UI - HTML-CSS-Javascript

Reading

课件以以下资料为基础，抓取了重难点来强调。讲课速度会很快。所以必须提前把reading熟读3遍以上。课后再熟读一遍。则能掌握的很夯实。对于学习要有敬畏之心，要坚持读够遍数。这是很高效省时间的学习方式。

It is recommended to go through the following links in the specified order for a comprehensive understanding.

- https://www.yuque.com/fairy-era/xurq2q/qflg1o
- https://www.yuque.com/fairy-era/xurq2q/yaa29o
- **CSS 基础**：https://www.yuque.com/fairy-era/xurq2q/bommr1#e1eef53c
- **CSS 进阶**：https://www.yuque.com/fairy-era/xurq2q/wiopgw
- **CSS 盒子模型**：https://www.yuque.com/fairy-era/xurq2q/mwwfum
- 浮动： https://www.yuque.com/fairy-era/xurq2q/drb5uu (不用看)

- **CSS 定位和装饰**：https://www.yuque.com/fairy-era/xurq2q/arvzq9
- https://www.yuque.com/fairy-era/xurq2q/vhhaiv#b5ee0e50
- https://www.w3schools.com/js/default.asp

Please make sure to read them in the listed order for better comprehension.

Practice：

- flex : https://flexboxfroggy.com/
- css: https://flukeout.github.io/

HTML Basic

HTML让鸟四肢健全

Reading

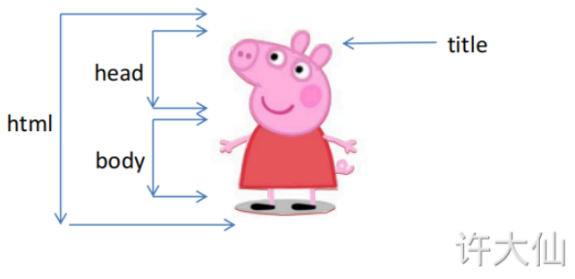
- https://www.yuque.com/fairy-era/xurq2q/qflg1o
- https://www.yuque.com/fairy-era/xurq2q/yaa29o

1.3 Web 标准

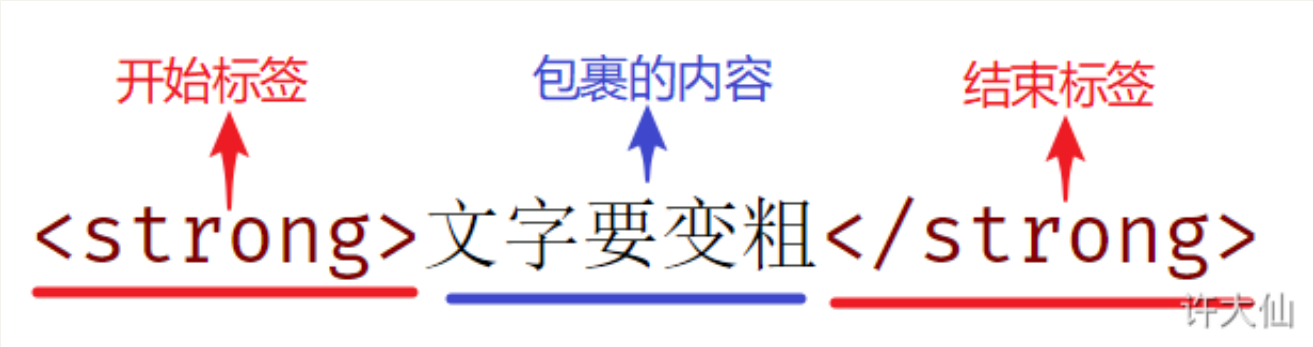
组成	语言	说明
结构	HTML	页面元素和内容
表现	CSS	网页元素的外观和位置等页面样式（如：文字大小、颜色等）
行为	JavaScript	网页模型的定义和页面交互

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>文字变粗</title>
</head>
<body>
    <strong>你猜，这段文字是加粗的吗？</strong>
</body>
</html>
```

```
<html>
  <head>
    <title>网页的标题</title>
  </head>
  <body>
    网页的主体内容
  </body>
</html>
```



3.2 HTML 标签的构成



hr是画一条线
单独符号，不需要close标签

3.3 HTML 标签的关系

- 父子关系（嵌套关系）

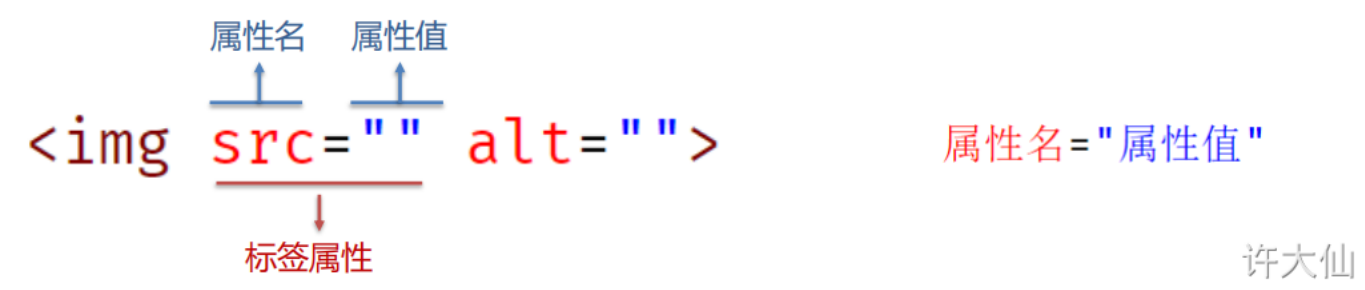
```
<head>
    <title>html的标题</title>
</head>
```

- 兄弟关系（并列关系）

```
<head></head>
<body></body>
```

4.3.1 图片标签

重点是理解标签有属性这个特点，以及属性的语法，以及了解常见属性。



- 属性的注意点：
  - ① 标签的属性写在 开始标签内部 。
  - ② 标签上可以同时存在多个属性。
  - ③ 属性之间以空格隔开。
  - ④ 标签名和属性之间 必须以空格隔开 。
  - ⑤ 属性之间没有顺序之分。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>图片标签</title>
</head>
<body>
    <img src="https://i0.wp.com/countryswag.com/wp-
content/uploads/2022/01/kane-brown-new-song.jpg?
resize=705%2C705&ssl=1" alt="So now I take my whiskey
sour,
Sittin' barside after hours" width="200">
</body>
</html>
```

- Question: html属性能设定多少事情？常见的HTML属性：
  - a. **class**：为元素指定一个或多个类名，通常用于配合 CSS 来设置样式。

b. **id**：为元素指定一个唯一的 id，这个 id 可以用于 JavaScript 操作或 CSS 样式设置。

c. **style**：直接在元素上应用 CSS 样式。尽管此属性可以使用，但一般建议将样式放在单独的 CSS 文件中。

d. **src**：通常用于 `<img>`, `<script>`, `<iframe>`, `<video>` 和 `<audio>` 等元素，指定资源的 URL。

e. **href**：在 `<a>` (链接) 元素中使用，指定链接的目标 URL。

f. **alt**：在 `<img>` 元素中使用，为图像提供一个替代的文本描述，当图像无法加载时显示。

g. **width** 和 **height**：设置元素的宽度和高度，常用于 `<img>`, `<table>`, `<td>`, `<th>`, `<video>`, 和 `<canvas>` 等元素。

h. **disabled**：禁用输入元素（如 `<input>`、`<button>`）。

i. **placeholder**：在 `<input>` 和 `<textarea>` 元素中使用，为用户输入提供一个提示。

j. **required**：在表单元素中使用，表示该字段是必填的。

k. **value**：在 `<input>`、`<button>`、和 `<option>` 元素中使用，指定元素的值。

l. **checked**：在 `<input type="checkbox">` 和 `<input type="radio">` 中使用，预先选择一个选项。

m. **selected**：在 `<option>` 元素中使用，预先选择一个选项。

4.4 超链接标签

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>超链接标签</title>
</head>
<body>
    <a href="https://www.baidu.com">百度一下，你就知道！</a> <br>
    <a href="./01 HTML 的概念.html">HTML 的概念</a>
</body>
</html>
```

超链接标签的 target 属性

- 属性名：target 。
- 属性值：目标网页的打开形式。

取值	说明
`_self`	默认值，在当前窗口中跳转（覆盖原网页）
`_blank`	在新窗口中跳转（保留原网页）

HTML Advanced

Reading

- https://www.yuque.com/fairy-era/xurq2q/yaa29o

1.2 无序列表

标签名	说明
ul	表示无序列表的整体，用于包裹 li 标签
li	表示无序列表的每一项，用于包含每一行的内容

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>无序列表</title>
</head>
<body>
    <h2>水果列表</h2>
    <ul>
        <li>榴莲</li>
        <li>香蕉</li>
        <li>苹果</li>
        <li>哈密瓜</li>
        <li>火龙果</li>
    </ul>
</body>
</html>
```

1.3 有序列表

标签名	说明
ol	表示有序列表的整体，用于包裹 li 标签
li	表示有序列表的每一项，用于包含每一行的内容

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>有序列表</title>
</head>
<body>
    <h2>成绩排行榜</h2>
    <ol>
        <li>张三：100分</li>
        <li>李四：80分</li>
        <li>王五：65分</li>
    </ol>
</body>
</html>
```

2.1 表格的基本标签

标签名	说明
table	表格整体，可以用于包裹多个 tr
tr	表格每行，可以用于包裹 td
td	表格单元格，可用于包裹内容

> *注意：标签的嵌套关系是 table > tr > td 。*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>表格标题和表头单元格标签</title>
</head>
<body>
    <table border="1" width="300" height="200">
        <!-- 表格标题：用于表示表格整体大标题。 -->
        <caption>学生成绩表</caption>
        <tr>
            <!-- 表头单元格：表示一列小标题。 -->
            <th>姓名</th>
            <th>成绩</th>
            <th>评语</th>
        </tr>
        <tr>
            <td>张三</td>
            <td>100</td>
            <td>非常好</td>
        </tr>
        <tr>
            <td>李四</td>
            <td>50</td>
            <td>还行</td>
        </tr>
    </table>
</body>
</html>
```

3.1 input 系列标签

- 场景：在网页中显示收集用户信息的表单效果，如：登录页、注册页。
- 标签名：input（input 标签可以通过 type 属性值的不同，展示不同的效果）。
- type 属性值：

标签名	TYPE 属性值	说明
input	text	文本框，用于输入单行文本
input	password	密码框，用于输入密码
input	radio	单选框，用于多选一
input	checkbox	多选框，用于多选多
input	file	文件选择，用于上传文件
input	submit	提交按钮，用于提交
input	reset	重置按钮，用于重置
input	button	普通按钮，默认无功能，需要配合 js 实现功能

input属性比较多，使得input非常丰富。对于学习者来说，只需要知道input的丰富的功能，都是通过属性来控制的。
html的难点就是属性。阅读代码时候多思考属性。

对于fullstack 来说，不用记忆各个属性，但是要有往属性上思考的意识。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>input-按钮</title>
</head>
<body>
    <form action="#" method="POST">
        用户名：<input type="text" name="username" id="" placeholder="请输入用户名"> <br>
        密码：<input type="password" name="password" id="" placeholder="请输入密码"> <br>
        <!-- 按钮 -->
        <input type="submit" value="提交">
```

```
        <input type="reset" value="重置">
        <input type="button" value="普通按钮">
    </form>
</body>
</html>
```

3.3 select 下拉菜单标签

- 场景：在网页中提供多个选择项的下拉菜单表单控件。
- 标签组成：
- - select 标签：下拉菜单的整体。
- - option 标签：下拉菜单的每一项。
- 常见属性：selected，表示下拉菜单的默认选中项。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>下拉菜单</title>
</head>
<body>
    所属城市：<select name="city" id="">
                <option value="北京">北京</option>
                <option value="天津">天津</option>
                <option value="南京" selected>南京</option>
                <option value="广州">广州</option>
            </select>
</body>
</html>
```

4.1 没有语义的布局标签（ div 和 span）

- 场景：实际开发网页的时候，会大量频繁的使用到 div 和 span 标签。
- div 标签：一行只显示一个（独占一行）。
- span 标签：一行可以显示多个。

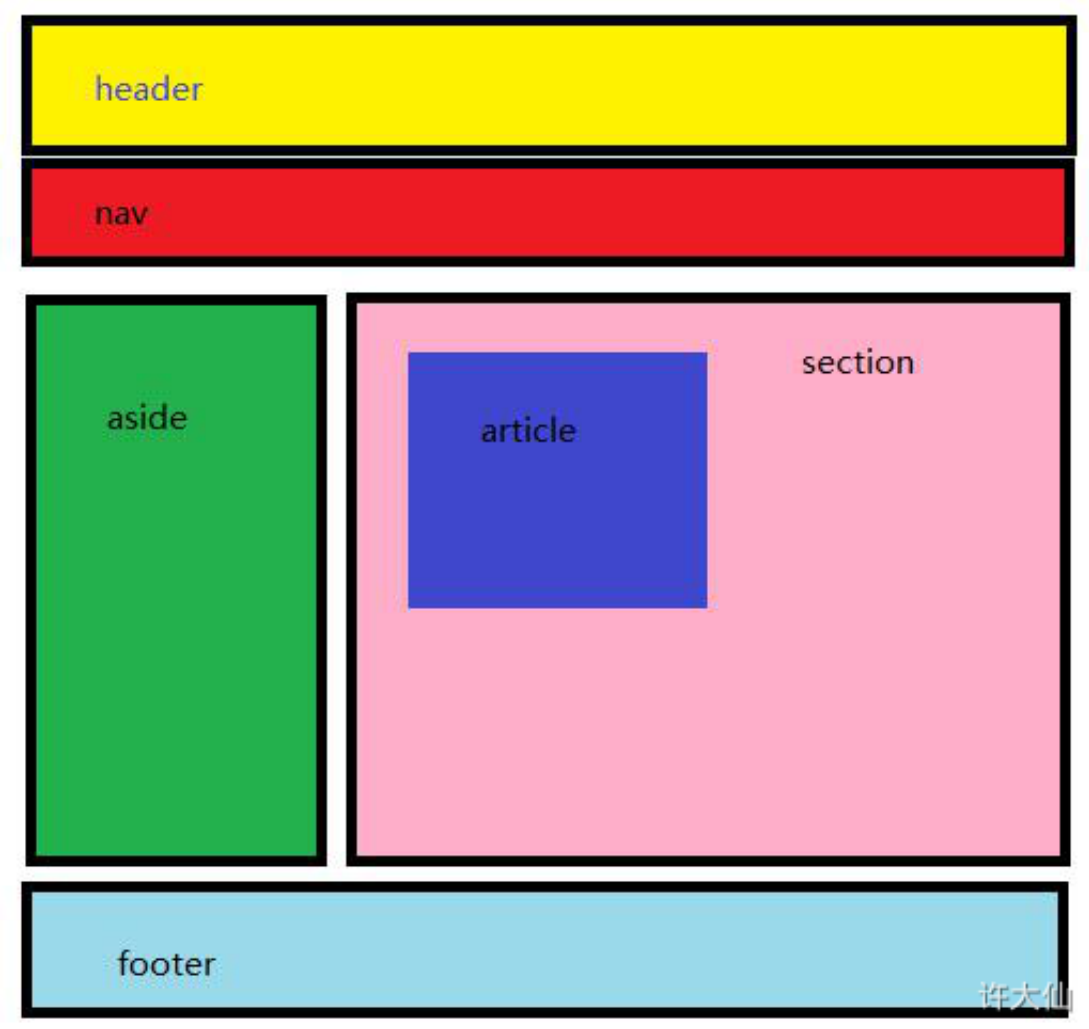
> *因为网页都是有layout的，所以就经常需要借助于di v来做layout。*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>div 和 span 标签</title>
</head>
<body>
    普通文字
    <div>这是 div 标签</div>
    <div>这是 div 标签</div>

    普通文字
    <span>这是 span 标签</span>
    <span>这是 span 标签</span>
</body>
</html>
```

4.2 有语义化的标签（了解）

标签名	语义
header	网页头部
nav	网页导航
footer	网页底部
aside	网页侧边栏
section	网页区块
article	网页文章

> *注意：以上标签显示特点和 div 一致，但是比 div 多了不同的语义。*




```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>有语义化的标签</title>
</head>
<body>
    <header>网页头部</header>
    <nav>网页导航</nav>
    <aside>网页侧边栏</aside>
    <section>网页区块</section>
    <article>网页文章</article>
    <footer>网页底部</footer>
</body>
</html>
```

CSS

CSS 让鸟儿漂亮

Question: CSS 怎么让网站漂亮?

Reading

- **CSS 基础**：https://www.yuque.com/fairy-era/xurq2q/bommr1#e1eef53c
- **CSS 进阶**：https://www.yuque.com/fairy-era/xurq2q/wiopgw
- **CSS 盒子模型**：https://www.yuque.com/fairy-era/xurq2q/mwwfum
- 浮动： https://www.yuque.com/fairy-era/xurq2q/drb5uu (不用看)
- **CSS 定位和装饰**：https://www.yuque.com/fairy-era/xurq2q/arvzq9

1.5 CSS 引入方式

CSS can be added to HTML documents in 3 ways:

- 内嵌式：CSS 写在 style 标签中。
  - **Internal** - by using a `<style>` element in the `<head>` section
  - > *注意：style 标签虽然可以写在页面的任何位置，但是通常约定写在 head 标签中。约定大于配置。*
- 外联式：CSS 写在一个单独的 `.css` 文件中。
  - **External** - by using a `<link>` element to link to an external CSS file
  - > *注意：需要通过 link 标签在网页中引入。*
- 行内式：CSS 写在标签的 style 属性中
  - **Inline** - by using the `style` attribute inside HTML elements
  - > *注意：这里不推荐，但是之后会配合 js 使用。*

Internal

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        p {
            /* 文字颜色 */
            color: red;
            /* 文字大小 */
            font-size: 30px;
            /* 背景颜色 */
            background-color: pink;
            /* 宽度 */
            width: 300px;
            /* 高度 */
            height: 300px;
        }
    </style>
</head>

<body>
    <p>你好，世界</p>
</body>

</html>
```

External

- 示例：外联式

CSS | Copy

```css
p {
    /* 文字颜色 */
    color: red;
    /* 文字大小 */
    font-size: 30px;
    /* 背景颜色 */
    background-color: pink;
    /* 宽度 */
    width: 300px;
    /* 高度 */
    height: 300px;
}
```

HTML | Copy

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <!-- 样式表 -->
    <link rel="stylesheet" href="./css/style.css">
</head>
<body>
    <p>你好，世界</p>
</body>
</html>
```

Inline

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>行内式</title>
</head>

<body>
    <p style="color: red;background-color: pink;font-size: 30px;width: 300px;height: 300px;">你好，世界</p>
</body>

</html>
```

基础选择器

选择器的作用

> *选择器的作用：选中页面中对应的标签，方便后续设置样式。*

Question:

- 怎么把

  的文字变成红色?
- 怎么把

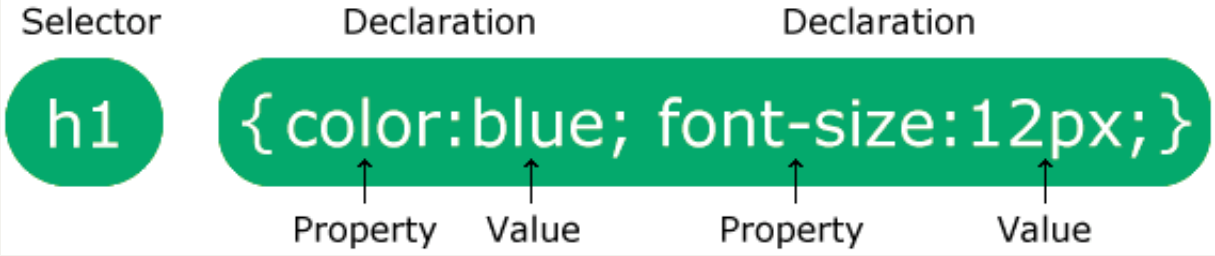
  里的

  变成背景变成黄色。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
```

```
    <style>
        /* 选择器 {} */
        /* 标签选择器就是以标签名命名的选择器 */
        /* 标签选择器会选中所有的标签，都生效 css */
        p {
            color: red;
        }
    </style>
</head>

<body>
    <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Vitae minima totam non consequatur officiis quo
        voluptatibus, ea ipsa nulla, cumque magni. Minus maiores temporibus cupiditate illo quaerat modi expedita
        facilis?</p>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Libero minima dolorum velit, labore incidunt aut architecto ducimus molestias omnis? Aliquid asperiores eveniet quibusdam impedit, facilis dolor id voluptatem! Numquam, maiores!</p>
    <div>
        <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Fuga, nisi adipisci. Esse ea inventore dolorem voluptas pariatur, suscipit, quam repellat animi eaque aperiam at alias nesciunt! Facere qui ut possimus?</p>
    </div>
</body>

</html>
```

标签选择器



- 结构：`标签名 {css属性名: css属性值; }`。
- 作用：通过标签名，找到页面中所有的这类标签，设置样式。

> *注意⚠*
>
> - *标签选择器选择的是一类标签，而不是单独的某一个*。
> - *标签选择器无论嵌套关系有多深，都能找到对应的标签*。

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        /* 选择器 {} */
        /* 标签选择器就是以标签名命名的选择器 */
        /* 标签选择器会选中所有的标签，都生效 css */
        p {
            color: red;
        }
    </style>
</head>

<bod>
```

```
20      <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit.
    Vitae minima totam non consequatur officiis quo
21          voluptatibus, ea ipsa nulla, cumque magni. Minus maiores
    temporibus cupiditate illo quaerat modi expedita
22          facilis?</p>
23      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    Libero minima dolorum velit, labore incidunt aut architecto
    ducimus molestias omnis? Aliquid asperiores eveniet quibusdam
    impedit, facilis dolor id voluptatem! Numquam, maiores!</p>
24      <div>
25          <p>Lorem ipsum dolor sit amet consectetur, adipisicing
    elit. Fuga, nisi adipisci. Esse ea inventore dolorem voluptas
    pariatur, suscipit, quam repellat animi eaque aperiam at alias
    nesciunt! Facere qui ut possimus?</p>
26      </div>
27  </body>
28
29  </html>
```

2.3 类选择器

- 结构：`.类名{css属性名: css属性值;}`。
- 作用：通过类名，找到页面中所有带这个类名的标签，设置样式。

> *注意：*
>
> - *① 所有标签上都有 class 属性，class 属性的属性值称为 `类名（类似于名字）`。*
> - *② 类名可以由数字、字母、下划线、中划线组成，但是不能以数字或中划线开头。*
> - *③ 一个标签可以同时有多个类名，类名之间以空格隔开。*
> - *④ 类名可以重复，一个类选择器可以同时选中多个标签。*

```
1   <!DOCTYPE html>
```

```
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        /* 类选择器 */
        /* .类名 {} */
        .two {
            color: red;
        }

        .size {
            font-size: 30px;
        }
    </style>
</head>

<body>
    <p>11111</p>
    <!-- 类：定义 和 使用 才能生效 -->
    <!-- 一个标签可以使用多个类名，需要空格隔开 -->
    <p class="two size">22222</p>
    <p>33333</p>
</body>

</html>
```

2.4 id 选择器

- 结构：`#id属性值{css属性名：属性值;}`。
- 作用：通过 id 属性值，找到页面中所有带这个 id 属性值的标签，设置样式。

> *注意：*
>
> - *① 所有标签上都有 id 属性。*
> - *② id 属性值类似于身份证号码，在一个页面中是唯一的，不可重复的。*
> - *③ 一个标签上只能有一个 id 属性值。*
> - *④ 一个 id 选择器只能选中一个标签。*

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        /* id 选择器 */
        /* #id属性值 {} */
        #two {
            color: red;
            font-size: 12px;
        }
    </style>
</head>

<body>
    <p>11111</p>
    <p id="two">22222</p>
    <p>33333</p>
```

```
</body>

</html>
```

2.5 通配符选择器

- 结构：`*{css属性名：属性值;}`。
- 作用：找到页面中所有的标签，设置样式。

> *注意：*
>
> - *开发中使用极少，只会在极其特殊的情况下才会使用。*
> - *后续会用来去除标签默认的 margin 和 padding*

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        * {
            color: red;
        }
    </style>
</head>

<body>
    <div>div</div>
    <p>p</p>
    <h1>h1</h1>
```

```
    <span>span</span>
</body>

</html>
```

4.1 颜色常见取值（了解）

- 属性名:
-
    - 如：文字颜色 `color` 。
-
    - 如：背景颜色 `background-color` 。
- 属性值:

颜色表示方式	表示含义	属性值
关键词	预定义的颜色名	red、green、blue、yellow.....
rgb表示法	红绿蓝三原色。每项取值范围:0~255	rgb(0,0,0)、rgb(255,255,255)、rgb(255,0,0).....
rgba表示法	红绿蓝三原色+a表示透明度，取值范围是:0~1	rgba(255,255,255,0.5)、rgba(255,0,0,0.3).....
十六进制表示法	#开头，将数字转换成十六进制表示	#000000、#ff0000、#e92322，简写:#000、#f00

```
p {
    color: red;
}

p {
    color: rgb(255, 0, 0);
}

p {
    color: rgba(255, 0, 0, 0.5);
```

```
}

p {
    color: #FF0000;
}
```

1.1 复合选择器

1.1.1 后代选择器

- 作用：根据 HTML 标签的嵌套关系，选择父元素 `后代中` 满足条件的元素。
- 选择器的语法：

```
选择器1 选择器2 { ... }
```

- 结果：在 `选择器1` 所找到的标签的后代（儿子、孙子、重孙子.....）中，找到满足 `选择器2` 的标签，设置样式。

> *注意：*
>
> - *① 后代包括：儿子、孙子、重孙子.....*
> - *② 后代选择器中，选择器和选择器之间通过 `空格` 隔开。*

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
```

```
    <style>
        /* 后代选择器 */
        /* 找到 div 标签中的 p 标签，设置文字颜色为红色 */
        div p {
            color: red;
        }
    </style>
</head>

<body>
    <!-- 后代包括：儿子、孙子、重孙子…… -->
    <p>这是一个 p 标签</p>
    <div>
        <p>这是 div 中的 p 标签</p>
    </div>
</body>

</html>
```

1.1.2 子代选择器

```
选择器1 > 选择器2 { ... }
```

作用：根据 HTML 标签的嵌套关系，选择父元素 `子代中` 满足条件的元素。

> *注意：*
>
> - *① 子代只包括：儿子。*
> - *② 子代选择器中，选择器和选择器之间通过 `>` 隔开*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        /* 子代选择器 */
        div > a {
            color: red;
        }
    </style>
</head>
<body>
    <a href="#">我是 a 标签</a>
    <div>
        父级
        <a href="#">我是 div 中的 a 标签</a>
        <p>
            <a href="#">我是 div 中的 p 中的 a 标签</a>
        </p>
    </div>
</body>
</html>
```

Question: 子代选择器和后代选择器的区别?

1.2 并集选择器

```
选择器1 , 选择器2 { ... }
```

作用：同时选择多组标签，设置相同的样式

> *注意：*
>
> - *① 并集选择器中的每组选择器之间使用 `,` 分隔。*

> - *② 并集选择器中的每组选择器可以是基础选择器或者复合选择器。*
> - *③ 并集选择器中的每组选择器通常一行写一个，提高代码的可读性。*

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        /* 并集选择器 */
        p,
        div {
            color: red;
        }
    </style>
</head>

<body>
    <p>ppp</p>
    <div>div</div>
    <span>span</span>
    <h1>h1</h1>
</body>

</html>
```

1.3 交集选择器

作用：选中页面中 `同时满足` 多个选择器的标签。

> *注意：*

> - *① 交集选择器中的选择器之间是紧挨着的，没有东西分隔。*
> - *② 交集选择器如果有标签选择器，标签选择器必须写在最前面。*

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        p.box {
            color: red;
        }
    </style>
</head>

<body>
    <!-- 找到第一个 p ，带 box 类的，设置文字颜色为红色 -->
    <p class="box">ppppp:box</p>
    <p>pppppp</p>
    <div class="box">div:box</div>
</body>

</html>
```

3.4 元素显示模式转换

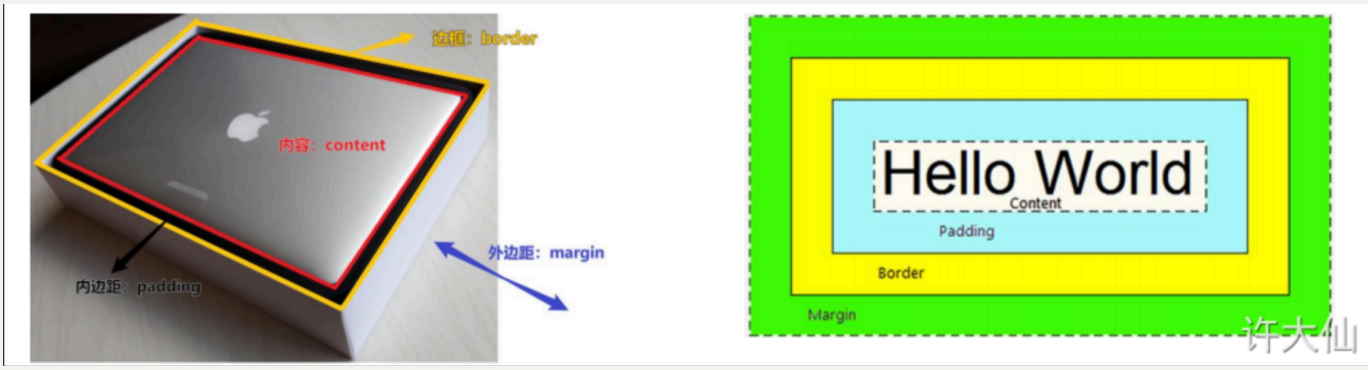
目的：改变元素默认的显示特点，让元素符合布局要求。其实就是改变它是否要占满一整行。

属性	效果	使用评率
`display: block;`	转换为块级元素	较多
`display: inline-block;`	转换为行内块元素	较多
`display: inline;`	抓换为行内元素	较少

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        div {
            /* 转换为行内块元素 */
            display: inline-block;
            width: 200px;
            height: 200px;
            background-color: pink;
        }
    </style>
</head>

<body>
    <div>11111</div>
    <div>22222</div>
</body>

</html>
```

Boxing Model

CSS 中规定每个盒子分别由：内容区域（content）、内边距区域（padding）、边框区域（border）、外边距区域（margin）构成，这就是 `盒子模型` 。




```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        div {
            width: 200px;
            height: 200px;
            background-color: pink;
            /* 边框，就相当于纸箱子 */
            border: 1px solid #000;
            /* 内边距，相当于填充泡沫，出现在内容和盒子边缘之间 */
            padding: 20px;
            /* 外边距，盒子和盒子之间的距离 */
            margin: 50px;
        }
    </style>
</head>
```

```
23
24 <body>
25     <div class="box">内容：笔记本电脑</div>
26     <div class="box">内容：笔记本电脑</div>
27 </body>
28
29 </html>
```

Box Sizing

`box-sizing` 是一个 CSS 属性，它决定了如何计算一个元素的总宽度和总高度。

在 CSS 中，一个元素的盒模型（Box Model）包括内容（content）、填充（padding）、边框（border）和外边距（margin）。这四个部分从内到外构成了元素的完整框架。

默认的 `box-sizing` 值是 `content-box`。当设置一个元素的 `width` 和 `height` 时，你只是设置了内容区的宽度和高度。如果你添加了 `padding` 或 `border`，它们会添加到总的宽度和高度上。这意味着元素的实际大小可能会超过你设置的 `width` 和 `height`。

```
1 div {
2     box-sizing: content-box;
3     width: 300px;
4     padding: 10px;
5     border: 5px solid black;
6 }
```

在上面的例子中，`<div>` 的总宽度实际上是 330px（300px 的内容宽度 + 20px 的填充 + 10px 的边框）。

如果你想让设置的 `width` 和 `height` 包含 `padding` 和 `border`，你可以将 `box-sizing` 设置为 `border-box`。

```css
div {
    box-sizing: border-box;
    width: 300px;
    padding: 10px;
    border: 5px solid black;
}
```

在这个例子中，`<div>` 的总宽度就是 300px，这是因为它包含了 260px 的内容宽度 + 20px 的填充 + 10px 的边框。

`box-sizing` 属性的这种特性在你需要精确控制元素大小时非常有用，尤其是在布局设计中。

Flexbox & Grid

Grid不推荐： https://www.w3schools.com/css/css_grid.asp

Flexbox 推荐：https://www.w3schools.com/css/css3_flexbox.asp

Bootstrap & Material UI

- https://www.w3schools.com/bootstrap5/bootstrap_carousel.php
- https://getbootstrap.com/docs/5.3/components/card/
- https://mui.com/material-ui/react-card/

Bootstrap 是一款开源的前端开发框架，它提供了许多预设的 CSS 样式、布局模板和交互式的 JavaScript 组件。Bootstrap 能够帮助开发者更快速地创建响应式、移动优先的网站。

Bootstrap 有以下几个主要的特点：

- 栅格系统：Bootstrap 提供了一个 12 列的灵活栅格系统，使得布局设计变得简单且直观。

- 预设的 **CSS** 样式：Bootstrap 预设了许多 CSS 样式，包括排版、表单、按钮、图像、导航等。
- 组件：Bootstrap 提供了许多预设的组件，如下拉菜单、导航条、警告框、弹出框等。
- **JavaScript** 插件：Bootstrap 提供了一些基于 jQuery 的 JavaScript 插件，可以添加更多的交互性。
- 响应式设计：Bootstrap 设计的所有组件都是响应式的，能够在不同的设备和屏幕尺寸上良好地展现。

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" rel="stylesheet">

    <title>Hello, Bootstrap!</title>
  </head>
  <body>
    <div class="container">
      <h1>Hello, Bootstrap!</h1>

      <div class="alert alert-success" role="alert">
        This is a success alert—check it out!
      </div>

      <button class="btn btn-primary">Primary Button</button>
    </div>

```

```
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
  </body>
</html>
```

可以认为bootstrap是CSS的frameworks。我们可以用该框架提供好的样式。

Material-UI 是Google出的CSS frameworks

JavaScript Basic

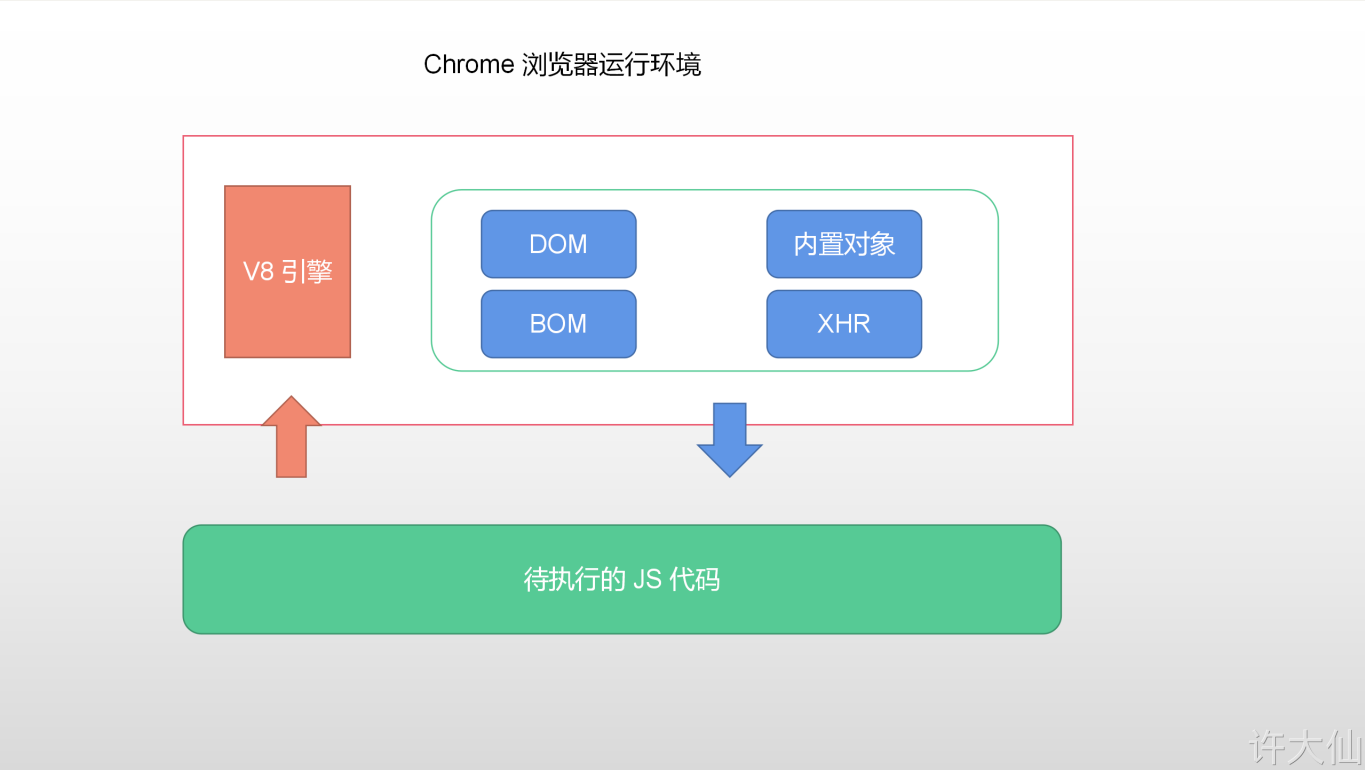
Reading

- https://www.yuque.com/fairy-era/xurq2q/vhhaiv#b5ee0e50
- https://www.w3schools.com/js/default.asp

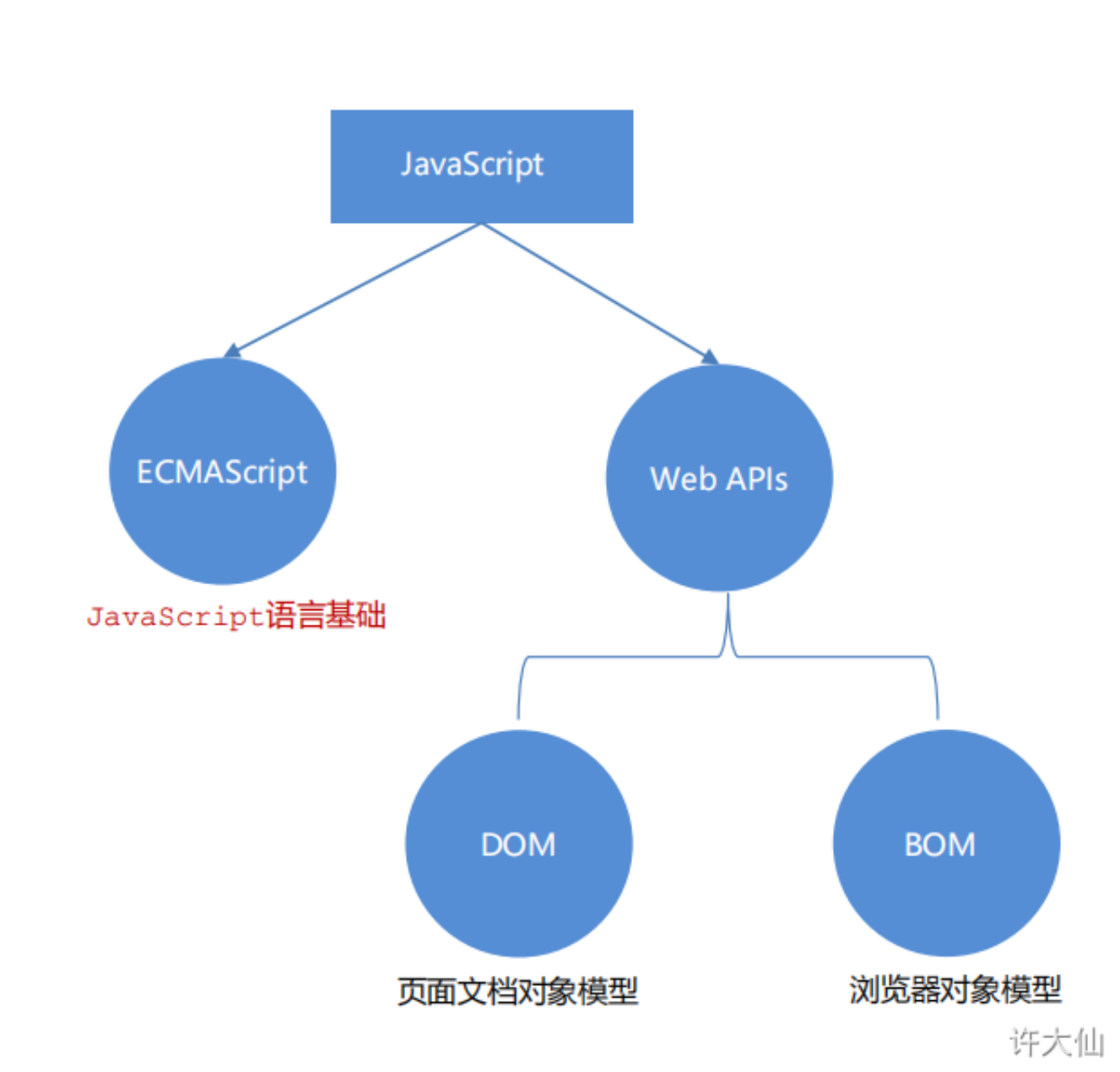
1.1 JavaScript 是什么?

- JavaScript 是一种运行在 `客户端（浏览器）` 的编程语言，实现 `人机交互效果` 。
- JavaScript 的作用：
    - ① 网页特效（监听用户的一些行为让网页作出对应的反馈）。
    - ② 表单验证（针对表单数据的合法性进行判断）。
    - ③ 数据交互（获取后台的数据，渲染到前端）。

- - ④ 服务端编程（node.js）。



- JavaScript 组成：
- - ECMAScript：规定了 JS 的基本语法，如：变量、分支语句、循环语句、对象等。
- - WebAPI：
- - - DOM：操作文档，如：对页面元素进行移动、删除等操作。
- - - BOM：操作浏览器，如：页面弹窗、检测窗口宽度、存储数据到浏览器等。



2.5 let 和 var 的区别

- let 是为了解决 var 的一些问题。
- var 声明：
-
    - 可以先使用再声明（不合理）。
-
    - var 声明过的变量可以重复声明（不合理）。
-
    - var 声明的变量有变量提升、全局变量、没有块级作用域等缺点。
- `总结：以后声明变量统一使用 let` 。

- JavaScript 中的数据类型分类：
    - ① 基本数据类型：
        - number：数字类型。
        - string：字符串类型。
        - boolean：布尔类型。
        - undefined：未定义类型。
            - 未定义类型是一个比较特殊的类型，只有一个值 undefined 。
            - 未定义类型：只声明变量，不赋值的情况下，变量的默认值就是 undefined 。
        - null：空类型。
            - undefined 表示没有赋值。
            - null 表示赋值了，但是内容为空。
    - ② 引用数据类型。
        - object：对象类型。
        - function：函数类型。
        - array：数组类型。

== vs. ===

`==`：左右两边是否相等

`===`：左右两边是否类型和值都相等

函数

```
具名函数
// 声明
function fn(){}
// 调用
fn()

  vs.
匿名函数：将匿名函数赋值给一个变量，并且通过变量名称进行调用，也称为函数表达式。
// 声明
let fn = function (){}
// 调用
fn()
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta content="IE=edge" http-equiv="X-UA-Compatible">
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <title>Title</title>
</head>
<body>
    <script>
        let fn = function (num1, num2) {
            return num1 + num2;
        };

        let res = fn(1, 2);
        console.log(res);

    </script>
</body>
</html>
```

6.2 立即执行函数（IIFE）

```
// 方式1
(function(){
    // 函数体
})();

// 方式2
(function(){
    // 函数体
} ());
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta content="IE=edge" http-equiv="X-UA-Compatible">
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <title>Title</title>
</head>
<body>
    <script>
        /* 立即执行函数：立即执行，无需调用 */
        // let fn = function (){}
        // fn()
        (function (num1, num2) {
            console.log(num1 + num2);
        })(1, 2);
    </script>
</body>
</html>
```

对象

> *可以用Java 的class的思想来理解这部分*

- 对象（object）：JavaScript 中的一种数据类型。
- 对象可以理解是一种无序的数据集合。
- 对象是用来描述某个事物，例如：描述一个人。
    - 人有姓名、年龄、性别等信息，还有吃饭、睡觉、写代码等功能。
    - 如果使用多个变量保存会比较松散，不利于管理；而使用对象保存，则会比较统一，且方便管理。
- 例如：描述 `人` 的信息：
    - 静态特征（类似于Java 的fields）：
        - 姓名 -- 字符串类型
        - 年龄 -- 数字类型
        - 身高 -- 数字类型
        - 性别 -- 字符串类型
        - 爱好 -- 数组类型
    - 动态行为（类似于Java的方法）：
        - 吃饭 -- 函数
        - 睡觉 -- 函数
        - 写代码 -- 函数

语法

```
let 对象名 = {
    属性名：属性值,
    方法名：函数
}
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta content="IE=edge" http-equiv="X-UA-Compatible">
    <meta content="width=device-width, initial-scale=1.0" name="viewport">
    <title>Title</title>
</head>
<body>
    <script>
        /* 对象的属性和方法 */
        let person = {
            name: '许大仙',
            age: 18,
            sex: '男',
            run: function () {
                console.log('跑路');
            }
        };
        /* 属性访问 */
        console.log(person.name); // 许大仙
        console.log(person.age); // 18
        console.log(person['sex']); // 男
        /* 方法调用 */
        person.run(); // 跑路
    </script>
</body>
</html>
```

这里的run方法。可以理解为前面学的。

```
run: function () {
                console.log('跑路');
            }
--->
let run = function () {
                console.log('跑路');
            }
--->
思考一下这里还可以怎么简化写?
```