HTML, CSS And JS

1. HTML 是一种标记语言，用来结构化我们的网页内容并赋予内容含义，例如定义段落、标题和数据表，或在页面中嵌入图片和视频
2. CSS 是一种样式规则语言，可将样式应用于 HTML 内容， 例如设置背景颜色和字体，在多个列中布局内容
3. JavaScript 是一种脚本语言，可以用来创建动态更新的内容，控制多媒体，制作图像动画，还有很多其它有趣的功能
4. HTML：
    a. HTML 不是一门编程语言，而是一种用于定义内容结构的标记语言；HTML 由一系列的元素 (elements) 组成，这些元素可以用来包围不同部分的内容，使其以某种方式呈现或者工作； 一对标签 (tags) 可以为一段文字或者一张图片添加超链接，将文字设置为斜体，改变字号，等等
    b. 在 HTML 中有两种重要的元素类别，块级元素和内联元素：
        - 块级元素在页面中以块的形式展现 —— 相对于其前面的内容它会出现在新的一行，其后的内容也会被挤到下一行展现；块级元素通常用于展示页面上结构化的内容，例如段落、列表、导航菜单、页脚等等；一个以 block 形式展现的块级元素不会被嵌套进内联元素中，但可以嵌套在其它块级元素中
        - 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：它通常出现在一堆文字之间例如超链接元素 <a> 或者强调元素 和
    c. HTML 头部是包含在 <head> 元素里面的内容，不像 <body> 元素的内容会显示在浏览器中，head 里面的内容不会在浏览器中显示，它的作用是包含一些页面的元数据，包含了像页面的 <title> (标题)，CSS (如果你选择用 CSS 来为 HTML 内容添加样式)，指向自定义图标的链接和其他的元数据 (描述HTML的数据，比如，作者，和描述文档的重要关键词)
    d. 关于粗体 <b>，斜体 <i>，下划线 <u> 最好的经验法则：使用 <b>，<i>，<u> 来传达传统意义上的粗体，斜体或下划线是合适的，没有其他元素更适合这样用了；然而，始终拥有可访问性的思维模式是至关重要的；斜体的概念对人们使用屏幕阅读器是没有帮助的，对使用其他书写系统而不是拉丁文书写系统的人们也是没有帮助的：
        - <i> 被用来传达传统上用斜体表达的意义：外国文字，分类名称，技术术语，一种思想……
        - <b> 被用来传达传统上用粗体表达的意义：关键字，产品名称，引导句……
        - <u> 被用来传达传统上用下划线表达的意义：专有名词，拼写错误……
    e. 超链接是互联网提供的最令人兴奋的创新之一，它们从一开始就一直是互联网的一个特性，使互联网成为互联的网络；超链接使我们能够将我们的文档链接到任何其他文档 (或其他资源)，也可以链接到文档的指定部分，我们可以在一个简单的网址上提供应用程序 (与必须先安装的本地应用程序或其他东西相比)
    f. 网页的外观多种多样，但一般都倾向于使用类似的标准组件：
        - 页眉：通常横跨于整个页面顶部有一个大标题 和/或 一个标志。 这是网站的

主要一般信息，通常存在于所有网页

- 导航栏：指向网站各个主要区段的超链接；通常用菜单按钮、链接或标签页表示；类似于标题栏，导航栏通常应在所有网页之间保持一致，否则会让用户感到疑惑，甚至无所适从
- 主内容：中心的大部分区域是当前网页大多数的独有内容，例如视频、文章、地图、新闻等；这些内容是网站的一部分，且会因页面而异
- 侧边栏：一些外围信息、链接、引用、广告等；通常与主内容相关（例如一个新闻页面上，侧边栏可能包含作者信息或相关文章链接），还可能存在其他的重复元素，如辅助导航系统
- 页脚：横跨页面底部的狭长区域；和标题一样，页脚是放置公共信息（比如版权声明或联系方式）的，一般使用较小字体，且通常为次要内容

g. id 属性只能在每个 HTML 文档中出现一次

5. CSS：即层叠样式表，HTML 用于定义内容的结构和语义，CSS 用于设计风格和布局；比如，我们可以使用 CSS 来更改内容的字体、颜色、大小、间距，将内容分为多列，或者添加动画及其他的装饰效果
   a. CSS 是用来指定文档如何展示给用户的一门语言——如网页的样式 (比如改变标题和链接的颜色及大小) 、布局 (比如将一个单列文本变成包含主要内容区域和存放相关信息的侧边栏区域的布局) 、等等
   b. 所有的标准 Web 技术 (HTML, CSS, JavaScript 等) 都被定义在一个巨大的文档中，称作 规范 specifications (或者简称为 "specs")，它是由 (像是 W3C, WHATWG, ECMA 或 Khronos) 这些规范化组织所发布的，其中还定义了各种技术是如何工作的；这些规范 (包括 CSS 规范) 使得即使技术是不断发展更新的，但是其核心是不变的
   c. 在文本中使用 CSS 一般有三种方式：
      - 外部样式表：外部样式表是指将 CSS 编写在扩展名为 .css 的单独文件中，并从HTML <link> 元素引用它的情况；这是将 CSS 附加到文档中的最常见和最有用的方法，因为我们可以将 CSS 链接到多个页面，从而允许您使用相同的样式表设置所有页面的样式；在大多数情况下，一个站点的不同页面看起来几乎都是一样的，因此我们可以使用相同的规则集来获得基本的外观
      - 内部样式表：内部样式表是指不使用外部 CSS 文件，而是将 CSS 放在 HTML 文件 <head> 标签里的 <style> 标签之中
      - 内联样式：内联样式表存在于 HTML 元素的 style 属性之中；其特点是每个 CSS 表只影响一个元素
   d. 在 CSS 中，属性和值都是区分大小写的；每对中的属性和值由冒号 (:) 分隔
   e. CSS 中的一些基本概念：
      - 层叠, 和它密切相关的概念是 specificity，决定在发生冲突的时候应该使用哪条规则；设计元素样式的规则可能不是期望的规则，因此需要了解这些机制是如何工作的：
        - 简单地说，CSS 规则的顺序很重要；当应用两条同级别的规则到一个元素的时候，写在后面的就是实际使用的规则
        - 优先级：浏览器是根据优先级来决定当不同选择器 (多个规则) 对应相同的元素的时候需要使用哪个规则 --- 一个元素选择器不是很具体，它会选择页面上该类型的所有元素，所以它的优先级就会低一些。而一个

类选择器稍微具体点，它会选择该页面中有特定 class 属性值的元素，所以它的优先级就要高一点

- 继承，也就是在默认情况下，一些 CSS 属性继承当前元素的父元素上设置的值，有些则不继承，这可能导致一些和期望不同的结果；

f. 在 HTML 中引用 CSS 样式有四种方式：
   - 行内样式：直接对 HTML 的标签使用 style 属性，然后将 CSS 代码直接写进去
   - 内嵌样式：将 CSS 写在 <head> 与 </head> 之间，并且用 <style>和 </style> 标记进行声明
   - 链接样式：在文件 <head> 和 </head> 标签之间加上 <link href = "sheet.css" type = "text/css" rel = "stylesheet">，将 CSS 文件链接到页面中，对其中的标签进行样式控制
   - 导入样式：与链接样式表的功能基本相同，仅在语法和运作方式上与链接样式表略有区别；采用 @import 方式导入的样式表，在 HTML 文件初始化时，会被导入到 HTML 文件内

g. 在 CSS 中有两种不同类型的字体系列：
   - 通用字体系列 - 拥有相似外观的字体系统组合 (比如 "Serif" 或 "Monospace")
     - Serif 字体
     - Sans-serif 字体
     - Monospace 字体
     - Cursive 字体
     - Fantasy 字体
   - 特定字体系列 - 具体的字体系列 (比如 "Times" 或 "Courier")：特定字体系列是具体的，包含于某个通用字体系列中，例如 "Times" 属于 Serif 字体

h. 背景应用于由内容和内边距、边框组成的区域，因此内边距区域的背景颜色等和内容区域颜色一样

i. 定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置

j. div、h1 或 p 元素常常被称为块级元素；这意味着这些元素显示为一块内容，即“块框”；与之相反，span 和 strong 等元素称为“行内元素”，这是因为它们的内容显示在行中，即“行内框”

k. CSS 中的定位机制：CSS 有三种基本的定位机制：普通流、浮动和绝对定位；除非专门指定，否则所有框都在普通流中定位，也就是说，普通流中的元素的位置由元素在 (X)HTML 中的位置决定；块级框从上到下一个接一个地排列，框之间的垂直距离是由框的垂直外边距计算出来；行内框在一行中水平布置，可以使用水平内边距、边框和外边距调整它们的间距；但是，垂直内边距、边框和外边距不影响行内框的高度；由一行形成的水平框称为行框（Line Box），行框的高度总是足以容纳它包含的所有行内框，并且，设置行高可以增加这个框的高度

l. CSS 中的 boxes 可分为四种：containing boxes (包含框，由行框组成)，inline boxes (行内框或者内联框，其中高度最大者的高度为 line-height，即 line boxes 的高度)，line boxes (行框)，content area (内容区域)

m. 伪类对元素进行分类是基于特征 (或者说状态) 而不是它们的名字、属性或者内

容；原则上特征 (状态) 是不可以从文档树上推断得到的

6. Javascript:
    a. 客户端（client-side）JavaScript 语言的核心包含一些普遍的编程特性，以让你可以做到如下的事情：
        - 在变量中储存有用的值
        - 操作一段文本 (在编程中称为“字符串” (string))
        - 运行代码以响应网页中发生的特定事件
        - 其它
    b. JavaScript 语言核心之上还构建了更高级的功能---应用程序接口 (Application Programming Interfaces (API)) 将为我们的代码提供额外的超能力；API 是已经建立好的一套代码组件，可以让开发者实现原本很难甚至无法实现的程序，通常分为以下几类：
        - 文档对象模型 API（DOM（Document Object Model）API） 能通过创建、移除和修改 HTML，为页面动态应用新样式等手段来操作 HTML 和 CSS
        - 地理位置 API（Geolocation API） 获取地理信息
        - 画布（Canvas） 和 WebGL API 可以创建生动的 2D 和 3D 图像
        - 诸如 HTMLMediaElement 和 WebRTC 等影音类 API 让我们可以利用多媒体做一些非常有趣的事，比如在网页中直接播放音乐和影片，或用自己的网络摄像头获取录像，然后在其他人的电脑上展示
        - 注：前面所提到的 API 均为浏览器内建的 API ，还有一些第三方 API
    c. 浏览器在读取一个网页时都发生什么；浏览器在读取一个网页时，代码 (HTML, CSS 和 JavaScript) 将在一个运行环境 (浏览器标签页) 中得到执行；就像一间工厂，将原材料 (代码) 加工为一件产品 (网页) ；在 HTML 和 CSS 集合组装成一个网页后，浏览器的 JavaScript 引擎将执行 JavaScript 代码。这保证了当 JavaScript 开始运行之前，网页的结构和样式已经就位；这样很好，因为 JavaScript 最普遍的用处是通过 DOM API 动态修改 HTML 和 CSS 来更新用户界面 (user interface) ，如果 JavaScript 在 HTML 和 CSS 就位之前加载运行，就会引发错误；每个浏览器标签页就是其自身用来运行代码的独立容器 (这些容器用专业术语称为“运行环境”) ；大多数情况下，每个标签页中的代码完全独立运行，而且一个标签页中的代码不能直接影响另一个标签页 (或者另一个网站) 中的代码
    d. 服务器端 (server-side) 和客户端 (client-side) 代码：客户端代码是在用户的电脑上运行的代码，在浏览一个网页时，它的客户端代码就会被下载，然后由浏览器来运行并展示，这就是客户端 JavaScript；而服务器端代码在服务器上运行，接着运行结果才由浏览器下载并展示出来；流行的服务器端 web 语言包括：PHP、Python、Ruby、ASP.NET 以及 JavaScript！JavaScript 也可用作服务器端语言，比如现在流行的 Node.js 环境
    e. 一般来说，JavaScript 运行在用户的终端网页上，而不是服务器上，所以我们称之为“前端语言”；就是服务于页面的交互效果、美化，不能操作数据库；后台语言是运行在服务器上的，比如 PHP、ASP、JSP 等等，这些语言都能够操作数据库，都能够对数据库进行“增删改查”操作
    f. Javascript 基础包括三个部分：
        - ECMAScript：JavaScript 的语法标准；包括变量、表达式、运算符、函

数、if 语句、for 语句等
    - DOM：Document Object Model (文档对象模型)，操作页面上的元素的 API；比如让盒子移动、变色、改变大小、轮播图等等
    - BOM：Browser Object Model (浏览器对象模型)，操作浏览器部分功能的 API；通过 BOM 可以操作浏览器窗口，比如弹框、控制浏览器跳转、获取浏览器分辨率等等
    - 通俗理解就是：ECMAScript 是 JS 的语法；DOM 和 BOM 浏览器运行环境为 JS 提供的 API
g. Javascript 有六种数据类型：
    - 基本数据类型 (值类型)：String 字符串、Number 数值、Boolean 布尔值、Null 空值、Undefined 未定义
    - 引用数据类型 (引用类型)：Object 对象；内置对象 Function，Array，Date，RegExp，Error 等都是属于 Object 类型；也就是说，除了那五种基本数据类型之外，其他的，都称之为 Object 类型
    - 值类型和引用类型的区别：在作参数赋值的时候，值类型的变量直接传递值而引用类型变量传递的是值的地址，即对同一内存空间操作；事实上引用类型变量保存的是引用对象的地址
h. 在 Javascript 中，所有的变量都保存在栈内存中；值类型变量的值，即对象也保存在栈内存中且值与值之间相互独立；引用对象则保存堆内存中，变量保存了对象的地址 (对象引用)，储存在栈内存中
i. 流程控制语句分类：
    - 顺序结构
    - 选择结构
    - 循环结构
j. 面对对象的特征：封装，继承和多态
k. 面对过程编程一般是先分析好的具体步骤，然后按照步骤，一步步解决问题；性能比面向对象高，适合跟硬件联系很紧密的东西，例如单片机就采用的面向过程编程，但是没有面向对象易维护、易复用、易扩展；面向对象编程则以对象功能来划分问题，而不是步骤；
   面对对象编程易维护、易复用、易扩展，由于面向对象有封装、继承、多态性的特性，可以设计出低耦合的系统，使系统更加灵活、更加易于维护；缺点则是性能比面向过程低
l. 面向对象的编程思想：对代码和数据进行封装，并以对象调用的方式，对外提供调用接口
m. 使用 var 关键字声明的变量（比如 var a = 1），会在所有的代码执行之前被声明 (但是不会赋值)，但是如果声明变量时不是用 var 关键字 (比如直接写a = 1)，则变量不会被提前声明
n. 闭包 (closure)：指有权访问另一个函数作用域中变量的函数
o. new 一个构造函数的执行过程：
    - 开辟内存空间，在内存中创建一个新的空对象
    - 让 this 指向这个新的对象
    - 执行构造函数里面的代码，给这个新对象添加属性和方法

- 返回这个新对象 (所以构造函数里面不需要 return)

p. var，let 和 const 的区别：
   - var 定义的变量被提升了并且初始化为 undefined ， let 和 const 定义的变量都会被提升，但是不会被初始化，不能被引用
   - let 和 const 定义的变量只在块级作用域中有效
   - const 定义的变量的引用不能被更改

q. 事件三要素：事件源、事件、事件驱动程序
   - 事件源：引发后续事件的html标签
   - 事件：js已经定义的操作，可以通过鼠标 (或键盘) 触发
   - 事件驱动程序：通过 Javascript 对样式和 html 的操作，也就是对 DOM 的调用

r. 节点 (Node) ： 构成 HTML 网页的最基本单元；网页中的每一个部分都可以称为是一个节点，比如：html标签、属性、文本 (包括标签之间的空格，回车) 、注释、整个文档等都是一个节点；常见分类如下：
   - 文档节点 (文档) ： 整个 HTML 文档
   - 元素节点 (标签) ： HTML 标签
   - 属性节点 (属性) ： 元素的属性
   - 文本节点 (文本) ： HTML 标签中的文本内容 (包括标签之间的空格、换行)