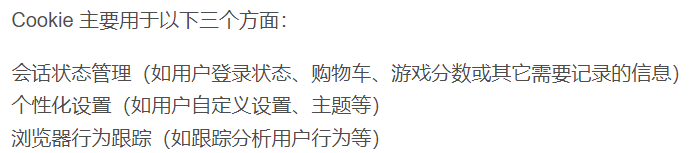
## cookie和session

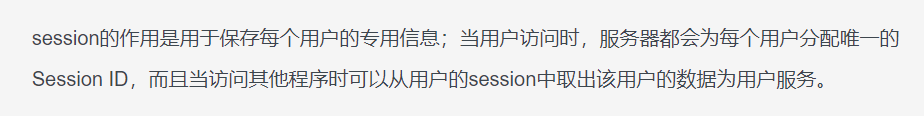
1. cookie和session是什么？

Cookie，有时也用其复数形式 Cookies，指某些网站为了辨别用户身份、进行 session 跟踪而储存在用户本地终端上的数据（通常经过加密）

Session：在计算机中，尤其是在网络应用中，称为“会话控制”。Session对象存储特定用户会话所需的属性及配置信息。

1. cookie和session的作用？





1. cookie和session的区别与联系？

**简答题:**

cookie和session本质的区别，一个位于客户端，一个位于服务端。

cookie与session区别有：1、对象不同；2、存储数据大小不同；3、生命周期不同；4、存储位置不同；5、数据类型不同；6、安全性不同。





**联系：Session 机制是建立在 Cookie 机制之上的, 即**

服务器会在用户登录成功后创建一个Session，然后将该Session的标识符存储在一个Cookie中，发送给客户端浏览器。之后，每当浏览器向服务器发送请求时，服务器都会检查请求中是否包含Session标识符的Cookie，从而找到相应的Session并检索相关信息。因此，Cookie和Session在Web应用程序中通常是相互配合使用的。

用户第一次请求服务器的时候，服务器根据用户提交的相关信息，创建创建对应的 Session ，请求返回时将此 Session 的唯一标识信息 SessionID 返回给浏览器，浏览器接收到服务器返回的 SessionID 信息后，会将此信息存入到 Cookie 中，同时 Cookie 记录此 SessionID 属于哪个域名。

当用户第二次访问服务器的时候，请求会自动判断此域名下是否存在 Cookie 信息，如果存在自动将 Cookie 信息也发送给服务端，服务端会从 Cookie 中获取 SessionID，再根据 SessionID 查找对应的 Session 信息，如果没有找到说明用户没有登录或者登录失效，如果找到 Session 证明用户已经登录可执行后面操作。

**补充:常见使用场景**

**1.**用户登录状态管理：当用户登录成功后，服务器会创建一个Session，然后将该Session的标识符存储在一个Cookie中，发送给客户端浏览器。之后，每次用户访问需要登录的页面时，浏览器都会携带该Cookie，服务器则可以根据Cookie中的Session标识符来找到对应的Session，从而判断用户是否已经登录，并返回相应的页面内容。

**2.**购物车管理：Web商城通常使用Cookie来存储用户的购物车信息，当用户添加商品到购物车时，服务器会将购物车内容存储在Cookie中。当用户浏览购物车页面时，浏览器会将购物车Cookie发送给服务器，服务器则可以从Cookie中读取购物车内容并返回相应的页面内容。

**3.**个性化设置管理：Web应用程序通常使用Cookie或Session来存储用户的个性化设置，例如网站主题、语言偏好等，当用户更改设置时，服务器会将新的设置存储在Cookie或Session中。之后，当用户再次访问网站时，服务器会读取Cookie或Session中的设置信息，并返回相应的页面内容。

**4.**记住密码功能：当用户勾选了“记住密码”选项并成功登录后，服务器可以将用户的登录凭证存储在Cookie中，下次用户访问该网站时，可以直接使用Cookie中的登录凭证自动登录，无需再次输入账号和密码。

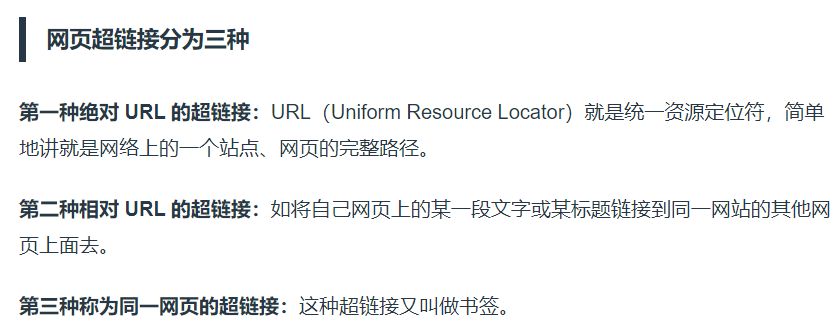
**5.**记录用户行为：Web应用程序可以使用Cookie或Session来记录用户的行为信息，例如用户浏览的页面、点击的链接等。这些行为信息可以用于分析用户行为、推荐相关内容等。

**6.**跨域通信：由于浏览器的同源策略限制，不同域名下的网页无法直接访问彼此的Cookie。为了实现跨域通信，可以使用类似于JSONP的方式，在请求参数中附带一个session ID，服务器在响应中返回该session ID对应的Session的信息，从而完成跨域通信。

## 关于URL

**超文本中的超链接**：超链接就是某个锚文本或者是网址，甚至是图片，其本身自带了链接属性，用户点击这个链接后就能跳转至其他页面，这个文本及包含的属性链接就是超链接。





从输入URL到网页呈现的过程大致如下：

​ 1、浏览器的地址栏输入URL并按下回车。

　　2、浏览器查找当前URL是否存在缓存，并比较缓存是否过期。

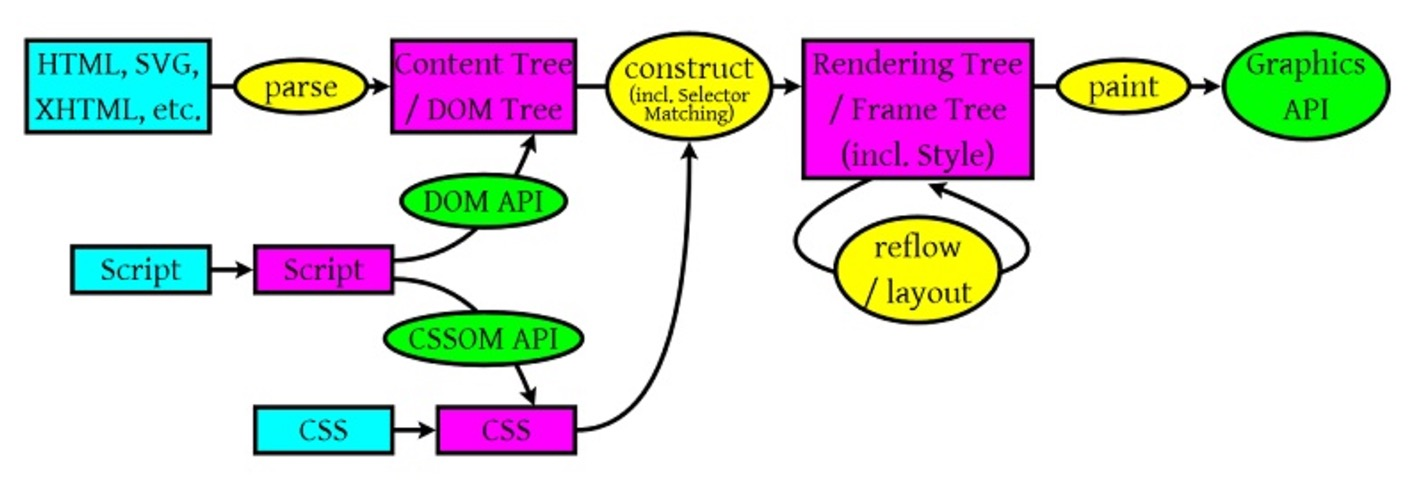
　　3、DNS解析URL对应的IP。

　　4、根据IP建立TCP连接（三次握手）。

　　5、HTTP发起请求。

　　6、服务器处理请求，浏览器接收HTTP响应。

7、渲染页面，构建DOM树。



　　8、关闭TCP连接（四次挥手）

用户单击鼠标后所发生的事件按顺序如下(以访问清华大学的网站为例)

例子：（很有用，可以看出各个部分都做了什么）

浏览器向DNS请求解析www.tsinghua.edu.cn的IP地址。

域名系统DNS解析出清华大学服务器的IP地址。

浏览器与该服务器建立TCP连接(默认端口号为80)。

浏览器发出HTTP请求:GET/chn/index.htm6)

服务器通过HTP响应把文件 index htm发送给浏览器。

释放TCP连接。

浏览器解释文件 index.html,并将Web页显示给用户。

## HTTP请求头和响应头

请求头：

请求的第一行:

GET /index.html HTTP/1.1

Host: www.example.com

在起始行中，有三段信息:

GET ：用于说明想要**服务器执行的操作**，此外还有PUT、POST等操作

/index.html ：**资源的路径**。这里指向服务器上的index.html文件。

HTTP/1.1： **协议的版本**。HTTP第一个广泛使用的版本是1.0，当前版本为1.1。

HTTP的请求必须有Host头信息，用于说明服务器的地址和端口。HTTP协议的默认端口是80，如果在HOST中没有说明端口，那么将默认采取该端口。

响应头：

HTTP/1.1 200 OK

Content-type: text/plain

Content-length: 12

Hello World!

回复的起始行同样包含三段信息

HTTP/1.1 协议版本

200 状态码(status code)

OK 状态描述

OK是对状态码200的文字描述，它只是为了便于人类的阅读。电脑只关心三位的状态码(status code)，即这里的200。200表示一切OK，资源正常返回。状态码代表了服务器回应动作的类型。

**1xx（信息性状态码）**

100：Continue，表示服务器已经接收到了请求头，并且客户端应该继续发送请求体。

101：Switching Protocols，表示服务器已经理解了客户端的请求，并将通过Upgrade头信息告诉客户端采用不同的协议。

**2xx（成功状态码）**

200：OK，表示服务器已经成功处理了请求。

201：Created，表示服务器已经成功创建了资源。

204：No Content，表示服务器已经成功处理了请求，但是没有返回任何内容。

**3xx（重定向状态码）**

301：Moved Permanently，表示所请求的资源已经被永久地移动到了新的URI。

302：Found，表示所请求的资源已经被临时地移动到了新的URI。

304：Not Modified，表示所请求的资源没有被修改，客户端可以使用缓存中的数据。

**4xx（客户端错误状态码）**

400：Bad Request，表示客户端发送的请求存在语法错误。

401：Unauthorized，表示客户端需要进行身份验证才能访问所请求的资源。

403：Forbidden，表示服务器拒绝了客户端的请求。

404：Not Found，表示所请求的资源不存在。

**5xx（服务器错误状态码）**

500：Internal Server Error，表示服务器在处理请求时发生了未知的错误。

503：Service Unavailable，表示服务器暂时无法处理请求。

## 有关CSS布局

CSS**正常布局流**是指元素按照文档流自上而下、从左到右的顺序排列的方式。CSS正常布局流的布局规则如下：

1. 块级元素独占一行，从上往下排列；
2. 行内元素按照顺序，从左到右排列，碰到父元素边缘则自动换行；
3. 块级元素和行内元素都有一个默认的宽度，块级元素默认为父元素的宽度，行内元素则根据其内容自适应宽度；
4. 块级元素和行内元素都有一个默认的高度，块级元素默认为内容的高度加上内边距和边框，行内元素则根据其字体大小自适应高度；
5. 块级元素可以设置宽度、高度、内边距和边框；
6. 行内元素可以设置内边距和边框，但是不可以设置宽度和高度；
7. 块级元素和行内元素都可以设置外边距。

CSS提供了3种机制来设置盒子的摆放位置，分别是**普通流（标准流）、浮动和定位**，其中：

一、普通流（标准流）

块级元素会独占一行，从上向下顺序排列；

常用元素：div、hr、p、h1~h6、ul、ol、dl、form、table

行内元素会按照顺序，从左到右顺序排列，碰到父元素边缘则会自动换行；

常用元素：span、a、i、em等

二、浮动

让盒子从普通流中浮起来，主要作用让多个块级盒子一行显示。

三、定位

将盒子定在浏览器的某一个位置——CSS离不开定位，特别是后面的js特效。

**补充1:**

1. 普通流（标准流）是 CSS 中默认的盒子摆放机制，它可以自动计算盒子的位置和大小，从而实现基本的页面布局。
2. 浮动可以让盒子沿着父元素的左侧或右侧浮动，多个浮动盒子可以在同一行中显示。浮动盒子的高度不受内容的限制，**会自动根据内容高度变化**，需要通过**清除浮动**来解决它对后续元素布局的影响。
3. 定位可以将盒子固定在某个位置，通过 top、right、bottom、left 四个属性来控制盒子的位置。相对定位（position: relative）会**保留盒子原本在标准流中**的位置，绝对定位（position: absolute）会将盒子**从标准流中完全移除**，固定定位（position: fixed）会将盒子固定在**视口**的某个位置，不随页面滚动而改变。

**补充2: 脱离文本流的方法和布局方法**

元素脱离文本流后，不再按照标准流的排列方式来排列，会导致元素位置的变化，通常表现为下列**特征**之一或多个：

1. 元素浮动后不再占据文档流中的位置，导致后续元素可以覆盖它；
2. 元素绝对定位后脱离了文档流，不再对后续元素进行布局，但对于内部子元素仍然遵循标准流的排列方式；
3. 元素设置了固定定位后脱离文档流，位置固定，不随页面滚动而变化。
4. 当元素脱离文本流后，需要采用特定的方式进行布局：
5. 对于浮动元素，可以通过设置其浮动方向，以及给其父元素添加 clear 属性来进行布局，防止后续元素覆盖；
6. 对于绝对定位元素，可以通过给其父元素添加相对定位属性，或者对其进行定位的偏移量进行布局；
7. 对于固定定位元素，一般是在固定位置进行展示，通过设置 top、right、bottom、left 等属性来确定其具体的位置。

#### 一、什么是浮动

**概念：元素的浮动是指设置了浮动属性的元素**会

1. 脱离标准普通流的控制
2. 移动到指定的位置。

**作用**：

1. 让多个盒子（div）水平排列成一行，使得浮动成为布局的重要手段。
2. 可以实现盒子的左右对齐等等
3. 浮动最早是用来**控制图片**，实现**文字环绕图片的效果**.

## 后端和前端系统的工作:

在一个 Web 应用程序中，后端和前端通常各自负责不同的任务。

前端主要负责的是用户界面的设计和开发，包括页面的布局、样式设计、交互功能实现等。前端通常使用 HTML、CSS 和 JavaScript 等技术来完成这些任务。前端的主要目标是提供用户友好的界面和交互方式，使得用户可以方便地使用和操作应用程序。

后端主要负责的是应用程序的业务逻辑和数据处理，包括但不限于以下任务：

* 设计数据库模型和数据存储方案；
* 实现业务逻辑，例如用户认证、权限管理、订单处理等；
* 设计和开发 API 接口，提供数据的输入和输出；
* 与其他系统进行数据交互，例如支付系统、第三方登录等；
* 性能优化、安全防护等。

## 后端服务器的作用(例如Tomcat)

在 Web 应用程序中，后端服务器的作用主要有以下几个方面：

* 接受和处理 HTTP 请求：后端服务器作为 Web 服务器，可以接受客户端发来的HTTP 请求，并将其转发给后端应用程序进行处理。
* 托管 Web 应用程序：后端服务器可以托管运行 Web 应用程序，例如基于 Java 技术栈的应用程序。
* 管理应用程序生命周期：后端服务器可以管理 Web 应用程序的生命周期，包括启动、停止、重启等操作。
* 处理并发请求：后端服务器可以处理大量并发的请求，支持高并发访问。
* 提供安全性保护：后端服务器可以提供一些安全性保护措施，例如 SSL 加密、认证、授权等。
* 负载均衡：后端服务器可以作为负载均衡器，将请求分发到多个应用服务器上，从而提高系统的性能和可靠性。
* 缓存：后端服务器可以通过缓存静态资源和动态数据，减少应用程序的响应时间和资源消耗。
* 日志记录：后端服务器可以记录 HTTP 请求和响应的日志，用于监控和调试应用程序。
* 数据库连接池：后端服务器可以维护数据库连接池，提高数据库访问效率。
* 服务容错：后端服务器可以通过多节点部署、热备份等方式提高服务的可用性和容错性。
* API 网关：后端服务器可以作为 API 网关，对外提供统一的 API 接口，同时对内管理多个微服务之间的访问。
* 分布式计算：后端服务器可以通过分布式计算技术，将任务分配到多个节点进行处理，提高计算效率。

## Js引入和执行差异

JavaScript在HTML中的引入位置会对其执行产生不同的影响，主要包括以下两个方面：

对DOM操作的影响：如果JavaScript代码在DOM元素之前被引入，那么代码中的DOM操作将会失败，因为DOM元素还没有被解析。如果JavaScript代码在DOM元素之后被引入，那么代码中的DOM操作将会成功。

对页面渲染的影响：当浏览器解析HTML文档时，它是按照自上而下的顺序进行解析的。如果JavaScript代码在页面底部引入，那么浏览器会在代码执行前先加载完整个页面，这样就可以加快页面加载速度，并且避免在页面渲染时出现延迟。相反，如果JavaScript代码在页面顶部引入，那么浏览器在执行JavaScript代码之前必须先加载整个JavaScript文件，这样会延迟页面的加载和渲染。

综上所述，一般建议将JavaScript代码放在HTML页面底部引入，这样可以加快页面的加载速度，避免延迟，同时还可以避免DOM操作失败的问题。