第一章 HTML5概述

内容安排

- 1.1 什么是HTML5
- 1.2 HTML5 发展史
- 1.3 新的认识
- 1.4 无插件范式
- 1.5 HTML5的新功能
- 1.6 课后思考
- 1.7 小结

1.1 什么是HTML5

◆**官方概念：**HTML5草案的前身名为Web Applications 1.0，是作为下一代互联网标准，用于取代HTML4和XHTML1的新一代标准版本，所以叫HTML5。它增加了新的标签和属性，加强了网页的标准、语义化与Web表现性能，同时还增加了本地数据库等Web应用的功能。

什么是HTML5

- **广义概念**：HTML5代表浏览器端技术的一个发展阶段。在这个阶段，浏览器呈现技术得到了一个飞跃发展和广泛支持，它包括：HTML5，CSS3，JavaScript，API在内的一套技术组合。

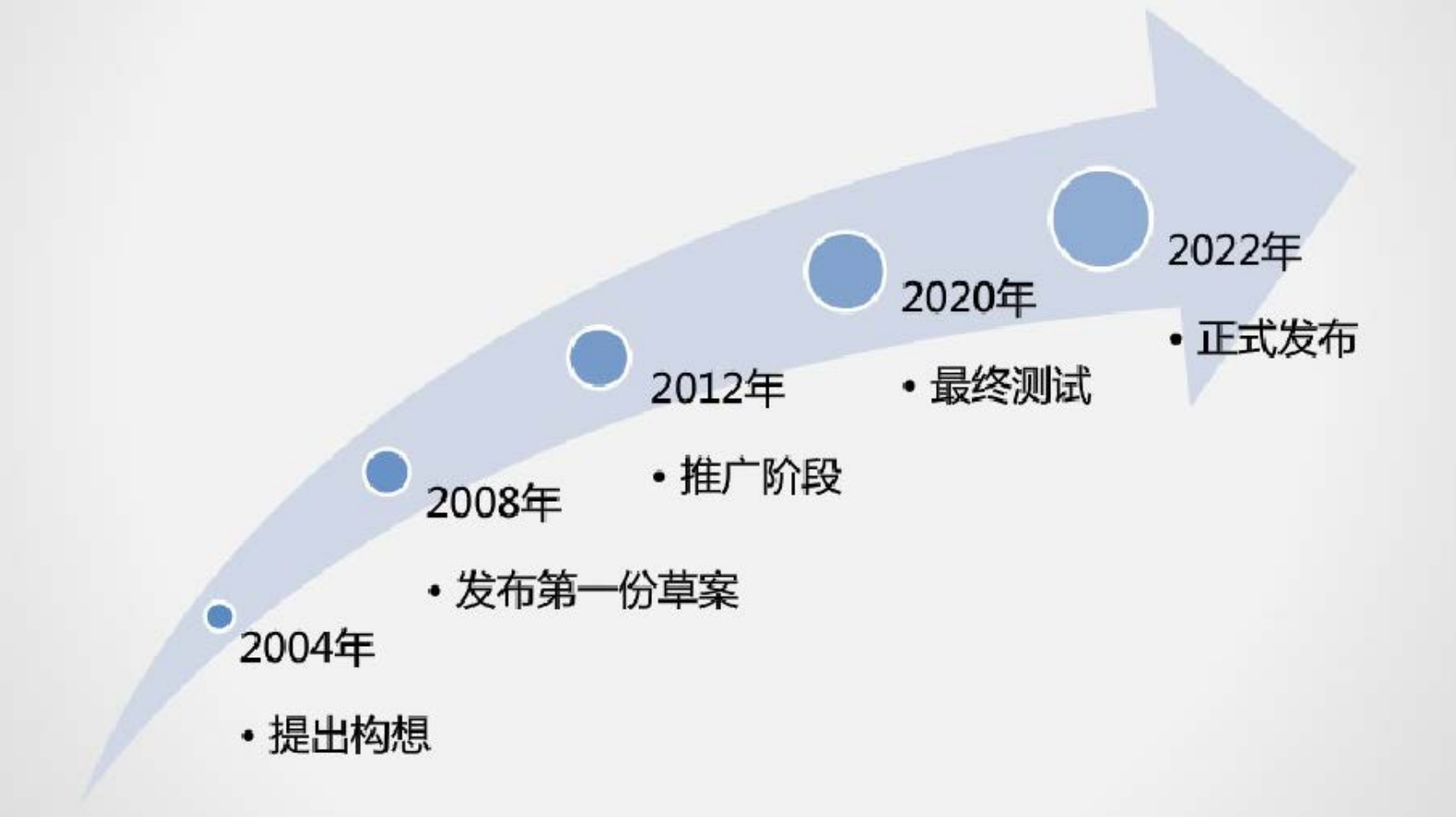
- 后面我们描述的HTML5就是基于广义来讲述

- **目标**：简单的Web程序、简洁的HTML代码、合理结构、规范统一

HTML5的发展史

HTML标签	1991年Tim Berners-Lee编写，包括20个HTML标签
HTML2.0	1995年由IEIF推出
HTML4.01	1999年W3C推出HTML4.01版本
HTML1.0/2.0	HTML4.01版本基础上衍变而来
HTML5	2009年W3C推出HTML5取代原有HTML版本

HTML5发展时间表



HTML5 浏览器支持

最新版本的Safari、Chrome、Firefox以及Opera支持某些HTML特性。Internet Explorer 9将支持某些HTML5特性。

谁在开发HTML5

- **WHATWG** (Web Hypertext Application Technology Working Group)：由来自Apple、Mozilla、Google、Opera等浏览器厂商的人组成，成立与2004年。WHATWG开发HTML和Web应用API，同时为各浏览器厂商以及其他有意向的组织提供开放式合作。

- **W3C** (The World Wide Web Consortium)：W3C下辖的HTML工作组目前负责发布HTML5规范。

谁在开发HTML5

- **IEIF** (Internet Engineering Task Force，因特网工程任务组)：这个任务组下辖HTTP等负责Internet协议的团队。HTML5定义的一种新API(WebSocket API)依赖于新的WebSocket协议，IEIF工作组正在开发这个协议。

1.3 新的认识

- 兼容性和存在即合理
- 效率和用户优先
- 化繁为简
- 通用访问

兼容性和存在即合理

1、兼容现存的HTML文档，平滑过渡。

2、对原有HTML的用法进行总结，创造一些新标签。例如：article、footer、header、nav、section等标签。

效率与用户优先

1、不严格的语法。

2、安全机制的设计。

3、表现和内容分离。

化繁为简

1、以浏览器原生能力代替复杂的JavaScript代码。

2、新的简化的DOCTYPE。

3、新的简化的字符集声明。

4、简单而强大的HTML5 API。

通用访问

1、可访问性。

2、媒体中立。

3、支持所有语种。

1.4 无插件范式

插件的方式存在很多问题：

- 插件安装可能失败；
- 插件可以被禁用或屏蔽（例如Apple的iPad就不支持Flash插件）；
- 插件自身会成为被攻击的对象；
- 插件不容易与HTML文档的其他部分集成（因为插件边界、剪裁和透明度问题）。

1.4 无插件范式

HTML5不仅仅是提供新元素支持新功能，更重要的是添加了对脚本和布局之间的原生交互能力，可以实现以前实现不了的效果。（例如canvas）

不需要安装插件

HTML5包括什么，不包括什么

HTML5不仅涵盖了核心的标记元素，同时也包括很多新的API。

- Canvas(2D和3D)
- Cross-document消息传送
- Geolocation
- Audio和Video
- Forms
- MathML
- Microdata
- Server-Sent Events
- WebSocket API及协议
- Web Origin Concept
- Scalable Vector Graphics(SVG)

HTML5包括什么，不包括什么

- Web Storage
- 应用缓存（离线Web应用）
- 拖放
- 索引数据库
- Web Workers
- XMLHttpRequest Level 2

1.5 HTML5的新功能

- 新的DOCTYPE和字符集
- 新元素和旧元素
- 语义化标记
- 使用Selectors API简化选取操作
- JavaScript日志和调试
- window.JSON
- DOM Level 3
- Monkeys、Squirrelfish和其他JavaScript引擎

新的DOCTYPE和字符集

新的DOCTYPE

- <!DOCTYPE>声明位于HTML文档中的最前面的位置，它位于<html>标签之前。
- 该标签告知浏览器文档所使用的HTML或XHTML规范。
- 在HTML4中，<!DOCTYPE>标签可以声明三种类型DTD类型，分别表示严格版本(Strict)、过渡版本(Transitional)和基于框架(Frameset)的HTML文档。

新的DOCTYPE

浏览器会根据DOCTYPE识别该使用哪种模式，以及使用什么规则来验证页面。

- **标准模式**：浏览器按W3C标准解析执行代码。

① 如果XHTML文档包含完整的DOCTYPE，那么它一般以标准模式呈现；

② 包含严格DTD的DOCTYPE和包含过渡的DTD和URI的DOCTYPE常常导致页面以标准模式呈现。

新的DOCTYPE

- **怪异模式**：兼用老页面。使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行方式不一样，所以我们称之为怪异模式。

① 不存在DOCTYPE或形式不正确会导致怪异模式；

② 有过渡/框架DTD没有URI会导致页面以怪异模式呈现；

③ IE中，如果DOCTYPE声明在xml之后，会导致怪异模式。

新的DOCTYPE

- **近标准模式**:与标准模式一致，除了在处理下面这种情况时：

如果一个块级元素除了空白文本（空格、tab等字符）外再无其它内容，则它的高度按0处理；如果有子元素，则它的高度不能比子元素大，无论它的font-size多大。

新的DOCTYPE

- HTML4的DOCTYPE代码

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

- HTML5的DOCTYPE代码

```
<!DOCTYPE html>
```

HTML5的字符集

HTML5的字符集也得到了简化，只需要使用UTF-8即可，使用一个meta标记就可以指定HTML5的字符集。

- HTML4的字符集：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

- HTML5的字符集：

```
<meta charset="utf-8">
```

新元素和旧元素

HTML5引入了很多新的标记元素，根据内容类型的不同，这些元素被分成了7大类。

表 HTML5的内容类型

内容类型	描述
内嵌	向文档中添加其他类型内容，例如audio、video、canvas和iframe等
流	在文档和应用的body中使用的元素，例如form、h1和small
标题	段落标题，例如h1、h2和hgroup
交互	与用户交互的内容，例如音频和视频控件、button和textarea等
元数据	通常出现在页面的head中，设置页面其他部分的表现和行为，例如script、style和title等
短语	文本和文本标记元素，例如mark、kbd、sub和sup等
片段	用于定义文档中片段的元素，例如article、aside和title等

新元素和旧元素

以下的 HTML 4.01 元素在HTML5中已经被删除:

- <acronym>
- <basefont>
-
- <font>
- <frameset>
- <strike>
- <applet>
- <big>
- <dir>
- <frame>
- <noframes>
- <tt>

语义化标记

思考：什么是语义化？

语义化标记

- **语义化：**是指用合理HTML标记以及其特有的属性去格式化文档内容。

通俗地讲，语义化就是对数据和信息进行处理，使得机器可以理解。

语义化标记

表 **HTML5中新的片段类元素**

元素名	描述
header	标记头部区域的内容（用于整个页面或页面中的一块区域）
footer	标记脚部区域的内容（用于整个页面或页面中的一块区域）
section	Web页面中的一块区域
article	独立的文章内容
aside	相关内容或者引文
nav	导航类辅助内容

语义化标记

- 语义化的优点：

a) 语义化的(X)HTML文档有助于提升你的网站对访客的易用性比如使用PDA、文字浏览器以及残障人士将从中受益。

b) 对于搜索引擎或者爬虫软件来说,则有助于它们建立索引,并可能给予一个较高的权值。

语义化标记

- 代码：HTML5示例页面--对应的HTML文件

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>HTML5</title>
  <link rel="stylesheet" href="html5.css">
</head>
<body>
  <!-- 头部 -->
  <header>
    <h1>Header</h1>
    <h2>Subtitle</h2>
    <h4>HTML5 Rocks!</h4>
  </header>
  <!-- 内容容器 -->
  <div id="container">
    <!-- 导航 -->
    <nav>
      <h3>Nav</h3>
      <a href="">Link 1</a>
      <a href="">Link 2</a>
      <a href="">Link 3</a>
    </nav>
```

语义化标记

- 代码：HTML5示例页面--对应的HTML文件（续）

```
<!-- Web页面中的一块区域 -->
<section>
  <!-- 独立的文章内容 -->
  <article>
    <!-- 头部 -->
    <header>
        <h1>Article Header</h1>
    </header>
    <!-- 段落 -->
    <p>Lorem ipsum dolor HTML5 nunc aut nunquam sit amet, consectetur
    adipiscing elit. Vivamus at est eros, vel fringilla urna.</p>
    <p>Per inceptos himenaeos. Quisque feugiat, justo at vehicula
    pellentesque, turpis lorem dictum nunc.</p>
    <!-- 脚部区域 -->
    <footer>
        <h2>Article Footer</h2>
    </footer>
  </article>
  <article>
    <header>
        <h1>Article Header</h1>
    </header>
```

语义化标记

- 代码：HTML5示例页面--对应的HTML文件（续）

```
      <p>HTML5: "Lorem ipsum dolor nunc aut nunquam sit amet, consectetur
      adipiscing elit. Vivamus at est eros, vel fringilla urna. Pellentesque
      odio</p>
      <footer>
        <h2>Article Footer</h2>
      </footer>
    </article>
  </section>
  <!-- 相关内容 -->
  <aside>
    <h3>Aside</h3>
    <p>HTML5: "Lorem ipsum dolor nunc aut nunquam sit amet, consectetur
    adipiscing elit. Vivamus at est eros, vel fringilla urna. Pellentesque odio
    rhoncus</p>
  </aside>
  <!-- 脚部区域的内容 -->
  <footer>
    <h2>Footer</h2>
  </footer>
 </div>
</body>
</html>
```

语义化标记

- 代码：HTML5示例页面--对应的CSS文件（续）

```
/* html5 css file, Copyright © Pro HTML5 Programming */
 body {
    background-color:#CCCCCC;
    font-family:Geneva,Arial,Helvetica,sans-serif;
    margin: 0px auto;
    max-width:900px;
    border:solid;
    border-color:#FFFFFF;
}
header {
    background-color: #F47D31;
    display:block;
    color:#FFFFFF;
    text-align:center;
}
header h2 {
    margin: 0px;
}
h1 {
    font-size: 72px;
    margin: 0px;
}
```

语义化标记

- 代码：HTML5示例页面--对应的CSS文件（续）

```
/* html5 css file, Copyright © Pro HTML5 Programming */
 body {
    background-color:#CCCCCC;
    font-family:Geneva,Arial,Helvetica,sans-serif;
    margin: 0px auto;
    max-width:900px;
    border:solid;
    border-color:#FFFFFF;
}
header {
    background-color: #F47D31;
    display:block;
    color:#FFFFFF;
    text-align:center;
}
header h2 {
    margin: 0px;
}
h1 {
    font-size: 72px;
    margin: 0px;
}
```

语义化标记

- 代码：HTML5示例页面--对应的CSS文件（续）

```css
h2 {
    font-size: 24px;
    margin: 0px;
    text-align:center;
    color: #F47D31;
}
h3 {
    font-size: 18px;
    margin: 0px;
    text-align:center;
    color: #F47D31;
}
h4 {
    color: #F47D31;
    background-color: #fff;
    -webkit-box-shadow: 2px 2px 20px #888;
    -webkit-transform: rotate(-45deg);
    -moz-box-shadow: 2px 2px 20px #888;
    -moz-transform: rotate(-45deg);
    position: absolute;
    padding: 0px 150px;
    top: 50px;
    left: -120px;
    text-align:center;
}
```

语义化标记

- 代码：HTML5示例页面--对应的CSS文件（续）

```
nav {
    display:block;
    width:25%;
    float:left;
}
nav a:link, nav a:visited {
    display: block;
    border-bottom: 3px solid #fff;
    padding: 10px;
    text-decoration: none;
    font-weight: bold;
    margin: 5px;
}
nav a:hover {
    color: white;
    background-color: #F47D31;
}
nav h3 {
    margin: 15px;
    color: white;
}
#container {
    background-color: #888;
}
```

语义化标记

- 代码：HTML5示例页面--对应的CSS文件（续）

```
section {
    display:block;
    width:50%;
    float:left;
}
article {
    background-color: #eee;
    display:block;
    margin: 10px;
    padding: 10px;
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    border-radius: 10px;
}
article header {
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    border-radius: 10px;
    padding: 5px;
}
article footer {
    -webkit-border-radius: 10px;
    -moz-border-radius: 10px;
    border-radius: 10px;
    padding: 5px;
}
```

语义化标记

- 代码：HTML5示例页面--对应的CSS文件（续）

```
article h1 {
    font-size: 18px;
}
aside {
    display:block;
    width:25%;
    float:left;
}
aside h3 {
    margin: 15px;
    color: white;
}
aside p {
    margin: 15px;
    color: white;
    font-weight: bold;
    font-style: italic;
}
footer {
    clear: both;
    display: block;
    background-color: #F47D31;
    color:#FFFFFF;
    text-align:center;
    padding: 15px;
}
```

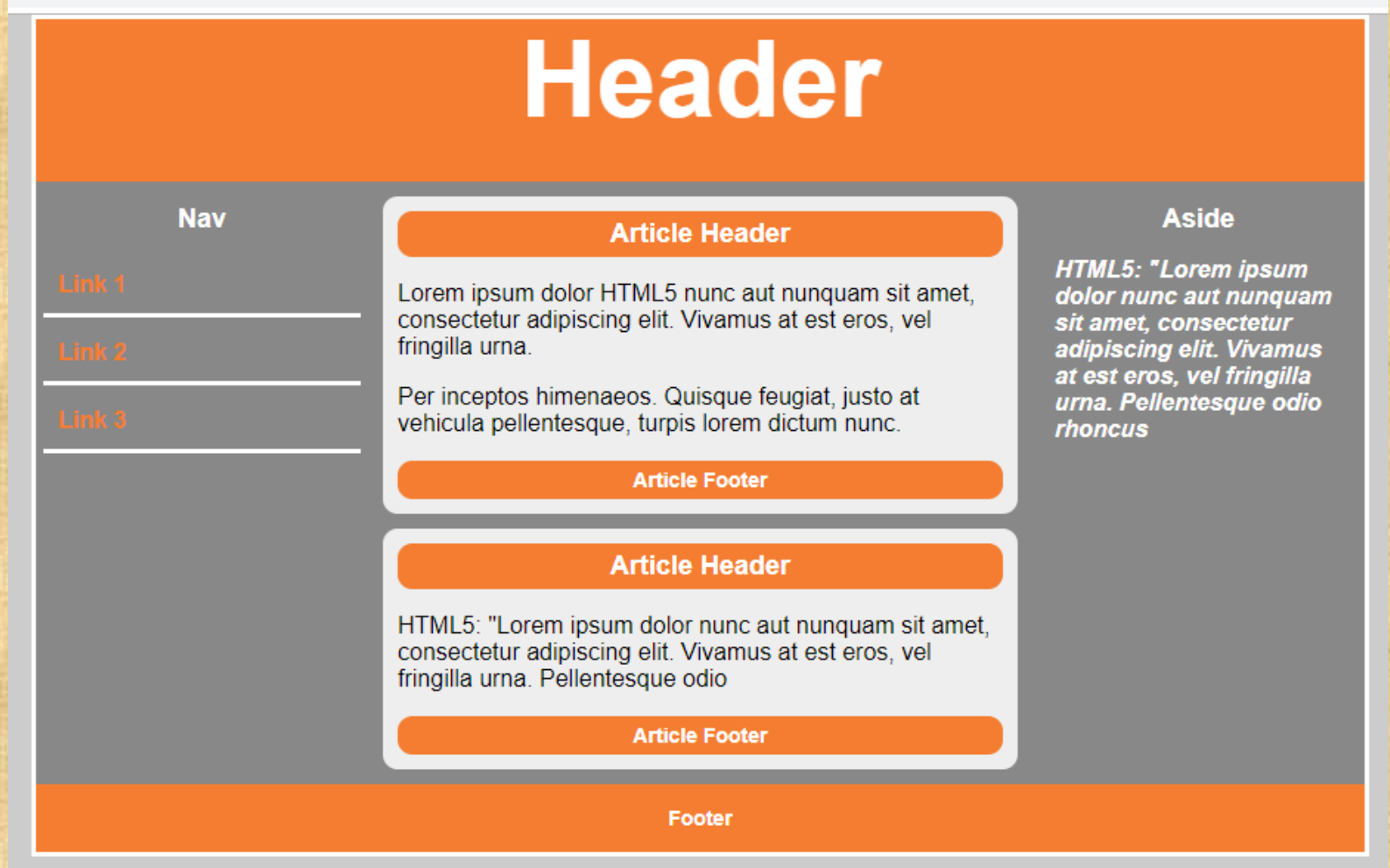
语义化标记

- 代码：HTML5示例页面--对应的CSS文件（续）

```
footer h2 {
    font-size: 14px;
    color: white;
}
/* links */
a {
    color: #F47D31;
}
a:hover {
    text-decoration: underline;
}
```

语义化标记

- 代码：HTML5页面对应的CSS文件（效果图）



使用Selectors API简化选取操作

表 以前用来查找元素的JavaScript方法

函数	描述	示例
getElementById()	根据指定的id特性值查找并返回元素	 getElementById("foo");
getElementsByName()	返回所有name特性为指定值的元素	<input type="text" name="foo"> getElementsByName ("foo");
getElementsByTagName()	返回所有标签名称与指定值相匹配的元素	<input type="text"> getElementsByTagName ("input");

使用Selectors API简化选取操作

表 新QuerySelector方法

函数	描述	示例	结果
querySelector()	根据指定的选择规则，返回在页面中找到的第一个匹配元素	querySelector (“input.error”);	返回第一个CSS类名为“error”的文本输入框
querySelectorAll()	根据指定规则返回页面中所有相匹配的元素	querySelectorAll (“#results td”);	返回id值为results的元素下所有的单元格

使用Selectors API简化选取操作

- 代码：使用Selectors API -- querySelector()

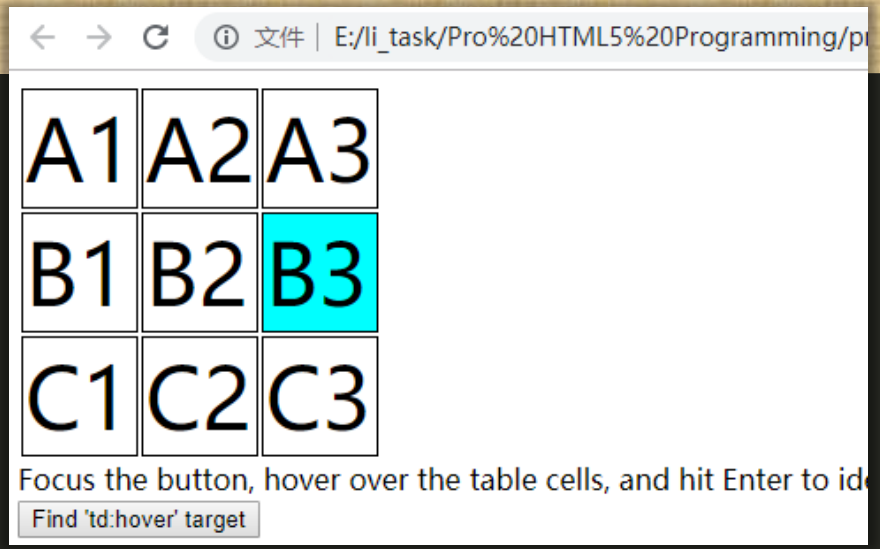
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Query Selector Demo</title>
  <style type="text/css">
    td {
      border-style: solid;
      border-width: 1px;
      font-size: 300%;
    }

    td:hover {
      background-color: cyan;
    }

    #hoverResult {
      color: green;
      font-size: 200%;
    }
  </style>
</head>
```

使用Selectors API简化选取操作

- 代码：使用Selectors API（续）




```
<body>
  <section>
    <!-- 创建一个3*3的表格 -->
    <table>
      <tr>
        <td>A1</td> <td>A2</td> <td>A3</td>
      </tr>
      <tr>
        <td>B1</td> <td>B2</td> <td>B3</td>
      </tr>
      <tr>
        <td>C1</td> <td>C2</td> <td>C3</td>
      </tr>
    </table>
    <div>Focus the button, hover over the table cells, and hit Enter to identify them using
    querySelector('td:hover').</div>
    <button type="button" id="findHover" autofocus>Find 'td:hover' target</button>
    <div id="hoverResult"></div>
  <script type="text/javascript">
    document.getElementById("findHover").onclick = function() {
      // 使用新QuerySelector方法快速找找到元素
      var hovered = document.querySelector("td:hover");
      if (hovered)
        document.getElementById("hoverResult").innerHTML = hovered.innerHTML;
    }
  </script>
  </section>
</body>
</html>
```

使用Selectors API简化选取操作

- 代码：使用Selectors API -- querySelectorAll()

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Query Selector All Demo</title>
  <style type="text/css">
    td {
      border-style: solid;
      border-width: 1px;
      font-size: 200%;
    }

    #checkedResult {
      color: green;
      font-size: 200%;
    }
  </style>
</head>
<body>
  <section>
    <table>
```

使用Selectors API简化选取操作

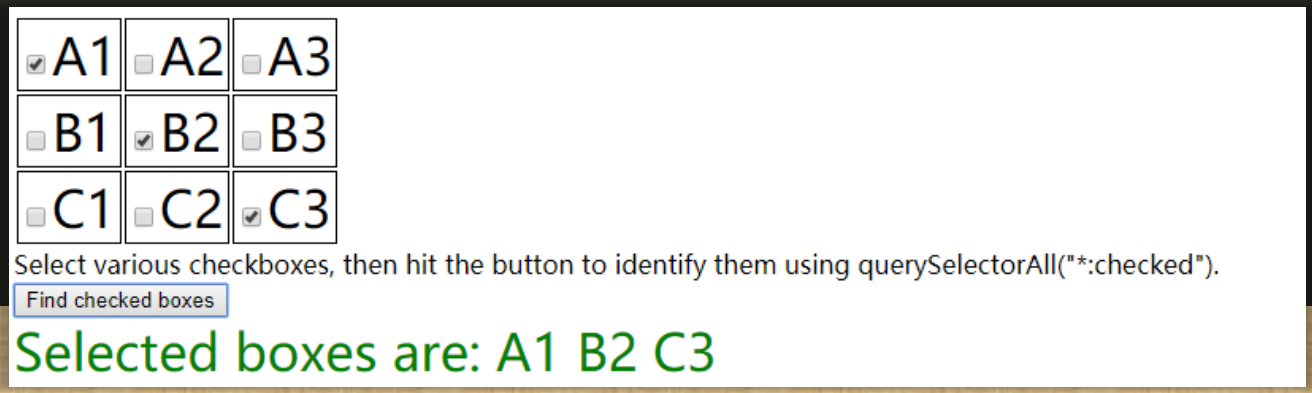
- 代码：使用Selectors API -- querySelectorAll()（续）

```
  <tr>
    <td><input type="checkbox" name="A1">A1</td>
    <td><input type="checkbox" name="A2">A2</td>
    <td><input type="checkbox" name="A3">A3</td>
  </tr>
  <tr>
    <td><input type="checkbox" name="B1">B1</td>
    <td><input type="checkbox" checked name="B2">B2</td>
    <td><input type="checkbox" name="B3">B3</td>
  </tr>
  <tr>
    <td><input type="checkbox" name="C1">C1</td>
    <td><input type="checkbox" name="C2">C2</td>
    <td><input type="checkbox" name="C3">C3</td>
  </tr>
</table>
<div>Select various checkboxes, then hit the button to identify them using
querySelectorAll("*:checked").</div>
<button type="button" id="findChecked" autofocus>Find checked boxes</button>
<div id="checkedResult"></div>
```

使用Selectors API简化选取操作

- 代码：使用Selectors API -- querySelectorAll()（续）

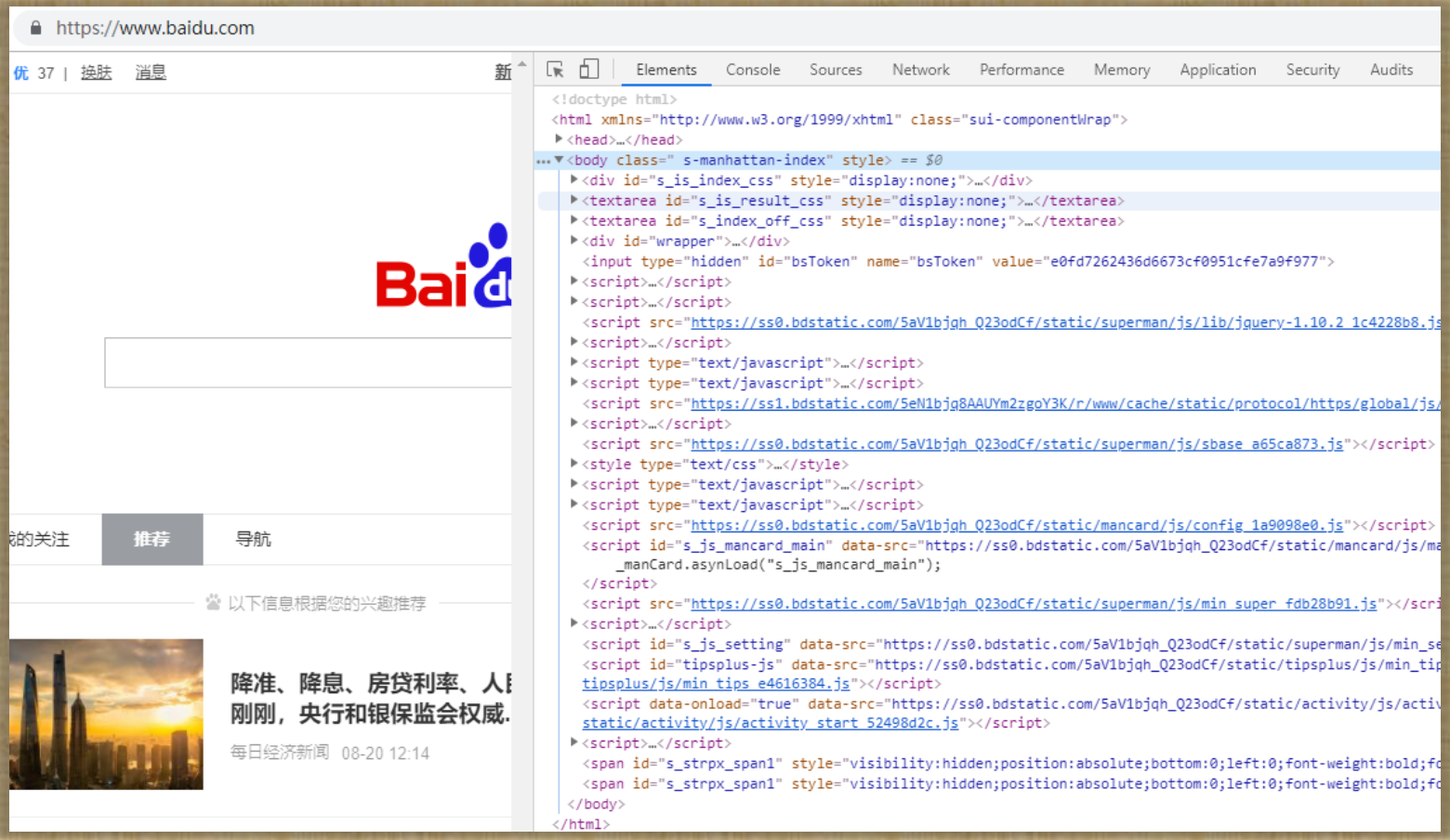
```
    <script type="text/javascript">
      document.getElementById("findChecked").onclick = function() {
        //根据指定规则返回页面中所有相匹配的元素
        var selected = document.querySelectorAll("*:checked");
        var result = "Selected boxes are: ";
        for (var i = 0; i < selected.length; i++) {
          result += (selected[i].name + " ");
        }
        //显示选择的所有的td的name
        document.getElementById("checkedResult").innerHTML = result;
      }
    </script>
  </section>
</body>
</html>
```



JavaScript日志和调试

- 浏览器的内嵌开发工具
  - Safari的Web Inspector
  - Google的Chrome开发工具(Developer Tools)
  - IE的开发工具(Developer Tools)
  - Opera的开发工具(Dragonfly)
- 调试代码
  - console.log API

JavaScript日志和调试



图：Google的Chrome开发工具(Developer Tools)截图

window.JSON

新浏览器提供了对json的原生支持

window.JSON

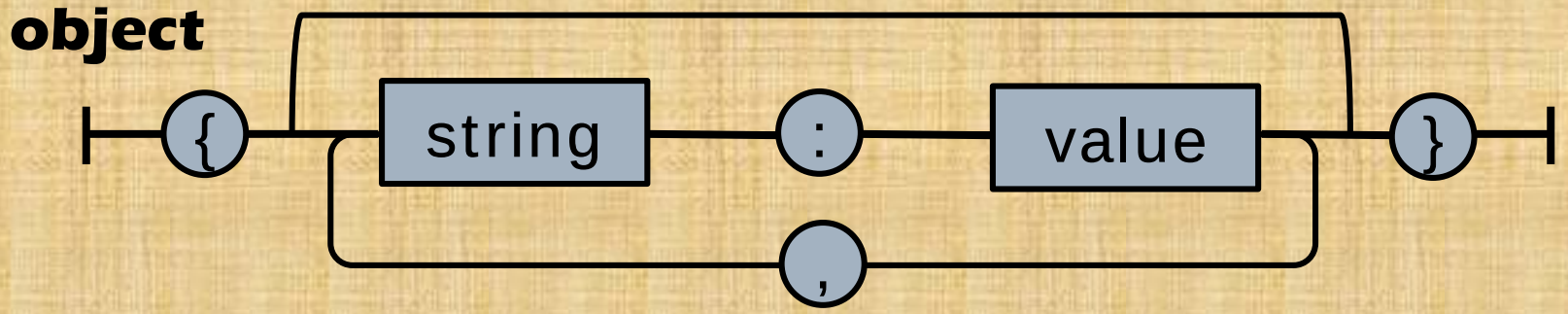
- JSON简介

```
{
  "学生": [
    { "姓名":"Bill" , "年龄":15 },
    { "姓名":"George" , "年龄":17 },
    { "姓名":"Thomas" , "年龄":16 }
  ]
}
```

JSON (Javascript Object Notation)是一种轻量级的数据交换格式。它基于ECMAScript的一个子集。JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C、C++、C#、Java、JavaScript、Perl、Python等）。这些特性使JSON成为理想的数据交换语言。易于人阅读和编写，同时也易于机器解析和生成（一般用于提升网络传输速率）

window.JSON

- JSON结构有两种结构
  - JSON结构一



- 例子：表示人的一个对象

```
{
   "姓名":"小明",
   "年龄":24
}
```

window.JSON

1、对象是一个无序的“‘名称/值’对”集合。

① 一个对象以“{”（左括号）开始，

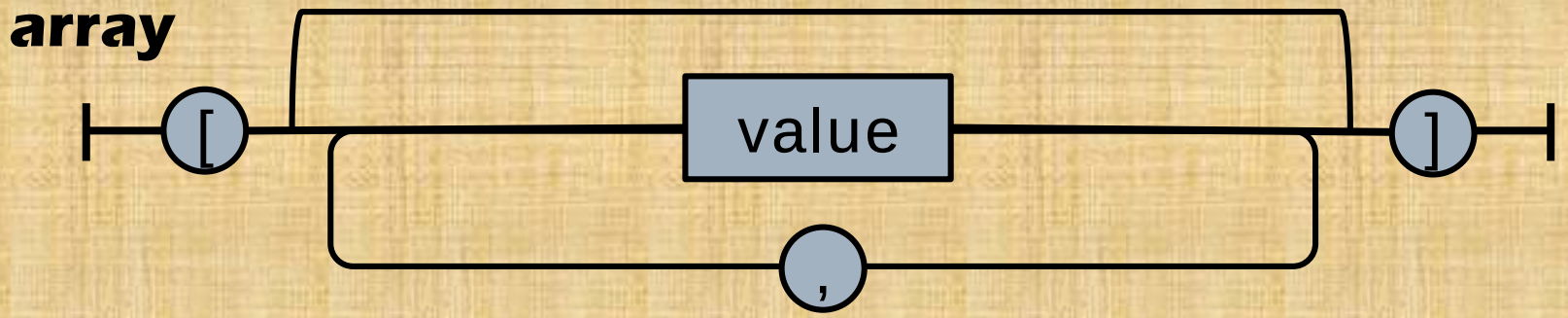
“}”（右括号）结束。

① 每个“名称”后跟一个“:”（冒号）；

② “‘名称/值’对”之间使用“,”（逗号）分隔。

window.JSON

- JSON结构有两种结构
  - JSON结构二



- 例子：一组学生

```
{
  "学生":[
    {"姓名":"小明", "年龄":24},
    {"姓名":"小米", "年龄":22}
  ]
}
```

window.JSON

2、数据是值（value）的有序集合。

① 一个数据以“[”（左中括号）开始，“]”（右中括号）结束。

① 值之间使用“,”（逗号）分隔。

DOM Level 3

新浏览器提供了统一的代码实现DOM操作和事件处理。

Monkeys Squirrelfish和其他JavaScript引擎

各浏览器厂商竞相提升JavaScript的执行性能，越来越接近于本地桌面应用程序。

Monkeys Squirrelfish和其他JavaScript引擎

表 **Web浏览器的JavaScript引擎**

浏览器引擎	引擎名称	备注
Apple Safari	Nitro（也被称作squirrel fish extreme）	Safari 4中发布，在Safari 5中提升性能，保罗字节码优化和上下文线程的本地编译器
Google Chrome	V8	自从Chrome 2开始，使用了新一代垃圾回收机制，可确保内存高度可扩展而不会发生中断
Microsoft Firefox	Chakra	注重于后台编译和高效的类型系统，速度比IE8快10倍

Monkeys Squirrelfish和其他JavaScript引擎

表 **Web浏览器的JavaScript引擎（续）**

浏览器引擎	引擎名称	备注
Mozilla Firefox	JagerMonkey	从3.5版本优化而来，结合了快速解释和源自追踪树（trace tree）的本地编译
Opera	Carakan	它采用了基于寄存器的字节码和选择性本地编译的方式，声称效率比10.50版本提升了75%

1.6 课后思考

- HTML与HTML5的主要区别有哪些？
- DOCTYPE作用？标准模式与兼容模式各有什么区别？它们有何意义？
- 你做的页面在哪些流览器测试过?这些浏览器的内核分别是什么?
- HTML5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？

1.7 小结

- HTML5的开发历史和即将迎来的几个重要时间点。
- HTML5的四个新设计准则：兼容性、实用性、互通性和通用访问性。
- HTML5无插件范式，回答了HTML5包括什么，不包括什么。
- HTML5的新功能：新的DOCTYPE和字符集、新元素和旧元素、语义化标记、使用Selectors API简化选取操作、JavaScript日志和调试、window.JSON、JavaScript引擎的竞争。