CSS Basic

CSS?

```
CSS 代表层叠样式表（Cascading Style Sheets）
```

关系选择器

```
后代选择器——典型用单个空格（" "）字符——组合两个选择器
.box p {
  color: red;
}

子代关系选择器是个大于号（>），只会在选择器选中直接子元素的时候匹配。继承关系上更远的后代则不会匹配。
ul > li {
  border-top: 5px solid red;
}

邻接兄弟选择器（+）用来选中恰好处于另一个在继承关系上同级的元素旁边的物件。
h1 + p {
  font-weight: bold;
  background-color: #333;
  color: #fff;
  padding: .5em;
}

通用兄弟(~)如果你想选中一个元素的兄弟元素，即使它们不直接相邻，你还是可以使用通用兄弟关系选择器（~）
h1 ~ p {
  font-weight: bold;
  background-color: #333;
  color: #fff;
  padding: .5em;
}
```

优先级

```
一个元素选择器不是很具体，则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。
一个类选择器稍微具体点，则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。

下面列表中，选择器类型的优先级是递增的：

1. 类型选择器（例如，h1）和伪元素（例如，::before）
```

```
2. 类选择器（例如，.example），属性选择器（例如，[type="radio"]）和伪类（例
如，:hover）
3. ID 选择器（例如，#example）
```

CSS 盒模型

```
整体上适用于区块盒子，它定义了盒子的不同部分（外边距、边框、内边距和内容）如何协同工作，
以创建一个在页面上可以看到的盒子。行内盒子使用的只是盒模型中定义的部分行为。

content => padding => border => margin
```

CSS 的值与单位

```
相对长度单位是相对于其他某些东西的。例如：

em 和 rem 分别相对于父元素和根元素的字体大小。
vh 和 vw 分别相对于视口的高度和宽度。

百分比:
如果将元素的字体大小设置为百分比，那么它将是元素父元素字体大小的百分比。
如果使用百分比作为宽度值，那么它将是父值宽度的百分比。

函数:
颜色部分函数的作用——rgb()、hsl() 等。
用于从文件返回图像的值——url()——也是一个函数。
calc():
.box {
  width: calc(20% + 100px);
}
```

: vs ::

```
Pseudo-classes (:) Pseudo-classes are keywords added to selectors that
specify a special state of the selected elements.
  :hover: Styles an element when the user hovers over it.
  :first-child: Selects an element that is the first child of its parent.
  :nth-child(n): Selects elements based on their position in a group of
siblings

Pseudo-elements (::) Pseudo-elements are used to style specific parts of an
element's content.
  ::before: Inserts content before the content of an element.
  ::after: Inserts content after the content of an element.
  ::first-line: Styles the first line of an element's content.
  ::first-letter: Styles the first letter of an element's content.
```

CSS 中哪些属性可以继承？

每一个属性在定义中都给出了这个属性是否具有继承性，一个具有继承性的属性会在没有指定值的时候，
会使用父元素的同属性的值来作为自己的值。

一般具有继承性的属性有，字体相关的属性，font-size和font-weight等。
文本相关的属性，color和text-align等。
表格的一些布局属性、列表属性如list-style等。还有光标属性cursor、元素可见性visibility。

当一个属性不是继承属性的时候，我们也可以通过将它的值设置为inherit来使它从父元素那获取同名的属性值来继承。

CSS Layout

正常布局流

正常布局流（normal flow）是指在不对页面进行任何布局控制时，浏览器默认的 HTML 布局方式。
当你使用 css 创建一个布局时，你正在离开正常布局流，但是对于页面上的多数元素，
正常布局流将完全可以创建你所需要的布局。从一个结构良好的 Html 文档开始是非常重要，
因为你可以按照默认的方式来搭建页面，而不是自造车轮。

display: inline ; display: block; display: inline-block

```
display: block
特点：
  元素独占一行，前后会自动换行。
  可以设置宽度（width）和高度（height）。
  支持所有方向的内边距（padding）和外边距（margin）。
  宽度默认填满其父容器的可用空间。
常见元素：
  <div>、<p>、<h1> 至 <h6>、<ul>、<ol>、<li> 等。

display: inline
特点：
  元素不会独占一行，多个行内元素会在同一行内排列，直到一行排满才会换行。
  无法设置宽度和高度，元素的尺寸由其内容决定。
  仅支持水平方向的内边距和外边距，垂直方向的内外边距对布局无影响。
  宽度随内容变化，不会自动填满父容器。
常见元素：
  <span>、<a>、<strong>、<em>、<img> 等。

display: inline-block
特点：
```

```
  元素在同一行内排列，不会独占一行。
  可以设置宽度和高度，且设置有效。
  支持所有方向的内边距和外边距。
  宽度由内容或设置的 width 决定，不会自动填满父容器。
常见用途：
  用于需要设置尺寸且希望多个元素在同一行排列的场景，如水平导航菜单中的按钮。
```

display: flex (Flexbox)

```
display: flex 是 CSS3 引入的一种布局方式，称为弹性盒布局（Flexbox），
用于简化复杂布局的实现。通过将容器的 display 属性设置为 flex，
其子元素（称为“项目”）的排列和对齐方式变得更加灵活和直观。

主要概念：
  容器（Container）：设置了 display: flex 的元素，称为弹性容器。
  项目（Item）：容器内的直接子元素，称为弹性项目。
  主轴（Main Axis）：定义项目排列的主方向，默认水平从左到右。
  交叉轴（Cross Axis）：垂直于主轴的方向。

常用属性：
  容器属性：
    flex-direction：定义主轴方向，如 row（默认，水平）、column（垂直）等。
    flex-wrap：定义项目是否换行，如 nowrap（默认，不换行）、wrap（换行）等。
    justify-content：定义项目在主轴上的对齐方式，如 flex-start（默认，起点对齐）、
center（居中）等。
    align-items：定义项目在交叉轴上的对齐方式，如 stretch（默认，拉伸占满）、
center（居中）等。
    align-content：定义多根轴线的对齐方式，适用于多行布局。
  项目属性：
    order：定义项目的排列顺序，数值越小，排列越靠前。
    flex-grow：定义项目的放大比例，默认为 0，即不放大。
    flex-shrink：定义项目的缩小比例，默认为 1，即空间不足时按比例缩小。
    flex-basis：定义项目在分配多余空间之前的初始大小。
    flex：flex-grow、flex-shrink 和 flex-basis 的简写形式。=>使用数字，用来控制比
例 (flex: 2;)
    align-self：允许单个项目有与其他项目不同的对齐方式，可覆盖 align-items 属性
```

display: grid (Grid)

```
display: grid 是 CSS 提供的一种强大的布局方式，称为网格布局（Grid Layout）。
它允许开发者将页面划分为行和列的网格，精确地控制元素在页面上的位置和排列方式。

主要特点：
  二维布局： 与 display: flex 的一维布局不同，网格布局同时处理行和列，适用于复杂的页面
布局。
  简洁的代码： 通过简洁的 CSS 代码，实现复杂的布局效果，减少对浮动（float）和定位
（position）的依赖。
  灵活的网格定义： 可以使用 grid-template-rows 和 grid-template-columns 定义行高
```

```
和列宽，使用 grid-template-areas 定义布局区域。

常用属性：
  容器属性：
    display: grid; 定义网格容器。
    grid-template-columns 定义列的数量和宽度。
    grid-template-rows 定义行的数量和高度。
    gap 设置行间距和列间距。
    grid-template-areas 定义命名的网格区域，便于布局。
  项目属性：
    grid-column 指定项目在网格中的列位置。 grid-column: 1 / 3; => 从第一个到第三个
    grid-row 指定项目在网格中的行位置。
    grid-area 指定项目所在的网格区域。
        example:
          grid-template-areas:
                "a a ."
                "a a ."
                ". b c";

与弹性盒子不同的是，在定义网格后，网页并不会马上发生变化。
因为 display: grid 的声明只创建了一个只有一列的网格，所以子项还是会像正常布局流那样，
自上而下、一个接一个的排布。

.container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}

除了长度和百分比，我们也可以用 `fr` 这个单位来灵活地定义网格的行与列的大小。
这个单位代表网格容器中可用空间的一份
.container {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}

.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
}
```

float

```
在 CSS 中，float 属性用于指定元素应沿其容器的左侧或右侧浮动，从而使后续的文本和内联元素
环绕在其周围。
最初，float 属性主要用于实现文字环绕图片的效果，但随着前端技术的发展，它也被广泛应用于布
局设计中

float 属性的取值：
  left：元素向左浮动。
  right：元素向右浮动。
```

```
  none：默认值，元素不浮动，按照正常文档流排列。
  inherit：元素继承其父元素的 float 属性值。

注意事项：
  脱离文档流： 浮动元素会脱离正常的文档流，可能导致父容器高度塌陷。
  为了解决此问题，可以对父容器使用清除浮动的方法，如设置 overflow: hidden; 或使用伪元素清除浮动。

  布局影响： 浮动元素会影响后续元素的排列方式，需要谨慎使用，尤其是在复杂布局中。

.box {
  float: left;
}
```

position

```
定位 (positioning) 能够让我们把一个元素从它原本在正常布局流 (normal flow) 中应该在的位置移动到另一个位置。

position 属性的常见取值：
  static：默认值，元素按照正常的文档流进行排列，不受 top、right、bottom、left 和 z-index 属性的影响。

  relative：相对定位，元素相对于其在正常文档流中的位置进行偏移，
    但仍占据原始空间。可使用 top、right、bottom、left 属性进行偏移。

  absolute：绝对定位，元素脱离正常文档流，相对于最近的已定位祖先元素（非 static）进行定位。
  如果没有已定位的祖先元素，则相对于初始包含块（通常是视口）定位。可使用 top、right、bottom、left 属性指定位置。

  fixed：固定定位，元素脱离正常文档流，相对于视口进行定位，即使页面滚动，元素也保持在指定位置。
  可使用 top、right、bottom、left 属性指定位置。

  sticky：粘性定位，元素在跨越特定阈值前为相对定位，之后为固定定位。
  即元素在滚动到某个位置时会“粘”在视口的指定位置。需配合 top、right、bottom、left 属性使用。
```

Multiple-column layout

```
The CSS Multi-column Layout module allows content to be divided into multiple columns,
similar to newspaper layouts.

1. column-count: Specifies the number of columns into which the content should be divided.
  .container {
```

```
    column-count: 3;
  }

2. column-width: Defines the ideal width for each column.
The browser will determine the optimal number of columns based on the
available space and the specified column width.
  .container {
    column-width: 200px;
  }

3. column-gap: Sets the space between columns.
4. column-rule: Adds a vertical line (rule) between columns, enhancing
visual separation.
5. column-span: Determines how many columns an element should span.
```

Responsive design

```
Responsive web design (RWD) is a web design approach to make web pages
render well
on all screen sizes and resolutions while ensuring good usability.
It is the way to design for a multi-device web.

Key Components of RWD:
  1. Fluid Grids: Utilize relative units like percentages instead of fixed
units such
  as pixels to define layout dimensions.
  This flexibility enables elements to resize proportionally based on the
screen size.

  2. Flexible Images: Implement images that can scale within their
  containing elements, preventing overflow and maintaining design integrity
across devices.

  3. Media Queries: Apply CSS techniques that detect the user's device
characteristics,
  such as screen width, and adjust the layout accordingly.
  Media queries facilitate the creation of responsive designs by applying
different styles based on device capabilities.
```

Media queries

```
Media queries allow us to run a series of tests
(e.g. whether the user's screen is greater than a certain width, or a
certain resolution) and apply CSS selectively to style the page
appropriately for the user's needs.

@media screen and (min-width: 80rem) {
  .container {
```

```
    margin: 1em 2em;
  }
}
@media media-type and (media-feature-rule) {
  /* CSS rules go here */
}
```

如何实现元素的垂直居中

父元素固定宽高，子元素设置position: absolute，margin：auto 平均分配margin

弹性布局display: flex。设置align-items: center; justify-content: center

BFC (Block Formatting Context)

BFC 也就是常说的块格式化上下文，这是一个独立的渲染区域，规定了内部如何布局，
并且这个区域的子元素不会影响到外面的元素，其中比较重要的布局规则有内部box 垂
直放置，计算BFC 的高度的时候，浮动元素也参与计算，触发BFC 的规则有根元素，
浮动元素，position 为absolute 或fixed 的元素，display 为inline-block，table-
cell，
table-caption，flex，inline-flex，overflow 不为visible 的元素

SASS

Sass is the most mature, stable, and powerful professional grade CSS
extension language in the world.

CSS 多列等高如何实现？

利用flex布局中弹性元素align-items属性默认为stretch，如果弹性元素未设置高度或设为
auto，
将占满整个容器的高度的特性，来实现多列等高。

width:auto 和 width:100%的区别

width:100%会使元素box的宽度等于父元素的content box的宽度。

width:auto会使元素撑满整个父元素，margin、border、padding、content区域会自动分配水
平空间。

display、position和float的相互关系？

总的来说，可以把它看作是一个类似优先级的机
制，"position:absolute"和"position:fixed"优先级最高，有它存在
的时候，浮动不起作用，'display'的值也需要调整；其次，元素的'float'特性的值不
是"none"的时候或者它是根元素
的时候，调整'display'的值；最后，非根元素，并且非浮动元素，并且非绝对定位的元
素，'display'特性值同设置值。

常见的元素隐藏方式？

-（1）使用 display:none;隐藏元素，渲染树不会包含该渲染对象，因此该元素不会在页面中占据位置，也不会响应绑定的监听事件。

-（2）使用 visibility:hidden;隐藏元素。元素在页面中仍占据空间，但是不会响应绑定的监听事件。

-（3）使用 opacity:0;将元素的不透明度设置为 0，以此来实现元素的隐藏。元素在页面中仍然占据空间，并且能够响应元素绑定的监听事件。

-（4）通过使用绝对定位将元素移除可视区域内，以此来实现元素的隐藏。

-（5）通过 z-index 负值，来使其他元素遮盖住该元素，以此来实现隐藏。

-（6）通过 clip/clip-path 元素裁剪的方法来实现元素的隐藏，这种方法下，元素仍在页面中占据位置，但是不会响应绑定的监听事件。

-（7）通过 transform:scale(0,0)来将元素缩放为 0，以此来实现元素的隐藏。这种方法下，元素仍在页面中占据位置，但是不会响应绑定的监听事件。