

创建动态网页

如今，大部分网页（例如注册和登录页面）都有某种方式来接受用户的输入。要设计此类网页，需要使用称为表单元素的几种元素，这些元素使用户能够输入值。使用这些表单元素输入的数据在提交时必须进行进一步处理。每个网页都应响应最终用户操作，例如单击提交按钮、更改字段中的值或从列表中进行选择。所有这些操作都称为事件，需要进行正确处理。可使用 JavaScript 对象模型来实现此动态功能。

本章讨论了如何设计 HTML 表单来接受用户输入。此外，还讨论了各种类型的浏览器和表单对象。

目标

在本章中，您将学习：

- 设计 HTML 表单
- 操作网页的组件



NITT | **training**  **-china**
.com

设计 HTML 表单

思考 FoodExpress.com 的场景。该网站帮助客户在线向各种饭店下单。为此，客户需要在名为 **OrderFoodOnline** 的在线表单中提供各种详细信息，例如姓名、联系电话和地址，如下图所示。

OrderFoodOnline

Name:

Date:

email ID:

Contact Number:

Select Food: Non-vegetarian ☐ Vegetarian ☐

Select Restaurant:

Drinks:

Soups:

Dishes:

Order For: Take Away ☐ Home Delivery ☐

Address:

State:

OrderFoodOnline 表单

表单是用于接受用户输入的交互式网页。这些表单包括一种或多种类型的输入字段，例如文本字段、复选框、单选按钮和提交按钮。这些输入字段使用户能够填写信息并向网站提交信息。**HTML** 支持各种用于创建表单和输入字段的标记。

创建表单

要在网页上创建表单，需要使用 `<FORM>` 标记。`<FORM>` 标记帮助您定义表单。它有一个开始 `<FORM>` 标记和一个结束 `</FORM>` 标记。以下语法用于指定 `<FORM>` 标记：

```
<FORM [attribute list]>
..
..
</FORM>
```

<FORM> 标记支持以下属性:

- name
- ID
- action
- method
- autocomplete
- novalidate
- target

name

name 属性用于指定表单的唯一名称。它用于表单提交时在 get 或 post 方法中唯一标识表单。此外, 多个表单可以显示在一个网页上。可使用 name 属性区分这些表单。以下语法用于指定表单的名称:

```
<FORM name= "User defined name">... </FORM>
```

ID

ID 属性用于指定网页上表单元素的唯一 ID。ID 属性在整个 HTML 文档中应是唯一的。ID 和 name 属性用于唯一标识网页上的表单元素。但是, 当您需要在样式表或脚本中引用表单元素时, 使用为该表单元素提供的 ID。您可以使用此属性向表单元素分配唯一 ID。

以下语法用于指定表单的 ID:

```
<FORM ID= "User defined ID">... </FORM>
```

action

action 属性指定表单内容要提交给的面面的 URL。如果该属性丢失, 那么就会把文档本身的 URL 假设为表单提交的位置。以下语法用于 action 属性来指定页面的 URL:

```
<FORM action="filename or URL">
```

method

method 属性用于指定将数据提交给 action 属性中指定的文件或 URL 的格式。它可以采用以下任一值:

- get
- post

method 属性的默认值是 get。

get

在表单提交时，get 值将表单数据作为“名称 - 值”对附加到表单的 URL。由于该数据被附加到 URL，所以它始终对用户可见。因此，当您需要提交敏感数据时，此方法不是正确的方法。而且，使用 get 方法可以提交的数据大小仅限于 3000 个字符。

post

在表单提交时，post 值不将表单数据附加到该表单的 URL。因此，数据不会显示在 URL 中，这是一种用于提交数据的安全方法。而且，可以使用 post 方法发送大量数据。该方法可用于发送文本数据和图像数据。

autocomplete

autocomplete 属性用于指定表单是应打开还是关闭自动完成功能。如果打开该功能，浏览器将根据用户以前已输入的值在字段中自动键入值。以下代码段用于指定表单的 autocomplete 属性：

```
<FORM ID="fileID" autocomplete= "on">
```

novalidate

novalidate 属性指定在提交数据时，浏览器不验证表单中的数据。它是不包含任何值的空属性。以下代码段用于指定表单的 novalidate 属性：

```
<FORM ID="fileID" novalidate>
```

target

target 属性用于指定在提交表单后必须显示所获得响应的框架或窗口的名称。以下语法用于指定表单的 target 属性：

```
<FORM target="_blank|_self|_parent|_top|frame_name">
```

target 属性可以具有以下任一值：

- **_blank**：指定应在新框架或窗口中显示响应。
- **_self**：指定应在同一框架中显示响应。
- **_parent**：指定应在父框架或窗口中显示响应。
- **_top**：指定应在窗口的完整正文中显示响应。
- **frame_name**：指定应在指定框架中显示响应。

在 BookYourHotel 场景中，以下代码用于创建 **OrderFoodOnline** 表单：

```
<FORM name= "OrderFoodOnline" ID= "Order_Food" method= "post">
.....
</FORM>
```

研究表单元素

您需要进一步设计 **OrderFoodOnline** 表单以便它可用于接受用户输入。为此，您需要在表单上添加各种输入字段。可以使用以下标记将这些字段添加到表单：

- `<INPUT>`
- `<SELECT>`
- `<LABEL>`
- `<FIELDSET>`
- `<TEXTAREA>`
- `<DATAList>`
- `<KEYGEN>`
- `<OUTPUT>`
- `<BUTTON>`

`<INPUT>`

`<INPUT>` 标记用于在表单中创建输入字段。这些字段用于接受用户的输入。输入字段有各种类型，例如文本框、单选按钮或复选框。输入字段的类型由其 `type` 属性的值确定。`<INPUT>` 标记具有一些重要属性，例如 `type`、`value`、`name`、`ID`、`autocomplete`、`autofocus`、`form`、`required`、`pattern` 和 `placeholder`。

定义 `type` 属性

`<INPUT>` 标记的 `type` 属性定义要添加到表单上的输入字段的类型。`type` 属性具有以下值：

- **text**：创建单行可编辑文本字段。当 `type` 属性的值为 `text` 时，还可以指定另外两个属性 `size` 和 `maxlength`。`size` 属性用于限制表单中文本字段的字符宽度。`maxlength` 属性定义可以输入文本字段的最大字符数。以下代码段用于创建文本字段：

```
First Name:<INPUT type="text" name="fname" size="20" maxlength="20">
Last Name:<INPUT type="text" name="lname" size="20" maxlength="20">
```

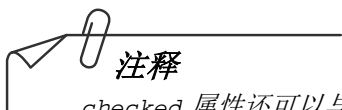
在上述代码段中，创建了两个文本字段 `fname` 和 `lname`。这两个字段的宽度都是 20 个字符，可以最多接受 20 个字符。

- **password**：创建密码字段，它将不显示用户正在输入的字符。它隐藏实际的字符，而显示每个字符的屏蔽值，例如 `****`。以下代码段用于创建密码字段：

```
<INPUT type="password" name="accountpasswrld">
```

- **radio**：创建单选按钮，它使用户能够从一组给定的选项中选择一个选项。当 `type` 属性的值为 `radio` 时，还可以指定另外一个属性 `checked`。`checked` 属性用于指定当页面加载时，单选按钮已预先选中。以下代码段用于创建单选字段：

```
<INPUT type="radio" name="Rating of Hotel" checked>
5-Star<BR/>
<INPUT type="radio" name="Rating of Hotel"> Budgeted
```



checked 属性还可以与 *checkbox* 输入字段一起使用。

在上述代码段中，创建了两个单选按钮 5-Star 和 Budgeted。checked 属性应用到第一个单选按钮 5-Star。因此，它显示为默认已选中。

为组中的单选按钮提供了同一名称，以确保用户一次只能选择一个单选按钮。

将显示单选按钮，如下图所示。

单选按钮

- **checkbox:** 创建复选框，它使用户能够从一组给定的选项选择一个或多个选项。以下代码段用于创建复选框字段：

```
<INPUT type="checkbox" name="Cuisine1" value="Continental Cuisine">
Continental Cuisine
<INPUT type="checkbox" name="Cuisine2" value="Chinese Cuisine"> Chinese
Cuisine
```

在上述代码段中，创建了两个复选框 Continental Cuisine 和 Chinese Cuisine，如下图所示。

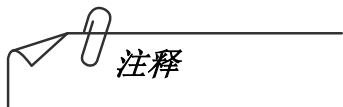
复选框



本章稍后将讨论 *value* 属性。

- **submit:** 创建提交按钮，它将表单数据提交到表单的 *action* 属性中指定的位置。当 *type* 属性的值为 *submit* 时，还可以指定一些其他属性，例如 *formaction*、*formmethod*、*formtarget* 和 *formnovalidate*。以下列表中提供了这些属性的描述：
 - **formaction:** 用于指定单击提交按钮时将把表单数据提交到的 URL。提交按钮的 *formaction* 属性中指定的 URL 将重写 *<FORM>* 标记的 *action* 属性中指定的 URL。因此，表单将被强制提交到提交按钮的 *formaction* 属性中指定的 URL，而不会提交到表单的 *action* 属性中指定的 URL。

- **formmethod**: 用于指定方法（例如 `get` 和 `post`），通过该方法表单数据将会被发送到表单的 `action` 属性中指定的文件或 URL。为提交按钮的 `formmethod` 属性指定的值将重写 `<FORM>` 标记的 `method` 属性的值。
- **formtarget**: 用于指定提交表单时将显示响应的框架或窗口的名称。为提交按钮的 `formtarget` 属性指定的值将重写 `<FORM>` 标记的 `target` 属性的值。
- **formnovalidate**: 默认情况下验证每个表单，除非您将 `novalidate` 属性与 `<FORM>` 标记一起使用。提交按钮的 `formnovalidate` 属性用于强制表单的行为与带有 `novalidate` 属性的表单一样。



注释

`formaction`、`formmethod`、`formtarget` 和 `formnovalidate` 属性也可与输入类型 `image` 和 `<BUTTON>` 标记一起使用。

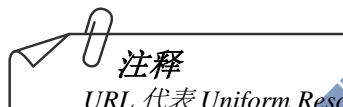
本章稍后将讨论输入类型 `image` 和 `<BUTTON>` 标记。

- **reset**: 创建重置按钮，它将清除表单字段中用户所输入的值。以下语法用于 创建重置字段：

```
<INPUT type="reset" name="reset">
```

- **URL**: 添加用于输入网站的 URL 的字段。提交表单时将自动验证此字段的值是否正确。以下语法用于 创建 URL 字段：

```
<INPUT type="url" name="locator">
```



注释

URL 代表 *Uniform Resource Locator*（统一资源定位符）。它是通过因特网访问的 Web 上提供的唯一文件或资源的地址。`www.w3schools.com` 是浏览器地址栏中指定的用于连接 `w3schools.com` 网站主页的 URL 的示例。

- **email**: 在 HTML 表单中创建字段来接受用户的电子邮件地址。您甚至可以 使用普通的文本字段来接受 电子邮件地址。但是，需要验证每个电子邮件地址是否为其正确格式 `<username>@<domainname>`。如果使用文本字段接受电子邮件地址，则很难执行此类验证。但是，如果使用电子邮件字段，那么在提交表单时将自动验证电子邮件地址。

当 `type` 属性的值为 `email` 时，还可以指定另外一个属性 `multiple`。`multiple` 属性用于指定用户可以在字段中输入多个值。以下代码段用于 创建电子邮件字段：

```
<INPUT type="email" name="email_id" multiple>
```

- **range**: 创建滑块控件以输入某一范围内的数值。滑块的默认范围是 0 到 100。当 `type` 属性的值为 `range` 时，还可以指定一些其他属性，例如 `min`、`max`、`value` 和 `step`。`min` 属性用于指定范围内的最小值。`max` 属性用于指定范围内的最大值。`value` 属性存储与范围字段关联的当前

值。step 属性用于指定当您移动滑块时范围字段的值将增加或减少的数字。以下代码段用于创建 范围字段：

```
<INPUT type="range" max="50" min="10" step="5" value="10">
```

上述代码段创建了默认值为 10、最小值为 10 并且最大值为 50 的范围。当您使用滑块指定数值时，该值将在最小值和最大值限制内增加 5。将显示范围字段，如下图所示。



范围字段

注释

min、max 和 step 属性还可以与输入类型（例如 number、date 和 time）一起使用。本章稍后将讨论输入类型 number、date 和 time。

- date: 用于在 HTML 表单中定义日期字段。它允许用户选择日期。以下代码段用于创建 日期字段：

```
<INPUT type="date" name="bday">
```

上述代码段创建了一个日期字段，如下图所示。

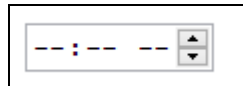


日期字段

- time: 用于在 HTML 表单中定义时间字段。它允许用户选择时间。以下代码段用于创建 时间字段：

```
<INPUT type="time" name="usr_time">
```

上述代码段创建了一个时间字段，如下图所示。

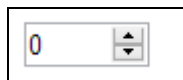


时间字段

- number: 用于创建输入数值的输入字段。以下代码段用于创建 数字字段：

```
<INPUT type="number" name="quantity" min="0" max="50">
```

上述代码段创建了最小值为 0、最大值为 50 的数字字段。将显示数字字段，如下图所示。



数字字段

- tel: 用于接受用户的电话号码。以下代码段用于创建 电话号码字段：

```
<INPUT type="tel" name="usrtel">
```

- image: 用于指定要用作 submit 按钮的图像。当 type 属性的值为 image 时，还可以指定一些其他属性，例如 height、width、alt 和 src。以下列表描述这些属性：
 - height 和 width: height 和 width 属性指定图像的高度和宽度。
 - alt: alt 属性用于指定在不能使用 src 属性中指定的图像时将显示的备用按钮。
 - src: src 属性指定将用作提交按钮的图像的 URL。

以下代码段用于 创建图像字段：

```
<INPUT type="image" src="img_submit.gif" alt="submit" width="48" height="48" />
```

上述代码显示将用作提交按钮的图像，如下图所示。



用作提交按钮的图像

定义 value 属性

value 属性指定字段的值。对于不同的输入类型它的行为不同，如以下列表所示：

- 对于 button、reset 和 submit 输入类型，value 属性定义在按钮表面显示的文本。
- 对于 text 和 password 输入类型，value 属性定义字段的默认值。
- 对于 checkbox、radio 和 image 输入类型，value 属性定义与输入字段关联的值。组中的所有单选按钮都有一个公用名。因此，value 属性用于区分组中的单选按钮。此外，在提交表单时，使用 value 属性标识选中的单选按钮。同样，用户可以选中一个或多个复选框并提交表单。可以使用 value 属性标识所有选中的复选框。

以下代码段用于指定 value 属性：

```
<INPUT type="text" name="fname" value="Enter Your First Name">
```

在上述代码段中，Enter Your First Name 赋给输入字段的 value 属性。

定义 name 属性

name 属性指定输入字段的名称。它用于在提交表单数据时标识表单字段。以下代码段用于指定输入字段的名称：

```
<INPUT type="text" name="Text1" />
```

在上述代码中，名称 Text1 赋给文本字段。

定义 ID 属性

ID 属性向输入字段提供唯一的 ID。此属性用于访问 CSS 或 JavaScript 代码中的输入字段。

以下代码段用于指定输入字段的唯一 ID：

```
<INPUT type="text" ID="value">
```

在上述代码中，ID value 赋给输入字段。

定义 autocomplete 属性

autocomplete 属性用于指定表单元素是应打开还是关闭自动完成功能。如果打开该功能，浏览器将根据用户早先已输入的值在表单字段中自动键入值。以下代码段用于指定表单的 autocomplete 属性：

```
<INPUT type="email" name="email" autocomplete="on">
```

在上述代码段中，email 输入字段的 autocomplete 属性设置为 on。

定义 autofocus 属性

autofocus 属性用于确保在网页加载时表单元素具有焦点。以下代码段用于指定输入字段的 autofocus 属性：

```
<INPUT type="text" name="lname" autofocus>
```

在上述代码段中，在页面加载时名称为 lname 的文本字段将具有焦点。

定义 required 属性

required 属性用于指定在提交表单时输入字段不得留空。required 属性可以与以下输入类型一起使用，例如 text、tel、email、password、number、checkbox 和 radio。以下代码段用于指定输入字段的 required 属性：

```
<INPUT type="text" name="username" required>
```

在上述代码段中，required 属性确保名为 username 的文本字段在表单提交时不得留空。

定义 pattern 属性

pattern 属性用于指定检查元素的值所依据的正则表达式。pattern 属性可以与以下输入类型一起使用，例如 text、url、tel、email 和 password。

下表列出了可应用到各种用于创建模式的输入字段的表达式。

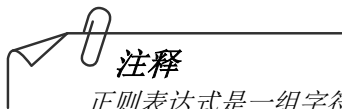
表达式	描述
[abc]	查找方括号内指定的字符。
[^abc]	查找方括号内未指定的字符。
[0-9]	查找介于0到9的任何数字。
[A-Z]	查找介于大写A到大写Z的字符。
[a-z]	查找介于小写a到小写z的字符。
A-z	查找介于大写A到小写z的字符。

创建模式可应用的表达式

以下代码用于指定输入字段的 pattern 属性：

```
<INPUT type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter code">
```

在上述代码中，pattern 属性的值确保文本字段仅接受大写或小写的三个字母的字母。它将不允许用户在文本字段中输入任何数字或特殊字符。



注释

正则表达式是一组字符，用于指定模式。

定义 placeholder 属性

placeholder 属性用于指定直到用户输入值才会显示的输入字段的示例值。placeholder 属性可以与以下输入类型一起使用，例如 text、tel、email、password 和 number。以下代码段用于指定输入字段的 placeholder 属性：

```
<INPUT type="text" placeholder="Type your first name" name="fname">
```

在上述代码段中，文本 Type your First name here 将显示在名为 fname 的文本字段中。当加载建议用户在字段中输入名字的页面时，将显示该文本。

将显示将 placeholder 属性与文本字段一起使用所派生的输出，如下图所示。

使用 Placeholder 属性所派生的输出

思考以下代码，它用于为 **OrderFoodOnline** 网页创建输入字段，例如 text、date、tel、email 和 radio:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:#FFEB CD;
}
.autostyle2{
color:red;
font-size:20px;
text-align:center;
}
</STYLE>
</HEAD>
<BODY bgcolor="#FFEB CD">
<FORM action = "registration.html">
<TABLE>
<TR class="autostyle2">
<TD colspan="4">OrderFoodOnline</TD></TR>
<TR>
<TD WIDTH="100%" COLSPAN="4">
</TD>
</TR>
<TR>
<TD> Name:</TD>
<TD><INPUT type="text" name="usrname" placeholder="Enter your name"
required></TD>
</TR>
<TR>
<TD>Date:</TD>
<TD><INPUT type="date" name="date" required></TD>
</TR>
<TR>
<TD>email ID:</TD>
<TD><INPUT type="email" name="usrmail" placeholder="Enter your email ID"
required></TD>
</TR>
<TR>
<TD>Contact Number:</TD>
<TD><INPUT type="tel" name="usrtel" placeholder="Enter your phone number"
required></TD>
</TR>
<TR><TD>Select Food:</TD>
<TD>Non-vegetarian<INPUT type="Radio" NAME="rd1">
Vegetarian<INPUT type="Radio" NAME="rd1" ></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

上述代码创建了一个文本框来接受客户的姓名、一个日期字段来接受日期、一个电子邮件字段来接受经过验证的客户电子邮件地址、一个电话号码字段来接受电话号码以及一个单选按钮来接受客户的食物偏好。

将显示 OrderFoodOnline 表单，如下图所示。



OrderFoodOnline 表单

<SELECT>

<SELECT> 标记是一个容器标记。它在表单上创建下拉列表。它具有以下属性：

- **multiple**：用于允许用户使用 Ctrl 键从下拉列表中选择多个值。
- **name**：用于指定将在提交表单时使用的选择列表的名称。
- **size**：用于指定选择列表或下拉列表中可见项的数目。默认值是 1。如果此属性的值大于 1，那么表单字段将是一个列表。
- **autofocus**：用于确保在页面加载时焦点在下拉列表上。
- **form**：用于指定 <SELECT> 标记所属的一个或多个表单的名称。

以下代码用于创建 <SELECT> 标记：

```
<SELECT name= "5-Star_Hotels" size=1 multiple></SELECT>
```

上述代码将创建名为 5-Star_Hotels 大小为 1 的下拉列表。multiple 属性允许用户从列表中选择多个项。但是，<SELECT> 标记仅创建下拉列表。它不将列表项嵌入列表中。要指定列表项，需要将标记 <OPTION> 和 <OPTGROUP> 与 <SELECT> 标记一起使用。

定义 <OPTION> 标记

它始终用于 <SELECT> 标记内，不能用作单独的标记。它用于在下拉列表中创建选项列表，具有以下属性：

- **selected**：用于指示在浏览器中加载页面时，特定选项已预先选中。
- **value**：用于指示当用户选择某个选项时，将在表单提交时发送的该选项的值。
- **disabled**：用于指示页面加载时应禁用的选项。

思考以下代码，它用于在 **OrderFoodOnline** 表单中创建下拉列表：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
.....</TR>
<TR>
<TD>Select Restaurant:</TD>
<TD><SELECT>
<OPTION value="opt1">Select Your Restaurants</OPTION>
<OPTION value="opt2">La Figa</OPTION>
<OPTION value="opt3">Benihana</OPTION>
<OPTION value="opt4">Gallipoli</OPTION>
<OPTION value="opt5">Kings Road SteakHouse</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD>Drinks:</TD>
<TD><SELECT>
<OPTION value="opt6">Select Your Drink</OPTION>
<OPTION value="opt7">Cappuccino</OPTION>
<OPTION value="opt8">Caffelatte</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD> Soups:</TD>
<TD><SELECT>
<OPTION value="opt9">Select the Soup of Your Choice</OPTION>
<OPTION value="opt10">Minestrone</OPTION>
<OPTION value="opt11">Fonduta</OPTION>
<OPTION value="opt12">Pasta e fagioli</OPTION>
</SELECT></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

在上述代码中，创建了下拉列表 Drinks、Soups 和 Restaurant。将显示使用 <OPTION> 标记所派生的输出，如下图所示。

使用 <OPTION> 标记所派生的输出

定义 <OPTGROUP> 标记

<OPTGROUP> 标记用于将相关选项分为一组。通常在选项列表较长并且您想要将相关选项分为一组以使其更简单时使用该标记。<OPTGROUP> 标记可以具有以下属性：

- disabled: 用于指示页面加载时应显示为禁用的选项组。
- label: 用于为选项组指定标签。

思考以下代码，它用于在 **OrderFoodOnline** 网页中创建下拉列表：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
.....</TR>
<TR>
<TD>Dishes:</TD>
<TD><SELECT>
<OPTION value="opt13">Select the Dishes of Your Choice</OPTION>
<OPTGROUP label="Italian">
<OPTION value="opt14">Pasta</OPTION>
<OPTION value="opt15">Fish</OPTION>
<OPTION value="opt16">Rice</OPTION>
</OPTGROUP>
<OPTGROUP label="Chinese">
<OPTION value="opt17">Chowmein</OPTION>
<OPTION value="opt18">Manchurian</OPTION>
<OPTION value="opt19">Water Chessnut Cake</OPTION>
```



```

</OPTGROUP>
</SELECT></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>

```

在上述代码中，<OPTGROUP> 标记用于将项分为两组：Italian 和 Chinese。而且，<OPTION> 标记括在<OPTGROUP> 标记之间，以指定类别中的项。

将显示使用 <OPTGROUP> 标记所派生的输出，如下图所示。

OrderFoodOnline

Name:

Date:

email ID:

Contact Number:

Select Food: ☐ Non-vegetarian ☐ Vegetarian

Select Restaurant:

Drinks:

Soups:

Dishes:

Select the Dishes of Your Choice

Italian

Pasta

Fish

Rice

Chinese

Chowmein

Manchurian

Water Chessnut Cake

使用 <OPTGROUP> 标记所派生的输出

<LABEL>

<LABEL> 标记用于为输入字段定义标签。您还可以为 <OUTPUT> 标记定义标签。<LABEL> 元素不为用户呈现任何特殊内容。但是，它为用户提供功能的方式是：如果用户单击 <LABEL> 元素内的文本，那么相应字段将会自动选中。为此，<LABEL> 标记的 for 属性应等同于相关输入字段的 ID 属性。

<LABEL> 标记具有以下属性:

- for: 用于将 <LABEL> 标记与输入字段绑定, 并且应与输入字段的 ID 属性具有相同的值。
- 指定 <LABEL> 标记所属的表单元素的 ID 名称。
- form: 用于指定 <LABEL> 标记所属的一个或多个表单的名称。

思考以下代码, 它用于为 **OrderFoodOnline** 网页中的输入字段创建标签:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:#FFEB CD;
}.autostyle2{
color:red;
font-size:20px;
text-align:center;
}
</STYLE>
</HEAD>
<BODY>
<FORM action = "registration.html">
<TABLE>
<TR class="autostyle2">
<TD colspan="2">OrderFoodOnline</TD></TR>
<TR>
<TD WIDTH="100%" COLSPAN="2">
</TD>
</TR>
<TR>
<TD><LABEL for="name">Name:</LABEL></TD>
<TD><INPUT type="text" name="usrname" ID="name" placeholder="Enter your
name" required></TD>
</TR>
<TR>
<TD><LABEL for="date">Date:</LABEL></TD>
<TD><INPUT type="date" name="date" ID="date" required></TD>
</TR>
<TR>
<TD><LABEL for="email">email ID:</LABEL></TD>
<TD><INPUT type="email" name="usrmail" ID="email" placeholder="Enter your
email ID" required></TD>
</TR>
<TR>
<TD><LABEL for="number">Contact Number:</LABEL></TD>
<TD><INPUT type="tel" name="usrtel" ID="number" placeholder="Enter your phone
number" required></TD></TR>
<TR><TD><LABEL for="food">Select Food:</LABEL></TD>
<TD><LABEL for="nonveg">Non-vegetarian</LABEL>
<INPUT type="radio" name="food" ID="nonveg">
<LABEL for="veg">Vegetarian</LABEL>
<INPUT type="radio" name="food" ID="veg"></TD>
</TR>
```

```

<TR>
<TD><LABEL for="restro">Select Restaurant:</LABEL></TD>
<TD><SELECT>
    <OPTION value="opt1">Select Your Restaurants</OPTION>
<OPTION value="opt2">La Figa</OPTION>
<OPTION value="opt3">Benihana</OPTION>
    <OPTION value="opt4">Gallipoli</OPTION>
    <OPTION value="opt5">Kings Road
SteakHouse</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="drinks">Drinks:</LABEL></TD>
<TD><SELECT>
    <OPTION value="opt6">Select Your Drink</OPTION>
<OPTION value="opt7">Cappuccino</OPTION>
<OPTION value="opt8">Caffelatte</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="soups">Soups:</LABEL></TD>
<TD><SELECT>
<OPTION value="opt9">Select the Soup of Your Choice</OPTION>
<OPTION value="opt10">Minestrone</OPTION>
    <OPTION value="opt11">Fonduta</OPTION>
<OPTION value="opt12">Pasta e fagioli</OPTION>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="dishes">Dishes:</LABEL></TD>
<TD><SELECT>
<OPTION value="opt13">Select the Dishes of Your Choice</OPTION>
<OPTGROUP label="Italian">
<OPTION value="opt14">Pasta</OPTION>
<OPTION value="opt15">Fish</OPTION>
<OPTION value="opt16">Rice</OPTION>
</OPTGROUP>
<OPTGROUP label="Chinese">
<OPTION value="opt17">Chowmein</OPTION>
    <OPTION value="opt18">Manchurian</OPTION>
<OPTION value="opt19">Water Chessnut Cake</OPTION>
</OPTGROUP>
</SELECT></TD>
</TR>
<TR>
<TD><LABEL for="Order For:">Order For:</LABEL></TD>
<TD><LABEL for="pickup">Take Away</LABEL>
<INPUT type="radio" name="pickup" ID="pickup">
<LABEL for="home">Home Delivery</LABEL>
<INPUT type="radio" name="pickup" ID="home"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>

```

在上述代码中，<LABEL> 标记用于标记 **OrderFoodOnline** 网页中的输入字段。可以通过单击输入字段的标签来选择字段。将显示使用 <LABEL> 标记所派生的输出，如下图所示。



The screenshot shows a web form titled "OrderFoodOnline" in red text. The form is enclosed in a light orange border. It contains the following fields and labels:

- Name:** A text input field with the placeholder "Enter your name".
- Date:** A date picker input field showing "mm/dd/yyyy".
- email ID:** A text input field with the placeholder "Enter your email ID".
- Contact Number:** A text input field with the placeholder "Enter your phone number".
- Select Food:** Two radio buttons labeled "Non-vegetarian" and "Vegetarian".
- Select Restaurant:** A dropdown menu with the text "Select Your Restaurants".
- Drinks:** A dropdown menu with the text "Select Your Drink".
- Soups:** A dropdown menu with the text "Select the Soup of Your Choice".
- Dishes:** A dropdown menu with the text "Select the Dishes of Your Choice".
- Order For:** Two radio buttons labeled "Take Away" and "Home Delivery".

使用 <LABEL> 标记所派生的输出

<FIELDSET>

<FIELDSET> 标记用于将表单中的相关字段进行合并和分组。它在所选字段周围创建一个框。您还可以使用 <LEGEND> 标记定义字段集的描述。<LEGEND> 标记与 <FIELDSET> 标记一起使用来定义字段集的标题。这是组织表单元素及其描述以使用户易于理解的最简单的方法。

<FIELDSET> 标记可以具有以下属性：

- **disabled:** disabled 属性用于指示页面加载时应显示为禁用的一组字段。
- **form:** form 属性用于指定 <FIELDSET> 标记所属的一个或多个表单的名称。
- **name:** name 属性用于指定字段集的名称。

思考以下代码，它用于分组 **OrderFoodOnline** 网页上字段集内的下拉列表：

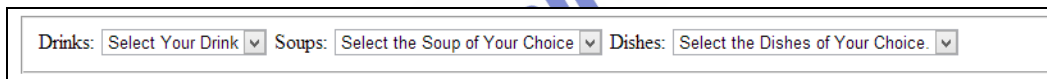
```
<!DOCTYPE HTML>
<HTML>
<BODY>
<FORM>
<FIELDSET>
Drinks:
<SELECT>
<OPTION value="opt1">Select Your Drink</OPTION>
<OPTION value="opt2">Cappuccino</OPTION>
<OPTION value="opt3">Caffelatte</OPTION>
</SELECT>
```

```

Soups:
<SELECT>
<OPTION value="opt4">Select the Soup of Your Choice</OPTION>
<OPTION value="opt5">Minestrone</OPTION>
<OPTION value="opt6">Fonduta</OPTION>
<OPTION value="opt7">Pasta e fagioli</OPTION>
</SELECT>
Dishes:
<SELECT>
<OPTION value="opt8">Select the Dishes of Your Choice.</OPTION>
<OPTGROUP label="Italian">
<OPTION value="opt9">Pasta</OPTION>
<OPTION value="opt10">Fish</OPTION>
<OPTION value="opt11">Rice</OPTION>
</OPTGROUP>
<OPTGROUP label="Chinese">
<OPTION value="opt12">Chowmein</OPTION>
<OPTION value="opt13">Manchurian</OPTION>
<OPTION value="opt14">Water Chessnut Cake</OPTION>
</OPTGROUP>
</SELECT>
</FIELDSET>
</FORM>
</BODY>
</HTML>

```

上述代码将食物项分组到一个框中。将显示使用 `<FIELDSET>` 标记所派生的输出，如下图所示。



使用 `<FIELDSET>` 标记所派生的输出

<TEXTAREA>

`<TEXTAREA>` 标记创建一个字段，用户可以在其中输入大量文本。`<TEXTAREA>` 标记具有以下属性：

- rows
- cols

rows

`rows` 属性帮助您设置在无须将字段向上或向下滚动的情况下就能看到的文本行数。

cols

`cols` 属性帮助您设置在无须将字段向右或向左滚动的情况下就能看到的文本列数。

思考以下代码，它用于在 **OrderFoodOnline** 网页中创建文本区域以接受用户的地址：

```

<!DOCTYPE HTML>
<HTML>
<HEAD>
.....</TR>
<TR>

```

```

<TD><LABEL for="Orderfor">Address:</LABEL></TD>
<TD><TEXTAREA rows="3" cols="16" ID="Orderfor">
Enter Your Address Here
</TEXTAREA></TD>
</TR></TABLE>
</FORM>
</BODY>
</HTML>

```

将在下图中显示使用 <TEXTAREA> 标记所派生的输出。

使用 <TEXTAREA> 标记所派生的输出

<DATA LIST>

<DATA LIST> 标记用于为输入字段创建预定义选项列表。它用于在输入字段上提供自动完成功能，以便用户在输入数据时可以查看预定义选项的下拉列表。

思考以下代码，它用于在 **OrderFoodOnline** 网页中创建名为 **State** 的 datalist:

```

<!DOCTYPE HTML>
<HTML>
<HEAD>
..... </TR>
<TR>
<TD><LABEL for="state">State:</LABEL></TD>
<TD><INPUT list="stat" name="stat" ID="state">
<DATA LIST ID="stat">
<OPTION value="Alabama">

```

```

<OPTION value="California">
<OPTION value="Delaware">
<OPTION value="Florida">
<OPTION value="Hawaii">
</DATA LIST><BR>
</TD>
</TR> </TABLE>
</FORM>
</BODY>
</HTML>

```

上述代码创建了一个 **datalist**，从而在用户输入州名称的前几个字符时，将在下拉列表中显示与这些首字母匹配的名称，如下图所示。

The screenshot shows a web form titled "OrderFoodOnline" in red text. The form has a light orange background and contains the following elements:

- Name:** A text input field with placeholder text "Enter your name".
- Date:** A date picker showing "mm/dd/yyyy".
- email ID:** A text input field with placeholder text "Enter your email ID".
- Contact Number:** A text input field with placeholder text "Enter your phone number".
- Select Food:** Two radio buttons labeled "Non-vegetarian" and "Vegetarian".
- Select Restaurant:** A dropdown menu with the text "Select Your Restaurants".
- Drinks:** A dropdown menu with the text "Select Your Drink".
- Soups:** A dropdown menu with the text "Select the Soup of Your Choice".
- Dishes:** A dropdown menu with the text "Select the Dishes of Your Choice".
- Order For:** Two radio buttons labeled "Take Away" and "Home Delivery".
- Address:** A text area with placeholder text "Enter Your Address Here".
- State:** A dropdown menu. The input field shows the letter "c", and a blue dropdown list is open, showing "California" as the selected option.

使用 `<DATA LIST>` 标记所派生的输出

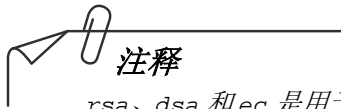
在上图中，当用户输入字符 **c** 时，将在列表中提示 **California**。

<KEYGEN>

<KEYGEN> 标记用于指定表单中密钥对生成的字段。只要提交表单，将生成私钥和公钥，其中私钥在本地存储，而公钥则发送给服务器。由于公钥存储在服务器中，所以它可用于在以后验证用户。

<KEYGEN> 标记可以具有以下属性：

- autofocus: 用于指定在网页加载时 <KEYGEN> 标记自动获取焦点。
- disabled: 用于指示 <KEYGEN> 标记在页面加载时应显示为禁用。
- name: 用于指定 <KEYGEN> 标记的名称。
- keytype: 用于指定密钥的安全算法。它接受各种安全算法的名称作为其值，例如 rsa、dsa 和 ec。
- form: 用于指定 <KEYGEN> 标记所属的一个或多个表单的名称。



rsa、dsa 和 ec 是用于公钥加密的不同安全算法。

可以使用以下代码段创建 <KEYGEN> 标记：

```
<KEYGEN name="key1" keytype="rsa">
```

上述代码使用安全算法 rsa 创建密钥对生成的字段。

<OUTPUT>

<OUTPUT> 标记用于表示计算的结果。<OUTPUT> 标记可以具有以下属性：

- for: 用于指定所使用的输入字段和为计算生成的结果之间的关系。
- form: 用于指定 <OUTPUT> 标记所属的一个或多个表单的名称。
- name: 用于指定 <OUTPUT> 标记的名称。

思考以下代码，它用于通过使用 <OUTPUT> 标记获取两个数字的相乘结果：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<FORM ONINPUT="mul.value=parseInt(val1.value)*parseInt(val2.value)">
Value1:<INPUT type="text" name="val1">
* Value2:<INPUT type="text" name="val2">
=<OUTPUT name="mul" for="val1 val2"></OUTPUT>
</FORM>
</BODY>
</HTML>
```

上述代码创建了两个输入字段 val1 和 val2 来接受用户输入。然后，ONINPUT 事件在表单上执行，并将输入字段中显示的值 val1 和 val2 相乘。接着，<OUTPUT> 标记用于显示计算的结果。将显示使用 <OUTPUT> 标记所派生的输出，如下图所示。

Value1: <input type="text" value="10"/>	* Value2: <input type="text" value="20"/>	=200
-----------------------------------------	-------------------------------------------	------

使用<OUTPUT> 标记所派生的输出

<BUTTON>

<BUTTON> 标记用于创建按钮。但是，与输入类型 submit 不同，您可以使用 <P> 或 标记指定按钮中的文本或图像。<BUTTON> 标记具有以下属性：

- type: 用于指定按钮的类型。它可以接受以下值，例如 button、submit 和 reset。button 值创建一个简单的按钮，submit 值创建提交表单数据的按钮，reset 值创建重置表单字段的按钮。
- name: 用于指定按钮的名称。
- form: 用于指定按钮所属的一个或多个表单的名称。
- autofocus: 用于指定在网页加载时按钮自动获取焦点。
- disabled: 用于指示按钮在页面加载时应显示为禁用。

思考以下代码，它用于在 **OrderFoodOnline** 网页中创建按钮 Submit 和 Reset:

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
#button{
Margin-left:50px;
}
....
</HEAD>
<BODY>
.....</TR>
<TR>
<TD>
<BUTTON ID="button" type="submit"><P>Submit</P></BUTTON></TD><TD>
<BUTTON type="reset"><P>Reset</P></BUTTON>
</TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

上述代码创建了两个按钮 Submit 和 Reset，如下图所示。

OrderFoodOnline

Name:

Date:

email ID:

Contact Number:

Select Food: Non-vegetarian ☐ Vegetarian ☐

Select Restaurant:

Drinks:

Soups:

Dishes:

Order For: Take Away ☐ Home Delivery ☐

Address:

State:

使用 `<BUTTON>` 标记所派生的输出



小问题:

以下哪个属性用于确保在网页加载时表单元素具有焦点?

1. `placeholder`
2. `autofocus`
3. `formtarget`
4. `formnovalidate`

答案:

2. `autofocus`



活动 5.1: 设计 HTML 表单

操作网页的组件

BookYourHotel.com 的网站使客户能够通过单击旅馆的图像来访问每家旅馆的在线预订表单。此外，还在网页上显示一个时钟，该时钟每秒都会动态更新以向用户显示当前时间。

为在网页上实现此功能，使用了 **JavaScript**。它是基于对象的语言，将浏览器窗口中的每个元素（例如图像或按钮）都当做对象。每个对象都有一组与其关联的预定义属性和事件。**JavaScript** 还提供各种类型的内置对象，例如浏览器对象和表单对象，它们有助于使网页变得动态且可交互。

使用浏览器对象

在浏览器中显示网页时，用户不仅可以访问关于当前网页的信息，还可以访问其他详细信息，例如有关窗口和屏幕的信息、用户在过去曾访问过的页面或者正用于查看文档的浏览器版本。

所有这些信息都可以使用浏览器对象进行访问。此外，浏览器将当前所显示的网页解析为多个对象，例如窗口、屏幕和文档。定义浏览器内容和浏览器本身的对象称为浏览器对象。通过浏览器对象能够检索和操作关于浏览器的信息，例如窗口大小、高度、宽度和名称。通过这些对象还能够访问信息，例如浏览历史记录和浏览器的当前版本。

JavaScript 定义网页上的以下浏览器对象：

- window
- document
- navigator
- screen
- history
- location

浏览器对象表示浏览器环境，并为其访问和操作提供属性和方法。浏览器环境指的是组件，例如显示文档的窗口和包含有关用户所访问网页的信息的历史记录列表。您可以使用 `window` 对象设置浏览器窗口的大小、名称和默认状态。您还可以使用 `location` 对象操作文档的 `URL`。同样，您可以使用 `navigator` 对象访问显示网页的浏览器版本和浏览器名称。该信息对于在浏览器中正确显示网页可能非常有用。

使用 `window` 对象

`window` 对象是 **JavaScript** 对象层次结构中最高级的对象之一。它表示显示文档的浏览器窗口。它还可以是多个框架的组合。窗口中的每个框架本身都是一个窗口。

下表列出了 `window` 对象的一些属性。

属性	描述	语法
<code>defaultStatus</code>	是字符串值，包含在窗口的状态栏上显示的默认文本。	<code>window.defaultStatus</code>
<code>document</code>	是对窗口中所显示文档的引用。	<code>window.document</code>
<code>frames[]</code>	是表示窗口中的所有框架的数组。您可以使用 <code>frames[]</code> 属性引用特定框架。	<code>window.frames[i]</code> ，其中 <i>i</i> 是窗口中特定框架的索引。
<code>frames.length</code>	是一个整数值，表示窗口中的框架数。	<code>window.frames.length</code>
<code>name</code>	返回或设置窗口的名称。	<code>window.name</code>
<code>parent</code>	返回当前窗口的父窗口。	<code>window.parent</code>
<code>self</code>	返回当前窗口。	<code>window.self</code>
<code>top</code>	返回最上层浏览器窗口。最上层窗口是覆盖所有打开窗口的当前活动窗口。	<code>window.top</code>
<code>status</code>	是字符串值，用于在窗口的状态栏上设置文本。	<code>window.status</code>


window 对象的属性

下表列出了 window 对象的一些方法。

方法	描述	语法	示例
<code>open()</code>	打开一个新的浏览器窗口。	<p><code>window.open(URL, Name, specs, replace);</code></p> <p>其中, URL 是要显示的网页的地址。</p> <p>Name 是在其中显示网页的窗口的名称。</p> <p>specs 以逗号分隔的属性列表形式定义窗口的规范, 例如其高度和宽度。</p> <p>replace 用于使用 <code>true</code> 或 <code>false</code> 值分别确定是替换当前文档还是在历史记录列表中创建新条目。</p>	<p><code>window.open</code> <code>("index.html",</code> <code>"Home",</code> <code>"height=100,width=200", false);</code></p> <p>其中, URL 是 <code>index.html</code>。</p> <p>窗口的名称是 <code>Home</code>。</p> <p>屏幕的高度和宽度分别是 100 和 200。</p>
<code>close()</code>	关闭当前窗口。	<code>window.close();</code>	<p><code>window.close();</code></p> <p>关闭当前窗口。</p>
<code>alert()</code>	在对话框中显示消息。	<p><code>window.alert(messageText);</code></p> <p>其中, <code>messageText</code> 是将在对话框中显示的文本。</p>	<p><code>window.alert("Hello");</code></p> <p>在对话框中显示消息 <code>Hello</code>。</p>
<code>setTimeout()</code>	在指定毫秒数后调用函数。	<code>window.setTimeout(function, time)</code>	<p><code>var t=setTimeout("alertMessage()", 3000);</code></p> <p>在 3 秒后调用 <code>alertMessage()</code> 函数。</p>

方法	描述	语法	示例
<code>clearTimeout()</code>	取消使用 <code>setTimeout()</code> 方法设置的计时器。它采用 <code>setTimeout()</code> 方法返回的 <code>timerID</code> 参数。	<code>window. clearTimeout(timerID)</code>	<code>t=setTimeout ("timedCount()",1000) ; clearTimeout(t);</code> 其中, <code>t</code> 是 <code>setTimeOut()</code> 方法返回的 <code>timerID</code> 参数。

window 对象的方法



注释
window 对象是编写 JavaScript 代码时的默认对象。因此, 不必使用点运算符显式限定 window 对象的方法和属性。

思考以下代码段, 它用于在用户单击旅馆图像时打开显示旅馆预订表单的新窗口:

```
function open_win()  
{  
window.open("HotelBooking1.html","height=100,width=200");  
}
```

在上述代码段中, 执行 `open_win()` 函数时, 它将在新窗口中打开 **purchase.html** 网页。

思考以下代码, 它用于使用 `setTimeout()` 方法在 **BookYourHotel.com** 网站的网页上显示时钟:

```
<!DOCTYPE HTML>  
<HTML>  
<HEAD>  
<SCRIPT type="text/javascript">  
function clockTime()  
{  
var todayDate=new Date();  
var hrs=todayDate.getHours();  
var mns=todayDate.getMinutes();  
var scs=todayDate.getSeconds();  
mns=check(mns);  
scs=check(scs);  
document.getElementByIdgetElementById('displayTime').innerHTML=hrs+":"+mns+":  
"+scs;  
t=setTimeout('clockTime()',1000);  
}  
  
function check(t)  
{  
if (t<10)
```

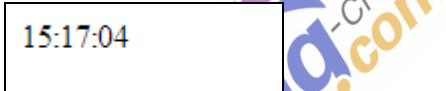
```

{
t="0" + t;
}
return t;
}
</SCRIPT>
</HEAD>
<BODY onload="clockTime()">
<DIV ID="displayTime"></DIV>
</BODY>
</HTML>

```

在上述代码中，调用了 `clockTime()` 函数来在网页上显示时钟。`clockTime()` 函数使用 `Date` 对象获取当前时间（以时、分、秒表示）。它使用 `setTimeout()` 函数每秒更新时间。`setTimeout()` 函数在每秒（1000 毫秒）后调用 `clockTime()` 函数。`clockTime()` 函数使用 `check(t)` 函数检查分钟数或秒数是否小于 10。如果为真，将在分钟数或秒数前显示 0。例如，如果当前时间是 14 时 45 分 3 秒，那么时间将显示为 14:45:03。

将显示在网页上显示时钟的上述代码的输出，如下图所示。



在网页上显示的时钟

使用 document 对象

`document` 对象从属于文档对象模型层次结构中的 `window` 对象。`document` 对象提供属性和方法来处理当前文档的很多方面，包括有关锚、表单、链接、标题、当前位置和 **URL** 以及当前颜色的信息。在 **Web** 浏览器窗口上加载的 **HTML** 文档充当 `document` 对象。

下表列出了 `document` 对象的一些属性。

属性	描述
<code>alinkColor</code>	是一个字符串值，表示活动链接的颜色。
<code>anchors[]</code>	是一个数组对象，包含对文档中所有锚元素的引用。
<code>bgColor</code>	是一个字符串值，表示文档的背景色。
<code>cookie</code>	是一个字符串值，包含数据的“名称 - 值”对，它会一直存在于客户机的内存中，直到清除 Web 浏览器或到达到期日为止。
<code>fgColor</code>	是一个字符串值，表示文档的文本颜色。
<code>forms[]</code>	是一个数组对象，包含对文档中每个表单的引用。表单元素包含在 <i>Form</i> 对象中。

属性	描述
<code>linkColor</code>	是一个字符串值，表示未访问链接的颜色。
<code>lastModified</code>	是一个字符串值，表示最后一次修改文档的日期和时间。
<code>links[]</code>	是一个数组对象，包含<A> 标记中所有元素的引用以及使用<AREA> 标记的元素。
<code>referrer</code>	是一个字符串值，表示从中访问当前文档的文档的 URL。
<code>Title</code>	是一个字符串值，表示文档的标题。
<code>vlinkColor</code>	是一个字符串值，表示已访问链接的颜色。

document 对象的属性

document 对象的一些广泛使用的方法有：

- `write()`
- `writeln()`
- `getElementsByName()`
- `getElementsByTagName()`
- `getElementById()`

write()

`document.write()` 方法使用户能够在网页上编写文本。以下语法用于使用 `document.write()` 方法在网页上编写文本 Hello!:

```
document.write("Hello!");
```

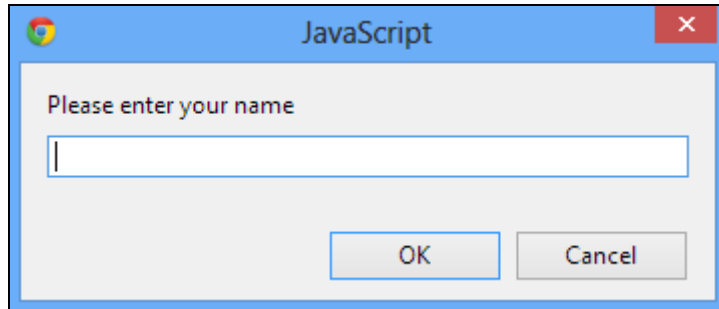
您还可以使用 `document.write()` 方法将 HTML 元素动态添加到网页。

例如，您可以接受名称作为输入，然后使用 `document.write()` 方法在网页上进行编写。您甚至可以在网页上动态指定内容的格式，如下代码所示：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<TITLE>JavaScript Write method Illustration</TITLE>
<STYLE>
body{
background-color:#DAA520;
}
</STYLE></HEAD>
<BODY>
<SCRIPT type="text/javascript">
{
var name= prompt("Please enter your name","");
document.write("<P>");
```

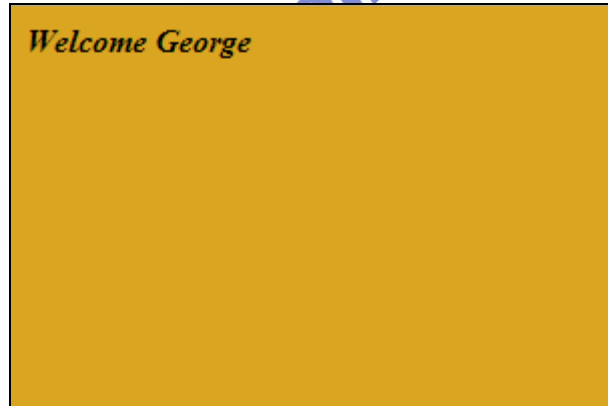
```
document.write("<I>");  
document.write("<B>");  
document.write("Welcome "+name);  
document.write("</B>");  
document.write("</I>");  
document.write("</P>");  
}  
</SCRIPT>  
</BODY>  
</HTML>
```

在上述代码中，使用 `prompt()` 方法提示用户输入其姓名，如下图所示。



提示框

假设用户在提示框中输入了姓名 **George**，将显示如下图所示的输出。



使用 `write()` 方法的代码的输出

在上述代码中，在 `<SCRIPT>` 标记中使用 `document.write()` 方法来在网页上显示欢迎消息。

`writeln()`

`writeln()` 方法还在网页上编写文本。唯一的区别是 `writeln()` 方法将回车符号附加到输出的末尾。回车符号用于在网页的单独行上编写数据。例如，思考以下代码：

```
<!DOCTYPE HTML>  
<HTML>
```

```

<BODY>
<PRE>
<SCRIPT type="text/javascript">
document.writeln("Hi!");
document.writeln("Welcome to our site!");
document.write("Have a ");
document.write("great day");
</SCRIPT>
</PRE>
</BODY>
</HTML>

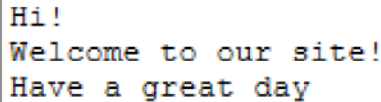
```

注释

`writeln()` 方法将换行符附加到文本的末尾。但是，在编写输出文本时 **HTML** 忽略了换行符。`<PRE>` 标记可用于在网页上显示预设格式的输出。当 `<SCRIPT>` 标记包含在 `<PRE>` 标记内时，将呈现使用 `document.writeln()` 方法附加的换行符。

执行上述代码段时，将显示消息 **Hi!**，并在此消息的末尾添加回车符号，以便在新行中显示下一条消息。

将显示上述代码段的输出，如下图所示。



使用 `writeln()` 方法所派生的输出

getElementsByTagName()

建议应使用唯一名称指定网页上的每一个元素。但是，多个元素还可以在网页中共享一个名称。`document.getElementsByTagName()` 方法用于访问具有 **HTML** 文档中所使用的指定名称的所有元素。此方法返回具有 **HTML** 文档中指定名称的所有元素的数组。以下语法用于 `getElementsByTagName()` 方法：

```
document.getElementsByTagName("name_of_the_element");
```

在上述语法中，`name_of_the_element` 指定要访问的元素的名称。

思考以下代码：

```

<!DOCTYPE HTML>
<HTML>
<SCRIPT type="text/javascript">
function count()
{
var x=document.getElementsByTagName("link");
alert(x.length + "Hyperlinks");
}

```

```

}
</SCRIPT>
<BODY>
<A name="link" href="" >Link 1</A><BR/>
<A name="link" href="" >Link 2</A><BR/>
<A name="link" href="" >Link 3</A><BR/>
<BR/>
<INPUT type="button" value="Count" onclick="count()" />
</BODY>
</HTML>

```

在上述代码中，length 属性用于计算其名称指定为 link 的所有元素的出现次数。因此，语句 x.length 将生成值 3，并在警报框中显示该值。

getElementsByTagName()

思考以下场景：用户可以在其中自定义网页上所有文本框的背景色。下拉列表用于选择背景色。如果用户从下拉列表中选择绿色，文档中所有文本框的背景色都将更改为绿色。这可通过使用 document.getElementsByTagName() 方法来实现。document.getElementsByTagName() 方法用于访问网页中具有相同类型的所有元素。在给定场景中，document.getElementsByTagName() 方法可用于返回文档中所有文本框的数组，然后可以使用 CSS 操作该数组以实现所需的功能。以下语法用于 getElementsByTagName() 方法：

```
document.getElementsByTagName("Tag_name");
```

在上述语法中，Tag_name 指定必须要访问的元素的标记名称。

思考以下代码段：

```

var x=document.getElementsByTagName("a");
alert(x.length + "Hyperlinks");

```

在上述代码段中，计算了网页上的锚或 <A> 标记的数目，并将该数目显示在警报框中。例如，如果在文档中有四个 <A> 标记，那么上述代码的输出将为 4 Hyperlinks。

getElementById()

众所周知，网页上使用的每个 HTML 元素都有一个可选属性 ID，它唯一地标识该元素。

document.getElementById() 方法使用 HTML 元素的 ID 访问和操作其内容。

以下语法使用 getElementById() 方法访问元素：

```
document.getElementById("id_of_the_element");
```

在上述语法中，id_of_the_element, 指定必须要访问的元素的 ID。

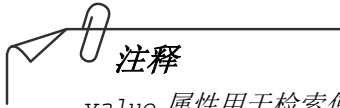
思考以下代码段，它用于在网页中使用文本框：

```
<INPUT type="text" ID="text1" />
```

要访问脚本中文本框的值，需要使用以下代码段：

```
var name = document.getElementById("text1").value;
```

在上述代码段中，使用指定为 text1 的文本框的 ID 获取文本框的值。然后该值存储在 name 变量中。



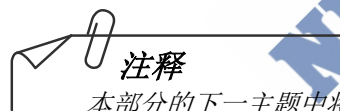
value 属性用于检索使用 <INPUT> 标记指定的元素的值。

使用 JavaScript，还可以更改 HTML 元素的内容，例如段落、超链接和标头。要检索和更改同时具有开始标记和结束标记（例如 <P>、<OPTION> 和 <DIV>）的 HTML 元素的文本，可以将 innerHTML 属性与 document.getElementById() 方法一起使用。

思考以下代码段：

```
<!DOCTYPE HTML>
<HTML>
<SCRIPT type="text/javascript"> function replacetext()
{
document.getElementById ("para1").innerHTML="Changed the content by using
the innerHTML property of getElementById() method";
}
</SCRIPT> <BODY>
<P ID="para1">A simple paragraph</P>
<INPUT type="button" value="Change text" onclick="replacetext()"/> </BODY>
</HTML>
```

在上述代码段中，单击 **Change text** 按钮时，将使用 onclick 事件调用 replacetext() 函数。replacetext() 函数将 <P> 元素的文本替换为文本 Changed the content by using the innerHTML property of getElementById() method。

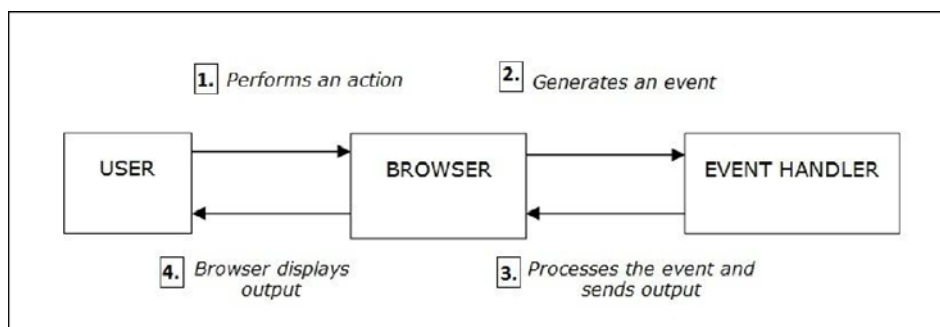


本部分的下一主题中将讨论事件。

处理事件

在 JavaScript 中，事件是在网页上发生的操作，例如鼠标单击或加载网页。Web 浏览器会等待事件发生，并在事件发生时执行为事件编程的处理。例如，当用户单击表单上的 **OK** 按钮时，Web 浏览器将执行设计用于执行该事件的那部分代码。这称为事件处理。为响应事件而执行的函数称为“事件处理程序”。

下图显示了事件处理过程。



事件处理过程

上图描述了用户与浏览器交互以及生成事件的过程。此事件由事件处理程序处理，并且输出被发送到浏览器。接着，浏览器向用户显示输出。

下表列出了 JavaScript 中支持的各类事件。

事件	描述	示例
<code>onblur</code>	在元素丢失焦点时发生。	<pre><INPUT type="text" name= "textbox1" onblur="show()" /></pre> <p>其中，<code>show()</code> 是事件发生时要执行的函数。</p>
<code>onchange</code>	在字段的内容更改时发生。	<pre><INPUT type="text" ID="fname" onchange="show()" /></pre> <p>其中，<code>show()</code> 是事件发生时要执行的函数。</p>
<code>onclick</code>	在用户单击对象时发生。	<pre><BUTTON onclick="show()">Show Text</BUTTON></pre> <p>其中，<code>show()</code> 是事件发生时要执行的函数。</p>
<code>onfocus</code>	在元素获得焦点时发生。	<pre><INPUT type="text" ID="firstname" onfocus="alert('hello');" /></pre>
<code>onkeydown</code>	在按下键盘的任何键时发生。	<pre><INPUT type="text" onkeydown="alert('key is down')" /></pre>

事件	描述	示例
onkeypress	在按下或按住键盘的字符键时发生。	<code><INPUT type="text" onkeypress="alert('key is pressed')"/></code>
onkeyup	在释放键盘键时发生。	<code><INPUT type="text" onkeyup="alert('key is released')"/></code>
onload	在完成页面或图像加载时发生。	<code><BODY onload="load()"></code> 其中, <code>load()</code> 是在事件发生时要执行的函数。
onmousedown	在按下鼠标按钮时发生。	<code></code>
onmousemove	在移动鼠标时发生。	<code></code>
onmouseout	在从元素上移走鼠标时发生。	<code><P onmouseout="alert('Mouse moved out of the paragraph')"> Move the mouse pointer out of my paragraph to display an alert box.</P></code>
onmouseover	在将鼠标移动到元素上时发生。	<code><P onmouseover="alert('Mouse moved over the paragraph')"> Move the mouse pointer over me to display an alert box.</P></code>
onmouseup	在释放鼠标按钮时发生。	<code></code>
onselect	在选择文本时发生。	<code>Select text:<INPUT type="text" value="Select me!" onselect="alert('You have selected the text.'')"/></code>
onunload	在用户退出页面时发生。	<code><BODY onunload="alert('The page is unloaded')"></code>
ondblclick	在用户双击对象时发生。	<code><BUTTON ondblclick="show()">Show Text</BUTTON></code> 其中, <code>show()</code> 是事件发生时要执行的函数。

事件	描述	示例
onerror	在加载文档或图像的过程中发生错误时发生。	<pre></pre>

JavaScript 中支持的事件

请考虑 BookYourHotel.com 网站的场景。用户注册页面显示注册表单，用户应必须先填写该表单才能在网站注册。管理层想要在注册页面上实现以下功能：

- 如果用户将任何文本框留空，其背景色应更改为粉红色，否则，它将保持为白色。
- 当用户在用户注册表单中填写必需的详细信息并单击 **Register me** 按钮后，应显示一个确认框来突出显示用户所输入的注册信息。

思考实现这些功能的以下代码：

```
<!DOCTYPE HTML>
<HTML> <HEAD>
<TITLE>REGISTRATION DETAILS</TITLE>
<STYLE>
h1,h3{
color: black;
font-size:40px;
text-align:center;
}
table{
margin-left:650px;
border:2px solid white;
background-color:beige;
}
td{
padding:10px;
border:2px solid white;}
#button{
margin-left:740px;}#button{
margin-left:740px;}

</STYLE>
<SCRIPT>
function show()
{
var fname = document.getElementById ("txtbox1").value;
var lname = document.getElementById("txtbox2").value;
var age = document.getElementById("age").value;
var address = document.getElementById ("address").value;
var gender=document.getElementById ("gender").value;
confirm("You have entered:"+ "\n Name :" + fname + " " + lname + "\n Age :" +
age + "\n Address :"+ address + "\n Gender :"+ gender + "\n\n Do you want to
confirm these details ?");
}
function changeColor(val)
{
```



```

if((val.value=="")||(val.value==null))
{
val.style.background="pink";
}
else
{
val.style.background="#FFFFFF";
}
}
</SCRIPT>
</HEAD>
<BODY>
<H1>USER REGISTRATION DETAILS<HR/></H1>
<H3>Please fill the following details and get registered !!</H3><BR/><BR/>
<TABLE>
<TR>
<TD> First name:</TD>
<TD> <INPUT type="text" name="name1" ID="txtbox1"
onblur="changeColor(this)"/> </TD>
</TR> <TR>
<TD> Last name:</TD>
<TD><INPUT type="text" name="name2" ID="txtbox2"
onblur="changeColor(this)"/></TD>
</TR> <TR>
<TD> Age:</TD>
<TD> <INPUT type="text" name="age_box" ID="age" onblur="changeColor(this)"/>
</TD>
</TR> <TR>
<TD> Address:</TD>
<TD><TEXTAREA rows="5" name="address_box" ID="address"
onblur="changeColor(this)"></textarea></TD>
</TR> <TR>
<TD> Gender:</TD> <TD>
<SELECT name="Gender" ID="gender"> <OPTION value="Male">Male</OPTION> <OPTION
value="Female">Female</option> </SELECT> </TD>
</TR> </TABLE>
<BR/><INPUT ID="button" type="button" value="Register Me" onclick="show()"/>
</BODY>
</HTML>

```

注释

关键字 *this* 指的是文档中的当前对象，例如文本框和按钮。

上述代码的网页在下图中显示。

USER REGISTRATION DETAILS

Please fill the following details and get registered !!

First name:	<input type="text"/>
Last name:	<input type="text"/>
Age:	<input type="text"/>
Address:	<input type="text"/>
Gender:	<input type="text" value="Male"/>

浏览器输出

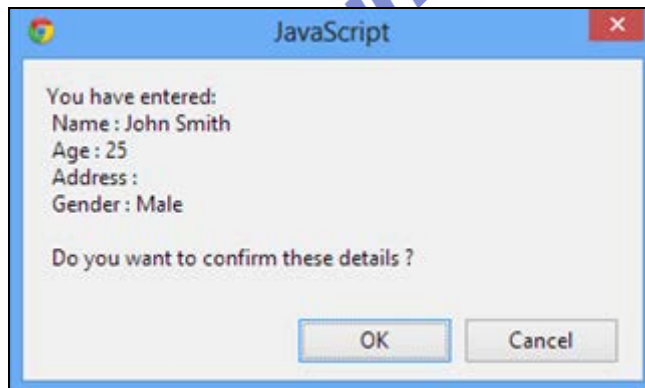
上述输出允许用户在注册表单中输入详细信息。在用户填写文本框并按 **Tab** 键以移动到下一个文本框时，未填写的文本框的背景色将更改为粉红色。

下图显示了已完成的用户注册表单。

USER REGISTRATION DETAILS	
Please fill the following details and get registered !!	
First name:	<input type="text" value="John"/>
Last name:	<input type="text" value="Smith"/>
Age:	<input type="text" value="25"/>
Address:	<input type="text"/>
Gender:	<input type="text" value="Male"/>
<input type="button" value="Register Me"/>	

已完成的用户注册表单

在用户单击 **Register Me** 按钮时，将显示一个确认对话框，如下图所示。



确认对话框

事件侦听器

在 JavaScript 中，您可以通过调用称为事件处理程序的特定方法来处理用户操作或事件。要处理每个用户操作或事件，将通知侦听器。侦听器是等候事件发生并执行与其对应的某些操作的对象。但是，为此，您需要向事件处理程序注册侦听器。向字段事件注册的事件处理程序函数处理事件并向侦听器提供适当的响应。然后，事件侦听器执行操作。

要注册处理程序，需要使用 `addEventListener()` 函数。该函数接收封装事件信息的事件对象。可使用以下语法创建 `addEventListener()` 函数：

```
addEventListener(type, listener[, useCapture]);
```

在上述语法中：

- **type**：指定表示事件类型的字符串。
- **listener**：指定事件发生时接收通知的对象。
- **useCapture**：是布尔变量，指定是否需要捕获事件。这是可选参数。

如果事件处理程序捕获一个事件，每次该事件在元素上发生时，都将调用事件处理程序。还可以通过调用 `removeEventListener()` 函数移除处理程序。但是，传递到该函数的参数必须与传递到 `addEventListener()` 函数的参数相同。

思考用于了解事件侦听器功能的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function OnClick () {
alert ("A click event has occurred on the Submit button.");
}
function AddEventHandler () {
var Button = document.getElementById ("Button");
if (Button.addEventListener) {
Button.addEventListener ("click", OnClick, false);
}
}
function RemoveEventHandler () {
var Button = document.getElementById ("Button");
if (Button.removeEventListener) {
Button.removeEventListener ("click", OnClick, false);
}
}
</SCRIPT>
</HEAD>
<BODY>
Click on the Submit button when the 'click' event has a listener and when it
does not.<BR/>
<BUTTON onclick="AddEventHandler();">Add a 'click' event listener to the
Submit button</BUTTON>
<BUTTON onclick="RemoveEventHandler();">Remove the event listener</BUTTON>
<BR /><BR />
<BUTTON ID="Button">Submit</BUTTON>
</BODY>
</HTML>
```

使用 navigator 对象

`navigator` 对象是 JavaScript 对象层次结构中高层对象之一。不像 `window` 或 `document` 对象，`navigator` 对象是有其自己的属性和方法集的独立对象。独立对象在自己的层次结构中没有任何子对象。

您可以使用 navigator 对象访问关于当前浏览器的信息，例如浏览器名称、版本和用户平台。下表列出了 navigator 对象的属性。

属性	描述	语法
appName	指定 Web 浏览器的名称。	navigator.appName
appVersion	指定有关 Web 浏览器版本、浏览器平台和发布 Web 浏览器的国家的信息。	navigator.appVersion
appCodeName	指定 Web 浏览器的内部代码名称。例如，为 Microsoft Internet Explorer 和 Netscape 浏览器返回的内部代码名称都是 Mozilla。	navigator.appCodeName
userAgent	将包含诸如浏览器版本、代码名称和平台之类信息的字符串从客户机发送到服务器。服务器使用此字符串识别客户机。	navigator.userAgent
pPlatform	指定客户机的操作系统。	navigator.platform

navigator 对象的属性

下表列出了与 navigator 对象相关联的方法。

方法	描述	语法
javaEnabled()	返回布尔值，指定在 Web 浏览器中是启用还是禁用 JavaScript。例如，值 True 表示当前 Web 浏览器中启用了 JavaScript。	navigator.javaEnabled()
taintEnabled()	返回一个布尔值，指定当前 Web 浏览器的安全功能是启用还是禁用。默认情况下，返回的值是 False。	navigator.taintEnabled()

navigator 对象的方法

思考使用 navigator 对象的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
document.write(navigator.appName);
document.write(navigator.appVersion);
</SCRIPT>
</BODY>
```

</HTML>

在上述代码中，navigator 对象的 appName 和 appVersion 属性分别用于显示浏览器的名称和版本。

使用 screen 对象

screen 对象使您能够访问用户屏幕的详细信息，例如其宽度、高度和分辨率。

下表列出 screen 对象的最常用属性。

属性	描述	语法
height	返回屏幕的总高度。	screen.height
pixelDepth	返回屏幕的颜色分辨率。	screen.pixelDepth
width	返回屏幕的宽度。	screen.width
availHeight	返回屏幕的高度，不包括 Windows 任务栏。	screen.availHeight
availWidth	返回屏幕的宽度，不包括 Windows 任务栏。	screen.availWidth
colorDepth	返回用于显示图像的调色板深度。	screen.colorDepth

screen 对象的属性

思考以下代码段，它描述 screen 对象的用法：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<SCRIPT type="text/javascript">
document.write("Height:" + screen.availHeight);
document.write("Width:" + screen.availWidth);
</SCRIPT>
</BODY>
</HTML>
```

上述代码使用 availHeight 和 availWidth 属性显示屏幕的高度和宽度，不包括 Windows 任务栏。

使用 history 对象

history 对象包含用户已访问的所有页面的列表。

下表列出了 history 对象的属性。

属性	描述	语法
length	是整数值，表示当前会话中 history 对象当前所引用的链接数。	history.length

history 对象的属性

下表列出 history 对象的最常用方法。

方法	描述	语法
back()	用于移动到前一张页面。	history.back()
forward()	用于移动到下一张页面。	history.forward()
go()	用于返回特定的页数。	history.go(x) 其中，x 是页数

history 对象的方法

思考使用 history 对象的以下代码：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT type="text/javascript">
function show()
{
alert(history.length)
}
</SCRIPT>
</HEAD>
<BODY>
<INPUT type="button" value="Show History" onclick="show()" />
</BODY>
</HTML>
```

上述代码显示用户单击按钮时当前浏览器会话中引用的链接数。

使用 location 对象

location 对象包含关于当前 URL 的信息。它还可用于实现到因特网上不同 URL 的导航。location 对象最常用的属性是 href 属性。href 属性指定当前文档的确切 URL。

location 对象的属性包含各种类型的信息，组成了一个完整的 URL。以下语法用于指定完整的 URL：

```
protocol://hostname:port/pathname/search#hash
```

上述语法包含以下属性：

- **protocol**：指定用于在因特网上传输数据的协议，例如：**HTTP** 和 **FTP**。
- **hostname**：指定 **URL** 的主机名，例如：**http://www.silvertech.com**。
- **port**：指定 **URL** 的端口号。这是可选的。如果没有为 **http** 请求指定端口号，浏览器将自动连接到端口 **80**。
- **pathname**：指定 **URL** 的路径，例如：**http://www.silvertech.com/careers.html**。
- **search**：指定 **URL** 任意部分的搜索字符串，后接一个问号，例如：**http://www.silvertech.com/careers.html/search?programmer**。问号用于在 **URL** 中嵌入参数。
- **#hash**：指定网页的内部超链接，例如：**http://www.silvertech.com/careers.html#position1**。

下表列出 **location** 对象的最常用属性。

属性	描述	语法
hash	指定或返回内部链接锚名称。在 URL 中内部链接跟在 # 后。	<code>location.hash</code>
host	指定或返回 URL 的主机名：端口部分。	<code>location.host</code>
hostname	指定或返回主机名。	<code>location.hostname</code>
href	指定或返回文件或网站的部分或全部路径。	<code>location.href</code>
port	指定或返回端口号。	<code>location.port</code>
protocol	指定或返回协议。	<code>location.protocol</code>

location 对象的属性

下表列出 **location** 对象的最常用方法。

方法	描述	语法
<code>assign()</code>	用于加载新文档。	<code>location.assign(URL);</code>
<code>reload()</code>	用于重新加载当前文档。	<code>location.reload();</code>
<code>replace()</code>	用于将当前文档替换为其他文档。	<code>location.replace(newURL);</code>

location 对象的方法

思考以下代码段，它用于在单击按钮时打开 **XYZ.com** 网站的主页：


```
<FORM>
<INPUT type = "button" VALUE = "visit us" onClick = "location.href =
'http://www.xyz.com'">
</FORM>
```

在上述代码段中，location 对象的 href 属性被设置为 XYZ 网站的 URL。因此，当用户单击按钮时，会显示 XYZ 网站的主页。

思考以下代码段，它用于检索有关当前 URL 的信息：

```
document.write(location.protocol);
```

在上述代码段中，使用 location 对象的 protocol 属性检索当前 URL 的协议。

使用表单对象

请考虑 BookYourHotel.com 网站的场景。要购买产品，客户必须填写购买订单表单。购买订单表单需要输入客户详细信息，例如姓名、性别和地址。该信息是使用 Form 对象（例如文本框和单选按钮）输入的。要访问和处理 Form 对象的内容，必须引用 Form 对象。

Form 对象是一个浏览器对象，充当收集用户信息的几个其他对象的容器。使用对象（例如命令按钮、单选按钮、文本框、组合框和复选框）收集的信息会提交到服务器进行处理。因此，Form 对象使您可以创建交互式网页。

JavaScript 的以下对象常在使用 Web 表单时被使用：

- form
- button
- checkbox
- text
- textarea
- radio
- select

form

表单接受用户输入。当在浏览器中查看包含多种表单的网页时，浏览器为网页上的每个表单创建一个 form 对象。还可以使用数组访问这些表单。例如，Web 浏览器可以使用 document.forms[0] 访问网页上的第一个表单，使用 document.forms[1] 访问第二个表单。

与 form 对象关联的方法有：

- submit()：向服务器提交表单进行处理。以下语法用于 submit() 方法：

```
form1.submit()
```

在上述语法中，form1 是当事件发生时（例如单击一个超链接）将提交到服务器的表单的名称。

- reset()：重置给定表单中的所有字段。以下语法用于 reset() 方法：

```
form1.reset()
```

在上述语法中，form1 是表单的名称。reset() 方法清空表单中多个字段中输入的所有信息。

与 form 对象关联的事件有：

- onsubmit: 在用户单击按钮或超链接以将表单提交到服务器进行处理时发生。
- onreset: 在用户通过单击重置按钮或超链接选择重置表单域时发生。

button

button 对象引用 HTML 按钮。它允许您基于用户操作执行多种任务。您可以在用户单击一个特定按钮时编写被执行的函数。

与 button 对象关联的方法有：

- click(): 模拟按钮的单击。例如，myButton.click()。
- focus(): 在按钮上设置焦点。
- blur(): 将按钮上的焦点删除。

与 button 对象关联的事件属性有：

- onclick: 在用户单击按钮时发生。
- onmousedown: 在按下鼠标按钮时发生。
- onmouseup: 在释放鼠标按钮时发生。

checkbox

checkbox 对象引用 HTML 复选框。

与 checkbox 对象关联的方法有：

- click()
- focus()
- blur()

可使用以下语法引用表单内的 checkbox 对象：

```
document.form1.checkbox1
```

在上述语法中，form1 是表单的名称，checkbox 是复选框的 ID。

思考以下代码段，它引用表单内的 checkbox 对象并对它执行验证：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:pink;
}
</STYLE>
<SCRIPT>
```

```

function validate()
{
if(document.f1.check1.checked==false)
{
alert("Please select the check box");
return false
}
else
return true
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="f1" method="get" action="thanks.html" onsubmit="return
validate();" >
<INPUT type="checkbox" ID="check1" />Do you want to accept?
<INPUT type="submit" name="submit" value="Submit"></INPUT>
<INPUT type="reset" name="reset" value="Reset"></INPUT>
</FORM>
</BODY>
</HTML>

```

上述代码段引用表单内的 checkbox 对象并在用户提交表单之前对其进行验证。它使用 document.f1.check1 语法引用 checkbox 对象。它使用 document.f1.check1.checked==false 条件检查用户是否已选中复选框。如果用户未选中复选框，那么 document.f1.check1.checked==false 条件求值为 false，并显示 **Please select the check box** 消息。

text

text 对象引用 HTML 文本框。

与 text 对象关联的方法有：

- blur()：将焦点从文本字段中移开。
- focus()：在文本字段上设置焦点。
- select()：选择文本字段。

与 text 对象关联的事件属性有：

- onfocus()：元素接收焦点时触发。
- onchange()：在 text 对象的值被修改时，blur 事件发生后触发。
- onselect()：在用户在文本字段中选择一部分文本时触发。

可使用以下语法引用表单内的 text 对象：

```
document.form1.text1
```

在上述语法中，form1 是表单的名称，text1 是文本框的 ID。

textarea

textarea 对象引用 HTML 文本区域。

textarea 对象与 text 对象有类似的方法和事件属性。

radio

radio 对象引用 HTML 单选按钮。

可使用以下语法引用表单内的 radio 对象：

```
document.form1.radio1
```

在上述语法中，form1 是表单的名称，radio1 是单选按钮的名称或 ID。

思考以下代码段，它引用表单内的 radio 对象并对它执行验证：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<STYLE>
body{
background-color:pink;
}
</STYLE>
<SCRIPT>
function validate()
{
if((document.f1.M.checked==false)&&(document.f1.F.checked==false))
{
alert("Please select the gender");
return false
}
else
return true
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="f1" method="get" action="thanks.html" onsubmit="return
validate();" >
Gender:
<INPUT type="radio" name="gender" ID="M" />Male
<INPUT type="radio" name="gender" ID="F" />Female
<INPUT type="submit" name="submit" value="Submit" align="center"></INPUT>
<INPUT type="reset" name="reset" value="Reset"></INPUT>
</FORM>
</BODY>
</HTML>
```

上述代码段引用表单内的 radio 对象并在用户提交表单之前对其进行验证。它检查是否已选中了用于表示性别的单选按钮 **Male** 或 **Female**。如果用户未指示性别，单选按钮的选中属性返回 false，并显示 **Please select the gender** 消息。

select

select 对象引用 HTML 下拉列表。

与 select 对象关联的方法有：

- blur()
- focus()

与 select 对象关联的事件属性有：

- onchange
- onfocus
- onblur

可使用以下语法引用表单内的 select 对象：

```
document.form1.select1
```

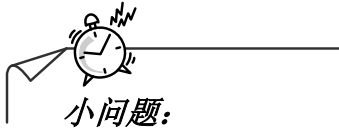
在上述语法中，form1 是表单的名称，select1 是 select 对象的名称或 ID。

思考以下代码段，它引用表单内的 select 对象并对它执行验证：

```
<!DOCTYPE HTML>
<HTML>
<HEAD>
<SCRIPT>
function validate()
{
if(document.f1.opt.value==0)
{
alert("Please select the country");

return false
}
else
return true
}
</SCRIPT>
</HEAD>
<BODY>
<FORM name="f1" method="get" action="thanks.html" onSubmit="return
validate();" >
Country:
<SELECT ID="opt" style="width:220">
<OPTION value=0>Select any country</OPTION>
<OPTION value="1">India</OPTION>
<OPTION value="2">USA</OPTION>
<OPTION value="3">UK</OPTION>
</SELECT>
<INPUT type="submit" name="submit" value="Submit">
<INPUT type="reset" name="reset" value="Reset">
</FORM>
</BODY>
</HTML>
```

上述代码段引用表单内的 `select` 对象并在用户提交表单之前对其进行验证。它使用 `document.f1.opt` statement 引用 `select` 对象。它使用 `document.f1.opt.value==0` 条件检查用户是否指出其国家或地区。如果用户未指出其国家或地区，将显示消息 **Please select the country**。



小问题:

以下哪个属性指定客户机的操作系统?

1. `platform`
2. `userAgent`
3. `appName`
4. `appVersion`

答案:

1. `platform`



活动 5.2: 操作网页的组件

练习问题

1. 以下哪个属性用于指定将数据提交给 action 属性中指定的文件或 URL 的格式?
 - a. ID
 - b. name
 - c. method
 - d. novalidate
2. target 属性的以下哪个值指定应在窗口的完整正文中显示响应?
 - a. _top
 - b. frame_name
 - c. _blank
 - d. _self
3. 以下哪个属性用于指定在提交表单时输入字段不得留空?
 - a. autofocus
 - b. required
 - c. value
 - d. target
4. 以下哪个标记用于表示计算的结果?
 - a. <OUTPUT>
 - b. <KEYGEN>
 - c. <DATA LIST>
 - d. <FIELDSET>
5. screen 对象的以下哪个属性用于返回屏幕的高度（不包括 Windows 任务栏）?
 - a. width
 - b. height
 - c. availHeight
 - d. pixelHeight

小结

在本章中，您学习了：

- 要在网页上创建表单，需要使用 `<FORM>` 标记。
- `<FORM>` 标记支持以下属性：
 - `name`
 - `ID`
 - `action`
 - `method`
 - `autocomplete`
 - `novalidate`
 - `target`
- 可以使用以下标记将这些字段添加到表单：
 - `<INPUT>`
 - `<SELECT>`
 - `<LABEL>`
 - `<FIELDSET>`
 - `<TEXTAREA>`
 - `<DATALIST>`
 - `<KEYGEN>`
 - `<OUTPUT>`
 - `<BUTTON>`
- JavaScript 定义网页上的以下浏览器对象：
 - `window`
 - `document`
 - `navigator`
 - `screen`
 - `history`
 - `location`
- 浏览器对象表示浏览器环境，并为它的访问和操作提供属性和方法。
- `Form` 对象是一个浏览器对象，充当收集用户信息的几个其他对象的容器。
- JavaScript 的以下对象常在使用 Web 表单时被使用：
 - `form`
 - `button`
 - `checkbox`
 - `text`
 - `textarea`
 - `radio`
 - `select`