软件工程基础 第三次实验

前端入门基础

实验大致内容

本次实验需要同学们学习 HTML 与 JS 的基础知识，并对 CSS 有一定的认识。

由于课程时间限制，介绍的内容非常有限，主要是为后边使用 Vue 做准备。

HTML 入门

HTML 是什么

HTML 指的是超文本标记语言 (Hyper Text Markup Language)。

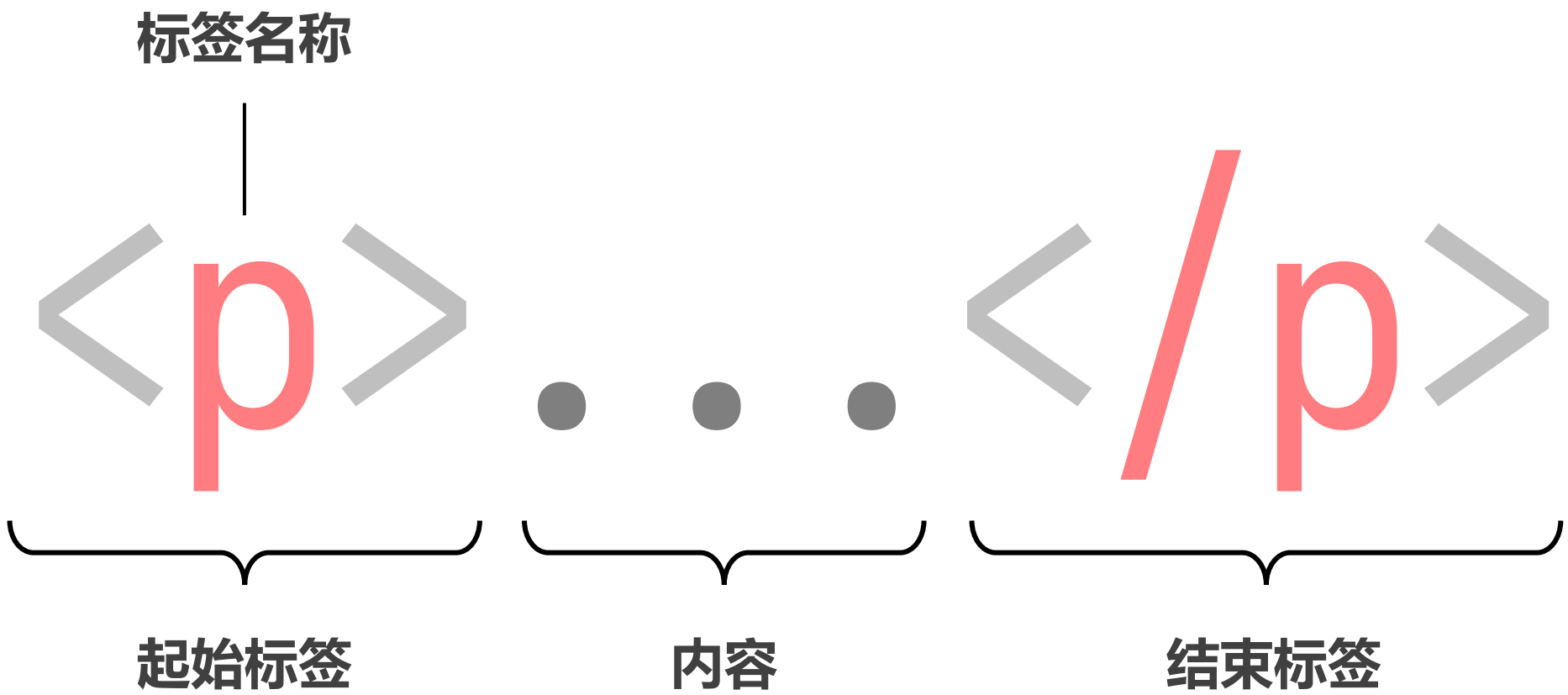
HTML 文档描述网页。

HTML 使用元素来描述页面


```
<html>
    <body>
        <h1>This is a title. </h1>
        <p>这是 p 标签，用于包裹一个段落的内容。</p>
    </body>
</html>
```

标签就相当于一个容器，成对标签之间包裹着标签中的内容。

标签的结构



标签的特性（属性）


```
<p style="color:red">...</p>
```

属性名　　属性值

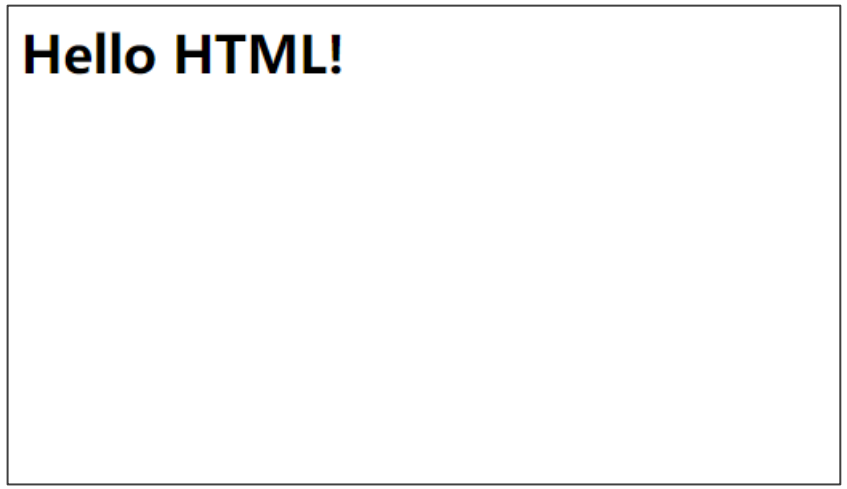
属性描述了元素的附加信息。

常用的标签（1） body head

<body>　　里面的内容会显示在页面中

<head>　　里面包含着页面的一些信息

```
<html>
    <head></head>
    <body>
        <h1>Hello HTML!</h1>
    </body>
</html>
```



常用的标签（2） h1 - h6


```
<html>
    <head></head>
    <body>
        <h1>Hello HTML!</h1>
        <h2>Hello HTML!</h2>
        <h3>Hello HTML!</h3>
        <h4>Hello HTML!</h4>
        <h5>Hello HTML!</h5>
        <h6>Hello HTML!</h6>
    </body>
</html>
```

<h1> 一级标题　　<h2> 二级标题

<h3> 三级标题　　<h4> 四级标题

<h5> 五级标题　　<h6> 六级标题

常用的标签（3） ol ul li

```
<ol>
    <li>有序列表项目1</li>
    <li>有序列表项目2</li>
    <li>有序列表项目3</li>
</ol>

<ul>
    <li>无序列表项目1</li>
    <li>无序列表项目2</li>
    <li>无序列表项目3</li>
</ul>
```

 有序列表

 无序列表

 列表项目

1. 有序列表项目1
2. 有序列表项目2
3. 有序列表项目3

- 无序列表项目1
- 无序列表项目2
- 无序列表项目3

常用的标签（4）

<a> 链接　　href 属性：点击后打开的网页地址

```
<a href="http://www.baidu.com">点击打开百度</a>
```

点击打开百度

可以使用相对URL，类似于Linux路径索引

img 标签是单标签，不需要成对出现

<img> 图片　　src 属性：图片的地址

```
<img src="https://7n.w3cschool.cn/statics/images/logonew2.png" />
```

可以使用相对URL

常用的标签（5） table tr th td

```
<table>
    <tr>
        <th></th>
        <th>星期六</th>
        <th>星期日</th>
    </tr>
    <tr>
        <th>门票售出量</th>
        <td>120</td>
        <td>135</td>
    </tr>
    <tr>
        <th>销售额</th>
        <td>$600</td>
        <td>$675</td>
    </tr>
</table>
```

<table>	**表格**
<tr>	**表格行**
<th>	**单元格（标题）**
<td>	**单元格**

	星期六	星期日
门票售出量	120	
销售额	$600	$675

常用的标签（6） input textarea

<input>　　**输入控件**

type 属性：　　输入控件的类型

text:	普通文本输入	file:	文件
password:	密码	radio:	单选框
submit:	提交按钮	checkbox:	多选框

name 属性：　　输入控件的名称，会与输入控件的内容一起提交到服务器

value 属性：　　输入控件的默认内容

placeholder 属性：输入控件的空白提示词

<textarea>　　**文本域**

name, value, placeholder ...

常用的标签（7） select option form

<select>　　**下拉选择框**

<option>　　**下拉选择选项**

value 属性：选项值

```
<select>
    <option value="0">请选择内容</option>
    <option value="1">第一个选项</option>
    <option value="2">第二个选项</option>
</select>
```

<form> **表单。里面包含有其他表单控件，点击里面的提交按钮之后表单会被提交。**

action 属性：提交发生时表单发往的 URL

method 属性：发送表单的 HTTP 方法，例如 get

注释

<!-- 注释的内容 -->

CSS 介绍

CSS 是什么

CSS 是一种描述 HTML 文档样式的语言。

CSS 描述应该如何显示 HTML 元素。

CSS 简介

CSS 用于控制网页的样式，利用规则规定 HTML 的元素应该如何显示

在 HTML 中引入 CSS 样式

使用外部样式

```
<html>
    <head>
        <link href="css/style.css" type="text/css" rel="stylesheet" />
    </head>
    <body></body>
</html>
```

CSS 文件的 URL

```
body, a{
    overflow-x: overlay;
    padding: 0 !important;
}

.hide_scroll{
    overflow: hidden !important;
}
```

一个 CSS 文件由多条规则组成

在 HTML 中引入 CSS 样式

使用内部样式

```
<html>
    <head>
        <style>
            body, a {
                padding: 0;
            }
        </style>
    </head>
    <body></body>
</html>
```

CSS 在页面的哪里引入或定义都没有特别大的限制，

但是 CSS 引入的顺序可能会影响样式之间作用的优先级

```
<a style="text-decoration: none;">example</a>
```

可以直接用 style 属性对一个元素规定其特定的布局

CSS 选择器

元素选择器（类型选择器）

按照标签类型选定

```
h1      a      input
```

类选择器

利用 HTML 的 class 属性定义类，然后对其选择

```
.highlight    .spinner
```

```
<a class="highlight small">example</a>
```

多个 class 可以用空格分开

同时满足多个条件

不加空格使用选择器可以取多个选择器选择出来的交集

```
p.highlight
```

class 属性包含有 highlight 的 p 元素

CSS 选择器

北京航空航天大学 软件学院

ID 选择器

```
<p id="introdution">example</p>
```

利用 HTML 的 id 属性定义 ID，然后对其选择

一个元素只能有一个 ID

一个 HTML 文档中不能有多个元素拥有同一个 ID

```
#mid
#introduction
```

后代选择器

在子元素中继续进行选择，用空格来分开

```
p #introduction
.highlight a
```

通用选择器

选择所有的元素

```
*
```

浅谈 CSS 规则优先级

CSS 规则的优先级有一套详细的法则，这里列出大部分适用的法则：

1. **直接在 HTML style 属性定义的规则优先；**
2. **选择器描述详细的规则优先；**
3. **越靠后的规则优先。**

使用 important 标记值

被 important 标记的规则都会被特殊对待，这些规则比任何没有被 important 标记的规则优先级都更高，被 important 标记的规则内部也遵循上述的优先级准则。

```
padding: 0 !important;
```

在属性值的最后使用 important 标记

div、span 与盒子模型

div 划分出一个块级区域，span 划分出一个行内区域。

```
<div></div>

<span></span>
```

盒子模型把 HTML 的每一个元素都看作一个盒子：

通过控制盒子的属性以及位置，CSS 可以操作每一个元素的样式，从而设定页面布局。

JavaScript 基础与 DOM

JavaScript 是什么

JavaScript 是属于 HTML 和 Web 的编程语言。

JavaScript 能够改变 HTML 内容、属性、样式。

JavaScript 变量

定义一个变量

```
var a = 10;
```

三种常用的数据类型

1. Number：数字数据类型

   10　　0.75　　-1

2. String：字符串数据类型

   'string'　　"JavaScript"　　' content '

3. Boolean：布尔数据类型

   true　　false

特殊量

1. undefined：未赋值变量
2. NaN：无法计算的数值
3. Infinity：非常大的数值

字符串操作

1. 拼接：'a'+'b'
2. 比较：'a'>'b'
3. 访问：'a'.charAt(0) 或 'a'[0]
4. 长度：'a'.length

JavaScript 毒瘤：==


```
'' == '0'                 // false
0 == ''                   // true
0 == '0'                  // true

false == 'false'          // false
false == 0                // true

null == undefined         // true

'\n' == 0                 // true
'\t' == 0                 // true
'\r\n' == 0               // true

0.1 + 0.2 == 0.3          // false
```

类型问题的可以用 === 和 !== 解决

JavaScript 对象、函数

定义一个对象

```
var object = {
    name: 'JavaScript',
    age: 15,
    'like JavaScript': true
};
```

访问对象的属性

```
object.name;

object['like JavaScript'];

object.sex;      // undefined
```

万物皆为对象

定义一个函数

```
function add(a, b) {
    return a + b;
}

var add = function(a, b) {
    return a + b;
};
```

匿名函数

```
function(a, b) {
    return a + b;
}
```

调用一个函数

```
add(0, 1);
```

```
var object = {
    func1: function() {
        ...
    },

    func2() {
        ...
    }
};
```

JavaScript 数组

定义一个数组

```
var arr = [1, 'JavaScript', {}];
```

```
var arr = new Array(10);
```

数组本质上也是一个对象

数组的操作

访问数组元素

```
arr[1];
```

添加数组元素

```
arr.push(0.6);
```

寻找数组元素

```
arr.indexOf(1);
```

将数组排序

```
arr.sort(function(a, b) {
    return a - b;
});
```

拼接两个数组

```
arr = arr.concat([2, 3]);
```

长度

```
arr.length;
```

JavaScript 语句

for

while

if - else if - else

switch - case

continue

break

return

try - catch

throw

... ? ... : ...

```
flag?'red':'blue'
```

函数调用模式与 this 变量（1）

函数的四种调用方式：方法调用模式、函数调用模式、构造器调用模式、Apply 调用模式。

调用模式会影响 this 的指向。

1. 方法调用模式

```
var object = {
    age: 18,
    func: function() {
        this.age;
    }
};

object.func();
```

this 指向整个 object 对象

函数调用模式与 this 变量（2）

函数的四种调用方式：方法调用模式、函数调用模式、构造器调用模式、Apply 调用模式。

调用模式会影响 this 的指向。

2. 函数调用模式

```
var func = function() {
    this;
};

func();
```

this 指向全局对象

全局对象是一个预定义的对象，通常和执行的环境有关。

直接在外部定义的变量和函数会被放入全局对象中。

函数调用模式与 this 变量（3）

函数的四种调用方式：方法调用模式、函数调用模式、构造器调用模式、Apply 调用模式。

调用模式会影响 this 的指向。

3. Apply 调用模式

```
var func = function(params) {
    this.age;
};

var object = {
    age: 18,
};

func.apply(object, 1);
```

this 指向 apply 的第一个参数

将 JavaScript 引入 HTML

引入外部的 JavaScript

```
<html>
    <head>
        <script src="js/index.js"></script>
    </head>
    <body></body>
</html>
```

JS 文件的 URL

在 HTML 中直接使用 JavaScript

```
<html>
    <head>
        <script>
            ...
        </script>
    </head>
    <body></body>
</html>
```

window 对象

window 对象作为浏览器的全局对象，提供了一系列的方法来操作页面的内容。

```
window = {
    onabort: null,
    onafterprint: null,
    onanimationend: null,
    onanimationiteration: null,
    onanimationstart: null,
    onappinstalled: null,
    onauxclick: null,
    onbeforeinstallprompt: null,
    ...
};
```

HTML DOM

HTML DOM 是关于如何获取、修改、添加或删除 HTML 元素的标准。

利用 JavaScript 创建一个元素

每个元素都是一个 DOM 结点

```
var a = document.createElement('a');
a.id = 'openBaidu';
a.classList.add('link-a');
a.setAttribute('href', 'http://www.baidu.com');
a.innerHTML = '点击打开百度';
parentNode.appendChild(a);
```

```
<a
    id="openBaidu"
    class="link-a"
    href="http://www.baidu.com">
    点击打开百度
</a>
```

将结点添加到另一个节点中

利用 JavaScript 获取一个元素

```
parentNode.getElementById('openBaidu');
```

获取对应的 ID 的元素

```
parentNode.getElementsByTagName('a');
```

获取标签名称对应的元素数组

```
parentNode.getElementsByClassName('link-a');
```

获取 class 中包含对应类别的元素数组

```
<a
    id="openBaidu"
    class="link-a"
    href="http://www.baidu.com">
    点击打开百度
</a>
```

Document 对象

```
document.getElementsByClassName('link-a');
```

利用 Document 对象可以从脚本对页面中的所有元素进行访问。

innerHTML 的使用

innerHTML 可以以字符串的形式来获取一个元素的内容，修改这个属性后，浏览器会以 HTML 的形式解析内容。

innerHTML 可以用来快速创建 HTML 元素

```
<ul id="nav"></ul>
```

```
document.getElementById('nav').innerHTML = '<li>index</li>';
```

- index

```
document.getElementById('nav').textContent = '<li>index</li>';
```

index

绝对不能信任任何用户可以编辑的内容！

使用 innerHTML 加载这些内容可能使得你的网站被攻击

设置监听器

通过在元素设置监听器，来实现页面与用户的交互。

点击时触发事件

```
document.getElementById('openBaidu').addEventListener('click', function(){
    ...
});
```

可以同时设置多个监听器

事件触发后所执行的函数

```
document.getElementById('openBaidu').removeEventListener('click', myFunction);
```

DOM 加载与 JavaScript 执行时机

HTML 会按照代码顺序来加载，这有可能使得 JavaScript 获取不到元素

```
<script>
    document.getElementById('mid');
</script>
<div id="mid"></div>
```

对外部引入的 JavaScript 代码也同理

这样的执行结果为 undefined，原因是此时下面的元素还没有生成。

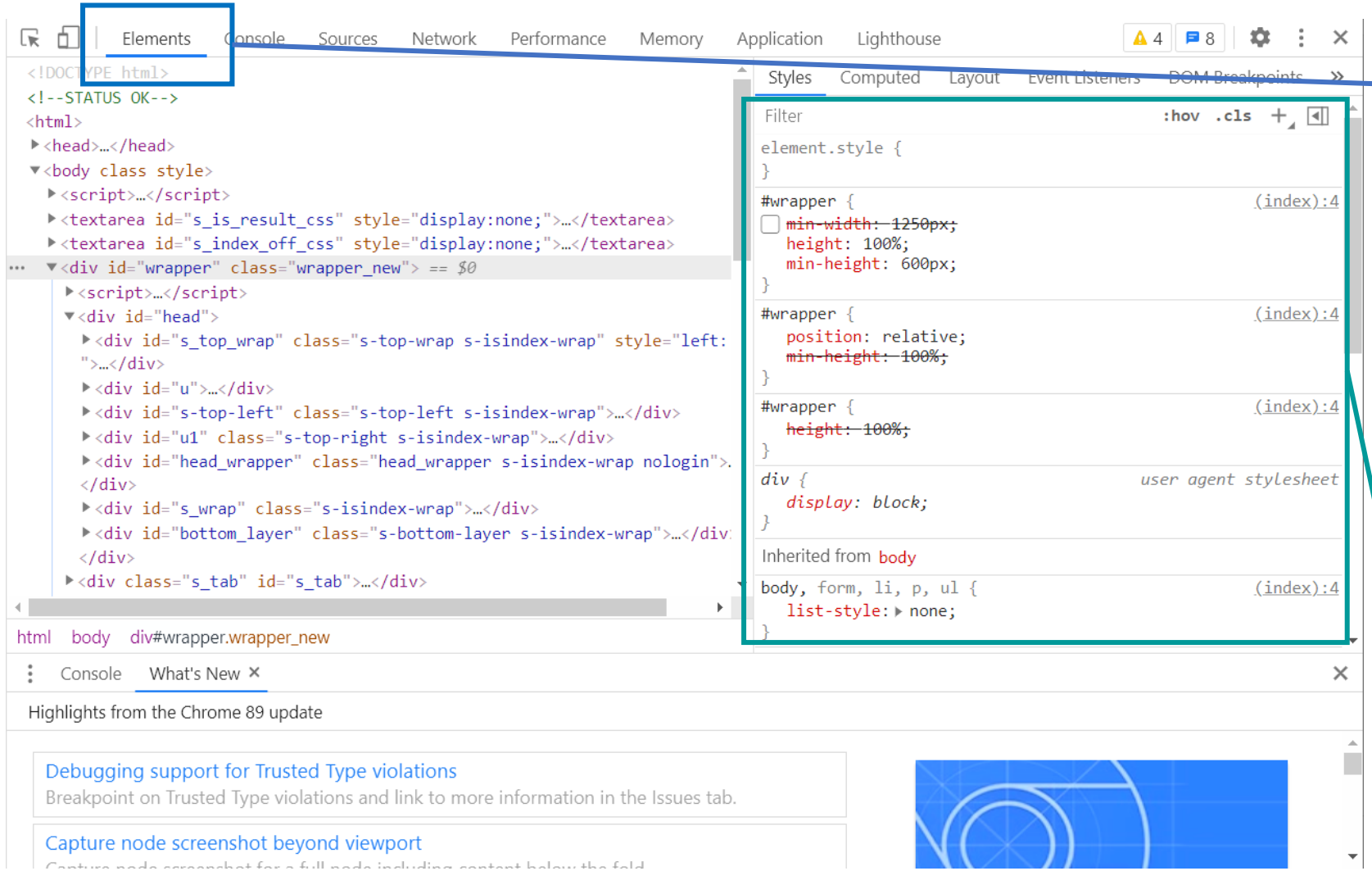
可以把 JavaScript 代码移到最后面，也可以使用 window.onload 函数：

```
<script>
    window.onload = function() {
        document.getElementById('mid');
    };
</script>
<div id="mid"></div>
```

window.onload 会在网页加载完成后执行

调试

在浏览器中右键 – 审查元素，或直接 F12，可以打开控制台窗口。

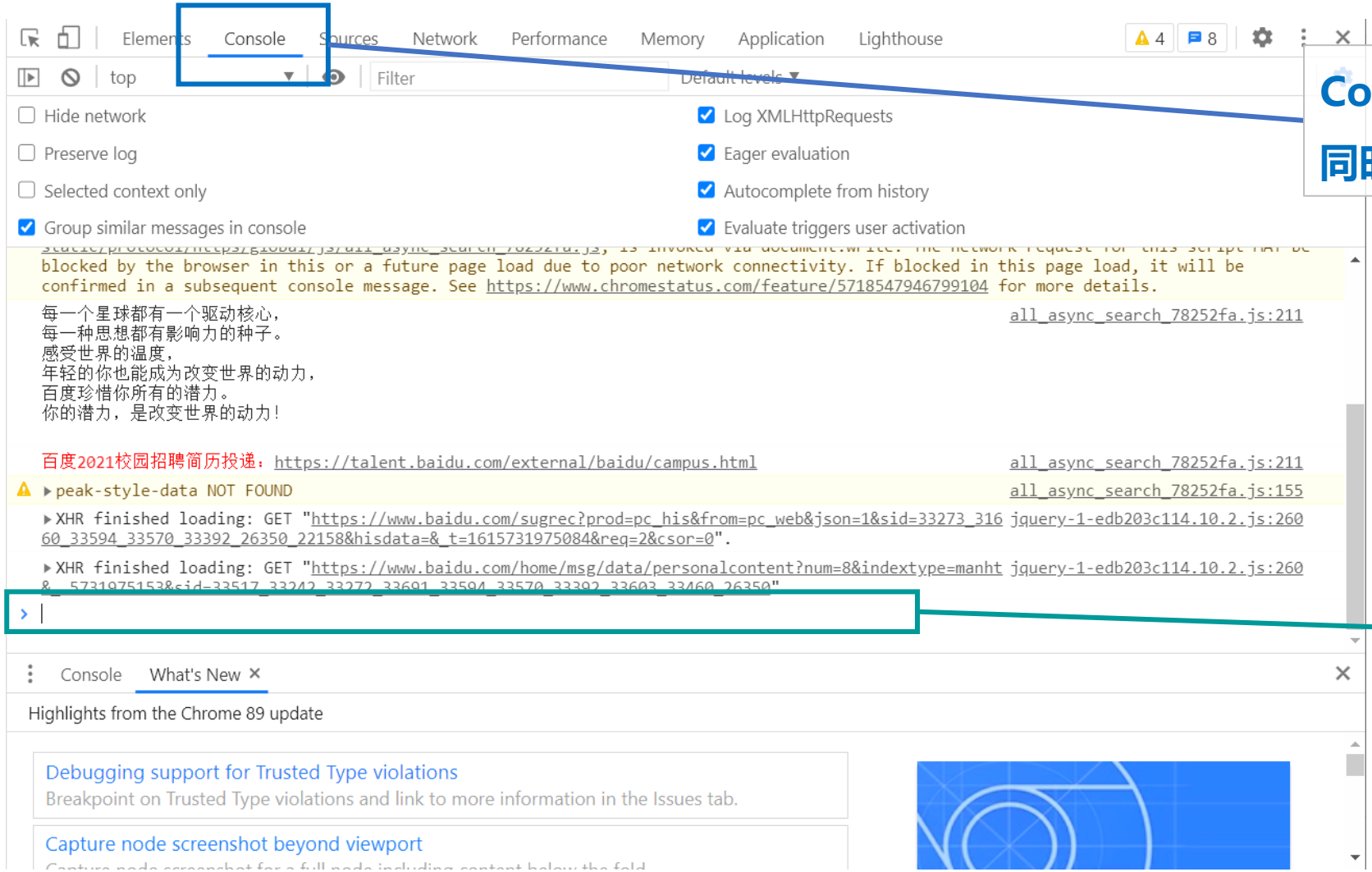


Elements 可以查看网页中的元素

选中元素后可以查看作用的 CSS

调试

在浏览器中右键 – 审查元素，或直接 F12，可以打开控制台窗口。



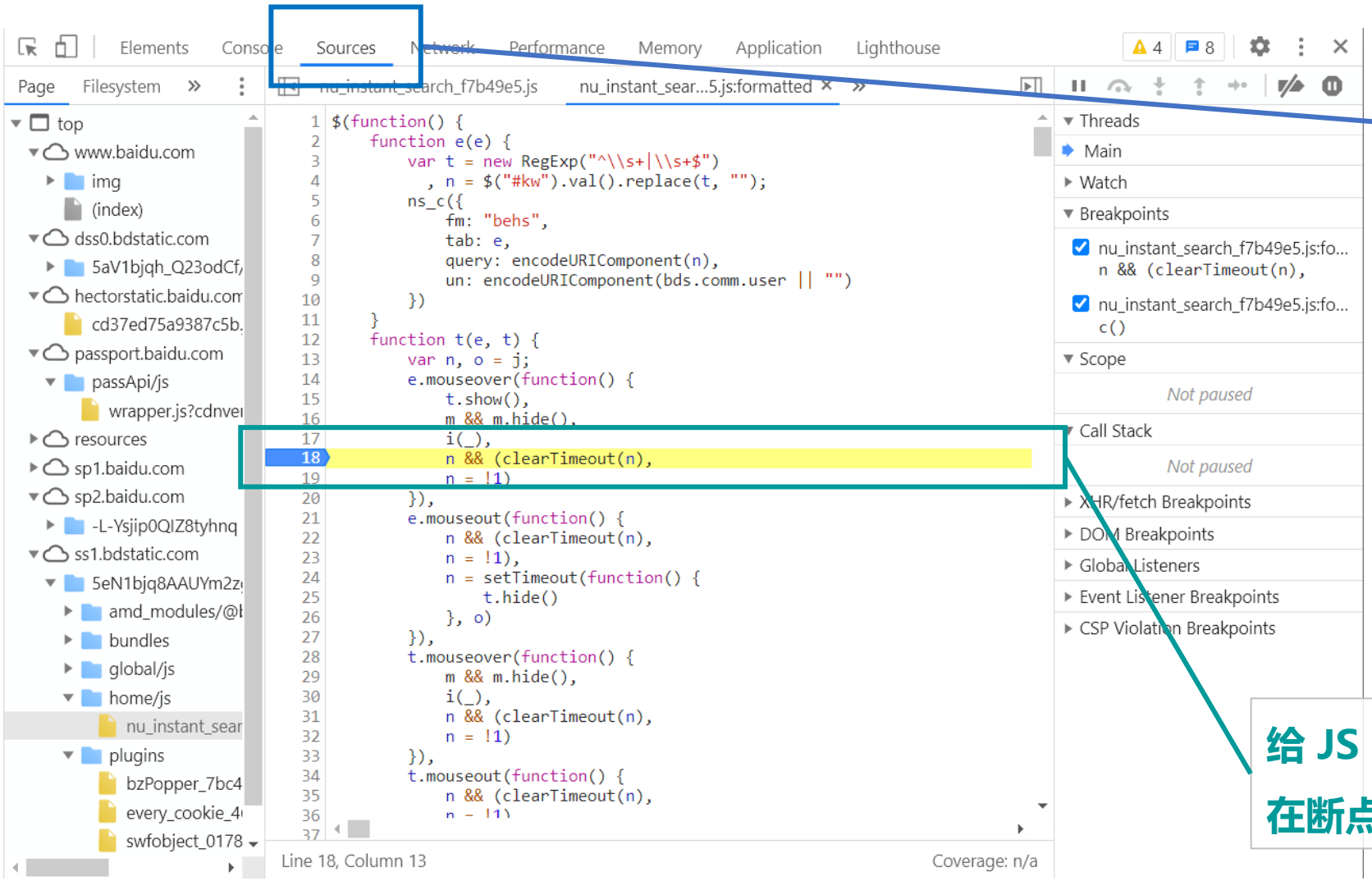
Console 可以来执行 JS 脚本，

同时也可以查看 JS 的控制台输出信息

可以随意输入 JS 脚本来执行

调试

在浏览器中右键 – 审查元素，或直接 F12，可以打开控制台窗口。

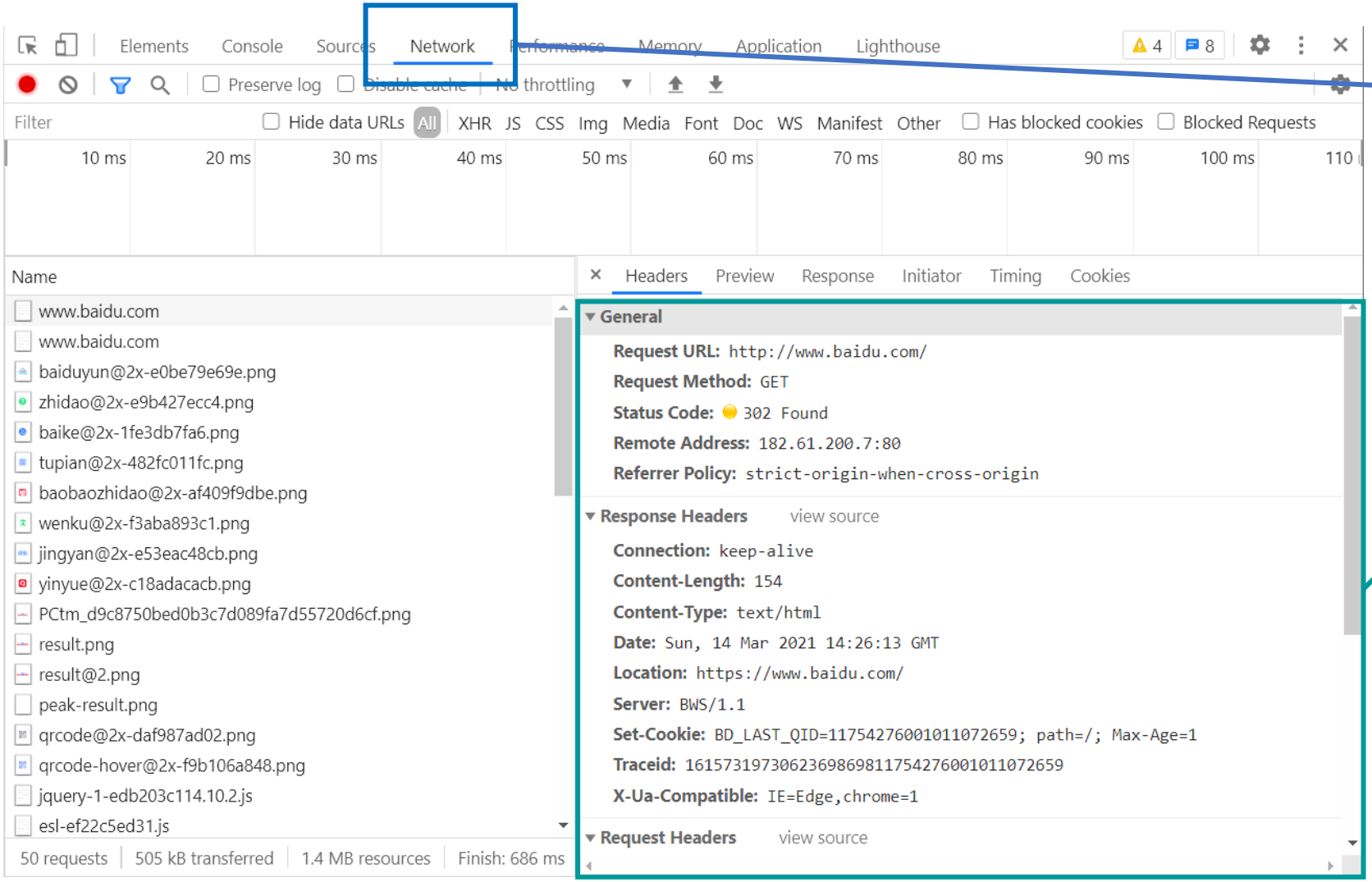


Sources 可以查看网页加载时所用到的文件

给 JS 代码打断点查看执行情况，在断点处可以使用控制台查看变量的信息。

调试

在浏览器中右键 – 审查元素，或直接 F12，可以打开控制台窗口。



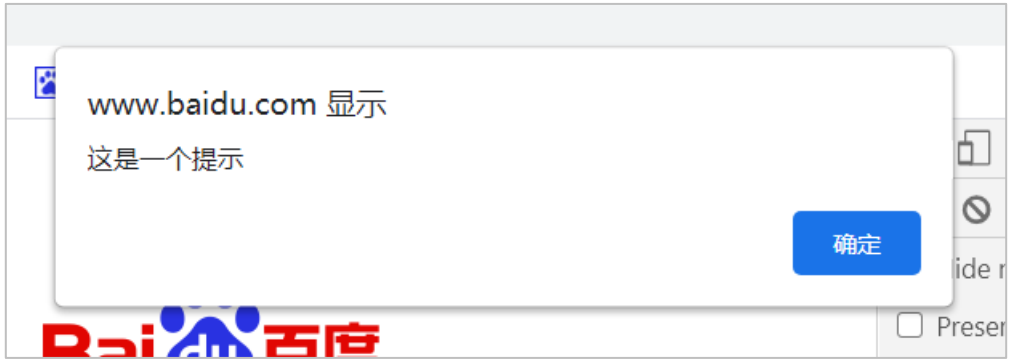
Network 可以查看网络连接记录

选中记录后可查看请求的目标地址,也可以查看请求的状态、返回的内容等。

alert 与 console

window.alert 函数可以弹窗提示

```
alert('这是一个提示');
```



console 是一个对象，里面的 log、warn、error 方法可以在控制台输出信息

```
console.log('控制台信息');
console.warn('控制台信息');
console.error('控制台信息');
```