户：**给我200号区块**服务器：给你：*0A 2F EE.../*** 抱歉，我找不到该块。/我不知道您在问什么。/我认为您指的是块#201，即*0A 2F EE...***协议：状态通信**

**OSI模型**

7：Application （您在这里）6：Presentation （例如，TL​​S）5：Session （例如，SOCKS）4：Transport （例如TCP，UDP）3：Network （例如 IP）2：Data Link （例如MAC）1：物理（例如蓝牙、以太网PHY）



TCP ports：80HTTP，443TLS，22SSH，8000, 8080, … development (unofficial)



**HTTP 状态通信**：请求和响应*标头*、状态代码

需要理解的是：元数据是 HTTP *协议*的重要组成部分。然而，*元数据与数据*是相互独立的。超文本文档并非*必须*通过 HTTP 传输。

**HTTP 消息结构**

start-line \*( header-field CRLF ) CRLF [ message-body ]

start-line：request-line（对于请求）或status-line（对于响应）。header是可选的，内部是field-name : field-value。

**HTTP 请求**

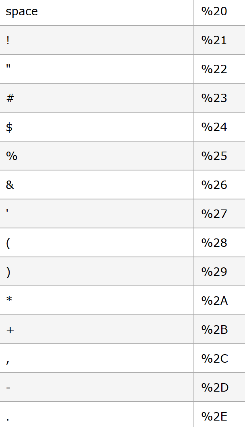
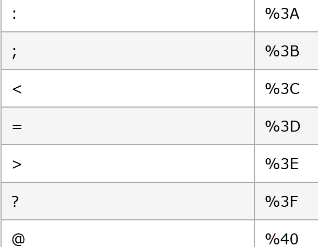
method SP request-target SP HTTP-version CRLF。SP= 空格。CRLF= 回车。HTTP-version很少改变（尽管它可以！）。

**HTTP 方法：**GET ：检索目标资源的副本POST ：向目标资源提交有效载荷数据HEAD ：检索相应 GET 的元数据PUT ：用有效载荷替换目标资源DELETE：删除目标资源**。**实际上，许多服务器不会实现或将忽略请求DELETE而PUT 倾向于使用POST请求的自定义语义。

**REST** Representational State Transfer：POST /files/README.txt?action=delete；GET /files?name=README.txt

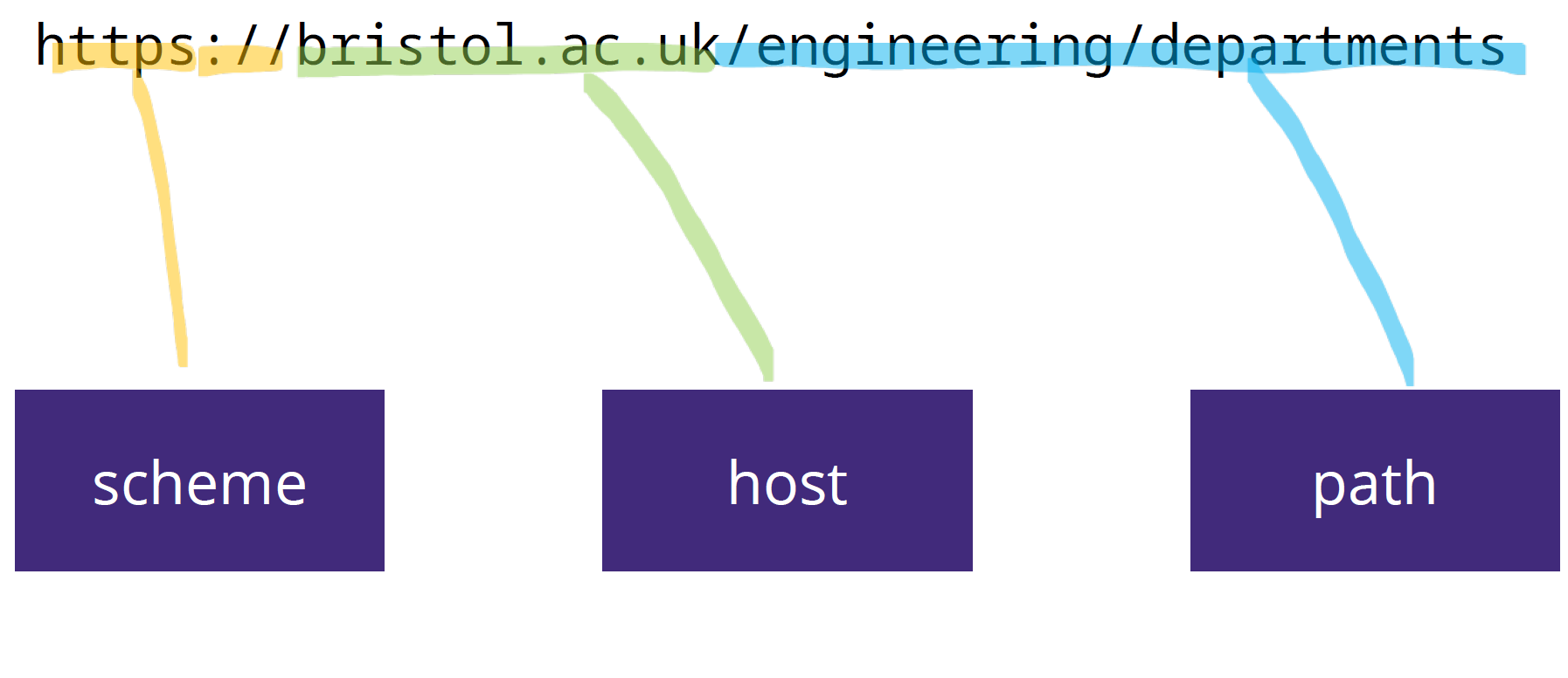
**HTTP 请求目标**RFC 3986

与**URI**（统一资源标识符）的path和query相关。pathA hierarchical identifier for the targeted resource.：/或/files/index.html或/user/george/。query：?name=welcome&action=view An optional mapping of values to specified parameters.

Scheme：A reference to the specification for the remainder of the URI.协议标识：告诉客户端（如浏览器、应用程序）如何处理该URI。http: 和 https:用于访问网页资源（超文本传输协议）。ftp:用于文件传输协议。mailto:用于发送电子邮件。file:指向本地文件系统资源。data:直接在URI中嵌入数据（如Base64编码的图片）。tel:用于拨打电话号码。urn:统一资源名称（不依赖位置的持久标识符）。

自定义Scheme（如 spotify:、zoommtg:）。

fragment ： URL 中 # 后面的部分（例如 #section1），用于指定页面内的某个锚点或位置。不会发送到服务器，仅由浏览器处理（如滚动到对应 HTML 元素的 id 或 name 属性处）。示例：https://example.com/page.html#footer → 直接跳转到页面底部。



**HTTP 状态行**

status-line = HTTP-version SP status-code SP reason-phrase CRLF

status code具有特定含义的3位代码reason-phrase解释状态代码的描述

**HTTP 状态代码**1xx信息（100 Continue）。2xx成功（200 OK201 Created,）3xx重定向（301 Moved Permanently）4xx客户端错误（400 Bad Request, 403 Forbidden, 404 Not Found）5xx服务器错误（500 Internal Server Error）

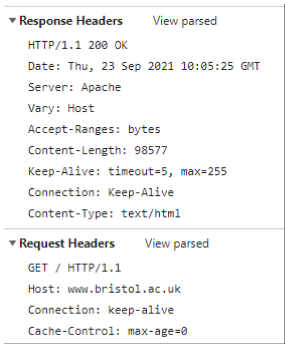
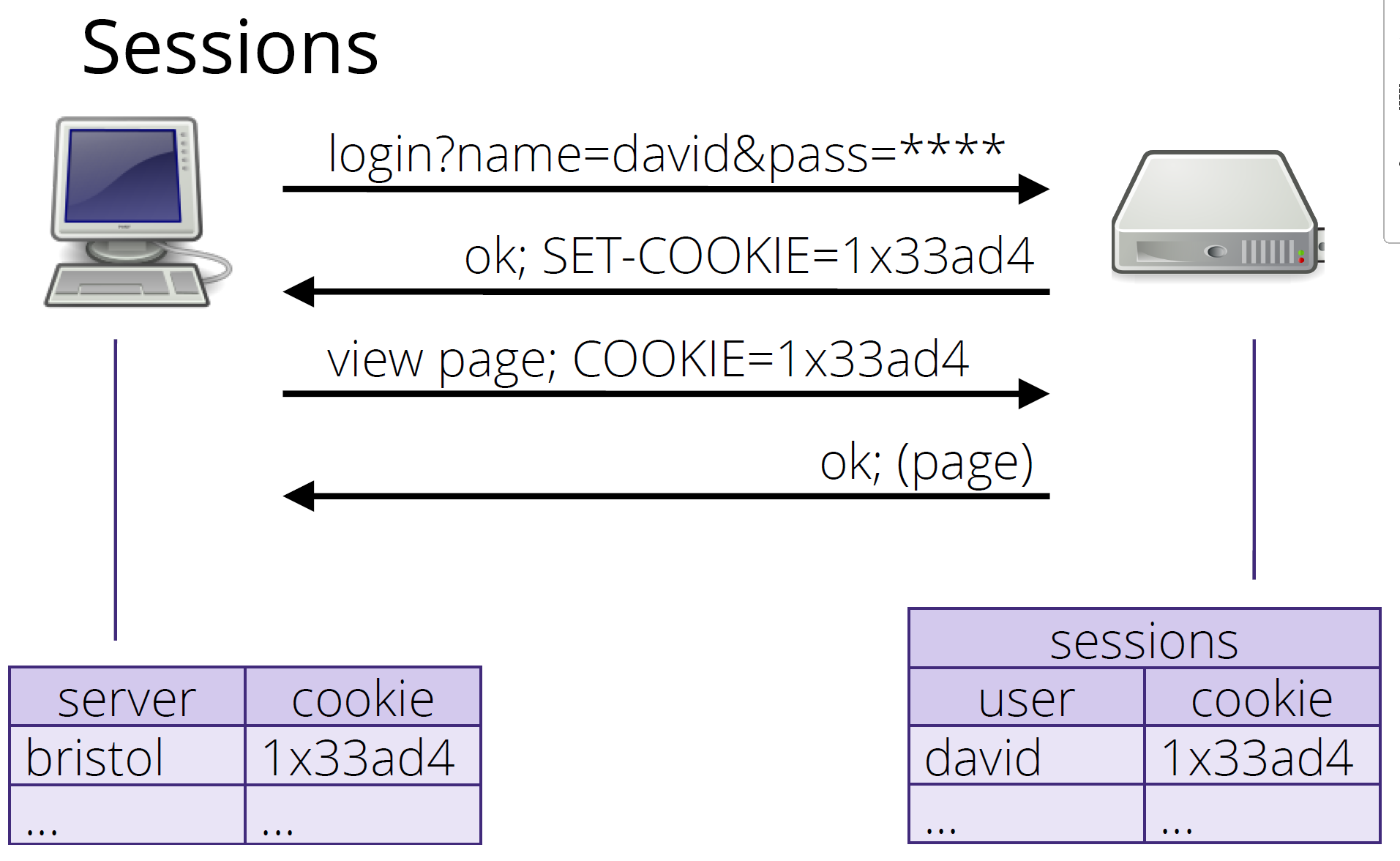
**HTTP 交换示例**

**要求**GET /index.html HTTP/1.1 Host: [www.bristol.ac.uk](http://www.bristol.ac.uk) Connection: close

**回复**HTTP/1.1 200 OK Content-Type: text/html; charset=UTF-8Content-Length: 1009<!DOCTYPE html>

**内容类型**Content-Type仍然是*元数据*。告诉客户端响应主体将包含*什么类型的数据。*

First priority for the browser! text/html; charset=UTF-8 text/plain, image/jpeg, application/pdf, video/mp4

**UserAgent**

nc -l -p 8000 < http-response：这将调用工具nc（“network cat”），并指示其监听 TCP 端口 8000（-l -p 8000），以在该端口运行服务器。nc默认情况下会从其标准输入读取并将其写入客户端，因此我们使用 shell 重定向使其改为从文件读取。请注意，在客户端连接*之前*nc，它将处于“阻塞”状态（即不执行任何操作）。

如果您发送一个带有.html扩展名但设置了 的文件Content-type: text/plain，浏览器不会将其解释为 HTML。

如果浏览器认为您正在访问一个刚刚下载的文件，它就会耍小聪明。服务器会发送一个Last-modified标头

**Javaweb**

src/main/resources有两个文件。首先，application.properties是一个配置为在 8000 端口运行的 Spring 文件（默认端口为 8080）。其次，是一个应用程序可以访问的 HTML 文件。src/main/java是源代码。虽然只有两个文件，但应用程序运行时整个 Spring 框架都会被激活。Server01Application是主类。@SpringBootApplication（在Server01Application类上）告诉 spring 这是要运行的主文件。@RestController告诉 Spring 此类包含处理 HTTP 请求的方法。之所以这样命名，是因为 Spring 提供了一些库，使得 REST 原则的实现变得尤为简单。@AutoWired字段上的 告诉 Spring，该字段由 Spring 在应用程序启动时设置。ResourceLoader是 Spring 的一部分，允许您访问 中的文件src/main/resources。@GetMapping(PATH)告诉 spring 应该调用这个方法来回复对所提供路径的 HTTP GET 请求（当然还有@PostMapping等等）。

回复 HTTP 请求的基本方法：设置您需要的任何标头（在本例中为内容类型）并返回一个ResponseEntity带有三个参数的标头：要返回的响应的 HTTP 主体（这将显示在您的浏览器中）、要设置的 HTTP 标头和响应代码（此处为 200（OK）。

**HTML**

**超文本**“带有超链接的文本”，*交互式文本*。它不是只能写和读的文档，而是读者可以与之交互的文档。背景：欧洲核子研究中心 (CERN) 研究实验室的物理学家对如何更有效地连接他们的工作很感兴趣 HTML 是超文本标记语言。它是一种用于标记文本以使其具有交互性（并且顺便提供文档结构）的语言。HTML 文档阅读器（如浏览器）必须解释标记并将结果呈现给用户。**关键要素：标签**表示有意义的文档组成部分。标签的嵌套和标签内的文本组织文档结构。标签可以具有影响其语义解释的**属性。**

**重要的 HTML 标签**

<!DOCTYPE html>: 声明文档类型。一切都嵌套在其中<html>。<head>包含<title>和<meta>标签， [<base>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/base)、[<link>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/link)、[<script>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/script)、[<style>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/style) 或 [<title>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/title)用于描述<body>。<body>包含文档的“可见”部分。大多数文档组件应放置在正文内。常见内容包括<p>，，<div>...<span> **<br />** new line <hr> 元素在文档中生成一条水平分割线

**属性**必须包含：一个空格，它在属性和元素名称之间。如果一个元素具有多个属性，则每个属性之间必须由空格分隔。属性名称，后面跟着一个等于号。一个属性值，由一对引号（""）引起来**。**标签a的*属性*href。这是最广泛使用的属性之一，因为它可以指示链接。您可能还会看到，a元素经常使用一个target属性来指示浏览器*如何*跟踪链接。例如，target="\_blank" 将在新标签页中显示链接。title 属性当鼠标悬停在超链接上面时，这部分信息将以工具提示的形式显示。form具有一个action控制提交时发生的情况的属性。ID 通常用于文档元素的编程操作（例如，使用 Javascript 更改文档），而class通常用于设置许多视觉属性。[**布尔属性**](https://developer.mozilla.org/zh-CN/docs/Learn_web_development/Core/Structuring_content/Basic_HTML_syntax#%E5%B8%83%E5%B0%94%E5%B1%9E%E6%80%A7)**：**没有值的属性

<a href=<https://www.mozilla.org/zh-CN/> title="了解 Mozilla 使命以及如何参与贡献的最佳站点。" download="firefox-latest-64bit-installer.exe" href="**mailto**:nowhere@mozilla.org">Mozilla 主页</a>的超链接。

**HTML 演示**GUI 浏览器有常见的视觉默认值（例如蓝色下划线）。但是文档的语义结构（应该）与其呈现形式css是*分开的。*

**Blockinner**

块级元素在页面中以块的形式展现。一个块级元素出现在它前面的内容之后的新行上。任何跟在块级元素后面的内容也会出现在新的行上。块级元素通常是页面上的结构元素。例如，一个块级元素可能代表标题、段落、列表、导航菜单或页脚。一个块级元素不会嵌套在一个内联元素里面，但它可能嵌套在另一个块级元素里面。 [<p>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/p)<div>

内联元素通常出现在块级元素中并环绕文档内容的一小部分，而不是一整个段落或者一组内容。内联元素不会导致文本换行。它通常与文本一起使用，例如，<a> 元素创建一个超链接，<em> 和 <strong> 等元素创建强调。<span>

[**嵌套元素**](https://developer.mozilla.org/zh-CN/docs/Learn_web_development/Core/Structuring_content/Basic_HTML_syntax#%E5%B5%8C%E5%A5%97%E5%85%83%E7%B4%A0)**：**把元素放到其他元素之中——这被称作*嵌套*。 [<strong>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/strong)

[**空元素**](https://developer.mozilla.org/zh-CN/docs/Learn_web_development/Core/Structuring_content/Basic_HTML_syntax#%E7%A9%BA%E5%85%83%E7%B4%A0)：通常用来在此元素所在位置插入/嵌入一些东西。例如：元素 [<img>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/img)

注释， <!-- 和 -->

<input type="email" id="email" name="email" multiple />[multiple](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/multiple) 属性，以允许在同一个 email 输入框中输入多个电子邮件地址，以英文逗号分隔

查询字段（Search，电话号码字段tel 值，URL 字段url

<input type="number" name="age" id="age" min="1" max="10" step="2" />step单击一次按钮所增长或减少的值为 2。允许浮点数输入，要指定 [step="any"](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/step)。如果省略了此值，step 会默认为 1，意味着只有自然数是有效的输入。

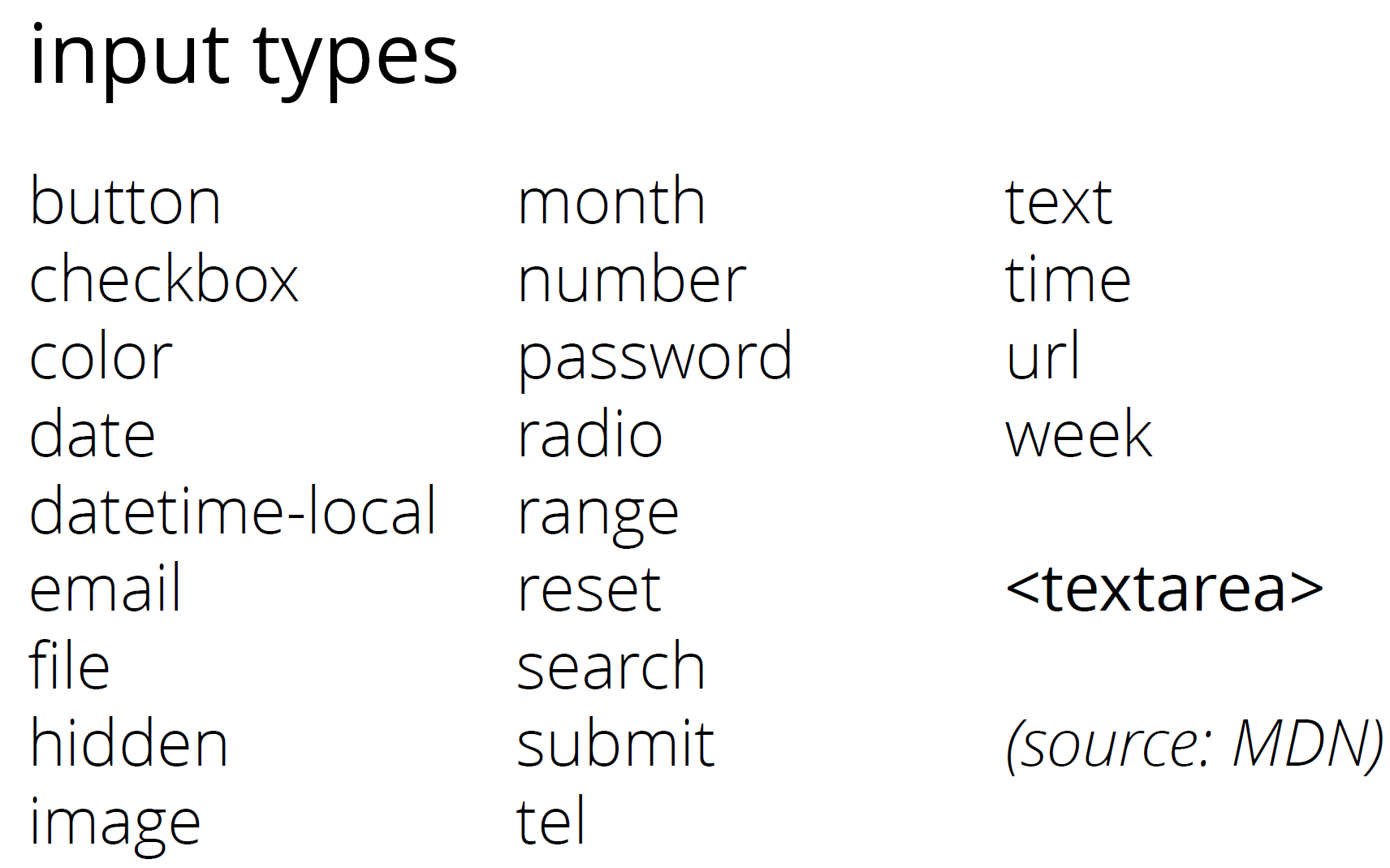
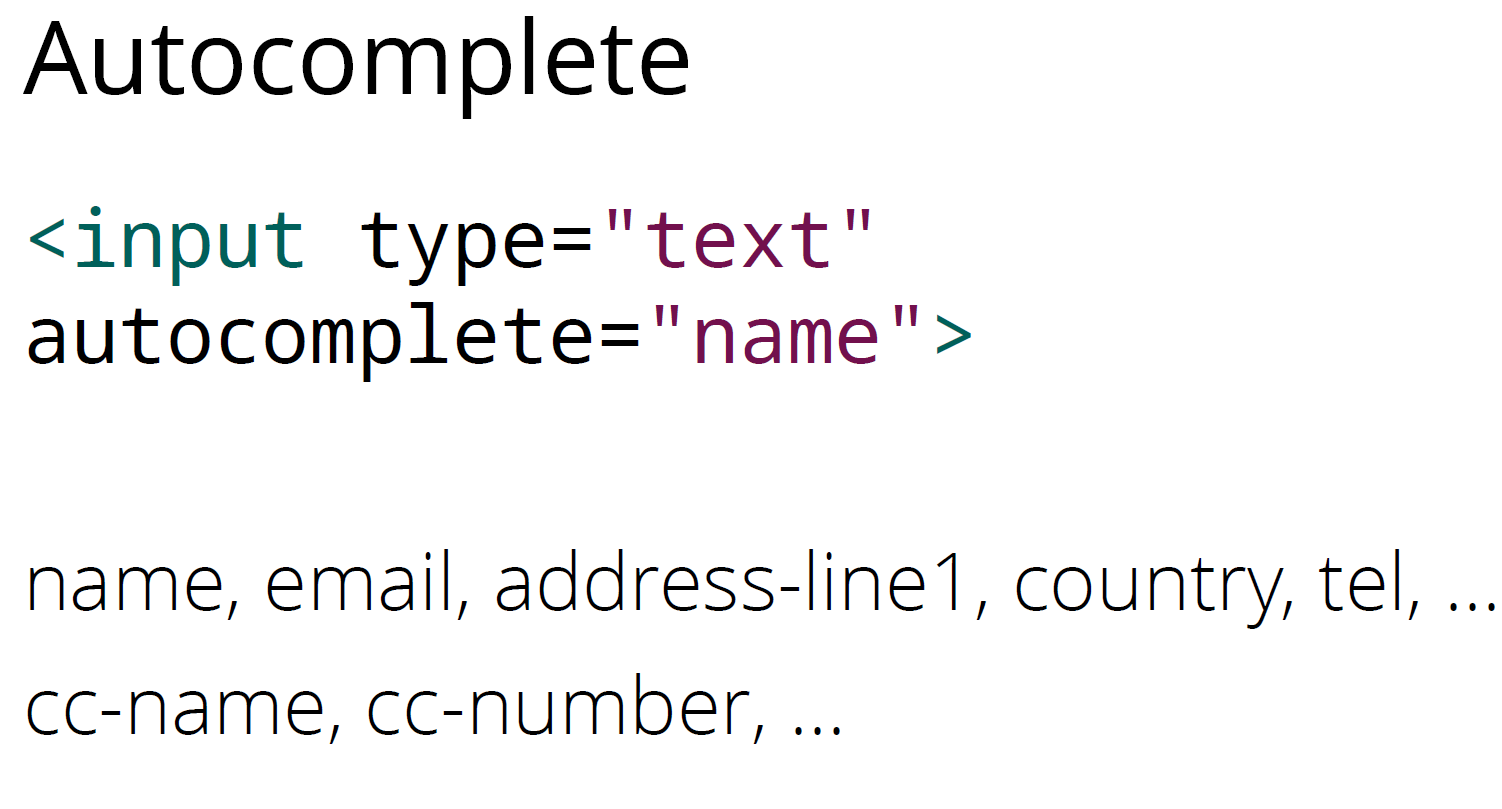
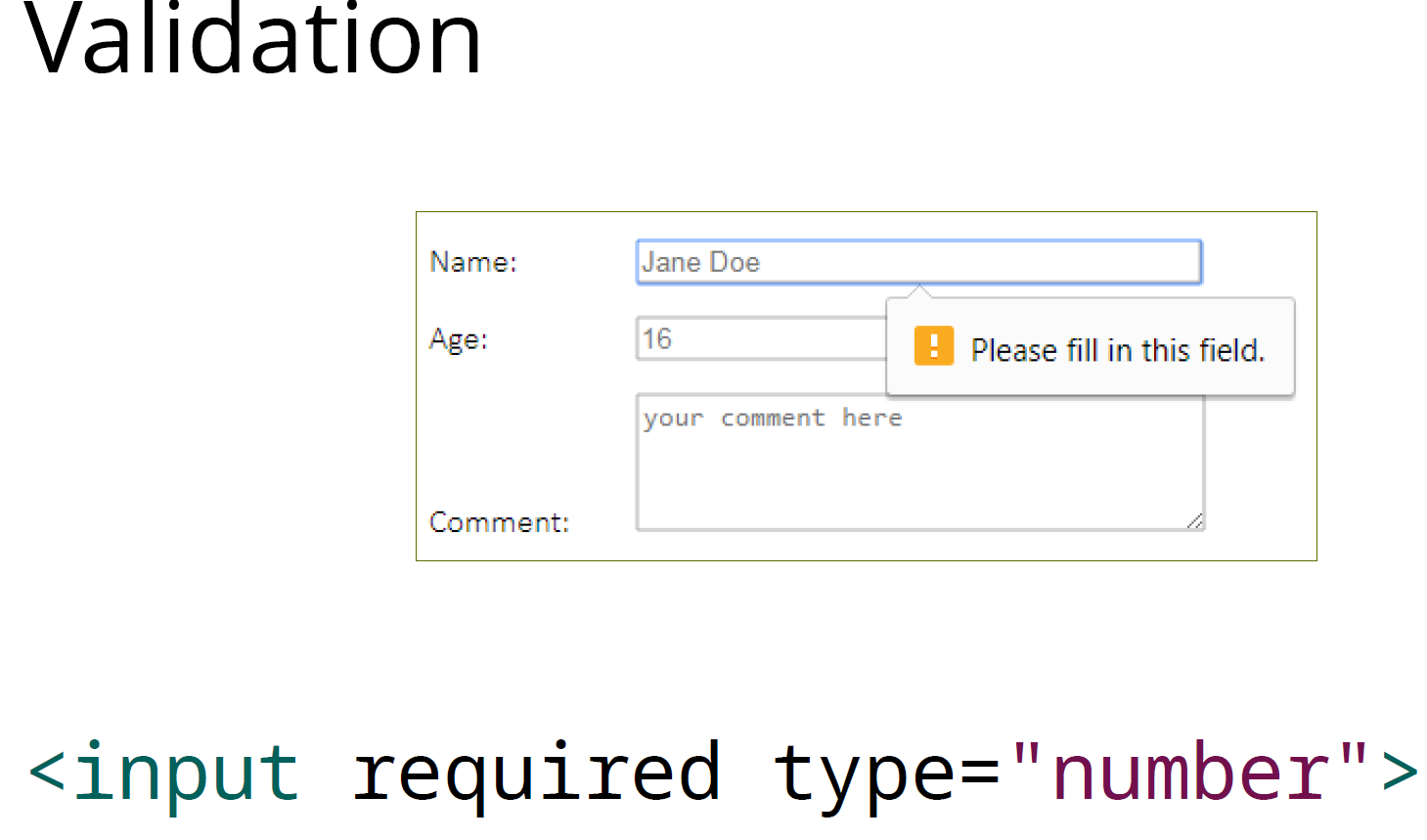
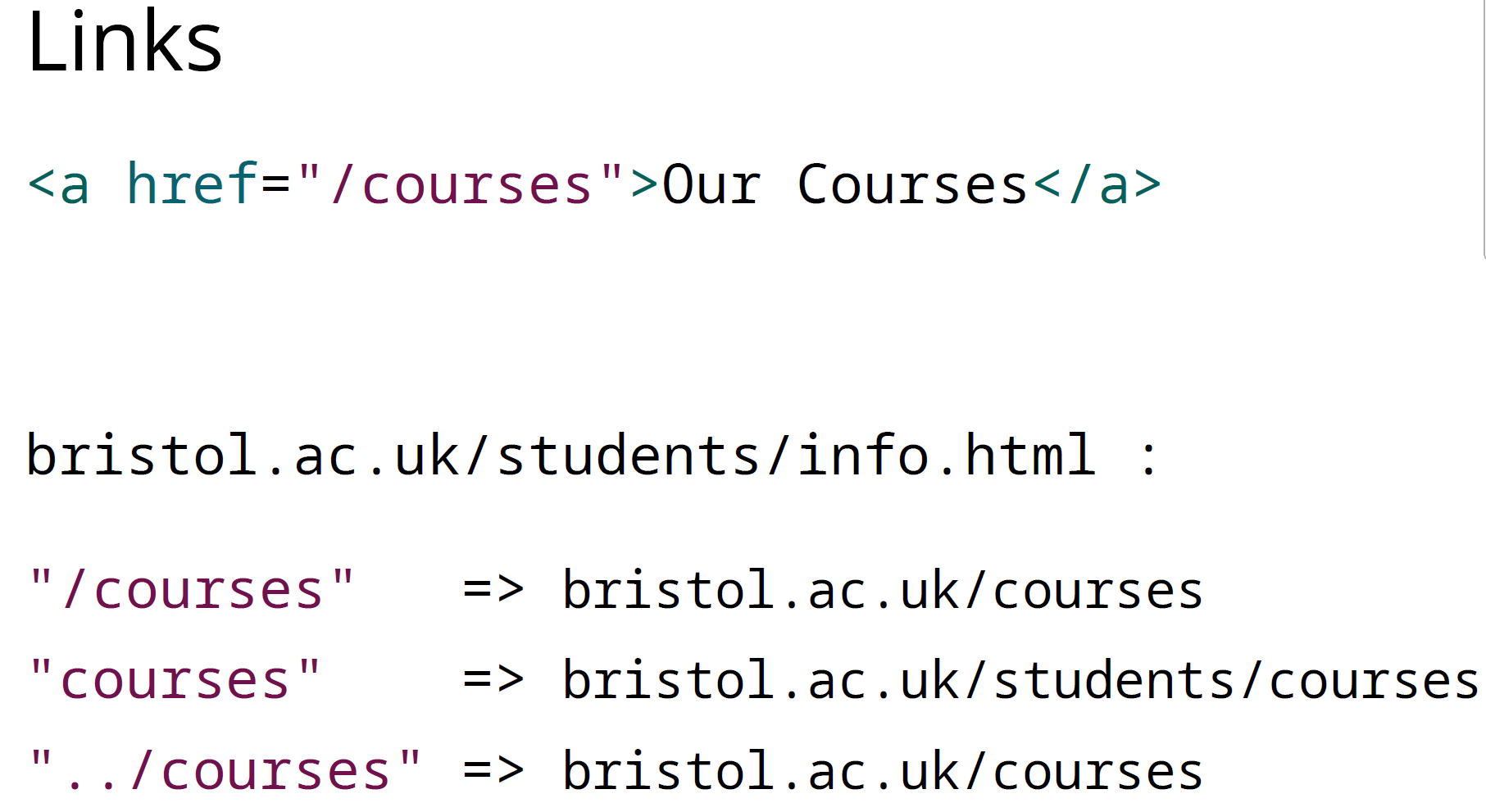
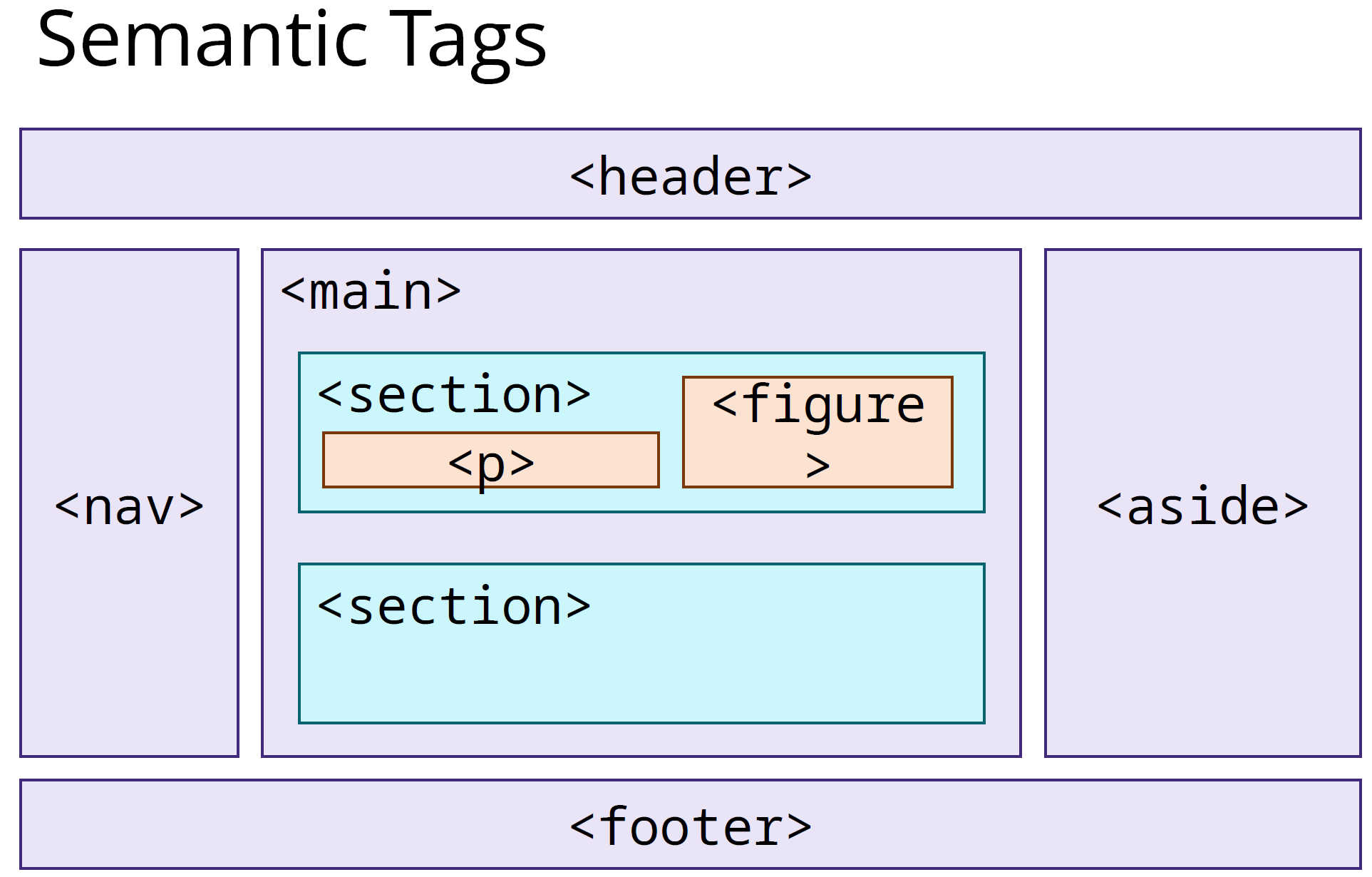
在 [<input>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input) 元素中使用 range 作为属性 [type](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input#type) 的值，就可以创建一个滑块, 配置 [min](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/min)、[max](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/max) 和 [step](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/step)

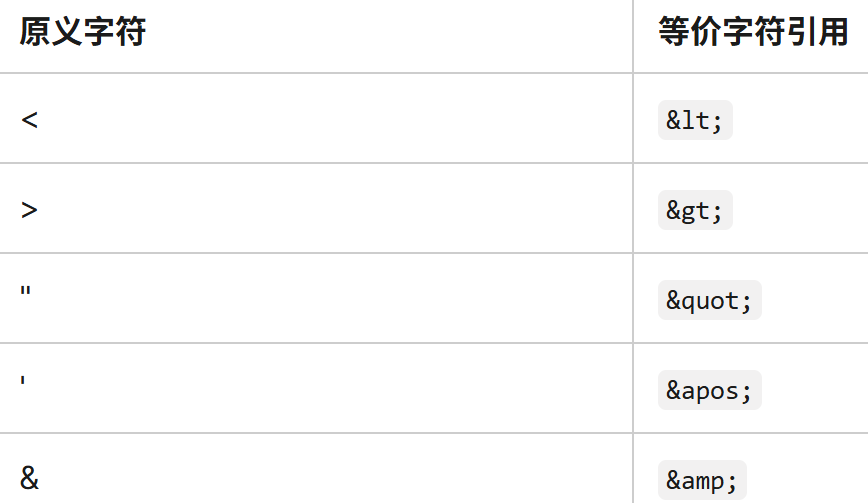
[<input type="datetime-local">](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input/datetime-local) 创建了显示和选择一个没有特定时区信息的日期和时间的控件。[<input type="month">](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input/month) 创建了显示和选择带有年份信息的某个月的控件。[<input type="time">](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input/time) 创建了显示和选择时间的控件。时间可能会以 *12 小时制*显示，但一定会以 *24 小时制*形式返回。[<input type="week">](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input/week) 创建了显示和选择一年中特定编号周的控件。所有的日期和时间控件总可以由 [min](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/min) 和 [max](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/max) 属性控制，可由 [step](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Attributes/step) 属性进一步做控制

颜色选择器控件<input type="color" name="color" id="color" />

[<em>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/em)斜体， [<strong>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/strong)粗体 。<b>、<i> 或 <u> 来表达传统上用粗体、斜体或下划线表达的意思

如果其他地方引用一个块级内容（一个段落、多个段落、一个列表等），你应该把它用 [<blockquote>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/blockquote) 元素包裹起来表示，并且在 [cite](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/blockquote#cite) 属性里用 URL 来指向引用的资源。[**行内引用**](https://developer.mozilla.org/zh-CN/docs/Learn_web_development/Core/Structuring_content/Advanced_text_features#%E8%A1%8C%E5%86%85%E5%BC%95%E7%94%A8) [<q>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/q) 和[cite](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/blockquote#cite) 。 [<cite>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/cite) 超链接蓝色下划线。<code>：用于标记计算机通用代码。<var>：用于标记具体变量名。





**CSS**层叠样式表，定义 HTML 元素如何显示的一组规则。从 HTML 派生的 DOM 对象。

语法：selector {property : value;}

<head><link rel="stylesheet" href="mystyle.css"/></head>

也可以直接指示浏览器默认将自定义样式表应用于 HTML 文档

**选择器**

单个标签的名称（例如，p，a，div）标签列表：p, div, main**。类**p.important 或.important**。ID**p#uniquebox或#uniquebox**。属性**p[name=tim]，p[class='important']将与 相同p.important。选择属性*包含*字符串“flower”的img[title~='flower']**。**

a:link没有被访问的链接a:visited访问过的链接a:hover 链接被鼠标悬停

::first-line是会选择一个元素中的第一行。\* 将匹配文档的所有元素。

无效的 class 选择器会被忽略。在用逗号组合起来以后，导致整个规则都会失效

**位置**： this that（空格）：选择this*的*所有后代元素that。this > that：选择this的所有初代子that元素。this ~ that： 匹配同一父元素下this 元素后的所有that 元素。this + that：选择紧跟某个this元素的所有元素that，并且共享同一个父节点。

**颜色值**设置background-color,border-color。red，和blue，#rrggbb，00 到 FF，rgba(r,g,b,a)，RGB 的值为 1-255，A 的值为 0-1。1 将使它完全不透明

**布局content** 是元素本身的原材料（例如， 中的文本空间或 中的图像空间**）**。**padding** 是内容和边框之间的空间**。border** 是元素周围的一条线（有时不可见），用于标记元素的边界。它可以具有一定的厚度**。margin** 是边界*之外*需要保留的空间**——其他元素不得侵占该空间。**

**布局值**margin-top: 10px;margin-right: 20px;margin-bottom: 10px;margin-left: 5px;按顺时针顺序集体指定。margin: 10px 20px 10px 5px; 先垂直后水平：margin: 20px 5px;border: 1px solid black; 内容：使用 inline-size、block-size、width、height 等属性

**测量单位：**“绝对”：1 px- 1‘像素’（ 1/96英寸）。1 pt- 1“点”（1/72英寸）1 cm- 1厘米（10 mm）1 in1英寸。

“相对”：1 vh- 视口高度的 1%（1 vw宽度）1 em- 1 x字体的大小（'m' 的宽度）。1 ex- 1 x ‘x’ 的高度。1 rem- 1 x 文档根元素的字体大小。1 %- 父元素相应尺寸的大小的 1%。lh 和 rlh ，前者是相对于元素自身的行高，而后者是相对于根元素（通常是 <html>）的行高。

PC screen: 72 / 96 (Mac / Win defaults)can be 120+ nowadays mobile: 200+ (iPhone 12 Pro: 460)

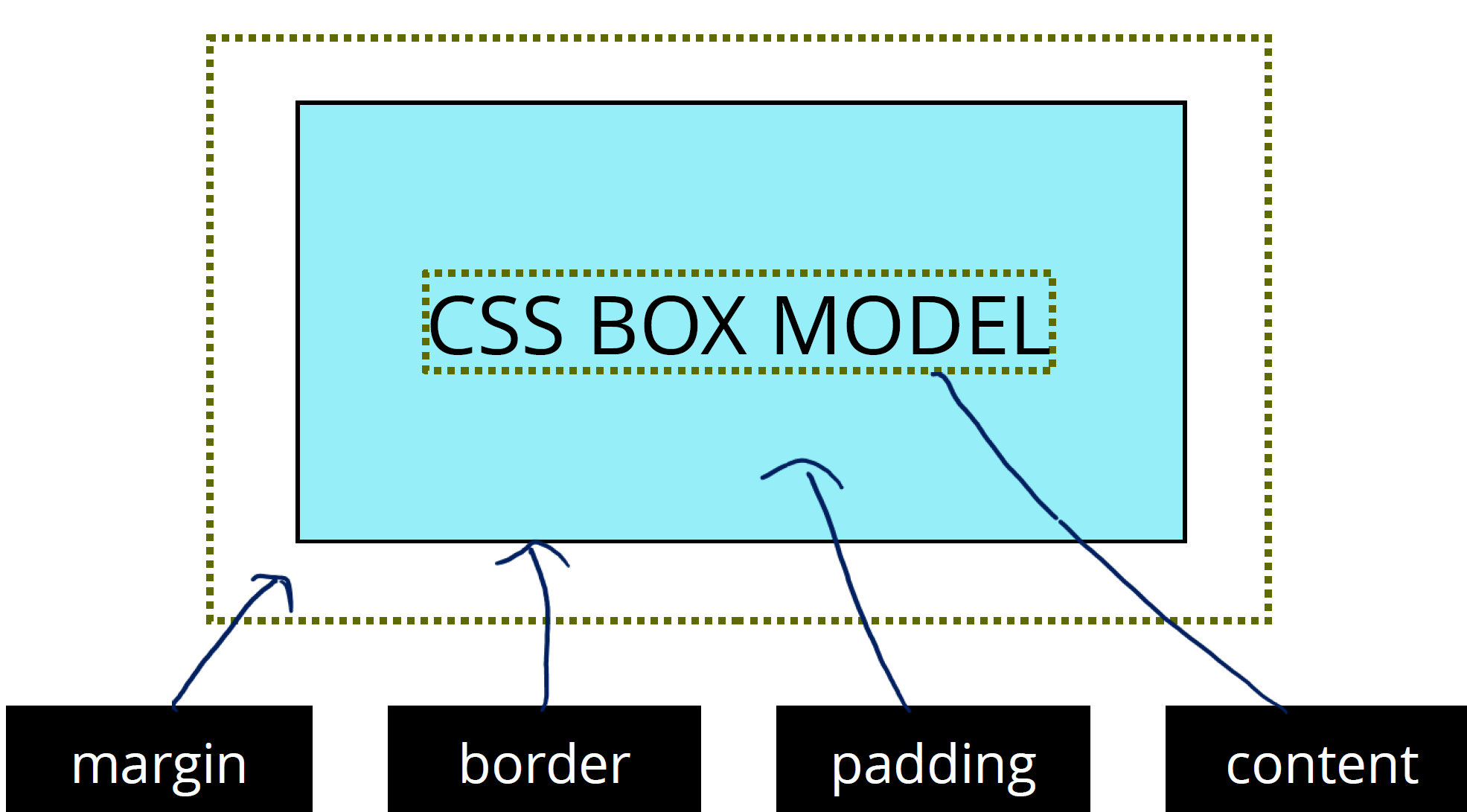
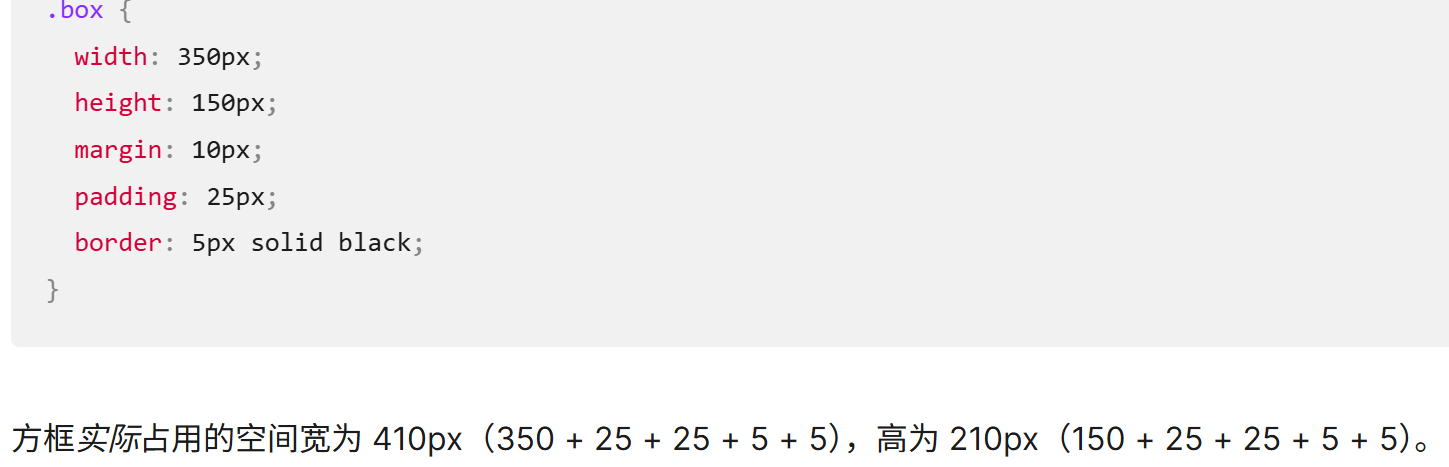
**盒模型**

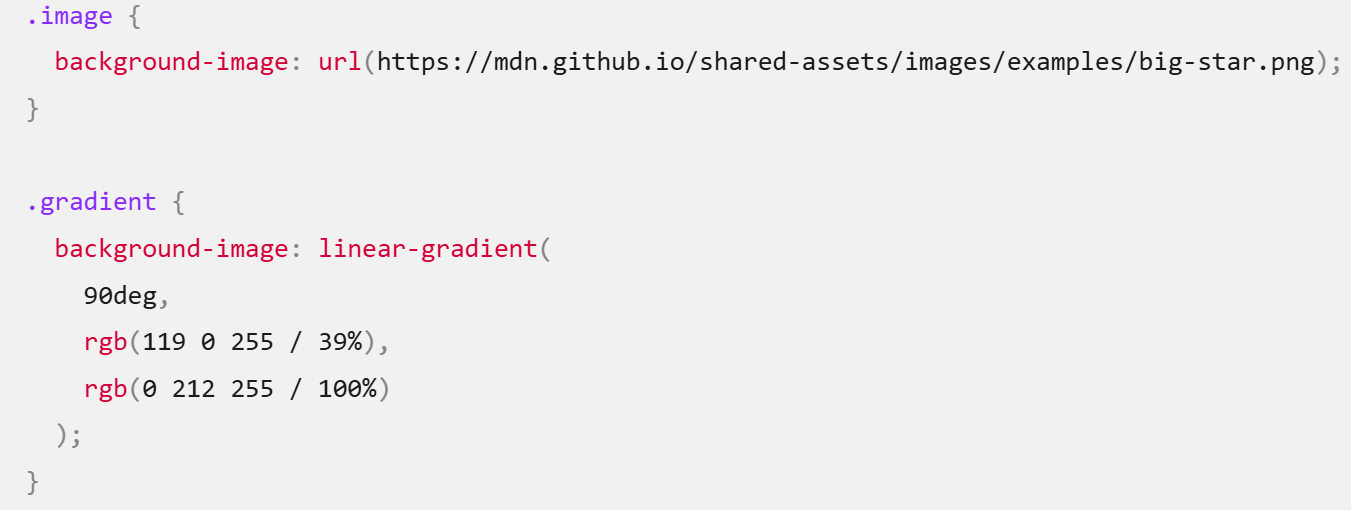
block 外部显示类型：盒子会产生换行。[width](https://developer.mozilla.org/zh-CN/docs/Web/CSS/width) 和 [height](https://developer.mozilla.org/zh-CN/docs/Web/CSS/height) 属性可以发挥作用。内边距、外边距和边框会将其他元素从当前盒子周围“推开”。如果未指定 [width](https://developer.mozilla.org/zh-CN/docs/Web/CSS/width)，方框将沿行向扩展，以填充其容器中的可用空间。 <h1> 和 <p>。

inline 外部显示类型：盒子不会产生换行。[width](https://developer.mozilla.org/zh-CN/docs/Web/CSS/width) 和 [height](https://developer.mozilla.org/zh-CN/docs/Web/CSS/height) 属性将不起作用。垂直方向的内边距、外边距以及边框会被应用但是**不会**把其他处于 inline 状态的盒子推开。水平方向的内边距、外边距以及边框会被应用**且会**把其他处于 inline 状态的盒子推开。<a>、 <span>、 <em> 以及 <strong>。display: block;改为block

 display: inline-block，设置 width 和height 属性会生效。padding、margin 和 border 会推开其他元素。它不会换行，只有在明确添加 width 和 height 属性后，才会变得比其内容大。

*内部*显示类型：区块和行内布局是默认行为方式。方框内的元素也表现为区块或行内盒子。可以通过设置 display: flex; 来更改内部显示类型。该元素仍将使用外部显示类型 block 但内部显示类型将变为 flex。



**使用 CSS 进行设计**

**网格：父级**display: grid。grid-template-columns定义网格列的数量和宽度。gap定义网格单元之间的间隙

**网格：孩子：**每个子元素占用下一个空闲的 1x1 空间。grid-row-start并grid-row-end让您指定元素应该在哪一行开始以及在哪一行结束（列相同）。简写为grid-row: 1 / 3——从第 1 行到第 3 行。或者span N可以根据 N 行/列给出元素的宽度（或高度），而不固定其位置。grid-row: 1 / 3;grid-column: 2 / 3;==grid-area: 1 / 2 / 3 / 3;

grid-template-columns: 200px 200px 200px; grid-template-columns: 1fr 1fr 1fr; grid-template-columns: repeat(2, 2fr 1fr)。隐式网格就是为了放显式网格放不下的元素，可以使用[grid-auto-rows](https://developer.mozilla.org/zh-CN/docs/Web/CSS/grid-auto-rows)和[grid-auto-columns](https://developer.mozilla.org/zh-CN/docs/Web/CSS/grid-auto-columns)属性手动设定隐式网格轨道的大小。grid-auto-rows: minmax(100px, auto);，那么尺寸就至少为 100 像素，并且如果内容尺寸大于 100 像素则会根据内容自动调整

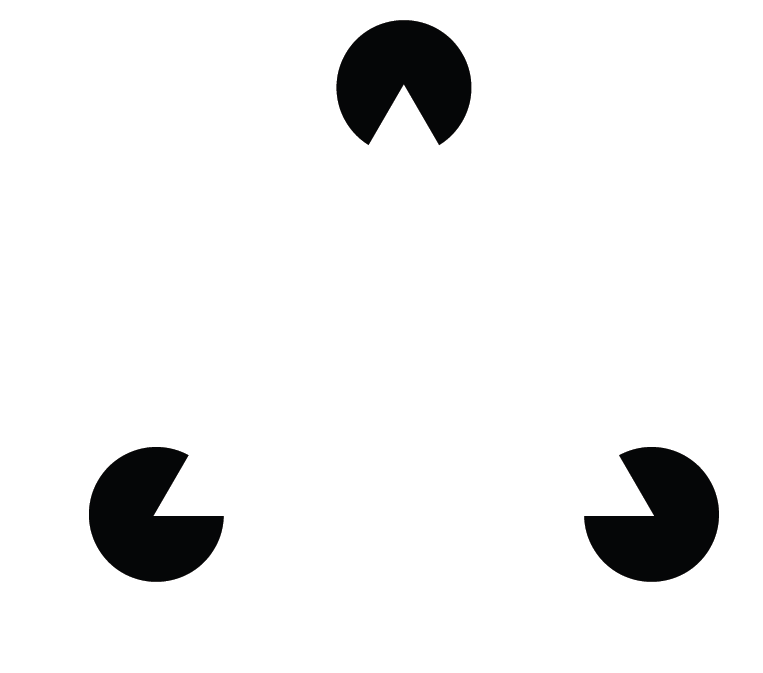
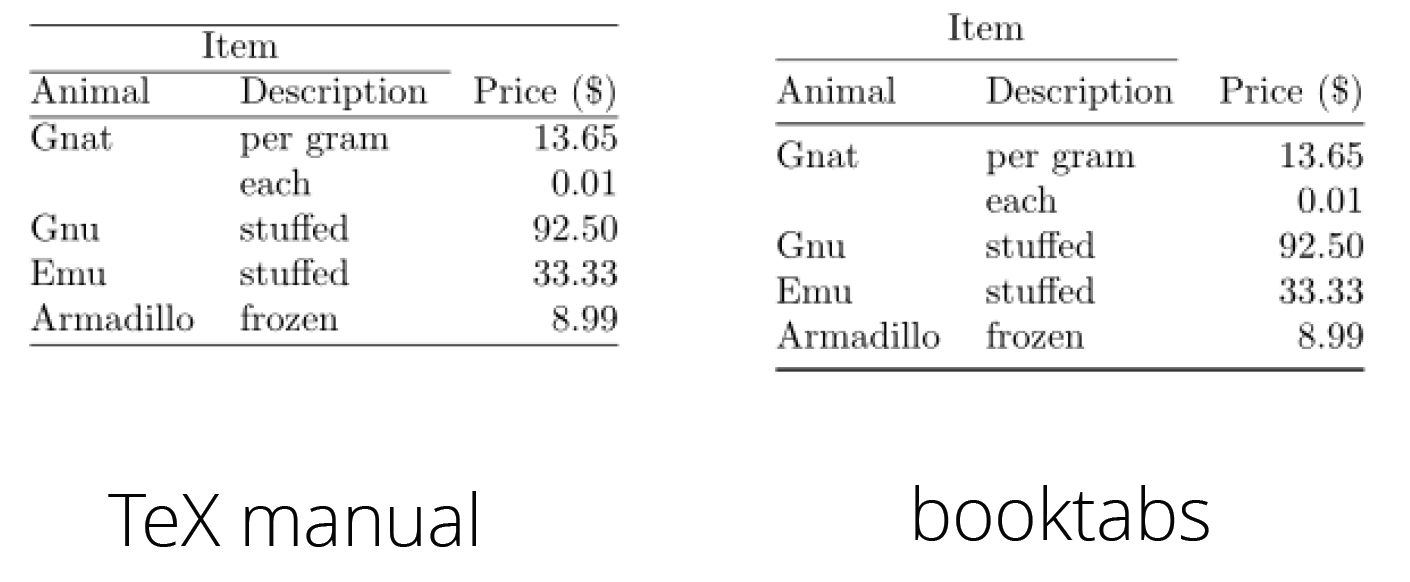
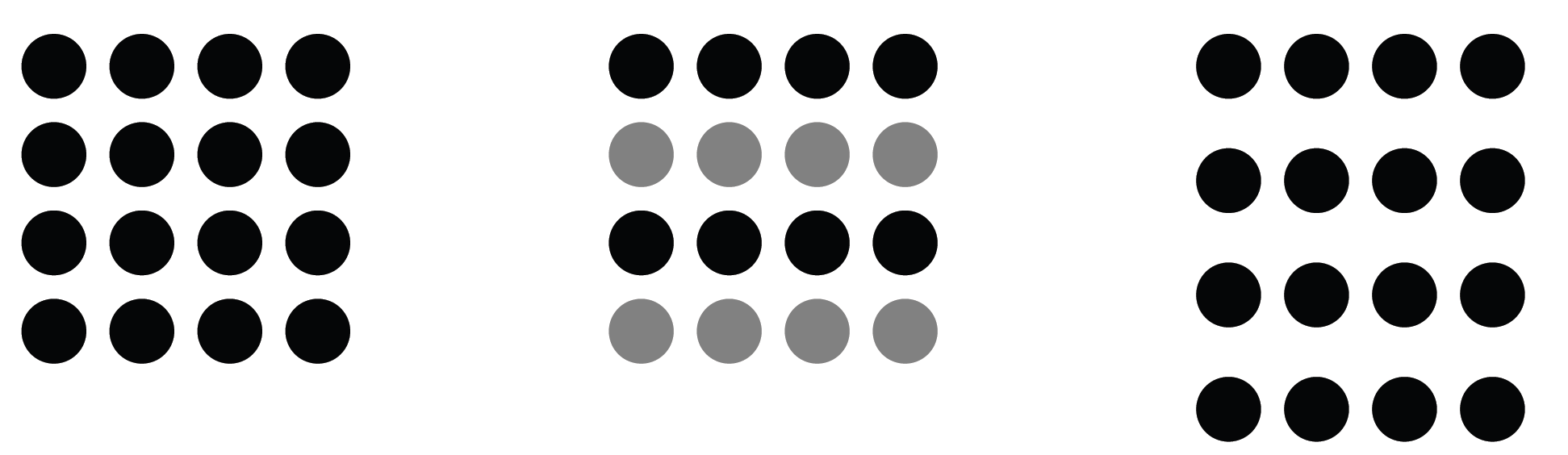
**媒体查询** @media media-type and (media-feature-rule) media-type主要是print、screen。media-feature-rule：

视口宽度（max-、min-和范围语法）方向[竖放（portrait mode）和横放（landscape mode）]、是否有指点设备(hover: hover)

@media screen and (min-width: 600px), screen and (orientation: landscape)

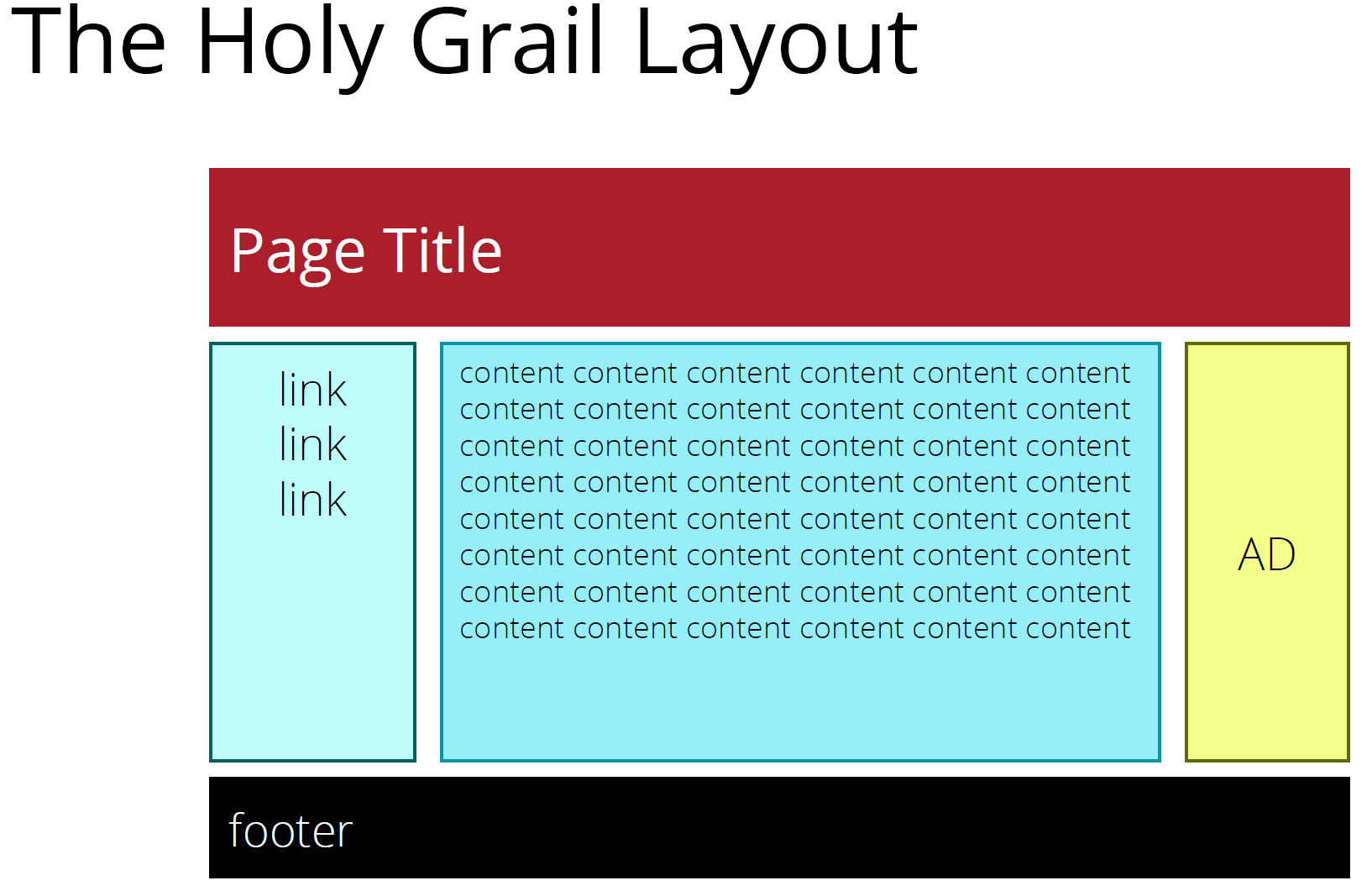
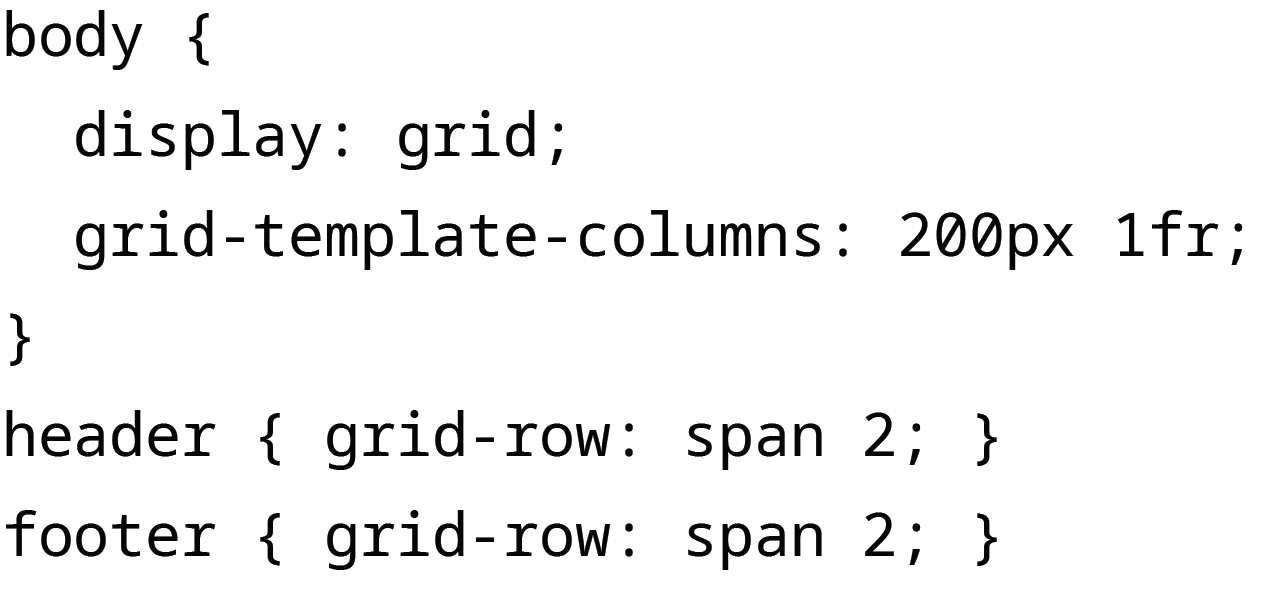
@media (not (width < 600px)) and (not (width > 1000px)) @media (30em <= width <= 50em)

**Design** principle 1: **test with your intended audience!2: on the web, use responsive design**

Gestalt  Whitespace

Text width:50–60 characters，two full alphabets，12 words，30 em

**History** of Web Layout：Tables，Float，Flexbox，Grid

**Flex**

当我们把display: flex添加到它的父元素时，这三个元素就自动按列进行排列。这是由于它们变成了*flex 项 (flex items)*，按照 flex 容器（也就是它们的父元素）的一些 flex 相关的初值进行 flex 布局：它们整整齐齐排成一行，是因为父元素上flex-direction的初值是row。它们全都被拉伸至和最高的元素高度相同，是因为父元素上align-items属性的初值是stretch。

**Float**

改变该元素本身和在正常布局流（normal flow）中跟随它的其他元素的行为。这一元素会浮动到左侧或右侧，并且从正常布局流 (normal flow) 中移除. left — 将元素浮动到左侧。right — 将元素浮动到右侧。none — 默认值，不浮动。inherit — 继承父元素的浮动属性。

Positioning

position: relative; top: 30px;left: 30px; 左边和顶部的元素被“推开”一定距离，这就导致了它的向下向右移动。

position: absolute; 指定元素应该从页面边界的顶部和左边的距离

position: fixed; 应用于视口 (viewport) 边框

position: sticky; 会在正常布局流中滚动，直到它出现在了设定的相对于容器的位置，这时候就会停止随滚动移动

[**多列布局**](https://developer.mozilla.org/zh-CN/docs/Learn_web_development/Core/CSS_layout/Introduction#%E5%A4%9A%E5%88%97%E5%B8%83%E5%B1%80)

使用 [column-count](https://developer.mozilla.org/zh-CN/docs/Web/CSS/column-count)属性来告诉浏览器我们需要多少列，也可以使用[column-width](https://developer.mozilla.org/zh-CN/docs/Web/CSS/column-width)来告诉浏览器以至少某个宽度的尽可能多的列来填充容器。

**JavaScript 基础知识**

一种使用**即时编译**的解释型语言,“古怪”但易于学习、**动态类型**、广泛支持。Web 和浏览器 API 的首选脚本语言。将复杂的“动态”元素引入网页的最常见方式，在某些情况下，如果没有 JS，网站就会完全崩溃。

**变量**const let var的是一个较旧的构造，它允许变量在使用后声明（“提升”）。接受同一标识符的多个声明。 (0-9,a-z,A-Z) 和下划线字符，变量名不要以下划线、数字开头。**动态类型：**根据赋值改变属性

**数学**可以接受其他数字基数系统：let ahexnum = 0xEF console.log(ahexnum)输出number，除号返回小数

还有指数运算：\*\*以及模数%和增量integer++

**字符串** 字符串可以用双引号、单引号或反引号定义。带反引号的字符串是*模板字面量*，可以嵌入代码。

**数组** [length](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/length) 属性获取数组的长度。split(",")。join(",")。toString()。[push()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/push) 和 [pop()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/pop)。[unshift()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift) 和 [shift()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/shift)是作用于第一个元素

**分支** switch(){case “”} ifelse

**For 循环**for(let i = 0; i < 3; i++) for (const item of list

对于处理数组，也有.map和.filter方法。要控制迭代，您可以使用break和continue。

**While 循环** while do / do while

**函数**function help(topic)

**对象**const planet = {}planet.name = "Earth"planet.rank = 3const moon = {name: "Moon", rank: 3.1}planet.moon = moon

**DOM**文档对象模型将内存中的网页表示为对象树。在浏览器中运行的 JavaScript 可以访问浏览器提供的 DOM API 来与页面元素进行交互。可以通过修改 DOM 来更改页面。您可以通过使用该 document对象以编程方式实现这一点。document.getElementById()**;**document.querySelector()**;**document.querySelectorAll()**;**document.createElement()；Element.innerHTML；Element.setAttribute()；Element.getAttribute()；EventTarget.addEventListener()；HTMLElement.style；Node.appendChild()；[Node.removeChild()](https://developer.mozilla.org/zh-CN/docs/Web/API/Node/removeChild) ；window.onload；window.scrollTo()

文档（document，在浏览器中用 DOM 表示）是加载到窗口的实际页面，在 JavaScript 中，它由 [Document](https://developer.mozilla.org/zh-CN/docs/Web/API/Document) 对象表示。树上的每个条目都被称为**节点**。你可以在上图中看到，一些节点代表元素（标识为 HTML、HEAD、META 等），另一些代表文本（标识为 #text）根节点: 树中顶层节点，总是一个 HTML 节点。子节点: 直接位于另一个节点内的节点。后代节点: 位于另一个节点内任意位置的节点。父节点: 里面有另一个节点的节点。兄弟节点: DOM 树中位于同一父节点下同一等级的节点

CSS 样式的 JavaScript 属性版本是用[小驼峰命名法](https://developer.mozilla.org/zh-CN/docs/Glossary/Camel_case)书写的，通过 [HTMLElement.style](https://developer.mozilla.org/zh-CN/docs/Web/API/HTMLElement/style) 属性实现para.style.backgroundColor = "black";改变属性设置：para.setAttribute("class", "highlight");

**找到 JavaScript 的地方**.js文件中，通过引用加载到网页上下文中<script type="text/javascript" src="target.js"/>，

<script>页面源本身的内部元素。作为某些元素属性的内容，例如按钮的onclick操作（不要这样做，请使用addEventListener）。

**脚本加载策略**

问题：您正在使用 JS 来操作页面元素，但是代码不起作用，因为*JavaScipt 在页面加载完成之前就已执行*。

修复：<script>页面底部的内联确保所有元素都已加载。外部<script>可以使用type="module"或defer属性。设置您的代码以从事件触发document.body.onload。



**异步 JavaScript**异步代码很难推理，因为切换可能发生在任何时候。

**异步/等待**async使函数异步。这意味着它在被调用时立即返回，在函数体执行完成之前。await是一条指令，您可以将其添加到异步调用之前以使其同步 - 您的代码将“等待”直到异步调用完成。

可以在调用一个返回 Promise 的函数之前使用 await 关键字。调用者得到的并不是 Promise，而是一个完整的 Response 对象。，避免了创建显式 Promise 链，并允许你像编写同步代码那样编写异步代码。请注意你只能在 async 函数中使用 await

**做出承诺：**异步函数返回什么？Promise结果的 A。A是Promise一个通用对象，可以具有以下三种状态之一：‘pending’ ：承诺尚未解决（执行仍在进行中）‘fulfilled’ ：执行已成功解决**。**‘rejected’ ：执行失败

最初，所有 Promise 都处于“待定”状态，当.then()执行完成时，它们的状态和内容都会更新。

有时你需要所有的 Promise 都得到实现，但它们并不相互依赖。在这种情况下，将它们一起启动然后在它们全部被兑现后得到通知会更有效率。这里需要 [Promise.all()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Promise/all) 方法。如果数组中有*任何一个* Promise 被拒绝。

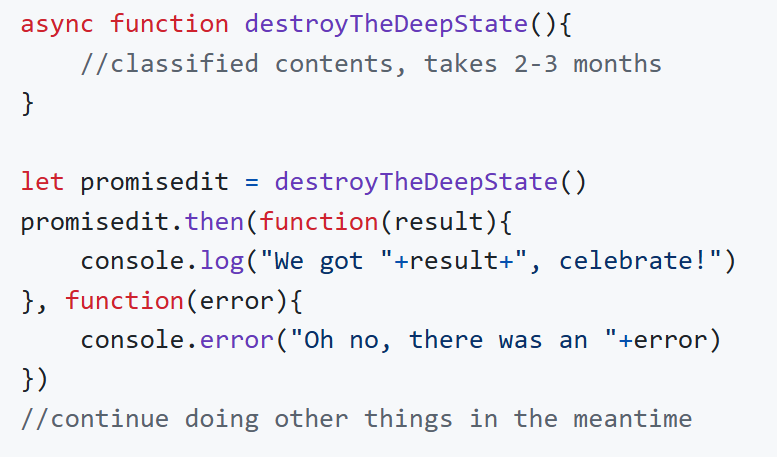
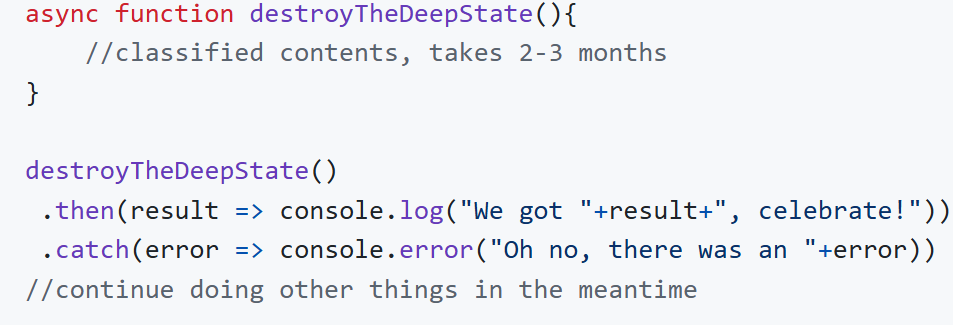
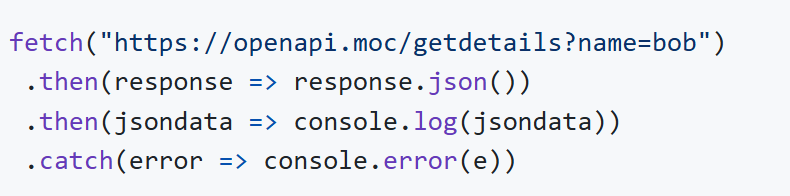
**信守承诺：**异步函数需要处理两个“事件”：成功完成（“成功处理程序”）或失败（“失败/错误处理程序”）。您可以使用方法将这些处理程序附加到 Promise .then(successCallback, failureCallback)。

**链接 Promises**一种常见的情况是，您希望承诺在前一个操作成功时执行某项操作（可能多次）。

**获取资源：**当某个函数需要很长时间（或不确定）时，通常需要使用异步。典型的例子是从较慢的附加组件（例如磁盘存储或网络）获取资源。fetch因此，检索资源的内置函数是async，并返回Promise的Response。

这意味着您可以调用fetch(url)和 使用.then来提供结果处理程序。成功处理程序将获得一个Response对象。要获取该响应的内容，您可以使用以下Response方法之一，例如 Response.json()或Response.text()。

如果您不介意等待结果，您可以await暂停直到结果 Response返回。

从 fetch() 返回的 Promise **不会被标记为 reject**，即使响应的 HTTP 状态码是 404 或 500。相反，它会将 Promise 状态标记为 resolve（如果响应的 HTTP 状态码不在 200 - 299 的范围内，则设置 resolve 返回值的 [ok](https://developer.mozilla.org/zh-CN/docs/Web/API/Response/ok) 属性为 false），仅当网络故障时或请求被阻止时，才会标记为 reject。fetch **不会发送跨域 cookie**

response 属性有：[Response.status](https://developer.mozilla.org/zh-CN/docs/Web/API/Response/status) — 整数（默认值为 200）为 response 的状态码。[Response.statusText](https://developer.mozilla.org/en-US/docs/Web/API/Response/statusText) — 字符串（默认值为 ""），该值与 HTTP 状态码消息对应。注意：HTTP/2 [不支持](https://fetch.spec.whatwg.org/#concept-response-status-message)状态消息[Response.ok](https://developer.mozilla.org/zh-CN/docs/Web/API/Response/ok) — 如上所示，该属性是来检查 response 的状态是否在 200 - 299（包括 200 和 299）这个范围内。该属性返回一个布尔值。

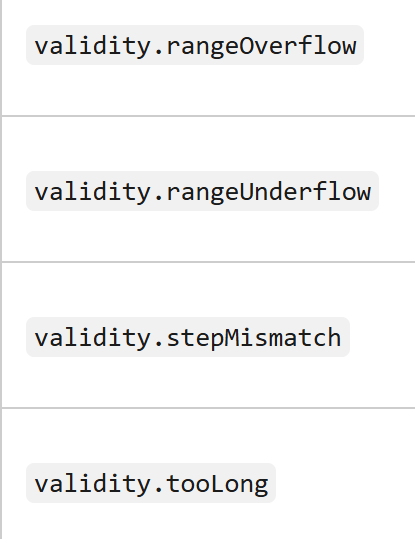
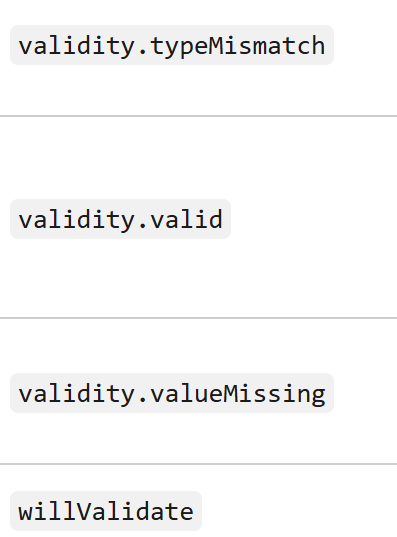
不管是请求还是响应都能够包含 body 对象。body 也可以是以下任意类型的实例。[ArrayBuffer](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/ArrayBuffer)，[ArrayBufferView](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/TypedArray) (Uint8Array 等)，[Blob](https://developer.mozilla.org/zh-CN/docs/Web/API/Blob)/File，string，[URLSearchParams](https://developer.mozilla.org/zh-CN/docs/Web/API/URLSearchParams)，[FormData](https://developer.mozilla.org/zh-CN/docs/Web/API/FormData)

获取 body 内容：Request.arrayBuffer() / Response.arrayBuffer()，Request.blob() / Response.blob()，Request.formData() / Response.formData()，Request.json() / Response.json()，Request.text() / Response.text()

[**表单数据校验**](https://developer.mozilla.org/zh-CN/docs/Learn_web_development/Extensions/Forms/Form_validation#%E4%B8%8D%E5%90%8C%E7%B1%BB%E5%9E%8B%E7%9A%84%E8%A1%A8%E5%8D%95%E6%95%B0%E6%8D%AE%E6%A0%A1%E9%AA%8C)

HTML5 **内置校验：**通过表单元素的[校验属性](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Guides/Constraint_validation)实现的，CSS 伪类 [:valid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:valid)  [:invalid](https://developer.mozilla.org/zh-CN/docs/Web/CSS/:invalid)进行特殊的样式化。required 属性。[pattern](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input#pattern) 属性，以 [Regular Expression](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Guide/Regular_expressions) 作为 value 值。使用[minlength](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input#minlength) 和 [maxlength](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Reference/Elements/input#maxlength) 属性来限制长度

JavaScript 校验表单：

**网页抓取**

为搜索或存档目的对网站进行索引。为其所有者执行特定任务（类似新闻阅读器的工具）。探测安全漏洞。冒充人类以达到邪恶目的（例如，为了在社交媒体上宣传某些东西）。生成虚假的“流量”，以便网站所有者可以欺骗广告网络。

**爬虫Crawlers**也可以称为“蜘蛛”**。**蜘蛛程序索引内容，通常用于搜索目的**。**其他“爬虫”可能对该内容有其他用途。爬虫是一个 HTTP 客户端，类似于浏览器，但自动化程度较高。爬虫有两种实现方式：编写自己的专用软件**，**使用现有工具

**wget 是一个爬虫**

wget -r -l N <url> The level of recursion to permit. (Default if omitted is 5). **不包含外部资源**

wget -m <url> Download *entire website* -w 1: Wait 1 second between each request (to avoid annoying a server).

**-i**：从指定文件读取 URL（每行一个），批量下载。**--force-html**：强制将输入文件视为 HTML。 **--spider**：模拟检查链接是否有效（不下载文件），返回 HTTP 状态码。--user-agent修改 User-Agent。-p：下载页面所需的所有资源（图片、CSS、JS 等），但不递归子页面。-D 指定允许的域名，--exclude-domains 排除域名。-nc（--no-clobber）：

不覆盖已存在的文件（若文件名冲突，跳过下载）。**默认行为**：同名文件会被覆盖（clobbering）

**robots.txt**网站用来告诉 HTTP 或 FTP 爬虫可以访问网站哪些部分的标准。

Simple text file stored in top-level directory of a site.

User-Agent: foobot Allow: /example/page/ Disallow: /example/page/disallowed.gif

User-Agent: bazbot Disallow: /example/page.html

爬虫程序旨在检查文件中自己的用户代理字符串（始终放在 webroot 中）并遵循规则。

**robots.txt 问题：**实际上并不强制执行访问限制。需要真正的安全机制来做到这一点（例如，身份验证）。通过列出站点组件，使内容可查找。更多的是“荣誉制度”，让好的机器人尊重网站所有者的意愿。

**抓取的其他限制：***您拥有通常不受限制地访问*公共网络内容的权利。重新发布内容的版权限制**。**道德限制，尤其是对于经过验证的内容**。**访问内容的方式也很重要——频繁下载大型网站可能会给服务器带来负担。服务器有时会通过将客户端列入黑名单来应对。通常情况下，在请求之间引入短暂的延迟是“礼貌的”，即使没有被要求。为自动访问而设计的 API 端点可能会施加不同的速率限制。始终先寻找 API。如果某个网站提供 JSON 端点来查询所需的结构化数据，那么使用该 API 比从专为人类阅读而设计的网页上抓取相同的数据更简单、更礼貌。

**BeautifulSoup**用于从 HTML 文件中提取数据的 Python 库。

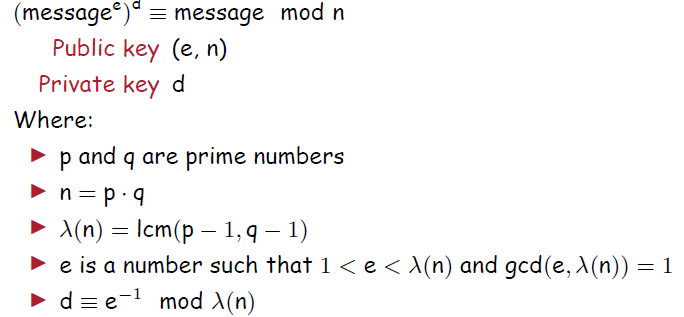
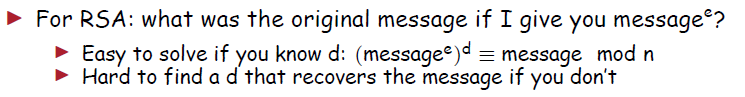
**公钥密码学**

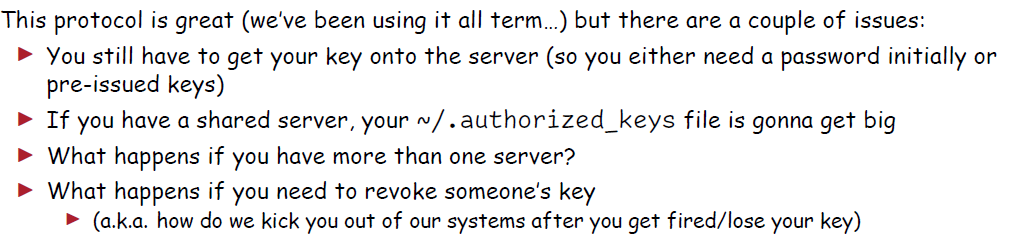
一种不依赖于预共享密码的数据加密机制。它依赖于一对密钥：一个公开，一个秘密。使用HTTPS连接到一个网站？验证你的软件没有被篡改？使用服务器进行身份验证？使用OpenSSL的TLS证书。使用GnuPG进行数字签名。用OpenSSH SSH密钥

加密：RSA, ElGammal, DSA, Eliptic Curve Cryptography

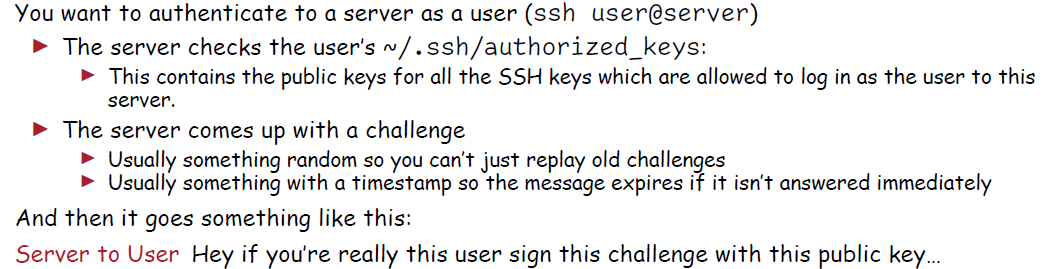
一个数学问题：很难解决（没有比猜测更好的方法）易于检查（您可以检查解决方案而不必找到解决方案）不会泄漏（知道一个解决方案并不会告诉你是如何解决的）公钥是问题的声明，每个人都能看出来。秘密钥匙是你需要知道的解决问题的诀窍，没有人能看到这个

**RSA**是一个经典的公钥密码系统，它基于很难计算出一个数字的质因数的信。1977年由Ron Rivest， Adi Shamir和Leonard Adleman发现，1973年由GCHQ的Clifford Cocks秘密发现



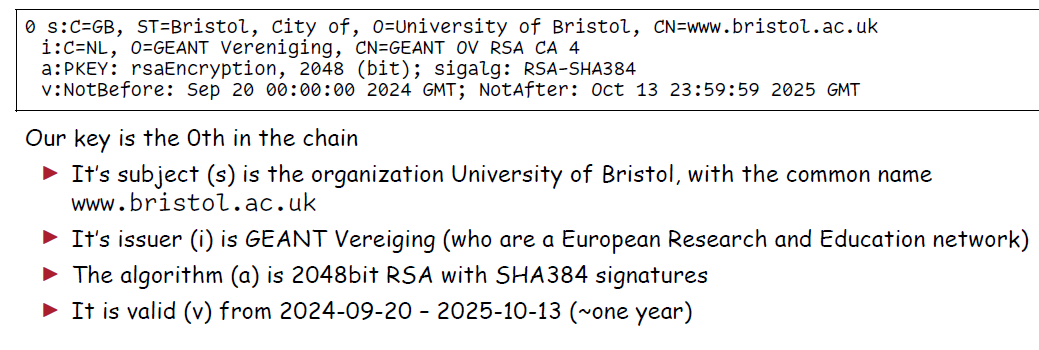
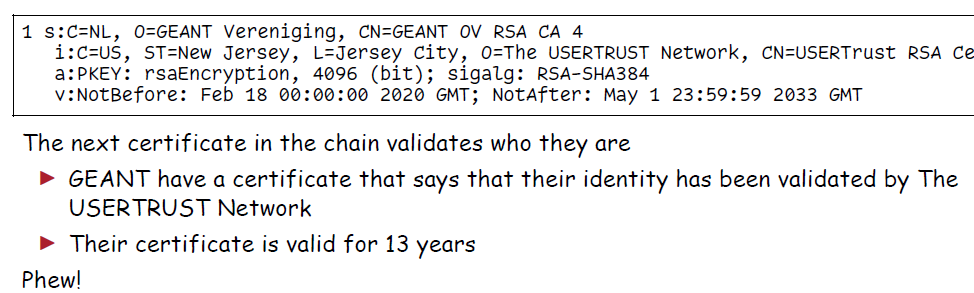
如果Alice有公钥和秘钥对。鲍勃只有爱丽丝的公钥，给爱丽丝发一条秘密信息，使用alice的公钥加密消息以设置一个问题，该问题只能在知道私钥的情况下解决。证明爱丽丝给你寄了东西：爱丽丝通过使用私钥创建一些东西来创建数字签名或证书，如果您知道公钥，则可以恢复原始文件

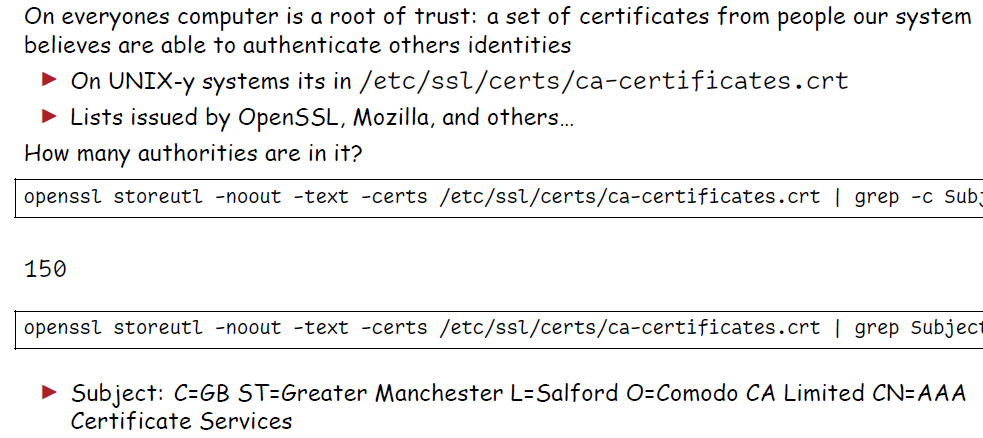
With OpenSSH We fell back to passwords



**certificate-based authentication：OpenSSL**

做加密软件的开源组织，包括openssl工具和库，以我们以前用来安全连接网站的SSL协议命名，所有版本的SSL协议现在都被破坏了，使用最新版本的TLS。有时会偷懒，把TLS称为SSL

Certificate chains：With OpenSSL We used a root of trust

客户端验证是第二个会话中必读内容的一部分。它不为服务器提供保护，也不可能强迫人们只写true，并且在大多数浏览器中很容易禁用。

大多数证书颁发机构都有多级身份验证。他们会给你签个证书。大多数情况下，他们只会给你签个证书，然后检查你的身份。证书颁发机构丢失了密钥吗？是。证书颁发机构是否签署了他们不应该签署的东西？是。证书颁发机构是否应政府的要求签署了文件？是。这是在闹着玩吗？是。

2012 Sascha Fahl **Why Eve and Mallory Love Android:An Analysis of Android SSL (In)Security**

假设我正坐在你的网络连接上，如果我能拦截你的DNS流量（把网站变成电话号码），我就能重定向你的网络流量，如果我能重定向你的网络流量，我就能看到里面有什么并修改它（除非它是加密的），如果我创建一个假证书，我可以解密你的流量，并将其转发到真实的网站，并劫持你所做的，但是你会看到一个警告，上面写着证书不好，除非你关闭了警告。这是中间人攻击！

1. 让我们禁用禁用证书检查的选项！2. 让我们引入证书固定来防止网站之前更改他们的证书到期日！3. 让我们免费获取SSL证书

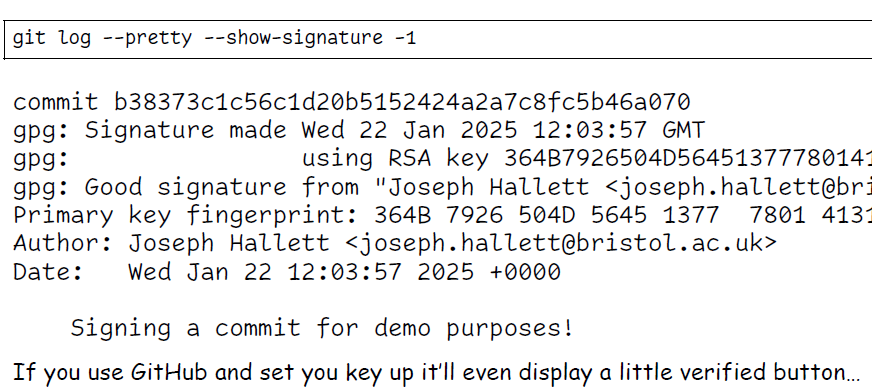
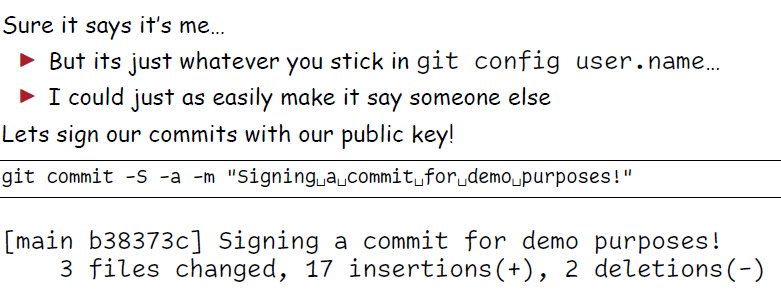
世界上最大的认证机构由Mozilla， EFF和Linux基金会（以及其他）创建，用于加密。只要您能完成他们的ACME协议，他们将免费向任何人颁发证书

在RFC 8555中标准化的ACME协议（2.0版）1. 您向ACME服务器注册您的公钥，并说它将用于管理您的域2. 您创建自己的证书，并要求ACME服务器对其签名3. ACME服务器向您发出挑战，并要求您使用注册的密钥对其进行签名4. 如果您可以将该签名托管在服务器上的. known/acme-challenge/地址上，他们将为您提供一个短暂的TLS证书5. 或者如果你能把那个签名放到DNS记录中他们会给你一个证书

ACME客户机自动执行证书请求和更新过程。RFC 8555中的全安全模型，只要你的密钥是安全的，那么同一个人至少在运行同一台服务器，如果它被泄露，那么至少他们的证书将在很长一段时间内无效

**PGP：We use a web of trust**

由Phil Zimmerman于1996年开发的Pretty Good Privacy。使用公钥加密进行操作的通用框架。通常用来验证下载和电子邮件加密。在一些国家这是非法的！一些国家不喜欢加密软件出口

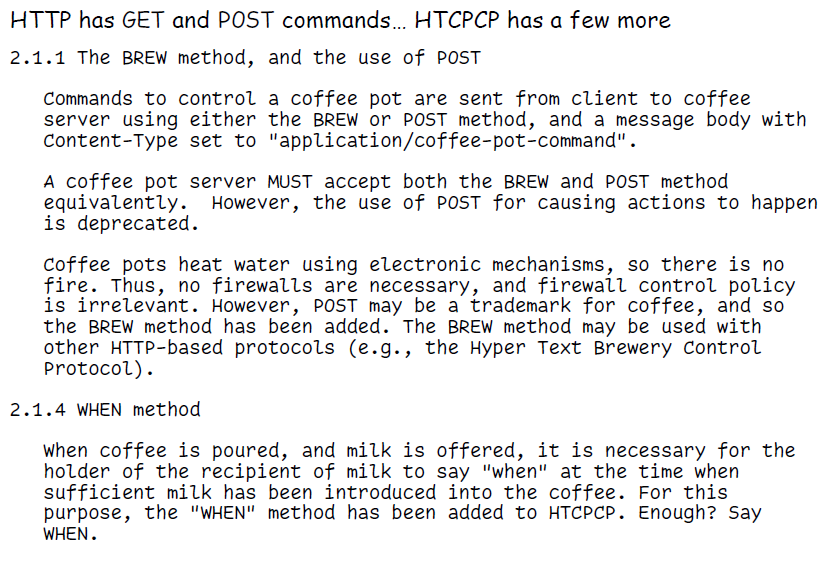
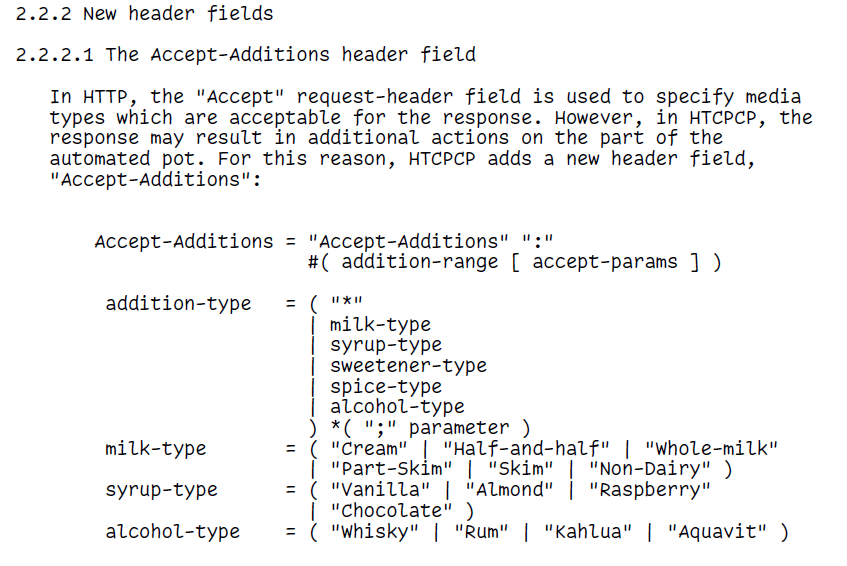


有了PGP，我们有了去中心化的信任，你相信自己（希望如此），如果有人检查你，他们会在你的钥匙上签名，并给你一个签名，如果你检查别人，你在他们的钥匙上签名并给他们签名，当你检查别人的钥匙时，我检查一下你是否信任他们，我检查你是否信任那些信任你的人。各种组织运行密钥服务器，您可以在其中存储密钥…例如<https://keyserver.ubuntu.com>，你可以从这里拿别人的钥匙……但也许在你核实之前不要相信他们？

**Protocols,**

HTTP 1.0 (the original) is documented in IETF RFC 1945。IETF是互联网工程任务组，他们是互联网的标准组织。它是非营利性的，任何人都可以参加，他们通过粗略的共识来决定互联网应该是什么样子，作为这项工作的一部分，他们发布了评论请求，描述了他们认为的协议应该是什么，任何人都可以参加讨论，但不是公司！

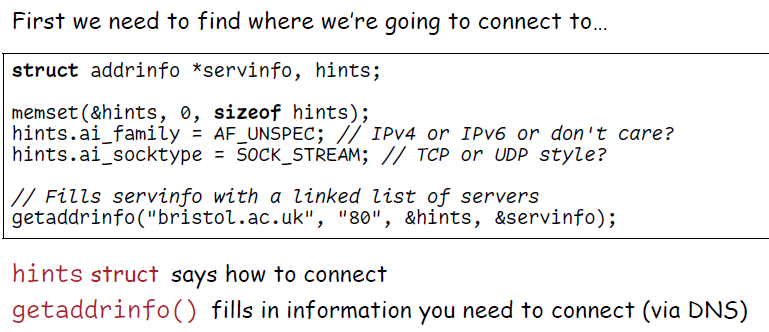
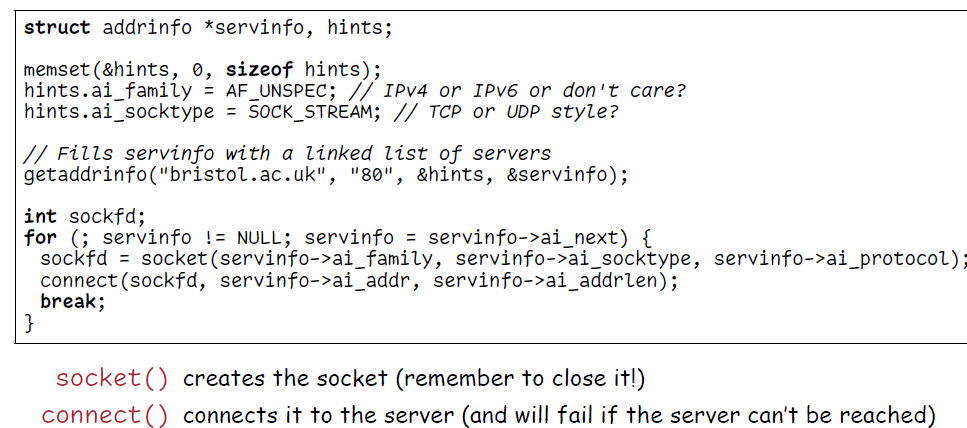
RFC 2324是超文本咖啡壶控制协议（HTCPCP/1.0）HTCPCP协议建立在HTTP之上，增加了一些新的方法、报头字段和返回码。所有的http服务器都应该使用“coffee：”URI方案（第4节）。

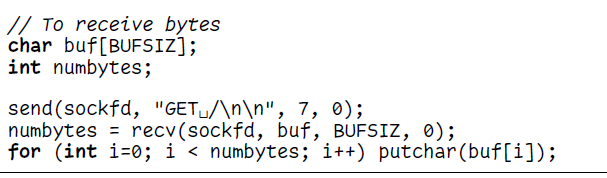
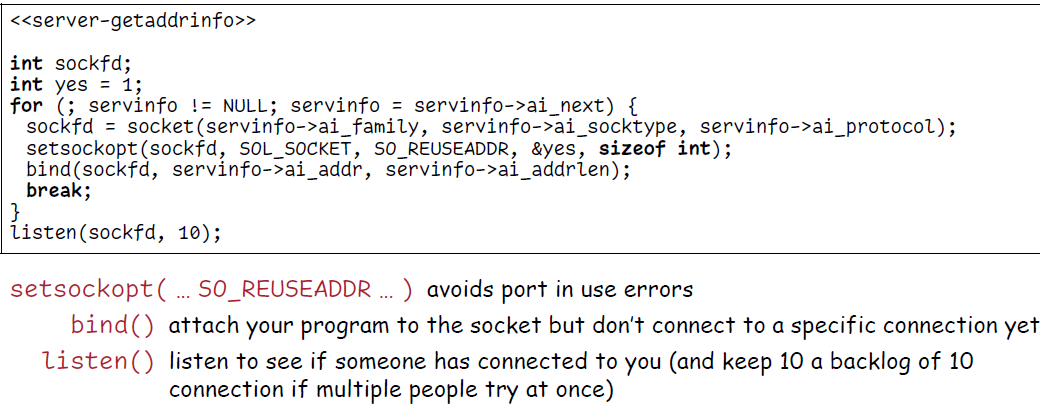
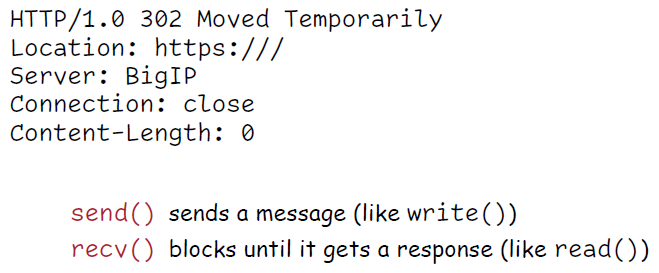
  

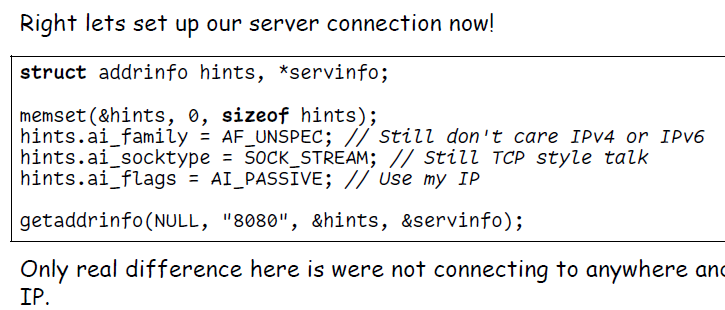
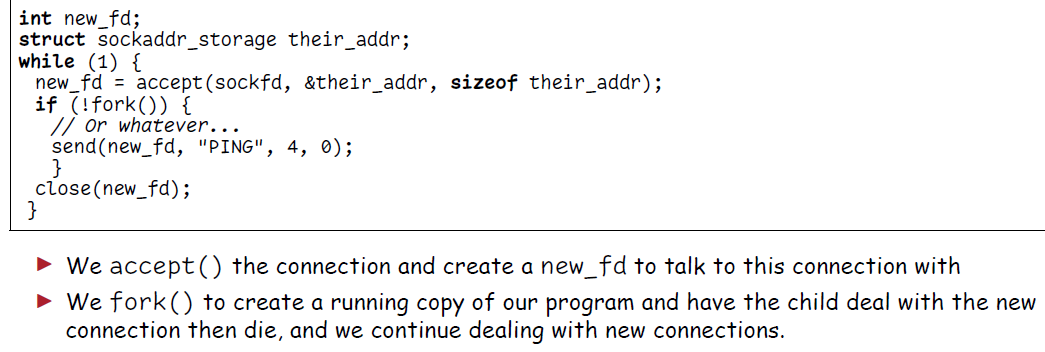
**Sockets**

POSIX的处理方式是使用套接字，类似于文件描述符（低级文件），但是读写是通过网络进行的

服务器（等待连接的东西）将需要打开一个套接字并开始侦听，看看要与谁交谈。Fork这样就不会被堵住

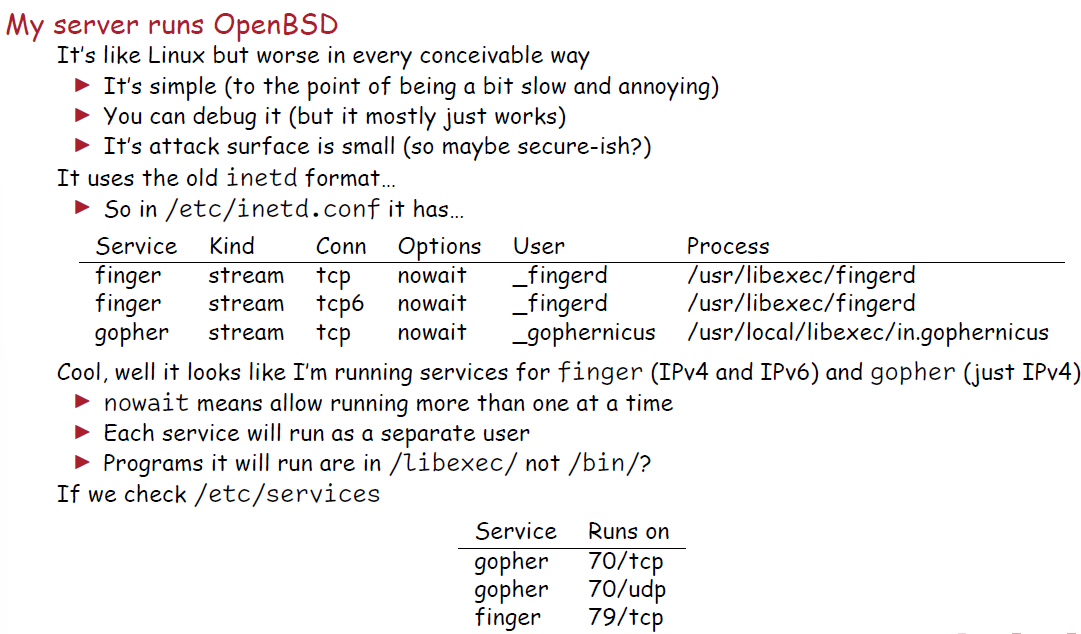
客户端（开始连接的东西）将需要知道它想要与谁交谈



请不要像这样写网络代码！没有错误处理（你会得到一大堆错误）死叉应该被收割，否则你会在进程列表中留下垃圾Beej的使用互联网套接字的网络编程指南

<https://beej.us/guide/bgnet/>

**Inetd**

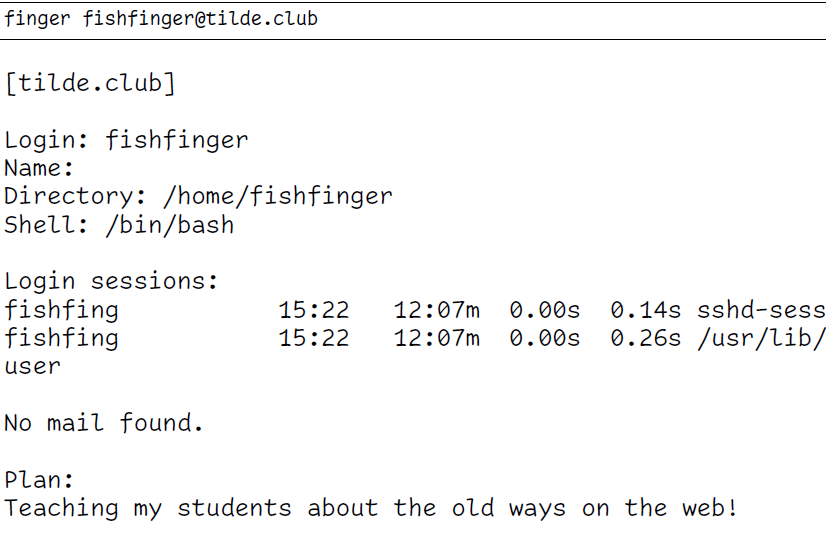
互联网服务守护进程！互联网超级服务（服务的服务）将用任何设置在套接字的stdin， stdout运行。首次出现于1978年，现在真的有点忘记了。觉得有点不安全。主要是因为它的攻击面又大又突出。“现代”版本是存在的

xinetd on Linux，Linux上systemd的类似功能。在MacOS上launchd的类似功能

它实际上是一个shell脚本inetd将标准输入、标准输出连接到套接字

**History**

**Early UNIX-social network**：在互联网出现之前的早期，只有十几台电脑联网，当外部用户拨入时，我很想知道你在和谁通话，如果你能指出他们是谁就太好了。就像一本旧地址簿……RFC 1288手指协议，您通过端口79发送TCP消息，有多少flags表示你想知道多少信息，如果是用户名，就指向那个用户，如果是@列表，表示该系统上的所有用户，如果是用户名@地址，则将手指信息转发到该地址，服务器向您返回它认为是响应的任何内容

我们得到了什么？我的名字和外壳，我登录的会话，我最近是否检查过我的电子邮件（嘿，那是90年代）。我的计划…你可以在你的主目录里放一个。plan文件，告诉别人你在做什么

John Carmack’s .plan files与John Romero一起编写了DOOM和Quake，并创建了ID Software现代FPS

RFC 7033 WebFinger

**Gopher-http html**：RFC 1436 Gopher是1991年在明尼苏达大学开发的，用于目录列表和文件检索，比HTML更加结构化。服务器可用（我使用Gophernicus）与更通用的HTML相比，Gopher最终失败了,在1993年尝试商业化，并收取使用协议的费用,Mimetypes击败了固定的链接类型列表，不能设计一个Gopherhole，加密吗?只有大约300台服务器还在运行。许多Tilde/Shell服务器仍然会给你一个Gopherhole。https://tilde.club https://tilde.town <https://sdf.org>。在Gopherspace上不存在广告，analytics也没有

**Gemini Protocol and the Smolweb**旨在简化，更加关注内容（没有CSS/JS），Gopher的简单文本，但是有一些HTML的好想法，<https://geminiprotocol.net>，几千台服务器

**Testing**

**Asserts** Every language has some version of it…In C it’s in <assert.h> If it’s not true your program crashes。Assert（通常）只是一个用于调试的宏…不希望它出现在产品代码中，因为它可能会导致崩溃，所以通常它会从发布代码中删除。如果使用-DNDEBUG=1编译，assert（）将被删除。Assert很适合在编写代码时测试程序员的假设……但是这对于检查你的代码是否总是工作是不好的，理想情况下，我们希望对已知的良好值和结果进行一系列检查。（可能针对已知的失败原因）

**Unit tests** 单元测试根据一系列已知输入对代码进行测试，并检查结果。很高兴看到你的代码是变得更好还是更糟。通过或不通过更多的测试。用一堆if语句。如果您正在为生产编写代码，这是您所期望的最低限度。

python -m pytest。They’re written in code。Not everyone can write code…

**Acceptance testing**如果我们能够接受我们被要求实现的规范并根据它进行测试，那不是很好吗？这样我们就不必编写自己的单元测试了。https://cucumber.io你用自然语言编写规范（特性），编写语法规则将规范转换为代码

**Property testing**与其编写单独的测试，不如尝试捕获全局属性。例如，如果我们取一个大于1的数的幂，它会变得更大。我们要检查这个属性是否适用于任何输入，我们不想自己生成函数输入。那么，哪些属性可能适合测试呢？与旧代码等价？可逆吗?如果我计算一个数的平方根，我应该得到相同的数，如果我打印一个数据结构，我应该能够读取相同的数据结构。没有输入我的程序应该崩溃…

**Fuzz testing**尝试随机输入，看看是什么触发了崩溃。尝试损坏的输入……看看通过检测代码会碰到什么逻辑路径。试着找到触发程序中每个条件的每个路径的输入！

**Formal proof**确保我们的代码正常工作……我们可以正式证明它是正确的。它永远不会崩溃（在任何输入下）它总能找到答案（在合理的时间内。停止问题是指你无法判断一个程序是否会完成。如果你有一个计算机的数学模型，你就可以证明一个算法的正确性。如果你有一个数学模型，你可以证明算法之间的等价…如果你有一个数学模型，你可以建立精确的界限，多少操作需要完成…如果你能以一种合理的方式将这个模型转换成代码，那么你就有了生成正确程序的途径。无需测试代码是否正确！密码学与现代网络、内存分配。Lean 4 is a proof assistant，它懂得数学推理……它可以检查你的证明，并帮助你写他们。我们可以证明我们的平方求幂算法等价于重复乘法。