中南大學
CENTRAL SOUTH UNIVERSITY

第四讲 CSS

盛 羽

中南大学计算机学院

shengyu@csu. edu. cn

目录

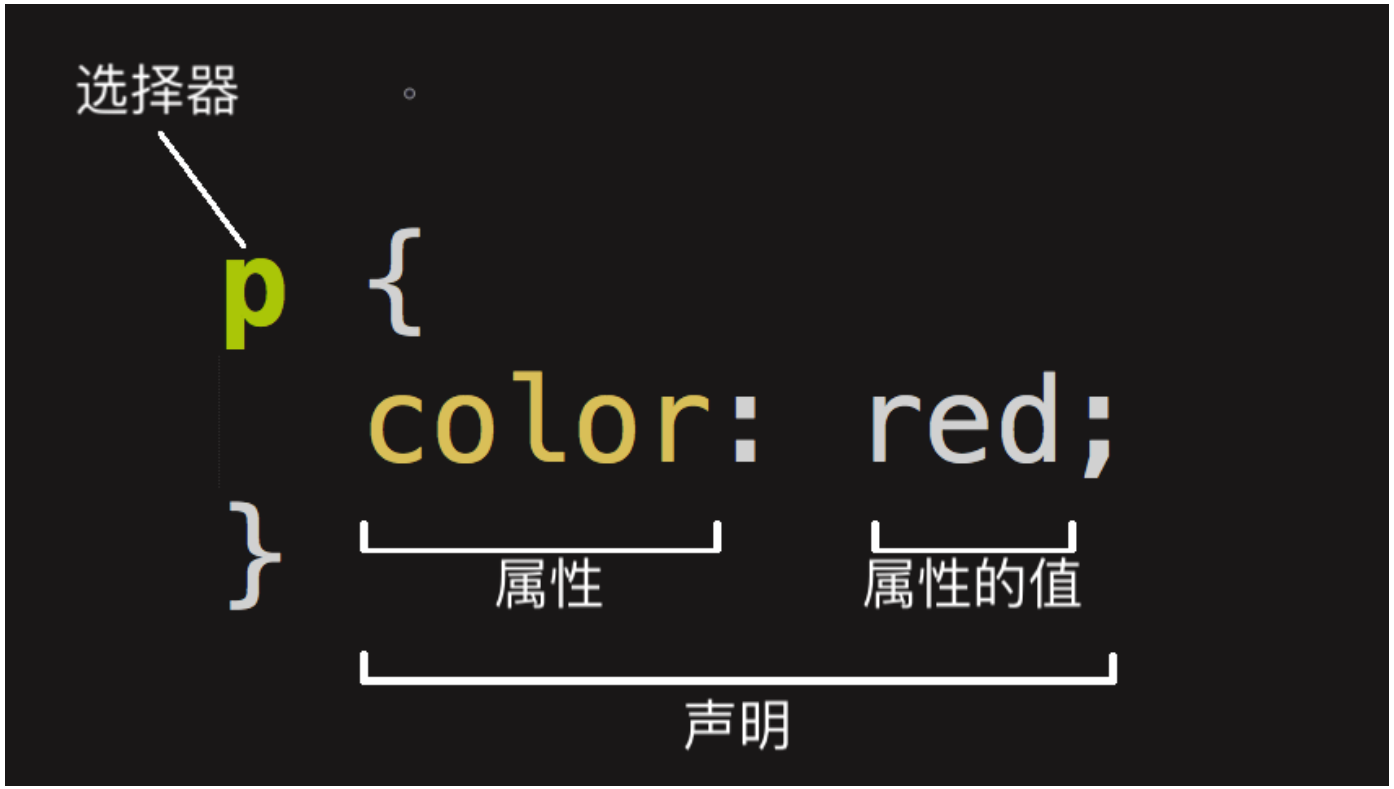
1 CSS简介

- CSS
  - Cascading Style Sheets
  - 层叠样式表
  - 定义如何显示（渲染）HTML (XML)元素
  - 表现与内容分离
- HTML
  - 定义文档的内容
  - 样式表可以应用到多个HTML文件中
- 示例
  - Demo
  - <link href="4-01-index.css" rel="stylesheet">
- 本讲内容主要来自MDN Web Docs
  - https://developer.mozilla.org/zh-CN/docs/Learn/CSS/

CSS的定义

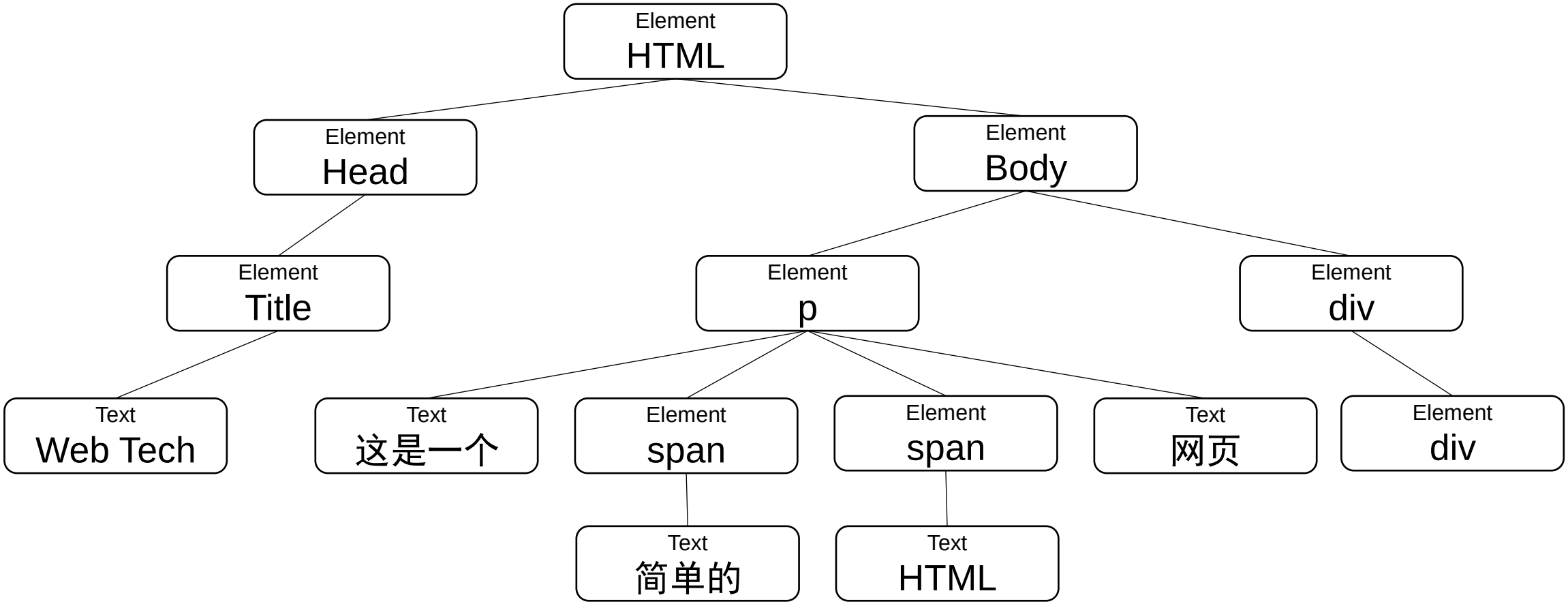
selector {declaration1; declaration2; ... declarationN }

- Selector/选择器：，需要改变样式的HTML元素
- Declaration/声明=属性：值



目录

- 选择器
  - 匹配文档中的元素，匹配的元素（Element）将会拥有规则指定的样式。



4-06-Selector.html

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <link href="4-06-Selector.css" rel="Stylesheet">
  </head>
  <body>
    <p>This is happy text.</p>
    <div class="warning">
      Be careful! There are wizards present,
      and they are quick to anger!
    </div>

    <div id="customized">
      <p>This is happy text.</p>

      <div class="warning">
        Be careful! There are wizards present,
        and they are quick to anger!
      </div>
    </div>
  </body>
</html>
```

4-06-Selector.css

```css
p {
  color: green;
}

div.warning {
  width: 100%;
  border: 2px solid yellow;
  color: white;
  background-color: darkred;
  padding: 0.8em 0.8em 0.6em;
}

#customized {
  font: 16px Arial, Helvetica, sans-serif;
}
```

This is happy text.

Be careful! There are wizards present, and they are quick to anger!

This is happy text.

Be careful! There are wizards present, and they are quick to anger!

2.选择器

- 元素（标签）选择器(ElementName)
  - 通过node节点名称匹配元素

```
<span>这里是由 span 包裹的一些文字.</span>
<p>这里是由 p 包裹的一些文字.</p>
```

```
<style>
    span {
        background-color: DodgerBlue;
        color: #ffffff;
    }
</style>
```

这里是由 span 包裹的一些文字.

这里是由 p 包裹的一些文字.

2.选择器

■类选择器（.classname）:根据元素的类属性(class)中的内容匹配元素

demo.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
        .classy {
            background-color: DodgerBlue;
            color: #ffffff;
        }
    </style>
  </head>
  <body>
    <span class="classy">这里是由 span 包裹的一些文字.</span>
    <p class="classy">这里是由 p 包裹的一些文字.</p>
    <p>这里是没有带class属性的 p 包裹的一些文字</p>
    <p class="classx">这里是class属性值不等于classy的 p 包裹的一些文字</span>
  </body>
</html>
```

这里是由 span 包裹的一些文字.

这里是由 p 包裹的一些文字.

这里是没有带class属性的 p 包裹的一些文字

这里是class属性值不等于classy的 p 包裹的一些文字

2.选择器

■ID选择器（#idname）:根据该元素的 ID 属性（id）中的内容匹配元素

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
        #demo{
            background-color: DodgerBlue;
            color: #ffffff;
        }
    </style>
  </head>
  <body>
    <p id="demo">这里是由 p 包裹的一些文字.</p>
    <p>这里是没有带id属性的 p 包裹的一些文字</p>
    <p id="demo1">这里是id属性值不等于demo的 p 包裹的一些文字</span>
  </body>
</html>
```

这里是由 p 包裹的一些文字.

这里是没有带id属性的 p 包裹的一些文字

这里是id属性值不等于demo的 p 包裹的一些文字

2.选择器

■通配选择器(*)：匹配任意类型的HTML元素

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      *{
        color:green;
      }
      *.warning {color:red;}
      *#maincontent {border: 1px solid blue;}
    </style>
  </head>
  <body>
    <p class="warning">
      a red paragraph.
    </p>
    <p id="maincontent">
      <span class="warning">A red span</span> in a green paragraph.
    </p>
  </body>
</html>
```

a red paragraph.

A red span in a green paragraph.

2.选择器

■组选择器

- ■并集选择器（,）：element, element, element { style properties }
- ■规则适用于列出的所有选择器，无先后顺序之分

```
h1,h2,h3,h4,h5,h6{
    font-family: helvetica;
}
```

```
#main,.content,article{
    font-size: 1.1em;
}
```

- 关系选择器
  - 邻近兄弟元素选择器（+）
    - 介于两个选择器之间，当第二个元素紧跟在第一个元素之后（在DOM树里的位置），并且两个元素都是属于同一个父元素的子元素，则第二个元素将被选中。
    - 紧挨着的两兄弟中的弟弟（不一定是同一类标签）
  - 语法
    - former_element + target_element { style properties }

2.选择器

```
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
       li + li {font-weight:bold;}
    </style>
  </head>
  <body>
    <ul>
      <li>List item 1</li>
      <li>List item 2</li>
      <li>List item 3</li>
    </ul>
    <ol>
      <li>List item 1</li>
      <li>List item 2</li>
      <li>List item 3</li>
    </ol>
 </body>
</html>
```

- List item 1
- **List item 2**
- **List item 3**

1. List item 1
2. **List item 2**
3. **List item 3**

关系选择器

- 兄弟元素选择器（ ~ ）
  - 兄弟选择符，位置无须紧邻，只须同层级，A~B 选择A元素之后所有同层级B元素。
  - 作为哥哥的A后面跟着所有选择器为B的弟弟
- 语法
  - former_element ~ target_element { style properties }

2.选择器

```html
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      p ~ span {
        color: red;
      }
    </style>
  </head>
  <body>
    <span>This is not red.</span>
    <p>Here is a paragraph.</p>
    <code>Here is some code.</code>
    <span>And here is a span.</span>
 </body>
</html>
```

This is not red.

Here is a paragraph.

`Here is some code.` And here is a span.

关系选择器

- 直接子元素选择器（>）
  - 匹配那些作为第一个元素的直接后代(子元素)的第二元素.
  - A>B：A为B的父节点，B为A的子节点（们）
- 语法
  - former_element > target_element { style properties }

2.选择器

```
<div>
  <span>Span 1. In the div.
    <span>Span 2. In the span that's in the div.</span>
    Span 1
  </span>
</div>
<span>Span 3. Not in a div at all</span>
<div>
  <div>Div 1.<div>Div 2.</div>Div 1.</div>
  <p>Para 1.</p>
  <div><p>Para 2. </p></div>
  <h1><p>Para 3.</p></h1>
</div>
```

```
<style>
  span { background-color: yellow; }
  div > span {
    background-color: DodgerBlue;
  }
  div{
    background-color: yellow;
  }
  div > div {
    background-color: DodgerBlue;
  }
  div > p {
    background-color: grey;
  }
</style>
```

Span 1. In the div. Span 2. In the span that's in the div. Span 1
Span 3. Not in a div at all
Div 1.
Div 2.
Div 1.

Para 1.

Para 2.

Para 3.

为什么Span 2.没有继承？

2.选择器

- 关系选择器
  - 后代元素选择器
    - 通常用单个空格（　）字符表示
    - 如果第二个选择器匹配的元素具有与第一个选择器匹配的祖先（不一定是父节点，父节点，父节点的父节点，父节点的父节点的父节点等）元素，则它们将被选择
    - A B: A节点包含的所有后代节点中满足B选择器的。
  - 语法

```
selector1 selector2 {
    /* property declarations */
}
```

2.选择器

```html
<ul>
  <li>
    <div>Item 1</div>
    <ul>
      <li>Subitem A</li>
      <li>Subitem B</li>
    </ul>
  </li>
  <li>
    <div>Item 2</div>
    <ul>
      <li>Subitem A</li>
      <li>Subitem B</li>
    </ul>
  </li>
</ul>
```

```css
li {
  list-style-type: disc;
}

li li {
  list-style-type: circle;
}
```

- Item 1
  - Subitem A
  - Subitem B
- Item 2
  - Subitem A
  - Subitem B

2.选择器

■属性选择器

4-07-Attribute selectors.html

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <link rel="stylesheet" href="4-07-Attr
  </head>
  <body>
    <div lang="en-us en-gb en-au en-nz">He
    <div lang="pt">Olá Mundo!</div>
    <div lang="zh-CN">世界您好！</div>
    <div lang="zh-TW">世界您好！</div>
    <div data-lang="zh-TW">世界您好！</div>
  </body>
</html>
```

4-07-Attribute selectors.css

```css
/* 将所有包含 `lang` 属性的 <div> 元素的字重设为 bold */
div[lang] {
  font-weight: bold;
}

/* 将所有语言为美国英语的 <div> 元素的文本颜色设为蓝色 */
div[lang~="en-us"] {
  color: blue;
}

/* 将所有语言为葡萄牙语的 <div> 元素的文本颜色设为绿色 */
div[lang="pt"] {
  color: green;
}

/* 将所有语言为中文的 <div> 元素的文本颜色设为红色
   无论是简体中文（zh-CN）还是繁体中文（zh-TW） */
div[lang|="zh"] {
  color: red;
}

/* 将所有 `data-lang` 属性的值为 "zh-TW" 的 <div> 元素的文本颜色设为紫色 */
/* 备注：和 JS 不同，CSS 可以在不使用双引号的情况下直接使用带连字符的属性名 */
div[data-lang="zh-TW"] {
  color: purple;
}
```

Hello World!
Olá Mundo!
世界您好!
世界您好！
世界您好!

2.选择器

- 伪类(pseudo-classes)与伪元素(pseudo-elements)[1]
  - 定义
    - css 引入伪类和伪元素概念是为了格式化文档树(DOM)以外的信息
    - 其他的选择器是对整个元素的所有状态发挥作用
  - 伪类
    - 选择处于特定状态的元素
    - 当已有元素处于的某个状态时，为其添加对应的样式
    - 比如说，当用户悬停在指定的元素时，我们可以通过:hover 来描述这个元素的状态。
  - 伪元素
    - 元素的一部分，如第一个字母，第一段等
    - 不在文档树中的元素，为其添加样式。如：以通过:before 来在一个元素前增加一些文本，并为这些文本添加样式

[1]AlloyTeam 腾讯全端AlloyTeam 团队Blog，http://www.alloyteam.com/2016/05/summary-of-pseudo-classes-and-pseudo-elements/

- 伪类与伪元素
  - 基本语法

```
Selector : pseudo-class
{
    property: value;
}

Selector :: pseudo-elements
{
    property: value;
}
```

[1]AlloyTeam 腾讯全端AlloyTeam 团队Blog，http://www.alloyteam.com/2016/05/summary-of-pseudo-classes-and-pseudo-elements/

2.选择器

■伪类与伪元素

■示例：伪类

- 我是第一个
- 我是第二个

```
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      li.first-item {
          color: orange
      }
    </style>
  </head>
  <body>
    <ul>
      <li class="first-item">我是第一个</li>
      <li>我是第二个</li>
    </ul>
  </body>
</html>
```

```
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      li:first-child {
          color: orange
      }
    </style>
  </head>
  <body>
    <ul>
      <li>我是第一个</li>
      <li>我是第二个</li>
    </ul>
  </body>
</html>
```

伪类与伪元素

示例：伪类（a标签）

4-08-Pseudo classes-a.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      a:link { color: blue; }          /* 未访问链接 */
      a:visited { color: purple; }     /* 已访问链接 */
      a:hover { background: yellow; }  /* 用户鼠标悬停 */
      a:active { color: red; }         /* 激活链接 */
      p:active { background: #eee; }   /* 激活段落 */
    </style>
  </head>
  <body>
    <p>This paragraph contains a link:
    <a href="#">This link will turn red while you click on it.</a>
      The paragraph will get a gray background while you click on it or the link.
    </p>
  </body>
</html>
```

2.选择器

■伪类与伪元素

■:not() ：否定选择器，
可以选择除某个元素之外的所有元素。

```
<p>我是一个段落。</p>
<p class="fancy">我好看极了！</p>
<div>我「不是」一个段落。</div>
```

我是一个段落。

我好看极了！

我「不是」一个段落。

```
<style>
  .fancy {
    text-shadow: 2px 2px 3px gold;
  }

  /* 类名不是 `.fancy` 的 <p> 元素 */
  p:not(.fancy) {
    color: green;
  }

  /* 非 <p> 元素 */
  body :not(p) {
    text-decoration: underline;
  }

  /* 既不是 <div> 也不是 <span> 的元素 */
  body :not(div):not(span) {
    font-weight: bold;
  }
</style>
```

2.选择器

- 伪类与伪元素
  - :target()选择器来对页面某个target元素指定样式
  - 只在用户点击了页面中的超链接
    并且跳转到target元素后起作用

```html
<body>
  <h3>Table of Contents</h3>
  <ol>
   <li><a href="#p1">Jump to the first paragraph!</a></li>
   <li><a href="#p2">Jump to the second paragraph!</a></li>
   <li><a href="#nowhere">This link goes nowhere,
     because the target doesn't exist.</a></li>
  </ol>

  <h3>My Fun Article</h3>
  <p id="p1">You can target <i>this paragraph</i> using a
    URL fragment. Click on the link above to try out!</p>
  <p id="p2">This is <i>another paragraph</i>, also accessible
    from the links above. Isn't that delightful?</p>
</body>
</html>
```

4-09-Pseudo classes-target.html

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      p:target {
        background-color: gold;
      }

      /* 在目标元素中增加一个伪元素*/
      p:target::before {
        font: 70% sans-serif;
        content: "►";
        color: limegreen;
        margin-right: .25em;
      }

      /*在目标元素中使用italic样式*/
      p:target i {
        color: red;
      }
    </style>
  </head>
```

2.选择器

- 伪类与伪元素
  - :nth-child(an+b) :所有当前元素的兄弟元素按照位置先后顺序从1开始排序
  - 选择的结果为CSS伪类:nth-child括号中表达式（an+b）匹配到的元素集合（n=0，1，2，3...)
    - 2n+0 或简单的 2n 匹配位置为 2、4、6、8...的元素（n=0时，2n+0=0，第0个元素不存在，因为是从1开始排序)。你可以使用关键字 even 来替换此表达式。
    - 2n+1 匹配位置为 1、3、5、7...的元素。你可以使用关键字 odd 来替换此表达式
    - 3n+4 匹配位置为 4、7、10、13...的元素。

2.选择器

■伪类与伪元素

■:nth-child(an+b)

AAAAAA	BBBBBB	CCCCCC
111111	111111	111111
222222	222222	222222
333333	333333	333333
444444	444444	444444
555555	555555	555555

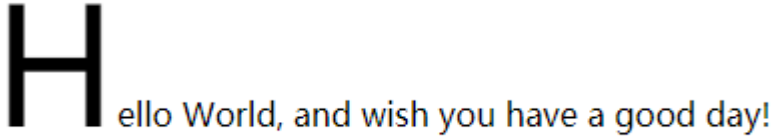
4-10-Pseudo classes-nth-child.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      table{
          border-collapse: collapse;
      }
      table,th,td{
        border: 1px black solid;
      }

      tr:nth-child(even){
        background-color: grey;
      }
    </style>
  </head>
  <body>
    <table>
      <tr><th>AAAAAA</th><th>BBBBBB</th><th>CCCCCC</th></tr>
      <tr><td>111111</td><td>111111</td><td>111111</td></tr>
      <tr><td>222222</td><td>222222</td><td>222222</td></tr>
      <tr><td>333333</td><td>333333</td><td>333333</td></tr>
      <tr><td>444444</td><td>444444</td><td>444444</td></tr>
      <tr><td>555555</td><td>555555</td><td>555555</td></tr>
    </table>
  </body>
</html>
```

2.选择器

■伪类与伪元素

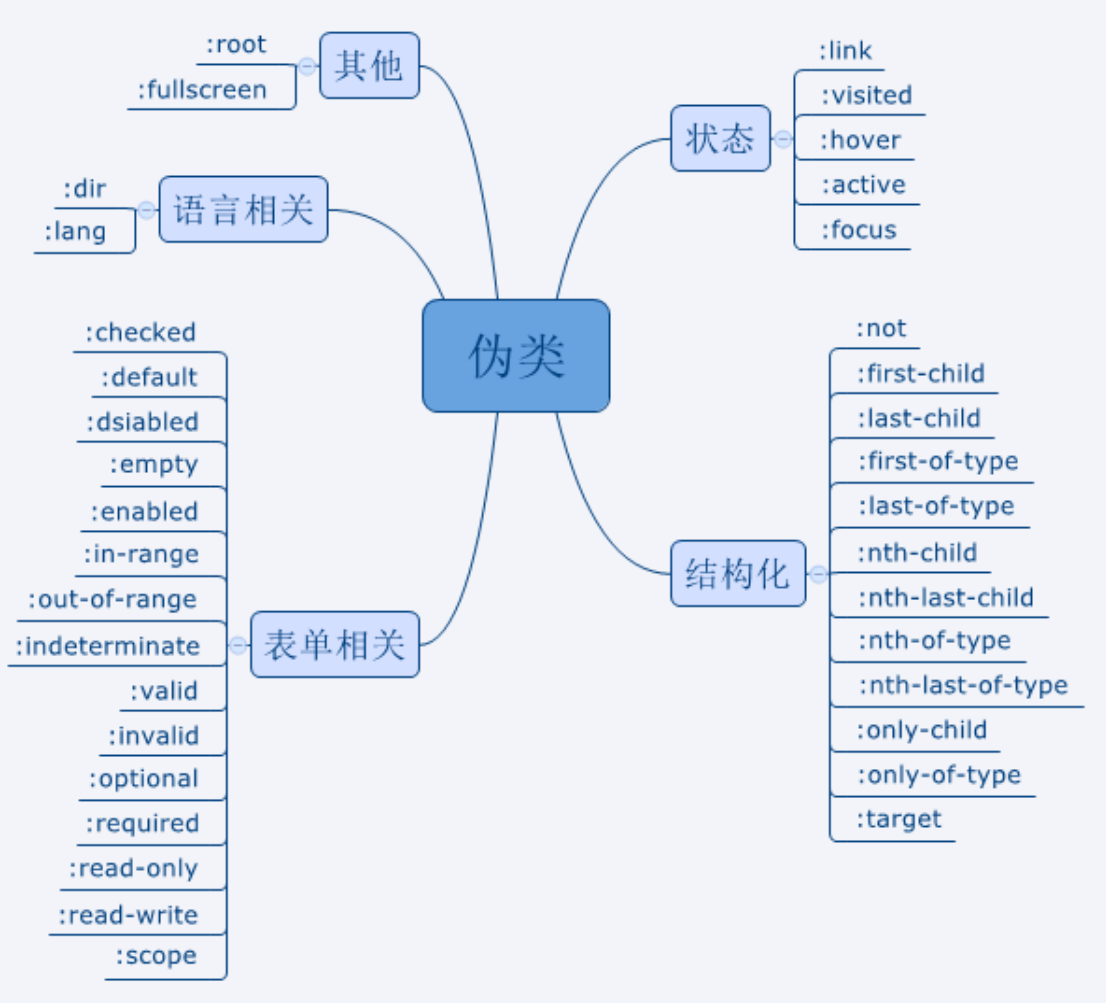


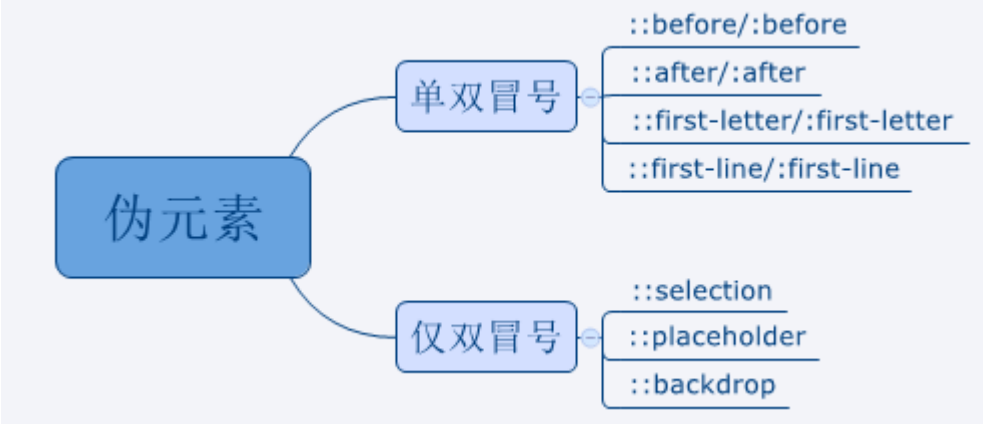

```
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .first {
          font-size: 5em;
      }
    </style>
  </head>
  <body>
    <p><span class="first">H</span>ello World,
      and wish you have a good day!
    </p>
  </body>
</html>
```

```
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      p::first-letter {
          font-size: 5em;
      }
    </style>
  </head>
  <body>
    <p>Hello World,
      and wish you have a good day!
    </p>
  </body>
</html>
```

2.选择器

- 伪类(pseudo-classes)与伪元素(pseudo-elements)[2]





[2] http://www.alloyteam.com/2016/05/summary-of-pseudo-classes-and-pseudo-elements/#prettyPhoto

2.选择器

- 层叠
  - 层叠样式表(Cascading Style Sheets)
  - 多个CSS规则应用于同一个元素
  - 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。

```
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      h1{
        color:red;
      }
      h1{
        color:blue;
      }
    </style>
  </head>
  <body>
    <h1>This is my heading</h1>
 </body>
</html>
```

This is my heading

3. 继承

- css属性继承当前元素的父辈元素上设置的值

4-05-Inheritance.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
        body {
            color: blue;
        }
        span {
            color: black;
        }
    </style>
  </head>
  <body>
    <p>As the body has been set to have a color of blue this is inherited through the descendants.</p>
    <p>We can change the color by targetting the element with a selector, such as this <span>span</span>.</p>
  </body>
</html>
```

As the body has been set to have a color of blue this is inherited through the descendants.

We can change the color by targetting the element with a selector, such as this span.

- 一些设置在父元素上的css属性是可以被子元素继承的，有些则不能（a width of 50%）

2. 优先级

- 当优先级与多个 CSS 声明中任意一个声明的优先级相等的时候，CSS 中最后的那个声明将会被应用到元素上。
- 一个元素选择器不是很具体，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，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。
- 看选择器可能覆盖的范围

```
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .main-heading{
        color:red;
      }
      h1{
        color:blue;
      }
    </style>
  </head>
  <body>
    <h1 class="main-heading">This is my heading</h1>
 </body>
</html>
```

This is my heading

2.选择器

- 优先级
  - 选择器类型
    - 下面列表中，选择器类型的优先级是递增的：
      - 类型选择器（例如，h1）和伪元素（例如，::before）
      - 类选择器 (例如，.example)，属性选择器（例如，[type="radio"]）和伪类（例如，:hover）
      - ID 选择器（例如，#example）。
    - 通配选择符（universal selector）（*）关系选择符（combinators）（+, >, ~, ' ', ||）和 否定伪类（negation pseudo-class）（:not()）对优先级没有影响。
    - 给元素添加的内联样式 (例如，style="font-weight:bold") 总会覆盖外部样式表的任何样式，因此可看作是具有最高的优先级。
    - !important 例外规则：当在一个样式声明中使用一个 !important 规则时，此声明将覆盖任何其他声明。

3.CSS的应用

■在HTML里面应用CSS

1. 外部样式表

■CSS编写在扩展名为.css 的单独文件中，并从HTML<link> 元素引用它

4-02-External stylesheet.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <link rel="stylesheet" href="4-02-External stylesheet.css">
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

4-02-External stylesheet.css

```
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
```

Hello World!

This is my first CSS example

■其他目录?

3.CSS的应用

■在HTML里面应用CSS

2. 内部样式表

- ■将CSS放在HTML文件<head>标签里的<style>标签之中
- ■难以复用

4-03-Internal stylesheet.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      h1 {
        color: blue;
        background-color: yellow;
        border: 1px solid black;
      }

      p {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

3.CSS的应用

■在HTML里面应用CSS

3. 内联样式

■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

4-04-Inline styles.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
  </head>
  <body>
    <h1 style="color: blue;background-color: yellow;border: 1px solid black;">Hello World!</h1>
    <p style="color:red;">This is my first CSS example</p>
  </body>
</html>
```

■除非你有充足的理由，否则不要这样做！

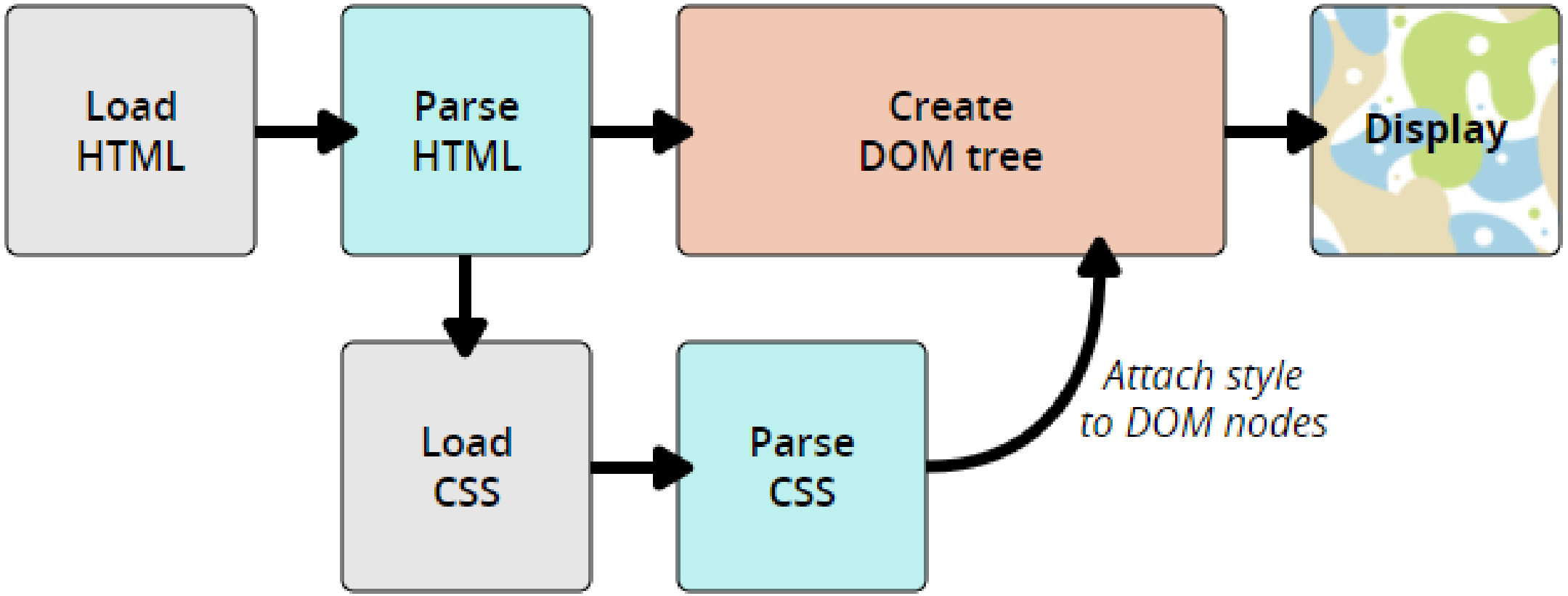
3.CSS的应用

CSS究竟是怎么工作的？

1. 浏览器载入HTML文件（比如从网络上获取）。
2. 将HTML文件转化成一个DOM（Document Object Model），DOM是文件在计算机内存中的表现形式。
3. 浏览器会拉取该HTML相关的大部分资源，比如嵌入到页面的图片、视频和CSS样式。JavaScript则会稍后进行处理。
4. 浏览器拉取到CSS之后会进行解析，根据选择器的不同类型（比如element、class、id等等）把他们分到不同的“桶”中。浏览器基于它找到的不同的选择器，将不同的规则应用在对应的DOM的节点中，并添加节点依赖的样式（这个中间步骤称为渲染树）。
5. 上述的规则应用于渲染树之后，渲染树会依照应该出现的结构进行布局。
6. 网页展示在屏幕上（这一步被称为着色）

3.CSS的应用

■CSS究竟是怎么工作的？



目录

目录

4.盒模型

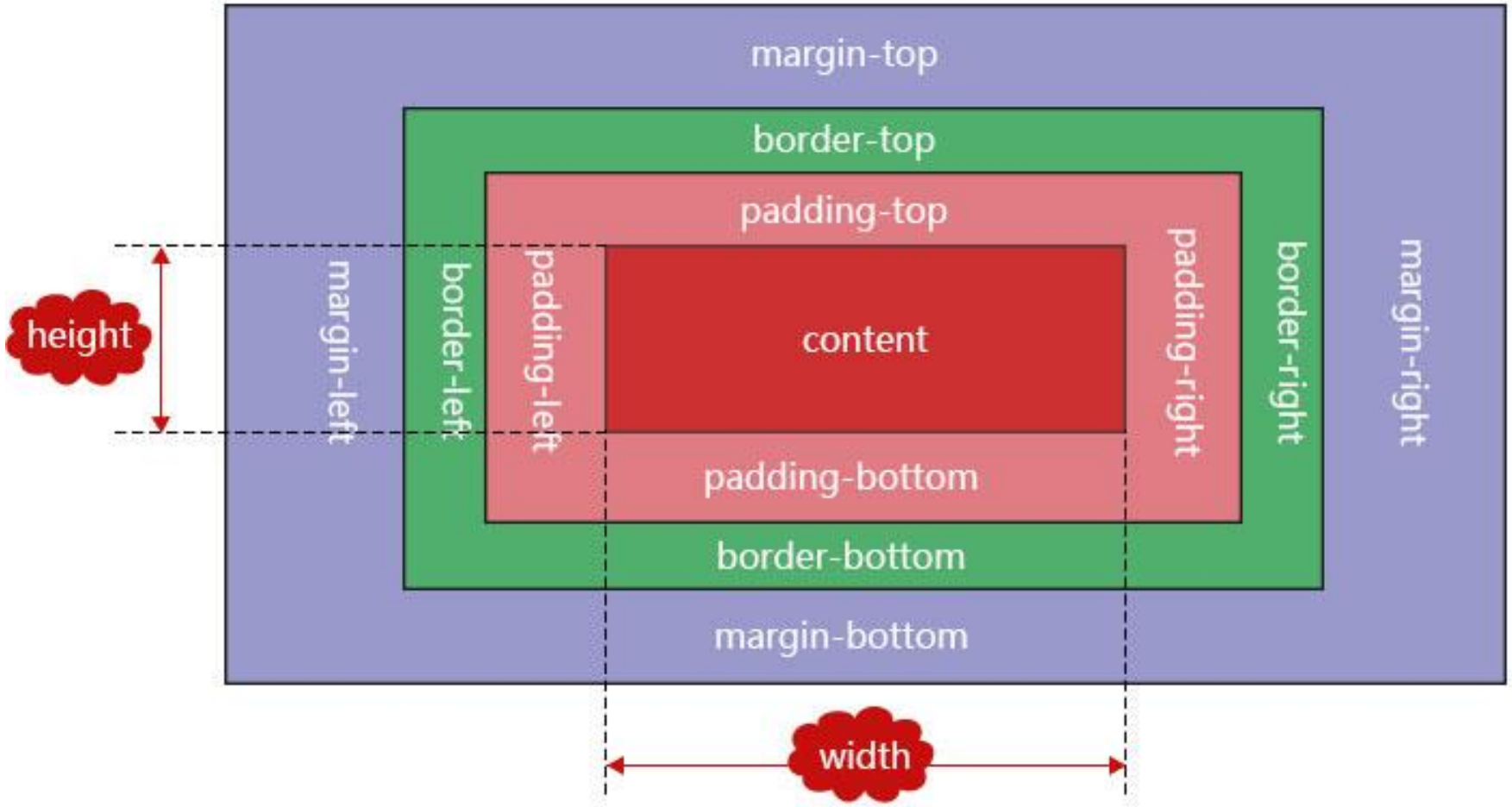
- 盒模型
  - 在 CSS 中，所有的元素都被一个个的“盒子（box）”包围着
  - 块级盒子（Block box） 和 内联盒子（Inline box）
  - 块级（block）盒子行为:
    - 盒子会在内联的方向上扩展并占据父容器在该方向上的所有可用空间，在绝大数情况下意味着盒子会和父容器一样宽
    - 每个盒子都会换行
    - width 和 height 属性可以发挥作用
    - 内边距（padding）, 外边距（margin） 和 边框（border） 会将其他元素从当前盒子周围“推开”
  - 内联盒子（Inline box）行为
    - 盒子不会产生换行
    - width 和 height 属性将不起作用。
    - 垂直方向的内边距、外边距以及边框会被应用但是不会把其他处于 inline 状态的盒子推开。
    - 水平方向的内边距、外边距以及边框会被应用且会把其他处于 inline 状态的盒子推开。
  - 通过对盒子display 属性的设置，比如 inline 或者 block ， 来控制盒子的外部显示类型
  - 正常文档流！

4.盒模型

- 盒模型
  - 完整的 CSS 盒模型应用于块级盒子，内联盒子只使用盒模型中定义的部分内容
  - 模型定义了盒的每个部分 —— margin, border, padding, and content
    - **Content box**: 这个区域是用来显示内容，大小可以通过设置 width 和 height.
    - **Padding box**: 包围在内容区域外部的空白区域； 大小通过 padding 相关属性设置。
    - **Border box**: 边框盒包裹内容和内边距。大小通过 border 相关属性设置。
    - **Margin box**: 这是最外面的区域，是盒子和其他元素之间的空白区域。大小通过 margin 相关属性设置。

■盒模型

W3C 盒子模型


margin-top
border-top
padding-top
margin-left
border-left
padding-left
content
padding-right
border-right
margin-right
padding-bottom
border-bottom
margin-bottom
height
width

4.盒模型

■盒模型

4-11-Block box.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        width: 400px;
        height: 150px;
        margin: 25px;
        padding: 25px;
        border: 5px solid black;
        font-size: 10px;
        background-color: grey;
        color:white;
        text-align: justify;
        line-height: 15px;
      }
    </style>
  </head>
  <body>
    <div class="box">This is a Block Box</div>
    <div class="box"></div>
    <div class="box"></div>
  </body>
</html>
```

4.盒模型

- 外边距
  - 盒子周围一圈看不到的空间
  - 外边距属性值可以为正也可以为负。设置负值会导致和其他内容重叠。

4-11-Block box-Margin.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .container{
        background-color: grey;
      }
      .box{
        margin-top: -40px;
        margin-right: 30px;
        margin-bottom: 40px;
        margin-left: 4em;
        width: 500px;
        height: 200px;
        background-color: yellow;
      }
    </style>
  </head>
  <body>
    <div class="container">
       Div 1.container
      <div class="box">Change my margin.</div>
    </div>
  </body>
</html>
```

4.盒模型

- 外边距折叠
  - 有两个外边距相接的元素，这些外边距将合并为一个外边距
  - 即最大的单个外边距的大小

I am paragraph one.

I am paragraph two.

4-12-Block box-Margin collapsing.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .one {
        margin-bottom: 50px;
      }

      .two {
        margin-top: 30px;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <p class="one">I am paragraph one.</p>
      <p class="two">I am paragraph two.</p>
    </div>
  </body>
</html>
```

4.盒模型

- 边框
  - 边距和填充框之间绘制的
  - 为边框设置样式时，有大量的属性可以使用
  - 有四个边框，每个边框都有样式、宽度和颜色

- border-top
- border-right
- border-bottom
- border-left

- border-width
- border-style
- border-color


```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .container {
        border-top: 5px dotted green;
        border-right: 1px solid black;
        border-bottom: 20px double rgb(23,45,145);
      }

      .box {
        border: 1px solid #333333;
        border-top-style: dotted;
        border-right-width: 20px;
        border-bottom-color: hotpink;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="box">Change my borders.</div>
    </div>
  </body>
</html>
```

4.盒模型

■内边距

- ■内边距位于边框和内容区域之间
- ■不能有负数量的内边距，所以值必须是0或正的值

- padding-top
- padding-right
- padding-bottom
- padding-left

4-14-Block box-Padding.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        padding-top: 0;
        padding-right: 30px;
        padding-bottom: 40px;
        padding-left: 4em;
        background-color: yellow;
      }

      .container {
        padding: 20px;
        background-color: grey;
      }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="box">Change my padding.</div>
    </div>
  </body>
</html>
```

- { box-sizing: border-box; }
  - Width包含：内容宽度+ border + padding

4.盒模型

- 盒子模型和内联盒子
  - 以上所有的方法都完全适用于块级盒子，有些属性也可以应用于内联盒子，例如由元素创建的那些内联盒子

4.盒模型

4-15-Inline boxes.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      span {
        margin: 20px;
        padding: 20px;
        width: 80px;
        height: 50px;
        background-color: lightblue;
        border: 2px solid blue;
      }
    </style>
  </head>
  <body>
    <p>
      I am a paragraph and this is a <span>span</span> inside that paragraph.
      A span is an inline element and so does not respect width and height.
    </p>
  </body>
</html>
```

I am a paragraph and this is a span inside that paragraph. A span is an inline element and so does not respect width and height.

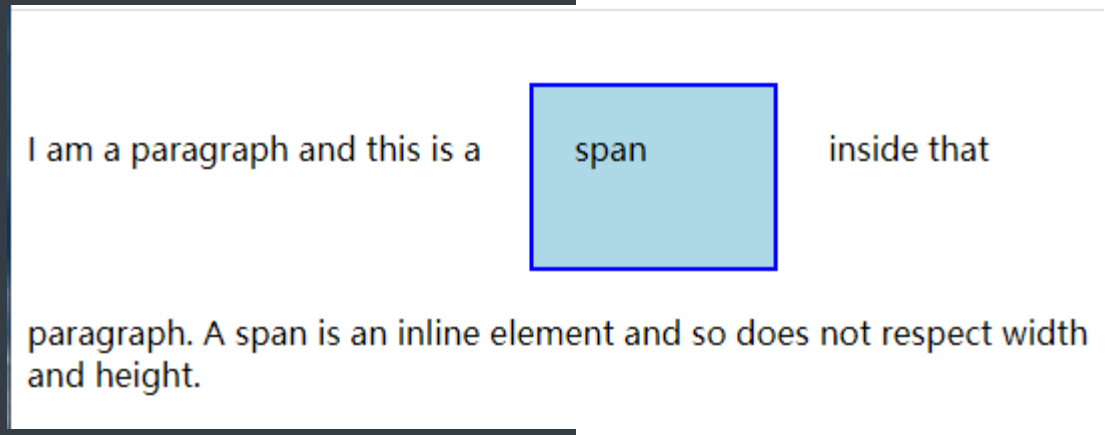
4.盒模型

- 使用display: inline-block
  - 内联和块之间的一个中间状态
  - 希望一个项切换到新行，但希望它可以设定宽度和高度，并避免上面看到的重叠
  - 一个元素使用 display: inline-block，实现我们需要的块级的部分效果：
    - 设置width 和height 属性会生效。
    - padding, margin, 以及border 会推开其他元素。

4.盒模型

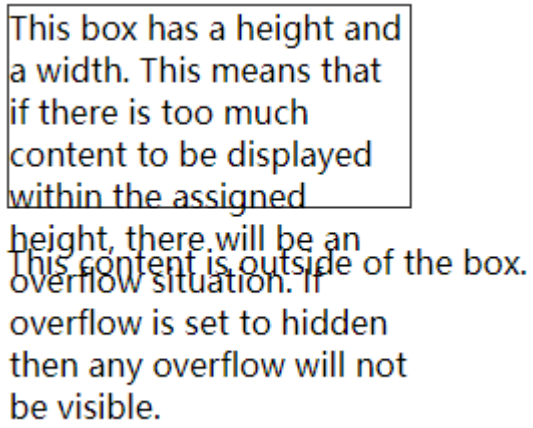
4-16-box-inline-block.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      span {
        margin: 20px;
        padding: 20px;
        width: 80px;
        height: 50px;
        background-color: lightblue;
        border: 2px solid blue;
        display: inline-block;
      }
    </style>
  </head>
  <body>
    <p>
      I am a paragraph and this is a <span>span</span> inside that paragraph.
      A span is an inline element and so does not respect width and height.
    </p>
  </body>
</html>
```



4.盒模型

- 溢出
  - 固定宽/高，内容过多
  - overflow属性
    - overflow: hidden
    - overflow: scroll
    - overflow-y: scroll



4-23-Overflow.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        border: 1px solid #333333;
        width: 200px;
        height: 100px;
      }
    </style>
  </head>
  <body>
    <div class="box">This box has a height and a width.
     This means that if there is too much content to be
     displayed within the assigned height, there
      will be an overflow situation.
      If overflow is set to hidden then any overflow
      will not be visible.</div>
    <p>This content is outside of the box.</p>
  </body>
</html>
```

目录

5.背景与边框

- 背景颜色
  - background-color属性
    - 定义CSS中任何元素的背景颜色
    - 属性接受任何有效的<color>值
    - 背景色扩展到元素的内容和内边距的下面

4-17-Background--Background Colors.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        background-color: #567895;
      }

      h2 {
        background-color: black;
        color: white;
      }
      span {
        background-color: rgba(255,255,255,.5);
      }
    </style>
  </head>
  <body>
    <div class="box">
      <h2>Background Colors</h2>
      <p>Try changing the background <span>colors</span>.</p>
    </div>
  </body>
</html>
```

5.背景与边框

- 背景颜色
  - background-image属性
    - 为一个元素设置一个或者多个背景图

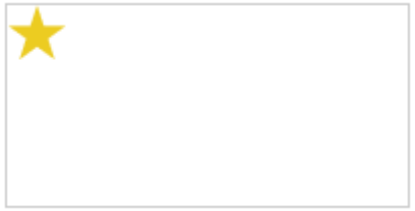
4-18-Background--Background Image--1.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .wrapper {
          display: flex;
      }
      .box {
          width: 200px;
          height: 100px;
          padding: .5em;
          border: 1px solid #ccc;
          margin: 20px;
      }
      .a {
        background-image: url(images/balloons.jpg);
      }

      .b {
        background-image: url(images/star.png);
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box a"></div>
      <div class="box b"></div>
    </div>
  </body>
</html>
```

5.背景与边框

- 背景颜色
  - 控制背景平铺
  - background-repeat属性用于控制图像的平铺行为。可用的值是
    - no-repeat — 不重复。
    - repeat-x —水平重复。
    - repeat-y —垂直重复。
    - repeat — 在两个方向重复。



4-19-Background--background-repeat.html

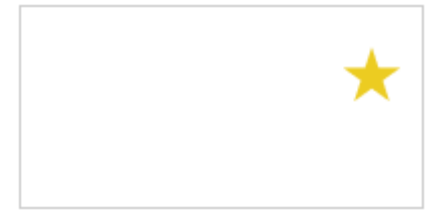
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        width: 200px;
        height: 100px;
        border: 1px solid #ccc;
        background-image: url(images/star.png);
        background-repeat: no-repeat;
      }
    </style>
  </head>
  <body>
    <div class="box"></div>
  </body>
</html>
```

5.背景与边框

■背景图像定位

■background-position属性

- ■择背景图像显示在其应用到的盒子中的位置
- ■它使用的坐标系中，框的左上角是(0,0)，框沿着水平(x)和垂直(y)轴定位。



4-20-Background--Background position.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        width: 200px;
        height: 100px;
        border: 1px solid #ccc;
        background-image: url(images/star.png);
        background-repeat: no-repeat;
        background-position: top 20px right 10px;
      }
    </style>
  </head>
  <body>
    <div class="box"></div>
  </body>
</html>
```

5.背景与边框

- 背景图像定位
  - 雪碧图（CSS Sprites）
    - 将多张图片合并到一张图片中，可以减小图片的总大小。
    - 将多张图片合并成一张图片后，下载全部所需的资源，只需一次请求。可以减小建立连接的消耗。

4-21-Background--Sprite.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .wrapper {
          display: flex;
      }
      .box {
          width: 80px;
          height: 80px;
          border: 1px solid #ccc;
          margin: 20px;
      }
      .a {
        background-image: url(images/Sprites.png);
        background-position: -10px -40px
      }

      .b {
        background-image: url(images/Sprites.png);
        background-position: -100px -40px
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box a"></div>
      <div class="box b"></div>
    </div>
  </body>
</html>
```

5.背景与边框

■圆角

■border-radius属性

- 使用两个长度或百分比作为值，
  第一个值定义水平半径，第二个值定义垂直半径

4-22-Background-radius.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        border: 10px solid rebeccapurple;
        border-radius: 1em;
        border-top-right-radius: 10% 30%;
      }
    </style>
  </head>
  <body>
    <div class="box">
      <h2>Borders</h2>
      <p>Try changing the borders.</p>
    </div>
  </body>
</html>
```

目录

6.值与单位

■数值

数值类型	描述
<integer>	<integer>是一个整数，比如1024或-55。
<number>	<number>表示一个小数——它可能有小数点后面的部分，也可能没有，例如0.255、128或-1.2。
<dimension>	<dimension>是一个<number>，它有一个附加的单位，例如45deg、5s或10px。<dimension>是一个伞形类别，包括<length>、<angle>、<time>和<resolution>类型。
<percentage>	<percentage>表示一些其他值的一部分，例如50%。百分比值总是相对于另一个量，例如，一个元素的长度相对于其父元素的长度。

6.值与单位

长度

绝对长度

单位	名称	等价换算
cm	厘米	1cm = 96px/2.54
mm	毫米	1mm = 1/10th of 1cm
Q	四分之一毫米	1Q = 1/40th of 1cm
in	英寸	1in = 2.54cm = 96px
pc	十二点活字	1pc = 1/16th of 1in
pt	点	1pt = 1/72th of 1in
px	像素	1px = 1/96th of 1in

6.值与单位

- 长度
  - 相对长度单位

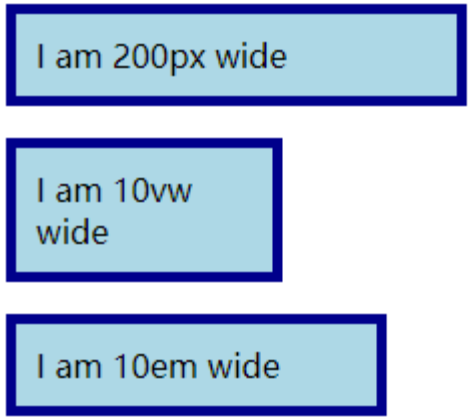
单位	相对于
em	在 font-size 中使用是相对于父元素的字体大小，在其他属性中使用是相对于自身的字体大小，如 width
ex	字符“x”的高度
ch	数字“0”的宽度
rem	根元素的字体大小
lh	元素的line-height
vw	视窗宽度的1%
vh	视窗高度的1%
vmin	视窗较小尺寸的1%
vmax	视图大尺寸的1%

6.值与单位

- 长度
  - 相对长度单位

4-24-Relative length.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .wrapper {
        font-size: 1em;
      }
      .box {
          background-color: lightblue;
          border: 5px solid darkblue;
          padding: 10px;
          margin: 1em 0;
      }
      .px {
        width: 200px;
      }
      .vw {
        width: 10vw;
      }
      .em {
        width: 10em;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box px">I am 200px wide</div>
      <div class="box vw">I am 10vw wide</div>
      <div class="box em">I am 10em wide</div>
    </div>
  </body>
</html>
```



6.值与单位

- em and rem
  - 在排版属性中 em 单位的意思是“父元素的字体大小”。
  - rem单位的意思是“根元素的字体大小”。

4-25--ems and rems.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      html {
        font-size: 16px;
      }
      .ems li {
        font-size: 1.3em;
      }
      .rems li {
        font-size: 1.3rem;
      }
    </style>
  </head>
```

- One
- Two
- Three
  - Three A
  - Three B
    - Three B 2

- One
- Two
- Three
  - Three A
  - Three B
    - Three B 2

```
<body>
  <ul class="ems">
    <li>One</li>
    <li>Two</li>
    <li>Three
      <ul>
        <li>Three A</li>
        <li>Three B
          <ul>
            <li>Three B 2</l
          </ul>
        </li>
      </ul>
    </li>
  </ul>

  <ul class="rems">
    <li>One</li>
    <li>Two</li>
    <li>Three
      <ul>
        <li>Three A</li>
        <li>Three B
          <ul>
            <li>Three B 2</l
          </ul>
        </li>
      </ul>
    </li>
  </ul>
</body>
```

6.值与单位

- 百分比
  - 相对于其他值设置的
  - 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
  - 每个<li>都有80%的字体大小，因此嵌套列表项在从父级继承其大小时将逐渐变小

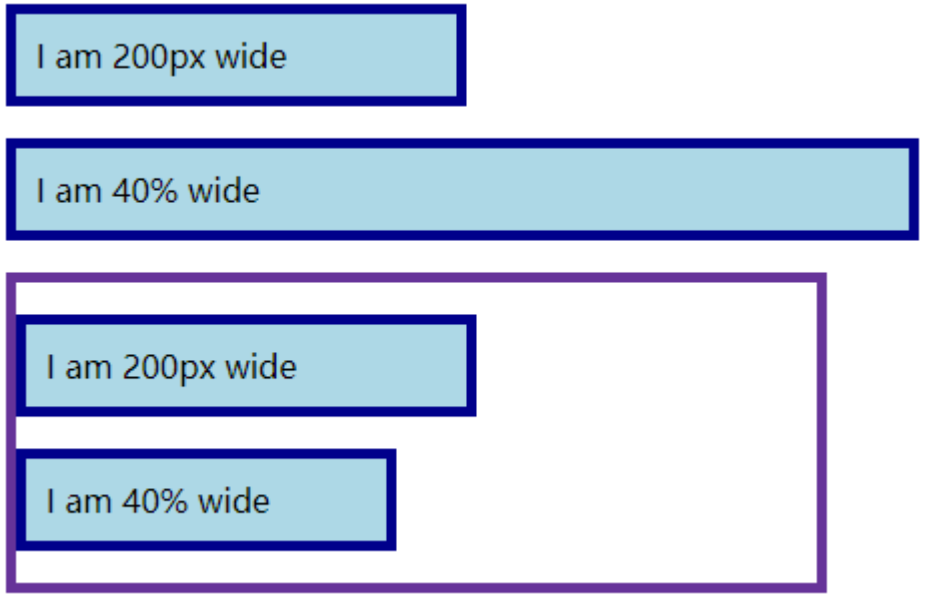
6.值与单位

■百分比

4-26--Percentages.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .wrapper {
        width: 400px;
        border: 5px solid rebeccapurple;
      }
      .box {
          background-color: lightblue;
          border: 5px solid darkblue;
          padding: 10px;
          margin: 1em 0;
      }
      .px {
        width: 200px;
      }

      .percent {
        width: 40%;
      }
    </style>
  </head>
  <body>
    <div class="box px">I am 200px wide</div>
    <div class="box percent">I am 40% wide</div>
    <div class="wrapper">
      <div class="box px">I am 200px wide</div>
      <div class="box percent">I am 40% wide</div>
    </div>
  </body>
</html>
```



6.值与单位

- 数字
  - 有些值接受数字，不添加任何单位。
  - 接受无单位数字的属性的一个例子是不透明度属性（opacity ），它控制元素的不透明度(它的透明程度)。此属性接受0(完全透明)和1(完全不透明)之间的数字。

6.值与单位

4-28-opacity.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        opacity: 0.6;
        background-color: lightblue;
        border: 5px solid darkblue;
        padding: 10px;
        margin: 1em 0;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box">I am a box with opacity</div>
    </div>
  </body>
</html>
```

6.值与单位

- 颜色
  - 现代计算机的标准颜色系统是24位的
  - 每个通道有256个不同的值(256 x 256 x 256 = 16,777,216)
  - 十六进制RGB值

6.值与单

■颜色

4-29-color.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        width: 200px;
        height: 100px;
        border: 1px solid #ccc;
      }
      .one {
        background-color: #02798b;
      }

      .two {
        background-color: rgb(197, 93, 161);
      }

      .three {
        background-color: rgba(18, 138, 125, .3);
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box one">#02798b</div>
      <div class="box two">background-color: rgb(197, 93, 161)</div>
      <div class="box three">background-color: rgba(18, 138, 125, .9)</div>
    </div>
 </body>
</html>
```

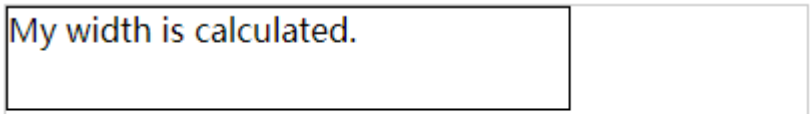
#02798b

background-color: rgb(197, 93, 161)

background-color: rgba(18, 138, 125, .9)

6.值与单位

- 函数
  - calc()函数：函数使您能够在CSS中进行简单的计算
  - calc()使框宽为20% + 100px。20%是根据父容器.wrapper的宽度来计算。



4-30--Functions.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .wrapper {
        width: 400px;
        height: 100px;
        border: 1px solid #ccc;
      }

      .box {
        width: calc(20% + 100px);
        height: 50px;
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box">My width is calculated.</div>
    </div>
  </body>
</html>
```

目录

7.在CSS中调整大小

- 原始尺寸，或固有尺寸
  - HTML元素有其原始的尺寸
  - img、div

4-31--Natural size.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      img {
        border: 5px solid darkblue;
      }
      .box {
        border: 5px solid darkred;
      }
    </style>
  </head>
  <body>
    <img src="images/star-1.png" alt="star">
    <div class="box"></div>
  </body>
</html>
```

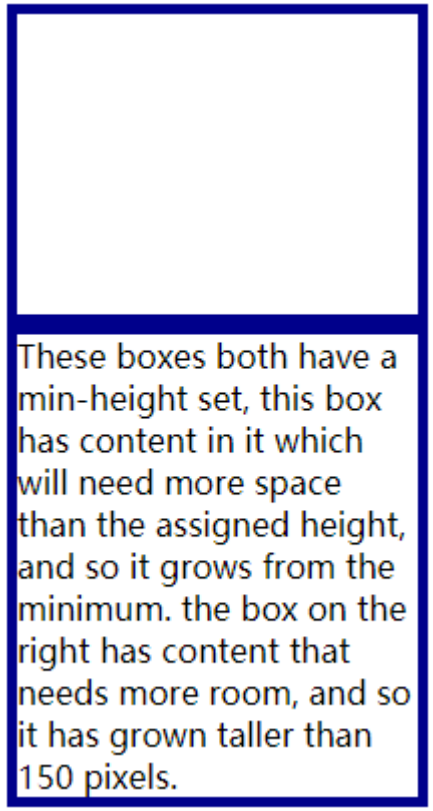
7.在CSS中调整大小

- 设置具体尺寸
  - width、height
- 使用百分数
  - 对于一个处于另外一个容器当中的盒子，如果你给予子盒子一个百分数作为宽度，那么它指的是父容器宽度的百分数。

7.在CSS中调整大小

■min-和max-尺寸

■min

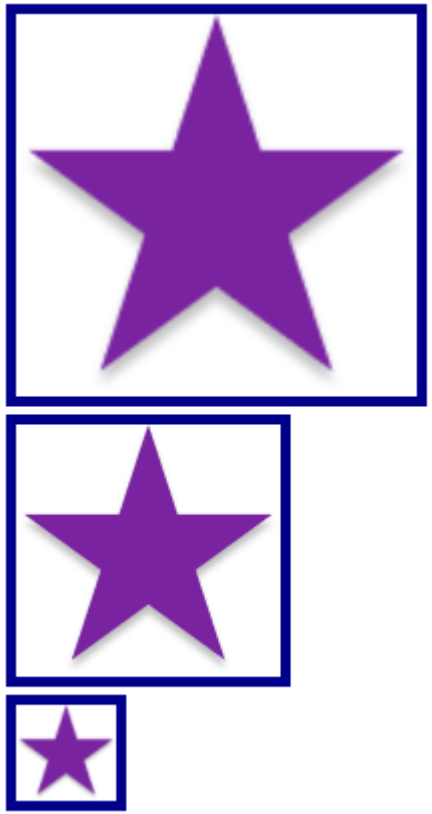


4-33--min size.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        border: 5px solid darkblue;
        min-height: 150px;
        width: 200px;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
    <div class="box"></div>
    <div class="box">These boxes both have a min-height set,
    this box has content in it which will need more space
    than the assigned height, and so it grows from the minimum.
      the box on the right has content that needs more room,
    and so it has grown taller than 150 pixels.</div>
  </div>
 </body>
</html>
```

7.在CSS中调整大小

- min-和max-尺寸
  - max



4-34--max size.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        width: 200px;
      }
      img {
        border: 5px solid darkblue;
      }
      .minibox {
        width: 50px;
      }
      .width {
        width: 100%;
      }
      .max {
        max-width: 100%;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box"><img src="images/star-1.png" alt="star" class="width"></div>
      <div class="box"><img src="images/star-1.png" alt="star" class="max"></div>
      <div class="minibox"><img src="images/star-1.png" alt="star" class="max"></div>
    </div>
  </div>
 </body>
</html>
```

7.在CSS中调整大小

- 调整图像大小
  - max-width: 100%
  - object-fit
    - contain:被替换的内容将被缩放，以在填充元素的内容框时保持其宽高比。（黑边）
    - cover：被替换的内容在保持其宽高比的同时填充元素的整个内容框。（裁剪）
    - fill：被替换的内容正好填充元素的内容框。（拉伸）
    - none
    - scale-down：内容的尺寸与 none 或 contain 中的一个相同，取决于它们两个之间谁得到的对象尺寸会更小一些。

7.在CSS中调整大小

■调整图像大小

4-35-object-fit.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My CSS experiment</title>
    <style>
      .box {
        width: 200px;
        height: 200px;
        border: 5px solid darkblue;
      }

      img {
        height: 100%;
        width: 100%;
      }

      .cover {
        object-fit: cover;
      }

      .contain {
        object-fit: contain;
      }
    </style>
  </head>
  <body>
    <div class="wrapper">
      <div class="box"><img src="images/balloons.jpg" alt="balloons" class="cover"></div>
      <div class="box"><img src="images/balloons.jpg" alt="balloons" class="contain"></div>
    </div>
  </div>
 </body>
</html>
```

目录

- 正常布局流
  - 取得元素的内容来放在一个独立的元素盒子中，然后在其周边加上内边距、边框和外边距
  - 默认的，一个块级元素的宽度是其父元素的100%，其高度与其内容高度一致。
  - 内联元素的height，width与内容一致（无法设置内联元素的height width）
  - 默认的，内部的块级元素会在垂直方向，从顶部开始一个接一个地放置，每个块级元素会在上一个元素下面另起一行
  - 独立容器，容器里面的子元素不会影响到外面的元素
  - 内联元素：只要在其父级块级元素的宽度内有足够的空间，它们与其他内联元素、相邻的文本内容（或者被包裹的）被安排在同一行

8.CSS布局

4-36-Normal Flow.html

```
      body {
        width: 500px;
        margin: 0 auto;
      }

      p {
        background: rgba(255,84,104,0.3);
        border: 2px solid rgb(255,84,104);
        padding: 10px;
        margin: 10px;
      }

      span {
        background: white;
        border: 1px solid black;
      }
    </style>
  </head>
  <body>
    <h1>Basic document flow</h1>

    <p>I am a basic block level element. My adjacent block level elements sit on new lin

    <p>By default we span 100% of the width of our parent element, and we are as tall as
    total width and height is our content + padding + border width/height.</p>

    <p>We are separated by our margins. Because of margin collapsing, we are separated by
    our margins, not both.</p>

    <p>inline elements <span>like this one</span> and <span>this one</span> sit on the same line as one another,
    and adjacent text nodes, if there is space on the same line. Overflowing inline elements will <span>wrap
    onto a new line if possible (like this one containing text)</span>, or just go on to a new line if not, much
    like this image will do: <img src="https://mdn.mozillademos.org/files/13360/long.jpg"></p>
  </div>
 </body>
</html>
```

Basic document flow

I am a basic block level element. My adjacent block level elements sit on new lines below me.

By default we span 100% of the width of our parent element, and we are as tall as our child content. Our total width and height is our content + padding + border width/height.

We are separated by our margins. Because of margin collapsing, we are separated by the width of one of our margins, not both.

inline elements like this one and this one sit on the same line as one another, and adjacent text nodes, if there is space on the same line. Overflowing inline elements will wrap onto a new line if possible (like this one containing text), or just go on to a new line if not, much like this image will do:

Flex布局

- 采用 Flex 布局的元素，称为 Flex 容器（Flex Container）
- 容器的所有子元素自动成为容器成员，称为 Flex 项目（Flex Item）

```
<div class=".flexcontainer">
    <div class=".flexitem">1</div>
    <div class=".flexitem">2</div>
    <div class=".flexitem">3</div>
</div>
```

```
.flexcontainer{
    display: flex;
}
```

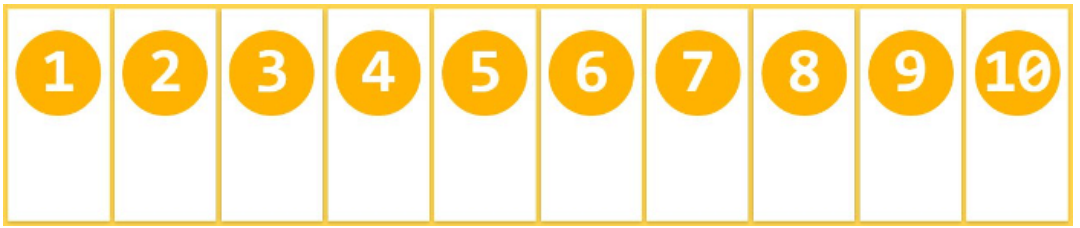
Flex布局

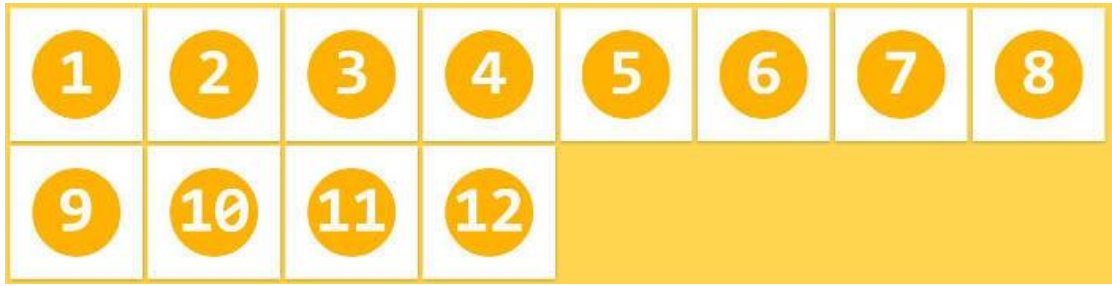
容器的属性

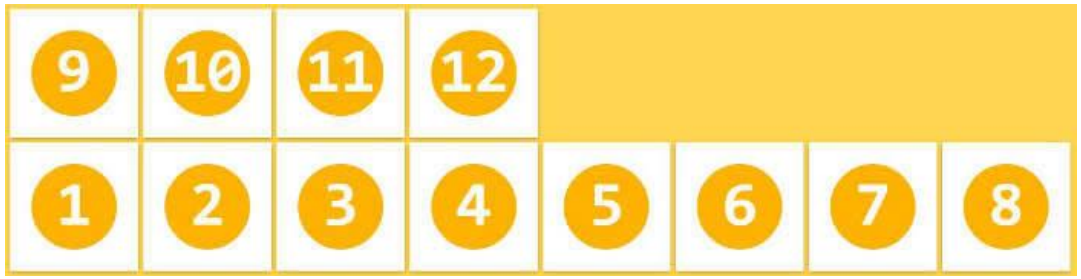
flex-direction：主轴的方向（怎么排）

- row（默认值）：主轴为水平方向，起点在左端。（横着排，从左到右）
- row-reverse：主轴为水平方向，起点在右端。（横着排，从右到左）
- column：主轴为垂直方向，起点在上沿。（竖着排，从上到下）
- column-reverse：主轴为垂直方向，起点在下沿。（竖着排，从下到上）

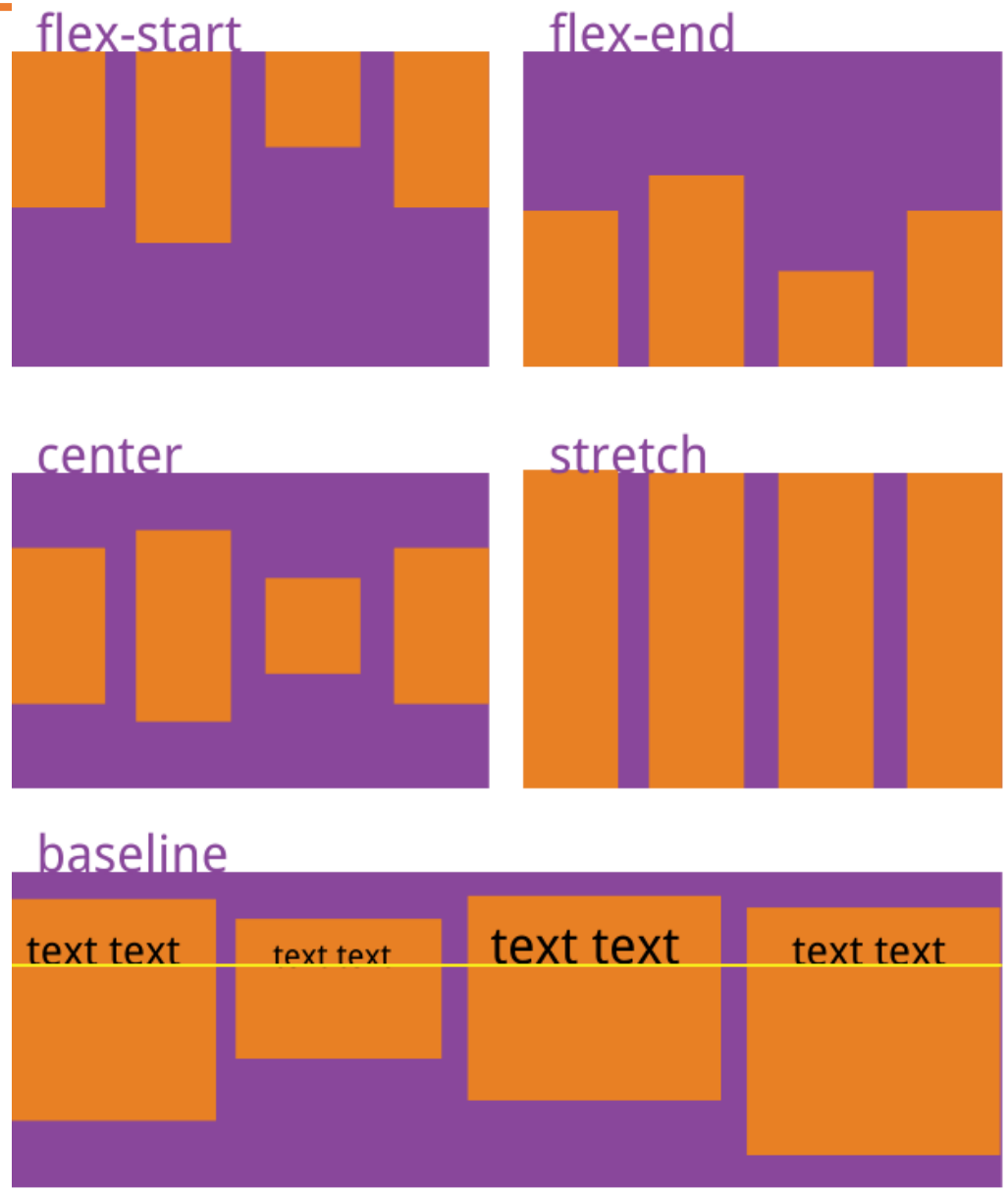
- Flex布局
  - 容器的属性
    - flex-wrap：
      - 如果一条轴线排不下，如何换行
      - nowrap（默认）：不换行
      - wrap：换行，第一行在上方。
      - wrap-reverse：换行，第一行在下方
    - flex-flow
      - flex-direction属性和flex-wrap属性的综合简写形式
      - 默认值为row nowrap。



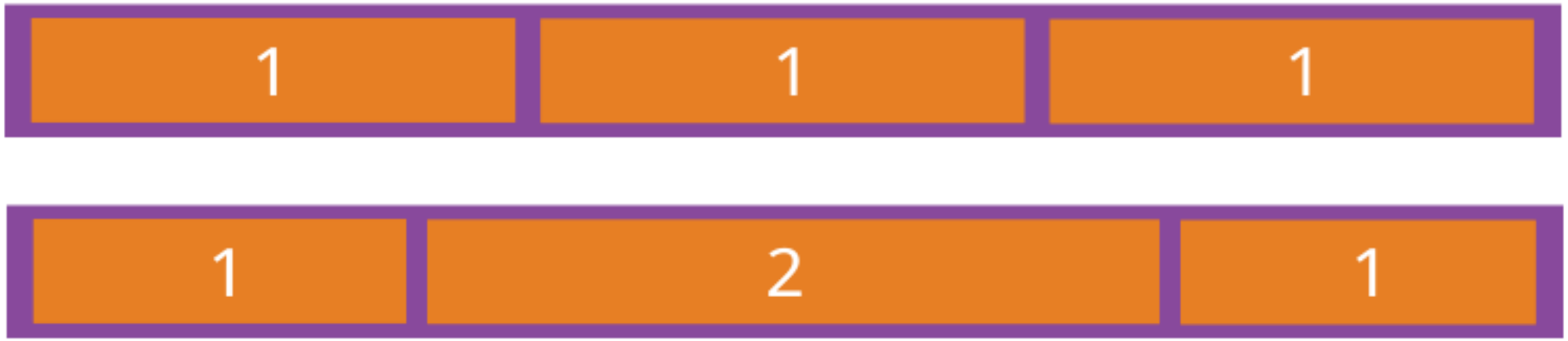




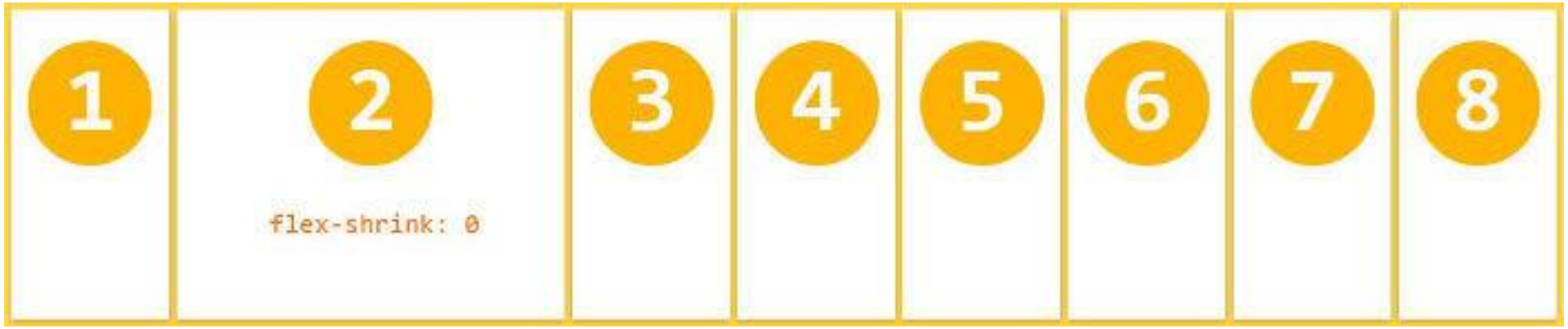
- Flex布局
  - 容器的属性
    - align-items
      - 定义项目在交叉轴上如何对齐
      - flex-start：交叉轴的起点对齐。（横着排，就是顶对齐）
      - flex-end：交叉轴的终点对齐。
      - center：交叉轴的中点对齐。
      - baseline: 项目的第一行文字的基线对齐。
      - stretch（默认值）：如果项目未设置高度或设为auto，将占满整个容器的高度。



- Flex布局
  - 项目(Flex Item)的属性
    - flex-grow
      - 容器还有剩余空间的情况下，各个项目如何放大。
      - 定义项目的放大比例，默认为0，即如果存在剩余空间，也不放大。
      - 如果所有项目的flex-grow属性都为1，则它们将等分剩余空间（如果有的话）。如果一个项目的flex-grow属性为2，其他项目都为1，则前者占据的剩余空间将比其他项多一倍。



- Flex布局
  - 项目的属性
    - flex-shrink
      - 如果容器空间小于各项目总空间之和的情况下，各个项目如何缩小
      - 定义了项目的缩小比例，默认为1，即如果空间不足，该项目将缩小
      - 如果所有项目的flex-shrink属性都为1，当空间不足时，都将等比例缩小。
        如果一个项目的flex-shrink属性为0，其他项目都为1，则空间不足时，前者不缩小。

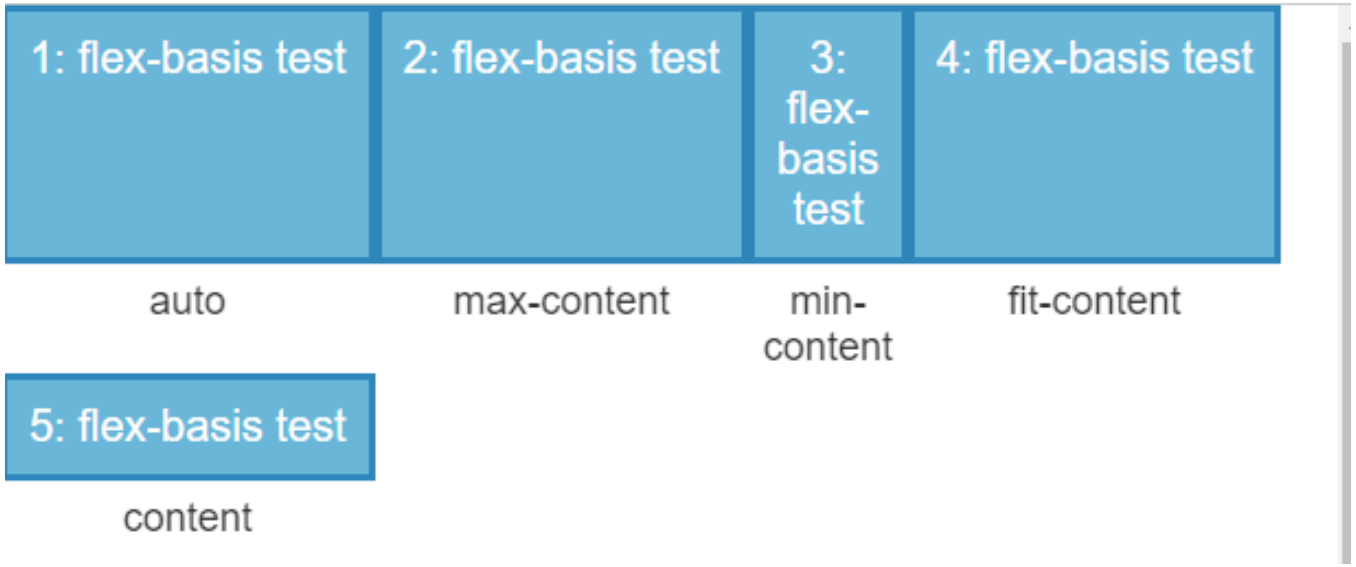


Flex布局

项目的属性

flex-basis

- 指定了 flex 元素在主轴方向上的初始大小，如何给各个项目分配空间（横向布局，就是宽度）。（
- 默认值为auto，即项目的本来大小，横向布局，就是各项目自己CSS定义的宽度
- 当一个元素同时被设置了 flex-basis (除值为 auto 外) 和 width (或者在 flex-direction: column 情况下设置了height) , flex-basis 具有更高的优先级。



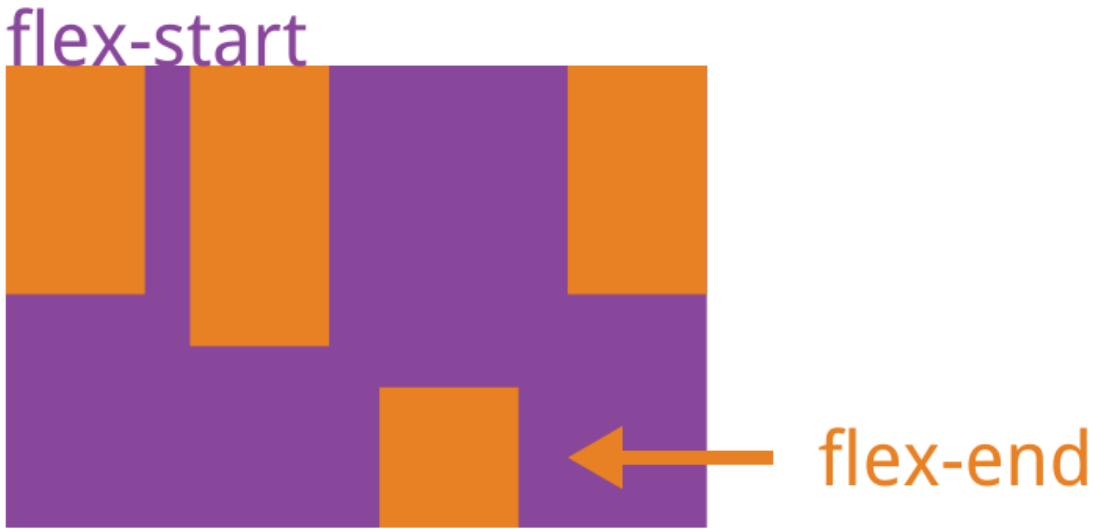
- Flex布局
  - 项目的属性
    - flex属性
      - flex-grow, flex-shrink 和 flex-basis的简写
      - 默认值为0 1 auto （不放大，等比例缩小，各项目自己定义的大小）
      - 后两个属性可选。
      - 该属性有两个快捷值：auto (1 1 auto) 和 none (0 0 auto)

Flex布局

项目的属性

align-self属性

- align-self属性允许单个项目有与其他项目不一样的对齐方式
- 可覆盖align-items属性
- 默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch



8.CSS布局

- 浮动
  - 示例
  - 使元素脱离文档流，按照指定方向发生移动，遇到父级边界或者相邻的浮动元素停了下来
  - 提升层级半层
  - 方向：左、右
  - 示例：首字母下沉
  - 多列浮动布局
    - 示例：两列布局
  - 清除浮动
    - 示例：清除浮动

- 定位
  - 从正常的文档流布局中取出元素，并使它们具有不同的行为
  - 静态定位(static)
    - 将元素放入它在文档布局流中的正常位置
  - 相对定位(relative)
    - 在正常的文档流中
    - 如果没有定位偏移量，对元素本身没有任何影响
    - 它原本所占的空间仍保留
    - top, bottom, left, 和 right 来精确指定要将定位元素移动到的位置
    - 示例
  - 绝对定位(absolute)
    - 完全脱离文档流
    - 如果有定位父级相对于定位父级发生偏移，没有定位父级相对于整个文档发生偏移

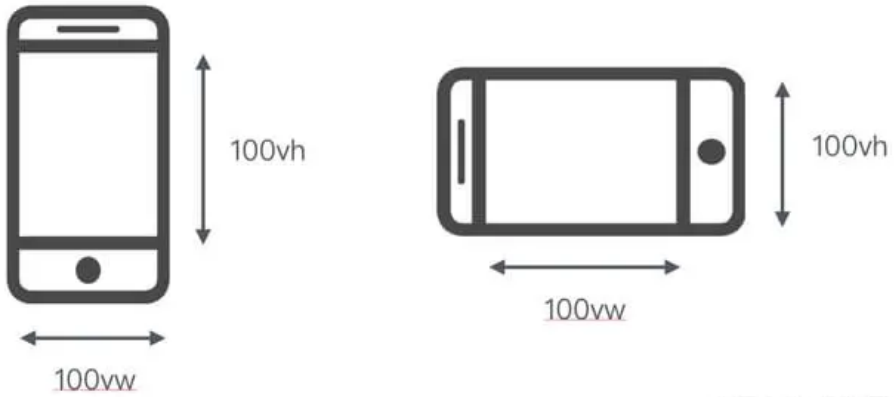
- 定位
  - 固定定位（fixed）
    - 绝对定位相对于其最近的定位祖先或者<html>
    - 固定定位相对于浏览器视口本身
    - 示例
  - 粘性：Sticky
    - 元素定位表现为在跨越特定阈值前为相对定位，之后为固定定位
    - 示例

- z-index
  - 元素位于Z轴上的位置
  - 数值越大，越在前面
  - 默认：auto（0）
  - 示例

- 响应式设计
  - 响应式网页设计（responsive web design，RWD）
    - 允许Web页面适应不同屏幕宽度因素等，进行布局和外观的调整
  - 媒体查询
    - 针对不同的媒体类型定义不同的样式


```
@media screen and (min-width: 800px) {
  .container {
    margin: 1em 2em;
  }
}
```

- 响应式设计
  - 百分比布局
  - rem布局
    - rem单位都是相对于根元素html的font-size来决定大小
    - 过js来动态控制根元素font-size的大小
  - 视口单位



    - 那么 1vw = 650 * 1% = 6.5px（这是理论推算的出，如果浏览器不支持0.5px，那么实际渲染结果可能是7px）

- 响应式设计
  - 响应式图像
    - 大小自适应
    - 使用max-width（图片自适应）

```
img {
    display: inline-block;
    max-width: 100%;
    height: auto;
}
```

    - 使用srcset

```
<img srcset="photo_w350.jpg 1x, photo_w640.jpg 2x" src="photo_w350.jpg" alt="">
```