SI339 Exam

▼ HTML 语法和标签，属性（不重要）

分为 head 和 body 两部分，head 描述页面属性，没有主要内容。body 是主要内容。

body 一般有 header main footer（但不是必须）。

添加 css，<link rel="stylesheet" type="text/css" href="css/style.css" />，在head里面

标题：h1 - h6
段落：<p>
换行：
1个框：
链接：<a href=>，target=”_blank”，在新标签页打开
列表： ，注意先要在外面有 ul 或 ol，里面每一项再是 li
图片：<img src=”” alt=””>，自闭合
导航：<nav>，没有实际意义，只是告诉搜索引擎
表格：
https://www.w3schools.com/html/html_tables.asp

https://www.runoob.com/tags/ref-byfunc.html 属性列表

class: 可以有多个，在一个引号内，用空格分开（所以有空格的是多个 class 名，不是 class 名包含空格）。id 只能有一个。

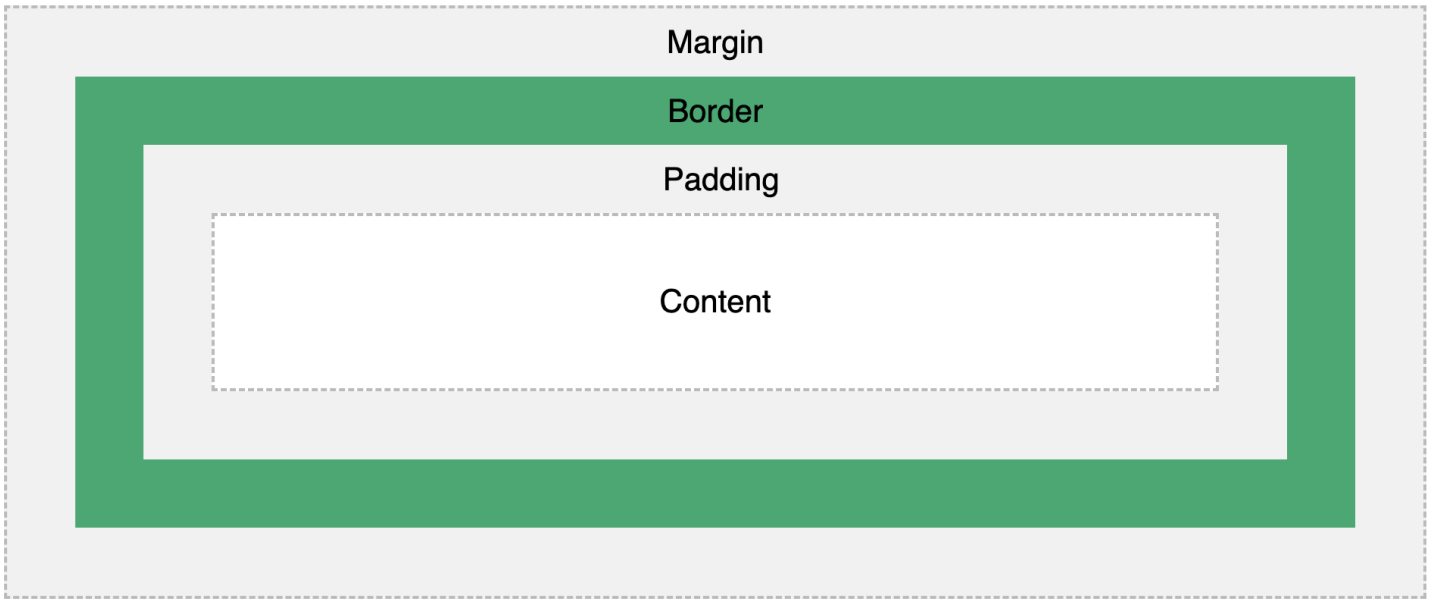
标签内定义 css，像这样， <body style="background-color:powderblue;">，或者在html里面定义 css，用 <style> </style>

▼ CSS 属性

太多了，没有合适的资源，并且根据大概名称 vs code 里都能找到，不知道的网上也容易查到，这里不做记录。只记录几个不方便记得。

line-weight: 字体粗细
text-transform: 首字母大写或全部大写
border-radius: 圆角效果
background-image 之类的要加上 url, background-image: url("../images/header.jpg");
text-decoration: underline

▼ Box Model



margin 计算：通常情况下，margin就是该元素占用空间的一部分，但是部分情况下，上下的 margin 可以合并。
元素实际尺寸不包含 margin，占用空间包含 maigin。
实际尺寸 content+2*border+2*padding
占用尺寸 content+2*border+2*padding+2*margin

margin 合并的情况：https://segmentfault.com/a/1190000013735912

给 1 个值：4 条边都是这个
给 2 个值：第 1 个上下，第 2 个是左右
给 4 个值：上 右 下 左
给 3 个值：a b c，上右下左 a b c b

box-sizing 属性: content-box (默认) 和 border-box
content-box 就是常规的那样, border-box 本身的 height 和 width 就包含了 padding 和 border, 不管 padding border 设置了多少, 都不会影响实际尺寸 (但是 margin 还是会影响, 但是计算尺寸本身就不算 margin)

https://www.w3schools.com/cssref/tryit.php?filename=trycss3_box-sizing

▼ CSS 选择器和继承

class 用 .xxx，id 用 #xxxx，html 标签名直接 xxxxx

空格 : 选择所有的子元素（无限层）
大于 > : 选择之后的第一代子元素
加号 + : 选择紧接在后面一个的（需要父元素相同）
波浪号 ~ : 选取 某个元素之后的所有相同元素（需要父元素相同）
星号 * : 选择所有的
[] : attribute selector, 比如 [href]，还可以运算判断相等之类的。
https://www.w3schools.com/css/css_attribute_selectors.asp
什么都没有（比如 a.b, a#c）: html 标签为 a 且 class 或 id 为 b 或 c
更多信息:
https://zhuanlan.zhihu.com/p/180011240

CSS 元素优先级：!important > inline > id > class (pseudo, attribute) > html_tag (pseudo element)
组合大于单个，相差一级相差 10 倍，选取和最大的，一样就选最后的

:nth-child(2): 选择第几个元素
:nth-last-child(2): 选择倒数第几个元素

:nth-child(2n): 选择是几的倍数的元素
所有的都从 1 开始，冒号前面必须有空格

伪类 Pseudo-classes
:hover : 鼠标悬停
:focus : 键盘 tab 上去
更多：
https://www.runoob.com/css/css-pseudo-classes.html
冒号前面最好有空格

Pseudo-elements：
两个冒号，没有空格。针对比如第一行、第一个字母、选中内容等。
更多：
https://www.w3schools.com/css/css_pseudo_elements.asp

去除属性：使用 属性名: unset

▼ CSS 几种 display (block, inline, grid, flex)， 排列方式

none: 完全不显示，不占用空间
block (默认): 占用一行，从上到下排列
inline: 在一行中显示，设置宽度和高度是无效的
inline-block: 在一行中显示，但是仍然可以像 block 一样设置宽度和高度（控制竖直方向对齐可以使用 vertical-align，top，center (默认)，bottom。
grid: 网格，可以设定每个元素的位置
flex: 弹性容器

float (注意，这不是 display，要用 float: left; 之类的)
可以使对象被包围，相当于 word 图片四周环绕
需要注意 float 的权限非常高，它可以破坏一个上级元素 (比如 main) 的完整形状

关于一些 block 下的说明
text-align只能让文字在它的对象里面居中 (比如 <p>)，无法让它的对象在上层对象里面居中，在 block 模式下这就是做不到的（不考虑手动设置 margin）（通过 margin-left: auto; margin-right: auto; 也可以实现）。如果要竖直居中，就一定只能 grid 或 flex，连 margin auto 都不行。
在没有 inline 的情况下，就是从上到下排列，全部靠左。
inline 如果宽度超出会换行，但本质上还是同一行，就相当于文本换行一样，如果屏幕足够宽就可以在同一行显示。

▼ display: grid

需要在上层容器设置 display: grid 或 inline-grid。

先要对父容器设置 grid-template-columns，接受多个参数，为每一列的属性，比如 grid-template-columns: 100px 200px 300px 或 20% 40% 40% 就是有 3 列，也可以用 fr，这样就是设置每列的比例，总和为 100%。如果设置了高度，也可以用 grid-template-rows。

针对父容器：
grid-column-gap: 20%, 可以设置列间的空隙
grid-auto-rows: 120px; 设置每行的高度（在column模式下）
可以通过 align-items 和 justify-items 来设置每个元素竖直和水平的居中，center 居中，self-start，self-end
grid-auto-flow: column; 先 从上到下 再从左到右 排列，需要确保设置了 grid-template-rows

对于每个 grid 元素：
align-self: 控制竖直方向，center 居中，self-start 靠上，self-end 靠下
justify-self: 控制水平方向，center 居中，self-start 靠左，self-end 靠右
grid-column / grid-row: 1 / 3; 或 1 / span 2;, 都表示从第 1 列开始，占用 2 列，类似 python 的半闭合，只不过从 1 开始。
也可以用 grid-area，grid-column: 1 / 3; grid-row: 2 / 4; <=> grid-area: 2 / 1 / 4 / 3;
order: 对元素排序，越小越靠前，默认值为 0，默认按照 html 里的顺序。比如，放到最前面，用 order: -1;

溢出时行为：
默认（什么都不设置）：超出屏幕宽度，可以左右滑动
如果想要自动换行，可以用 repeat(auto-fit, ...px/%), auto-fit 也可以用 auto-fill，基本没有区别。
如果是元素超出它格子的范围：
有可能影响布局和元素排列，会显示到外面去，有可能会覆盖其它内容。彻底禁止是做不到的（设置元素的 overflow 没用，grid 没法设置 overflow）。一定要在元素里设置 **overflow: hidden**。

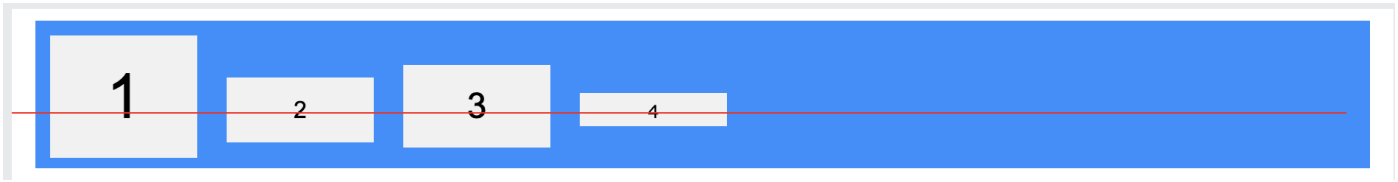
其它有用函数：
repeat: repeat(4, 200px) 不用手动输入 4 个 200px
minmax: minmax(200px, 1fr) 让浏览器自动判断，但是必须在这个范围内

▼ display: flex

可以在里面自由地水平或竖直排列，可以设置居中效果，不受外面影响。

父元素：

- **flex-direction**: 决定主轴的方向，即Flex项目的排列方向。常见值有`row`（默认值，水平方向）、`column`（垂直方向）、`row-reverse`、`column-reverse`。
- **flex-wrap**: 不设置的话默认 (nowrap) 全都挤在一行里，即使宽度超过（压缩元素宽度），设置了`wrap`就会换行。还有 wrap-reverse 可选。
- **justify-content (row 模式 水平)**: 在主轴上如何分布Flex项目 (如果 direction 为 row，就是左右)。例如`flex-start`（靠左）、`flex-end`（靠右）、`center`（居中）、`space-between`（两端对齐，项目之间的间隔相等添加空格）、`space-around`（每个项目两侧的间隔相等）。为 column 就是换个方向。**没有 stretch**。
- **align-items (row 模式 竖直)**: 在交叉轴上如何对齐Flex项目 (这里还是考虑为 row 模式)。例如`flex-start` **(靠上)**、`flex-end` **(靠下)**、`center`、`baseline`（项目的基线对齐）、`stretch`（默认值，仅针对没有设置 height 的情况，设置了无效，还是靠上）。为 column 就是换个方向。 Baseline 如下图：



- **align-content**: 多行Flex项目在交叉轴上的对齐方式，仅当有多行时生效。类似`align-items`，但适用于整个容器的多行。见下面说明。
- align-items 描述的是一个元素在那一行里面的行为，align-content 描述的是不同行之间的行为。只是说它本身的方向是竖直，工作模式类似于 justify-content，从水平变成竖直，可选名称也和 justify-content 一样。需要注意 align-items: stretch 只有在 align-items也是 stretch 且元素没有设置高度才有效。
- flex-flow: 没用，就是 同时设置 flex-direction 和 flex-wrap, 比如 flex-flow: row wrap;

每个 flex 元素：

- **flex-grow**: 定义Flex项目的放大比例，默认值为0。只要没有把一行占满就有效，优先级高于 width，按照各个元素设置的值按比例分配剩余空间。如果换行了，只要还有剩余空间，也有效。
- **flex-shrink**: 定义Flex项目的缩小比例，默认值为1。在空间不足时，此属性决定项目的缩小程度。只有在空间不足的情况下才有效（即不 wrap 且按照原本排列会超出）
- **flex-basis**: 代替 width，且权限比 width 更高，不管有没有 grow shrink 表现都一样，就是对 width 的替代和 overload。
- **align-self**: 自己的竖直对齐方式。相当于 align-items 只对自己作用，可取值一样。注意 flex 里面没有 justify-self。
- **flex**: 没用，就是`flex-grow`、`flex-shrink`和`flex-basis`的简写，默认值为`0 1 auto`。你可以使用简写来设置这三个属性。

▼ @media

@media screen {} : 针对屏幕，还有 print，speech
@media (max-width: 600px) : 根据宽度判断
@media screen and (min-width: 600px) and (max-width: 1200px) : 多条件组合，or 用逗号代替。不能部分地用 not，not 只能放在整个最前面，表示整个的 not。
@media (prefers-color-scheme: dark) : 深色模式
prefers-reduced-motion: reduce ：减少动画

其它内容

html 标签列表：https://www.runoob.com/tags/ref-byfunc.html

Position & Overflow

Position:

- `static` 默认
- `relative` 相对于 原本（static）的位置
- `fixed` 在页面上固定，即使滑动也不变
- `absolute` 相对于上层设定了 position 的容器
- `sticky` 滚动时不会超出页面

Z-index: 越大越上层
只对 positioned elements 和 flex items 有效
普通元素为 0，相同就按 html 里定义的顺序

Overflow：
visible (默认): 在超出范围内可见
hidden: 隐藏
scroll: 出现滑动框（没有超出也有）
auto: 只有超出了才会有滑动框
可以设置 overflow-x 和 overflow-y 来区分水平和垂直方向

Accessibility POUR Principle

Perceivable
Operable
Understandable
Robust

“Responsive” options

- **Responsive Web Design (RWD)** – fluid measurements, flexible grids, and varying CSS rules
- **Adaptive Design (dynamic serving)** – returns one of multiple versions of a page based on the type of device
- **Separate Mobile Site (.m)**a separate page URL for the mobile site

文字上下居中：做不到，只能外面再包裹一层容器

css 使用 **column-count 或 column-width** 可以分列，不需要 grid 或 flex，默认先从上到下再从左到右

em 和 rem

em 相对于最近的父元素，1 em 就是和父元素大小相同。rem 是相对入最外层的父元素（root em）

html 中特殊符号替换

If you want....	Then use...
<	&lt;
>	&gt;

©	©
blank space	
¢	¢
&	&

▼ 特殊文本形式

<b> - Bold text
 - Important text
<i> - Italic text
 - Emphasized text
<mark> - Marked text
<small> - Smaller text
<del> - Deleted text
<ins> - Inserted text
<sub> - Subscript text
<sup> - Superscript text

▼ <meta name="viewport" content="width=device-width, initial-scale=1.0">

这段 HTML 代码中的 `<meta name="viewport" content="width=device-width, initial-scale=1.0">` 是用来优化网页在移动设备上的显示效果的。

具体来说，这个 `meta` 标签有两个主要作用：

1. **控制视口的宽度**：`width=device-width` 这部分指示浏览器将页面的宽度设置为跟设备的屏幕宽度一样。这意味着网页的布局会根据不同设备的屏幕宽度进行调整，从而在各种尺寸的屏幕上都能保持较好的显示效果。
2. **控制初始缩放比例**：`initial-scale=1.0` 这部分设定了页面初始加载时的缩放级别。值为1.0意味着网页将以100%的缩放比例显示，也就是说，网页的显示大小将与其原始大小一致，不会进行任何缩放。这有助于确保用户在打开网页时看到的内容大小是设计者预期的大小，提升用户体验。

综上，这个 `meta` 标签主要是为了改善移动设备上的网页浏览体验，通过自动调整页面布局和缩放比例，使得网页能够在不同尺寸的屏幕上都能呈现出良好的可读性和操作性。