

# 7.CSS

## 1. CSS

### CSS

Python

```
<p>Some Text</p>
```

```
p { color: red }
```

Some Text

### CSS

<p>Some Text</p>

+

p { color: red }

=

Some Text

syntax

## selectors

```
p      <p>
.important      <div class="important">
#title      <div id="title">
```

```
p.important      <p class="important">
h1#title      <h1 id="title">
```

当然，这些是 CSS 选择器，我来用中文解释一下：- `p`：这是一个类型选择器，它会选择 HTML 文档中的所有 `<p>` 元素。- `.important`：这是一个类选择器，它会选择所有带有 `class="important"` 的元素。

## selectors

Python

```
.container p
.container > p
.container ~ p
.container + p

p, div

<div class="container">
<p>direct child</p>
```

```
<div>

<p>descendant</p>
</div>

</div>

<p>para  one</p>
<p>para  two</p>
```

以下是对给定选择器的解释：1. `.container p`：这个选择器选择了类名为 "container" 的元素内部的所有段落 (p) 元素。它将选择任何嵌套在类名为 "container" 的元素内部的段落元素。`<p>direct child</p><p>descendant</p>`2. `.container > p`：这个选择器选择了类名为 "container" 的元素直接子级中的所有段落 (p) 元素。它将选择任何作为类名为 "container" 元素的直接子级的段落元素。`<p>direct child</p>`3. `.container ~ p`：这个选择器选择了紧跟在类名为 "container" 的元素之后的所有段落 (p) 元素。它将选择紧跟在类名为 "container" 的元素之后的任何段落元素，即它们具有相同的父级并且处于同一级别。`<p>para one</p><p>para two</p>`4. `.container + p`：这个选择器选择了紧跟在类名为 "container" 的元素之后的下一个段落 (p) 元素。它将选择紧跟在类名为 "container" 的元素之后的下一个段落元素，即它们具有相同的父级并且处于同一级别。`<p>para one</p>`5. `p, div`：这个选择器选择了所有段落 (p) 元素和所有 div 元素。它将选择 HTML 文档中所有的段落元素和所有的 div 元素。

## Values

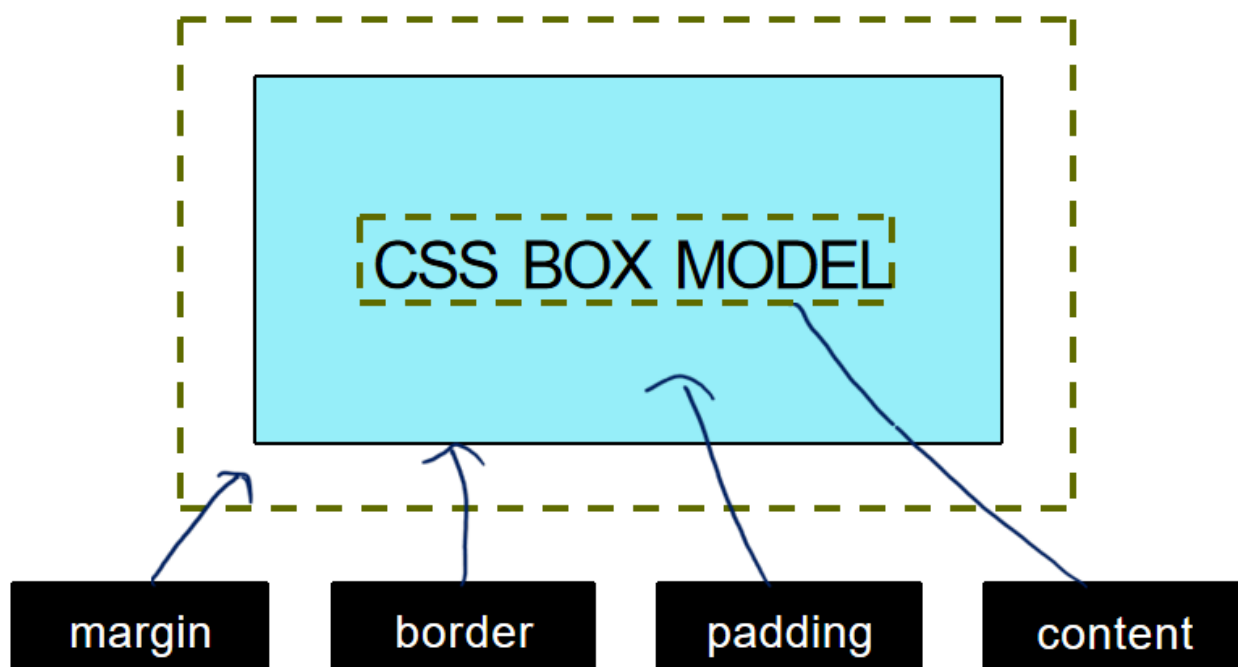
Python

```
color: red;
background-color: #ff0000;

border-color: rgba(255, 0, 0, 1.0);
```

## Box Model

# Box model



当然，CSS 的盒模型（Box Model）是一个在网页设计中非常重要的概念。它描述了元素在网页布局中所占的空间方式。

盒模型主要包括四个部分，从内到外依次是：

内容（Content）：这是元素的实际内容，如文本、图片等，通过 `width` 和 `height` 属性可以设置其大小。

内边距（Padding）：内边距是内容周围的空白区域，它清晰地隔开了元素的内容和边框。

边框（Border）：边框是围绕在内边距和内容外的线。边框的大小和样式可以通过 CSS 进行设置。

外边距（Margin）：外边距是边框外的空白区域，用于隔开不同元素之间的空间。

这四个部分一起构成了一个元素的完整视觉呈现，也决定了元素在网页布局中所占据的空间大小。理解盒模型对于掌握 CSS 布局非常重要。希望这个解释对你有所帮助！

## Box model

F12

## Admissions

If you are a prospective student or have an admissions query:

### Undergraduate

Email: [choosebristol-ug@bristol.ac.uk](mailto:choosebristol-ug@bristol.ac.uk) Tel: [+44 \(0\)117 394 1649](tel:+44(0)1173941649)

### Postgraduate (taught)

Email: [choosebristol-pg@bristol.ac.uk](mailto:choosebristol-pg@bristol.ac.uk) Tel: [+44 \(0\)117 394 1649](tel:+44(0)1173941649)

### Postgraduate (research)

Email: [sceem-pgr-admissions@bristol.ac.uk](mailto:sceem-pgr-admissions@bristol.ac.uk) Tel: [+44 \(0\)117 331 4753](tel:+44(0)1173314753) or [+44 \(0\)117 331 5232](tel:+44(0)1173315232)

## Current students

Contact the faculty [for letters \(banks, student status, council tax\), transcripts and other documents](#). Please also see the faculty's [current student pages](#) for further information and the [taught programme handbook](#).

Elements Network Sources

<https://www.bristol.ac.uk/engineering/departments/computerscience/contact/>

## Admissions

If you are a prospective student or have an admissions query:

### Undergraduate

Email: [choosebristol-ug@bristol.ac.uk](mailto:choosebristol-ug@bristol.ac.uk) Tel: [+44 \(0\)117 394 1649](tel:+44(0)1173941649)

### Postgraduate (taught)

Email: [choosebristol-pg@bristol.ac.uk](mailto:choosebristol-pg@bristol.ac.uk) Tel: [+44 \(0\)117 394 1649](tel:+44(0)1173941649)

### Postgraduate (research)

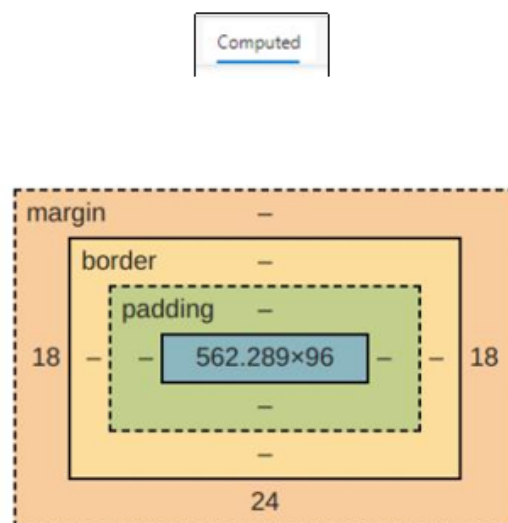
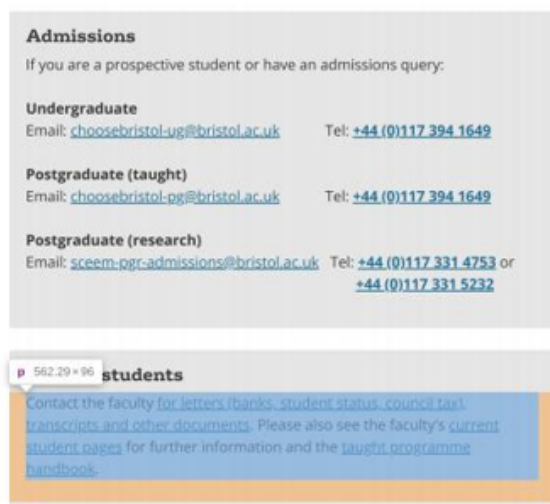
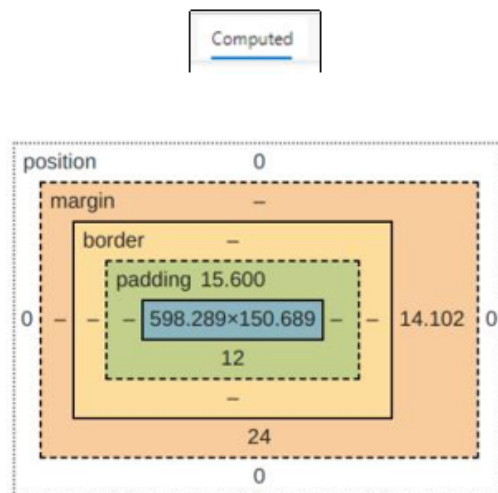
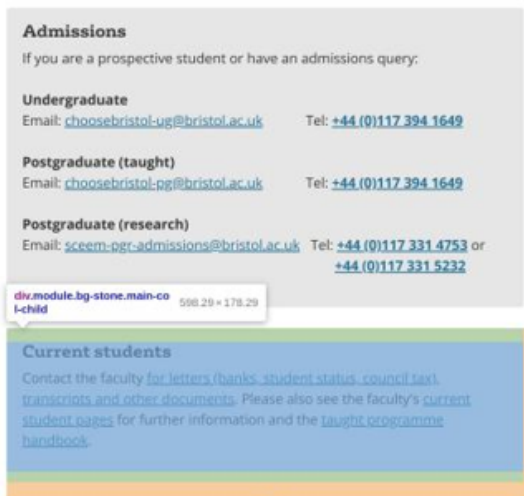
Email: [sceem-pgr-admissions@bristol.ac.uk](mailto:sceem-pgr-admissions@bristol.ac.uk) Tel: [+44 \(0\)117 331 4753](tel:+44(0)1173314753) or [+44 \(0\)117 331 5232](tel:+44(0)1173315232)

## Current students

Contact the faculty [for letters \(banks, student status, council tax\), transcripts and other documents](#). Please also see the faculty's [current student pages](#) for further information and the [taught programme handbook](#).

Elements

```
<div class="module bg-stone main-col-child ">_</div>
<div class="module bg-stone main-col-child ">
  <h2 class="module-heading">
    <span class="mh-text">Current students</span>
  </h2>
  <p>
    "Contact the faculty&nbsp;&nbsp;&nbsp;"
    <a href="/engineering/current-students/transcripts-and-documents/">_</a>
    ". Please also see the faculty's&nbsp;&nbsp;&nbsp;"
    <a href="/engineering/current-students/">current student pages</a>
    "&nbsp;&nbsp;&nbsp;for further information and the&nbsp;&nbsp;&nbsp;"
    <a href="/engineering/current-students/taught-programme-handbook/">taught
    programme handbook</a>
    ".
  </p>
</div>
```



## Box model

```
margin-top: 10px;
padding-left: 20px;
margin: 10px 5px 15px 20px;
border: 1px solid black;
```

Python

`margin-top: 10px;`: 这个属性设置了元素的上外边距（Margin）为 10 像素。外边距是元素边框外的空白区域，用于隔开不同元素之间的空间。

`padding-left: 20px;`: 这个属性设置了元素的左内边距 (Padding) 为 20 像素。内边距是内容周围的空白区域, 它清晰地隔开了元素的内容和边框。

`margin: 10px 5px 15px 20px;`: 这个属性是一个简写属性, 用于同时设置元素的上、右、下、左外边距。四个值分别对应上、右、下、左的外边距, 所以这个属性设置了上外边距为 10 像素, 右外边距为 5 像素, 下外边距为 15 像素, 左外边距为 20 像素。

`border: 1px solid black;`: 这个属性是一个简写属性, 用于同时设置元素的边框宽度、样式和颜色。所以这个属性设置了元素的边框宽度为 1 像素, 样式为实线, 颜色为黑色。

## 2.CSS 2

### Units of measurement

#### dpi (ppi)

dots (points) per inch

PC screen: 72 / 96 (Mac / Win defaults) can be 120+ nowadays

mobile: 200+ (iPhone 12 Pro: 460)

DPI (每英寸像素数, 即 PPI) 是指在一英寸长度内, 显示设备上的像素数量。在 PC 屏幕上, 通常使用的是 72 到 96 DPI (Mac 和 Windows 的默认值), 而现在一些高分辨率的显示器可以达到 120 以上。在移动设备上, 通常使用的是 200 以上的 DPI, 一些高端手机如 iPhone 12 Pro 甚至可以达到 460 DPI。

在图形设计中, 了解目标设备的 DPI 是很重要的, 因为它会影响到设计的清晰度和质量。

#### Units

```
px      pixel
pt      point cm, in, ...
em      width of letter m
ex      height of letter x
lh      line height (+ space)
```

在图形设计中，通常使用以下单位来表示尺寸：

- **px（像素）**：像素是图像中最小的单元，它是数字图像的基本构建块。在 Web 设计和数字图形中，像素是最常用的单位，表示屏幕上的点。
- **pt（点）**：点是印刷行业常用的单位，特别是在排版和印刷设计中。1 点等于 1/72 英寸，它与像素之间的转换通常取决于输出设备的 DPI。
- **cm（厘米）、in（英寸）等**：这些单位通常用于打印和印刷设计中，用来表示物理尺寸。在设计中，你可以使用这些单位来指定元素在纸张或其他介质上的尺寸。在设计过程中，你可以根据需要选择合适的单位来确保设计在不同设备上的呈现和打印效果符合预期。在排版和网页设计中，以下单位用于定义字体相关的尺寸和间距：

- **em**：em 是相对长度单位，它的值是相对于当前元素的字体尺寸。例如，如果一个元素的字体大小为 16 像素，1em 将等于 16 像素。
- **ex**：ex 也是相对长度单位，它的值是相对于小写字母 x 的高度。一般来说，ex 约等于字体大小的一半。
- **lh**：lh 是一种相对单位，代表行高（line height），也称为行间距。它定义了文本行的基线到基线之间的距离，包括文本的高度和行间距。通常，lh 的值是字体大小的倍数，例如 1.2 表示字体大小的 120%。这些单位在设计中用于定义字体大小、间距和行高，可以帮助设计师更精确地控制文本的呈现和排版。

## More units

```
vw, vh      1% of viewport width/height
rem      root em (html)
```

Python

```
%      parent width/height
```



在网页设计中，还有一些其他常用的单位，用于相对于视口大小或父元素大小来定义尺寸：

- **vw**：vw 代表视口宽度的百分比，1vw 等于视口宽度的 1%。例如，如果视口宽度为 1000 像素，则 1vw 等于 10 像素。
- **vh**：vh 代表视口高度的百分比，1vh 等于视口高度的 1%。例如，如果视口高度为 800 像素，则 1vh 等于 8 像素。
- **rem**：rem 代表根元素（html 元素）的字体大小。这是相对于文档的根元素而言的，可以帮助在整个文档中保持一致的尺寸。例如，如果根元素的字体大小为 16 像素，则 1rem 等于 16 像素。
- **%**：百分比单位用于相对于父元素的宽度或高度来定义尺寸。例如，如果父元素的宽度为 200 像素，子元素的宽度设置为 50%，则子元素的宽度将为 100 像素。

这些单位可以帮助设计师根据视口大小或父元素大小来动态调整元素的尺寸，从而实现响应式设计和布局。

## example

```
html { font-size: 12pt; }
p      { font-size: 1rem; }
h1 {
  font-size: 1.8rem;
  margin-top: 2ex;
  margin-bottom: 1ex;
}

h2      { font-size: 1.4rem; }
```

Python

这个例子中，我们首先设置了根元素（html）的字体大小为 12pt。然后，我们定义了段落（p）、标题 1（h1）、标题 2（h2）的字体大小和间距：

- **p**：段落的字体大小设置为 1rem，这意味着它将等于根元素（html）的字体大小，也就是 12pt。
- **h1**：标题 1 的字体大小设置为 1.8rem，即根元素字体大小的 1.8 倍，因此它将等于 1.8 乘以 12pt，即 21.6pt。此外，标题 1 的上边距设置为 2ex，表示相对于小写字母 x 的高度的两倍。下边距设置为 1ex，表示相对于小写字母 x 的高度的一倍。
- **h2**：标题 2 的字体大小设置为 1.4rem，即根元素字体大小的 1.4 倍，因此它将等于 1.4 乘以

12pt，即 16.8pt。这样的设置可以让文档的字体大小和间距保持一致，并根据根元素的字体大小来动态调整。

## example

```
html, body {  
  margin: 0;  
  padding: 0;  
}  
h1 {  
  background-color: yellow;  
  width: 100%;  
}
```

Python

这个例子中，我们设置了 HTML 和 body 元素的外边距和内边距都为 0，以确保页面没有默认的边距和填充。然后，我们对标题 1 (h1) 元素进行了样式设置：– 背景颜色被设置为黄色 (yellow)，这将导致标题 1 元素的背景颜色变为黄色。– 宽度被设置为 100%。这意味着标题 1 元素的宽度将填充其父元素的整个宽度，因此标题 1 将占据一行的全部宽度。这样的设置可以帮助设计师控制页面的外观和布局，确保元素的样式在不同浏览器和设备上呈现一致。

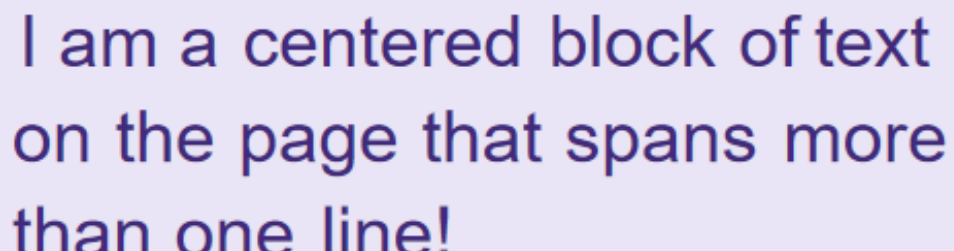
## 3.CSS grids

### History of Web Layout

这些都是用于网页布局的技术或方法：– **Tables（表格）**：传统的 HTML 表格布局方式，通过<table>、<tr>和<td>等标签来创建表格结构。虽然在某些情况下仍然有用，但通常不推荐作为主要的布局工具，因为它们的语义性更适合用于显示数据而不是页面布局。– **Float（浮动）**：通过将元素浮动到容器的左侧或右侧，使得其他元素可以环绕在其周围，从而实现页面布局。尽管浮动被广泛使用，但它会导致一些布局问题，例如清除浮动、父元素塌陷等，因此现代布局技术更多地向 Flexbox 和 Grid 发展。– **Flexbox**：Flexbox 是一种用于页面布局的 CSS3 模块，旨在提供更加灵活的布局方式。通过在父容器上应用 flex 布局，可以轻松地控制子元素的排列顺

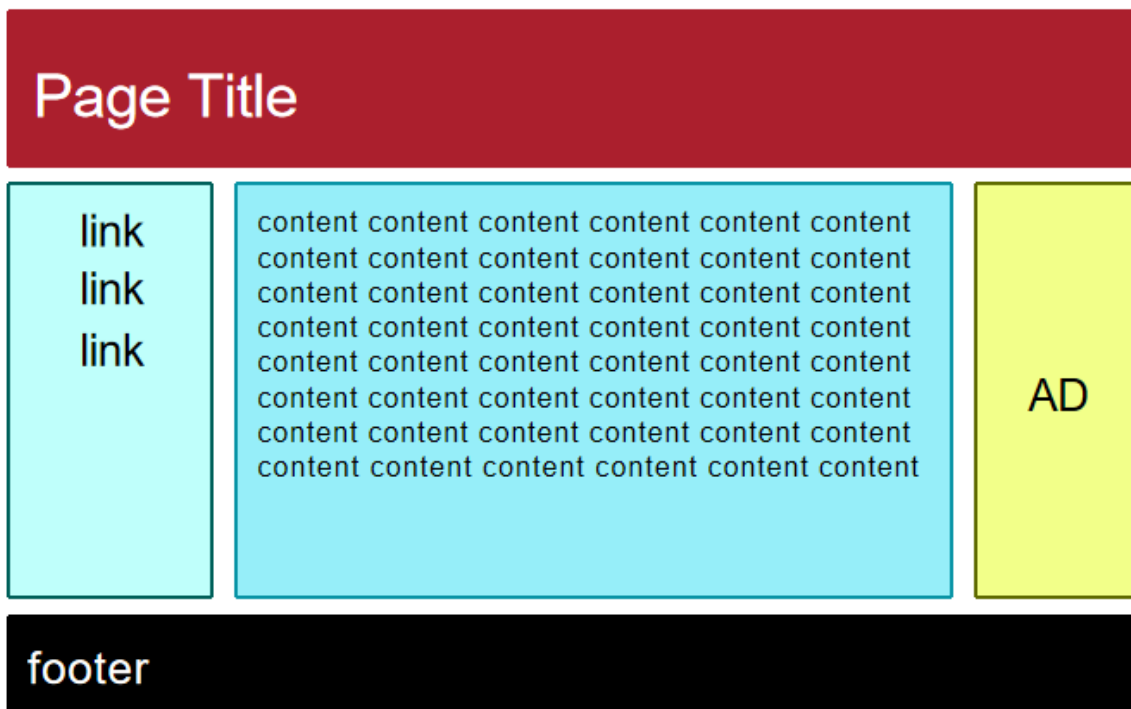
序、对齐方式、间距等属性，从而实现响应式和动态布局。– **Grid（网格）**：CSS Grid Layout 是一种二维网格布局系统，允许将页面划分为行和列，并在这些行和列上放置元素。Grid 布局提供了对元素之间空间和位置的更细粒度的控制，使得设计师可以更轻松地创建复杂的网格布局。这些技术都有各自的优点和用途，具体选择取决于设计需求、浏览器支持和开发者的偏好。

## Centering



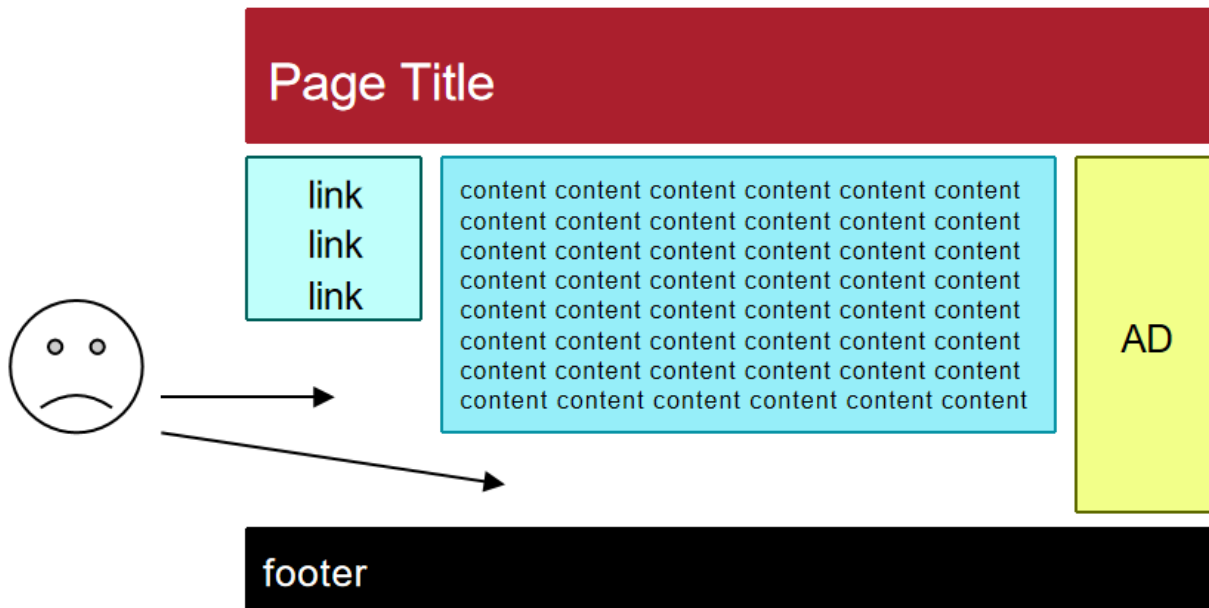
I am a centered block of text  
on the page that spans more  
than one line!

## The Holy Grail Layout 圣杯布局



圣杯布局是一种常用的网页布局技术，旨在实现具有三个主要区域的页面布局：一个固定宽度的中间内容区域（称为“主体”），以及两个侧边栏（左侧和右侧）。这种布局的目标是使中间内容区域优先加载以保持网页主要内容的重要性，并且允许侧边栏在视觉上排列在主内容区域的两侧。圣杯布局的关键是使用 CSS 中的浮动和定位属性来实现这种布局结构。典型的圣杯布局通常包括以下几个步骤：1. 将 HTML 分成三个主要部分：一个包含主体内容的 div，以及左侧和右侧侧边栏的 div。2. 使用 CSS 设置中间内容区域的宽度，并将其放置在页面的中心位置。3. 使用浮动属性将左右两个侧边栏浮动到主体内容区域的两侧。4. 使用负边距将左右侧边栏拉回到其相应的位置，使它们不会遮挡主体内容。5. 使用相对定位或左右内边距将内容推送到侧边栏之下，以确保整体布局的正确性。通过这些步骤，圣杯布局可以实现一个灵活且易于管理的网页布局，使得网页内容和导航元素都能得到合适的展示，并且在不同尺寸的设备上都能够呈现良好的效果。

# Not Holy



```
HTML
1 <div class="container">
2   <p>Child one</p>
3   <p>Child two</p>
4   <p>Child three</p>
5   <p>Child four</p>
6   <p>Child five</p>
7   <p>Child six</p>
8 </div>

CSS
1 p {
2   padding: 5px;
3   margin: 0;
4   border: 1px solid black;
5 }
6
```

Child one
Child two
Child three
Child four
Child five
Child six

## Grid layout

```

<div class= "container">

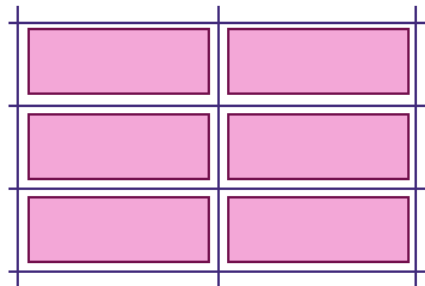
<p>Item 1</p>
<p>Item 2</p>
</div>

.container {
display: grid;
}

```

网格布局 `html<div class="container"> <p>Item 1</p> <p>Item 2</p></div>` `` `` ``css.container { display: grid;}` 这个示例中，`.container` 类被设置为 `grid` 布局，这意味着其中的元素将按照网格的方式进行排列。

## Grid



Grid Layout 是一种在网页设计中用于创建多行多列布局的强大 CSS 布局系统。它允许开发人员以类似于表格的方式对页面进行布局，但比传统的表格布局更灵活、更强大。使用 Grid Layout，您可以定义网格容器（grid container）和网格项目（grid item）。网格容器是包含网格布局的元素，而网格项目是网格布局中的子元素。您可以在网格容器上定义行和列，然后在网格项目上放置到相应的行和列中。下面是一个简单的示例，展示了如何使用 Grid Layout：

Python

```
HTML : html<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

Python

```
css.container {
display: grid;
grid-template-columns: 100px 100px 100px;
/* 定义三列，每列宽度为 100 像素 */
grid-gap: 10px; /* 定义网格间隔为 10 像素 */
}

.item {
background-color: lightblue;
padding: 20px;
text-align: center;
}
```

在这个例子中，`.container` 元素被设置为网格布局，并定义了三列，每列宽度为 100 像素。`.item` 元素是网格容器中的子元素，它们被放置到相应的网格中。

好的，让我们举一个简单的例子来说明这些属性的用法：假设我们有一个包含三个网格项目的网格布局，每个项目都被放置在不同的行和列上。

Python

```
<div class="container">
<div class="item">Item 1</div>
<div class="item">Item 2</div>
  <div class="item">Item 3</div>
</div>
```

CSS

```
.container {
display: grid;
grid-template-columns: repeat(3, 100px); /* 定义三列，每列宽度为 100px */
grid-template-rows: repeat(3, 100px); /* 定义三行，每行高度为 100px */
}
```

```
grid-gap: 10px; /* 定义网格间隔为 10px */`

.item {
background-color: lightblue;
padding: 20px;
text-align: center;
}

.item:nth-child(1) {
grid-row-start: 1;
grid-row-end: 3;
grid-column-start: 1;
grid-column-end: 3;
}

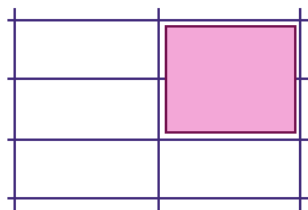
.item:nth-child(2) {
  grid-row: 1 / 3; /* 通过简化方式定义起始行和结束行 */
  grid-column: 3; /* 通过简化方式定义起始列和结束列 */
}

.item:nth-child(3) {
grid-area: 2 / 2 / 3 / 4; /* 通过 grid-area 属性一次性定义位置 */`
```

在这个例子中，我们有一个 `.container` 元素，其中包含三个 `.item` 子元素。每个项目都有不同的位置。– 第一个项目跨越了第一行和第二行，以及第一列和第二列。– 第二个项目从第一行到第二行，以及第三列。– 第三个项目占据了第二行和第三行之间的第二列到第三列。这就是如何使用 `grid-row-start`、`grid-row-end`、`grid-column-start`、`grid-column-end`、`grid-row`、`grid-column` 和 `grid-area` 属性来定义网格项目在网格布局中的位置。

## Rows and columns





CSS

Rows and columns

```
grid-row-start: 1;  
grid-row-end: 3;  
grid-column-start: 2;  
grid-column-end: 3;
```

```
grid-row: 1 / 3;  
grid-column: 2 / 3;
```

```
grid-area: 1 / 2 / 3 / 3;
```

这些属性用于定义网格项目在网格布局中的位置和尺寸：– `grid-row-start`：指定网格项目开始的行号。– `grid-row-end`：指定网格项目结束的行号。– `grid-column-start`：指定网格项目开始的列号。– `grid-column-end`：指定网格项目结束的列号。您还可以使用简化的方式同时定义起始行、结束行、起始列和结束列：– `grid-row`：通过指定起始行和结束行的编号来定义网格项目的行。– `grid-column`：通过指定起始列和结束列的编号来定义网格项目的列。最后，还可以使用 `grid-area` 属性一次性指定网格项目的四个位置：– `grid-area`：通过指定起始行、结束行、起始列和结束列的编号来定义网格项目的位置。这些属性的值可以是行号或列号，也可以是关键字，例如 `span`，表示跨越多个行或列。

## Holy Grail

```

<body>
<header>Site Logo . . .</header>
<nav>Links . . .</nav>

<main>Lots of content . . .</main>
<footer>Stuff . . .</footer>
</body>

```

"Holy Grail" 布局是一种经典的网页布局模式，通常由顶部的标头（header）、底部的页脚（footer）、左侧或右侧的导航（nav）以及位于中间的主要内容区域

（main）组成。这种布局旨在实现页面内容的合理排列，使得用户界面易于阅读和导航。在你提供的示例中，我们可以将各个部分映射到 Holy Grail 布局的不同部分：–

`<header>` 标签代表页面的顶部，通常用于显示网站的 logo、标题等内容。–

`<nav>` 标签代表导航栏，用于放置网站的链接和导航菜单。– `<main>` 标签代表主要内容区域，用于放置页面的核心内容。– `<footer>` 标签代表页面的底部，通常用于放置版权信息、联系方式等内容。这种布局可以使页面内容更加结构化和易于理解，同时也能提供良好的用户体验。

## Holy Grail

```

body>
  <header>Site Logo ...</header>
  <nav>Links ...</nav>
  <main>Lots of content ...</main>
  <footer>Stuff ...</footer>
</body>

```

```

body {
  display: grid;

```

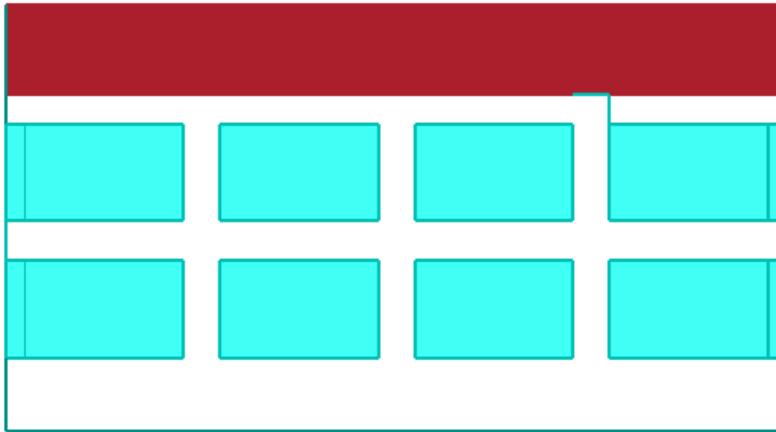
```
grid-template-columns: 200px 1fr;
}
header { grid-row: span 2; }
footer { grid-row: span 2; }
```

这个 CSS 代码定义了一个基本的网页布局，使用了 CSS Grid 布局模型。让我解释一下每一部分的含义：– `body { display: grid; grid-template-columns: 200px 1fr; }`：这段代码将 body 元素设置为网格容器，并定义了网格的列布局。具体来说，它指定了两列，第一列的宽度为 200px，第二列的宽度为剩余空间的一部分（1fr）。– `header { grid-row: span 2; }`：这行代码设置了 `<header>` 元素跨越两行。`grid-row: span 2;` 意味着 `<header>` 元素将占据网格布局中两个行的高度。– `footer { grid-row: span 2; }`：这行代码设置了 `<footer>` 元素跨越两行，效果与 `<header>` 类似，都是占据两行的高度。这个布局的效果是，在页面的左侧有一个固定宽度的列（200px），右侧是剩余空间的一部分，header 和 footer 跨越了两行，占据了一定的高度。

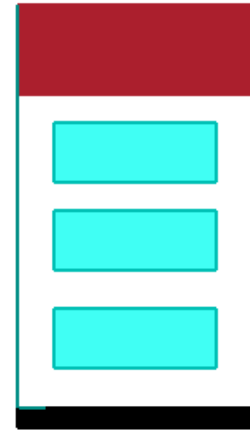
## 4.Responsive CSS

### Responsive

## Desktop



## Mobile



## General

Bad: `main { width: 800px; }`

OK: `main { max-width: 800px;  
margin: 0 auto; }`

## Media queries

```
@media ... {  
  h1 {  
    font-size: 20px;  
  }  
  
}  
  
@media (min-width: 600px)
```

CSS

```
and (max-width: 800px) { ... }
```

```
@media screen and ...
```

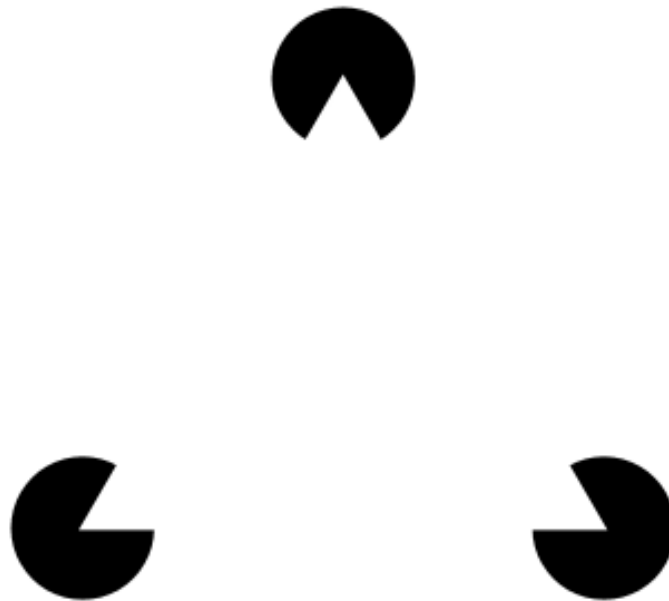
## 5.Web Design

### Design principle 1:

test with your intended audience!

### Some More Design Principles

#### Gestalt (1)



## Whitespace

gnats	gram	\$13.65
	each	.01
gnu	stuffed	92.50
emu		33.33
armadillo	frozen	8.99

Item		
Animal	Description	Price (\$)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Item		
Animal	Description	Price (\$)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

TeX manual

booktabs

## Text width

# Text width

## Guidelines you may see:

- 50–60 characters
- two full alphabets
- 12 words
- 30 em



文本宽度是指文本行的长度，通常由字符数、字母数、单词数或 em 单位来衡量。以下是一些关于文本宽度的常见指南：

- 50–60 个字符：**这是一种常见的建议，旨在确保文本行的长度适中，易于阅读且不会太长或太短。
- 两个完整字母表：**有时候将文本宽度定义为足够容纳两个完整的字母表，这确保了文本行的长度适中，同时避免了过长或过短的行。
- 12 个单词：**另一个常见的指导原则是文本行应该包含大约 12 个单词。这个数字可以确保文本行长度适中，易于阅读。
- 30 个 em 单位：**em 单位是相对于父元素字体大小的长度单位。将文本宽度定义为 30 个 em 单位可以确保文本长度适应用户设置的字体大小，保持相对的一致性。这些指南旨在帮助设计师和作者创建易于阅读且视觉上吸引人的文本布局。选择合适的文本宽度可以提高文本的可读性，并确保读者在阅读时有良好的体验。

## Responsive Design

### Design principle 2:

on the web, use responsive design

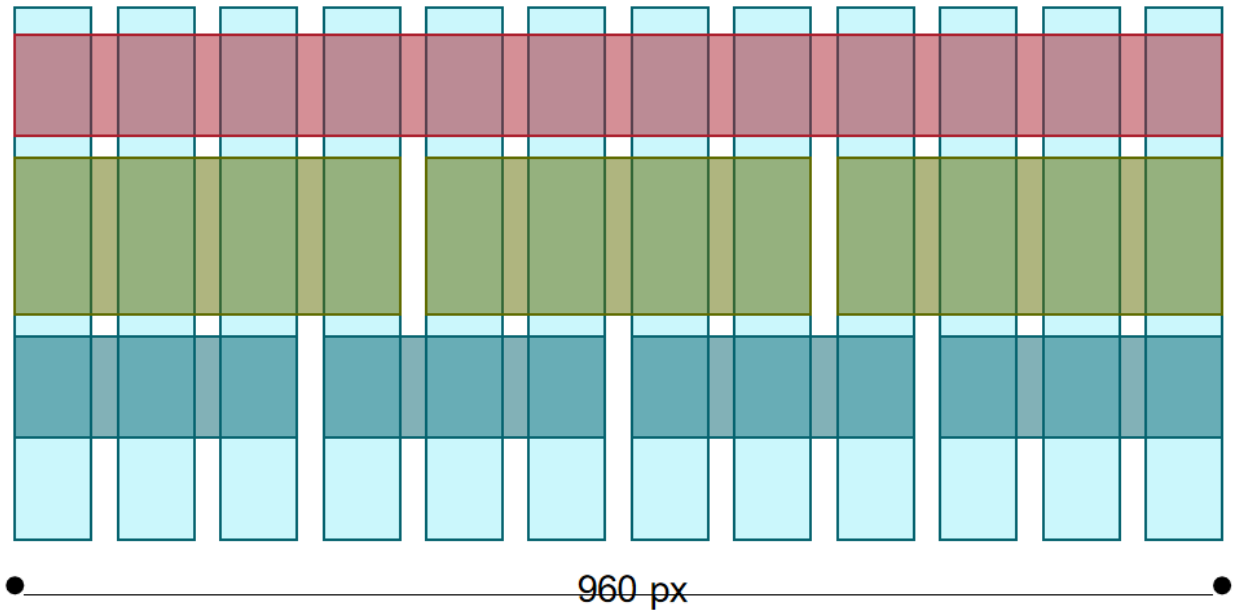
## Design — the easy part



- **Evenly:** 这表示元素之间的间距是均匀的，没有明显的偏向或不均匀的感觉。在网格布局中，这意味着网格单元之间的水平和垂直间距相等，使整个布局看起来均匀和平衡。
- **Spaced:** 这表示元素之间的间距是有意设置的，而不是随意的。在网格布局中，间距的大小可能是设计师根据特定需求和审美考虑进行调整的结果。这种间距的设置可以影响到布局的整体外观和感觉，以及用户对页面内容的感知。
- **Rectangular:** 这指的是网格单元的形状是矩形的，而不是其他形状。矩形形状的网格单元通常更容易布局和管理，因为它们的边界和角落是直角的，这使得元素在网格中的定位更加简单和直观。
- **Grids:** 这指的是将页面划分成网格，以便更好地组织和排列内容。网格可以是二维的，包括水平和垂直方向的线条，也可以是更复杂的多维网格。使用网格可以使设计师更容易地对页面元素进行定位，提高设计的一致性和可读性。



# 960.gs



## 7.3.1 Styling Text

Bash

```
body {  
    margin: 0 auto;  
    max-width: 40em;  
    line-height: 125%;  
}  
  
document.body.style.fontSize="24px";  
  
background-color: rgba(0, 0, 255, 0.25);padding-left: 0.5em;  
  
ul {  
    padding-left: 2em; /* the bullets appear in the padding */  
    list-style-type: disc;  
    list-style-position: outside;  
}
```

### 7.3.2 framework

Bash

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
</div>

-----

.grid-container {
  display: grid;
  grid-template-columns: repeat(3, 100px); /* 定义三列，每列宽度为 100px */
  gap: 10px; /* 定义网格间隔为 10px */
}

.grid-item {
  background-color: lightblue;
  padding: 20px;
  text-align: center;
}

-----

.item {
  grid-column: span 2; /* 子元素跨越 2 列 */
}
```

当然，CSS 的盒模型（Box Model）是一个在网页设计中非常重要的概念。它描述了元素在网页布局中所占的空间方式。盒模型主要包括四个部分，从内到外依次是：内容（Content）：这是元素的实际内容，如文本、图片等，通过 width 和 height 属性可以设置其大小。内边距（Padding）：内边距是内容周围的空白区域，它清晰地隔开了元素的内容和边框。边框（Border）：边框是围绕在内边距和内容外的线。边框的大小和样式可以通过 CSS 进行设置。外边距（Margin）：外边距是边框外的空白区域，用于隔开不同元素之间的空间。这四个部分一起构成了一个元素的完整视觉

呈现，也决定了元素在网页布局中所占据的空间大小。理解盒模型对于掌握 CSS 布局非常重要。希望这个解释对你有所帮助！

你提供的是一些 CSS 属性，它们都是与 CSS 盒模型（Box Model）相关的。让我来解释一下：  
`margin-top: 10px;`：这个属性设置了元素的上外边距（Margin）为 10 像素。外边距是元素边框外的空白区域，用于隔开不同元素之间的空间。  
`padding-left: 20px;`：这个属性设置了元素的左内边距（Padding）为 20 像素。内边距是内容周围的空白区域，它清晰地隔开了元素的内容和边框。  
`margin: 10px 5px 15px 20px;`：这个属性是一个简写属性，用于同时设置元素的上、右、下、左外边距。四个值分别对应上、右、下、左的外边距，所以这个属性设置了上外边距为 10 像素，右外边距为 5 像素，下外边距为 15 像素，左外边距为 20 像素。  
`border: 1px solid black;`：这个属性是一个简写属性，用于同时设置元素的边框宽度、样式和颜色。所以这个属性设置了元素的边框宽度为 1 像素，样式为实线，颜色为黑色。