CS 50 Web Design

APCSP Module 2: Internet

Unit 1: HTML (Chapter 8-10)

LECTURE 3: TABLE AND FORM DESIGN

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- The design of tables, table caption, table span, colgroup, rowgroup.
- The design of a form and all the input elements
- Other HTML embedded elements.

Table

SECTION 1

How to use Tables?

- HTML tables were created for instances when you need to add tabular material (data arranged into rows and columns) to a web page.
- Tables may be used to organize schedules, product comparisons, statistics, or other types of information, as shown in FIGURE 8-1. Note that “data” doesn’t necessarily mean numbers.
- A table cell may contain any sort of information, including numbers, text elements, and even images and multimedia objects.

How to use Tables?

- In visual browsers, the arrangement of data in rows and columns gives readers an instant understanding of the relationships between data cells and their respective header labels.
- Bear in mind when you are creating tables, however, that some readers will be hearing your data read aloud with a screen reader or reading Braille output.
- Later in this chapter, we'll discuss measures you can take to make table content accessible to users who don't have the benefit of visual presentation.

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Minimal Table Structure

- Let's take a look at a simple table to see what it's made of. Here is a small table
- with three rows and three columns that lists nutritional information.

Menu item	Calories	Fat (g)
Chicken noodle soup	120	2
Caesar salad	400	26

FIGURE 8-2 reveals the structure of this table according to the HTML table model. All of the table's content goes into cells that are arranged into rows. Cells contain either header information (titles for the columns, such as "Calories") or data, which may be any sort of content.

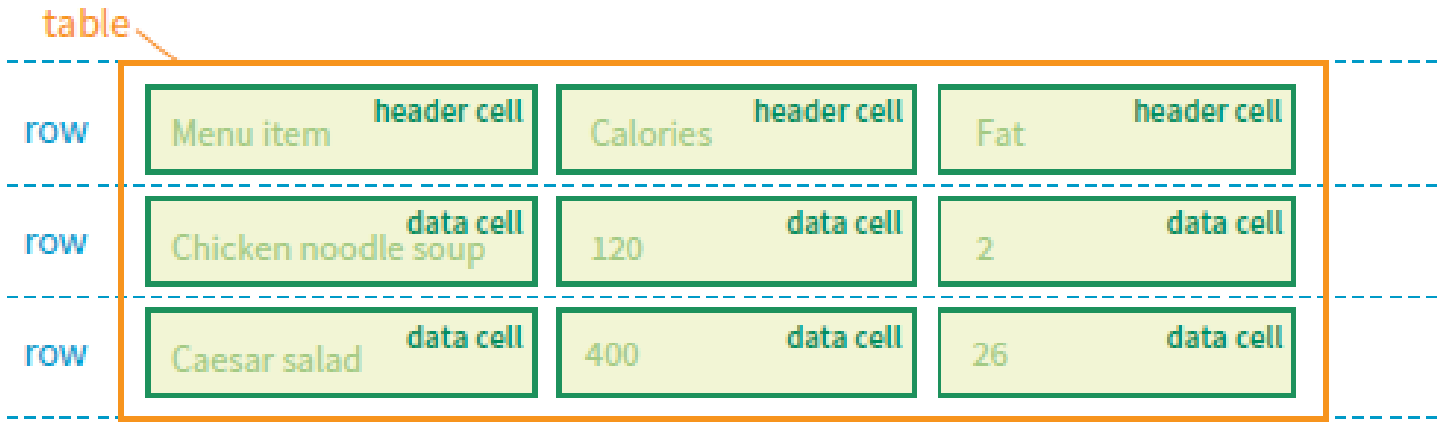


FIGURE 8-2. Tables are made up of rows that contain cells. Cells are the containers for content.

table

<table>…</table>

Tabular content (rows and columns)

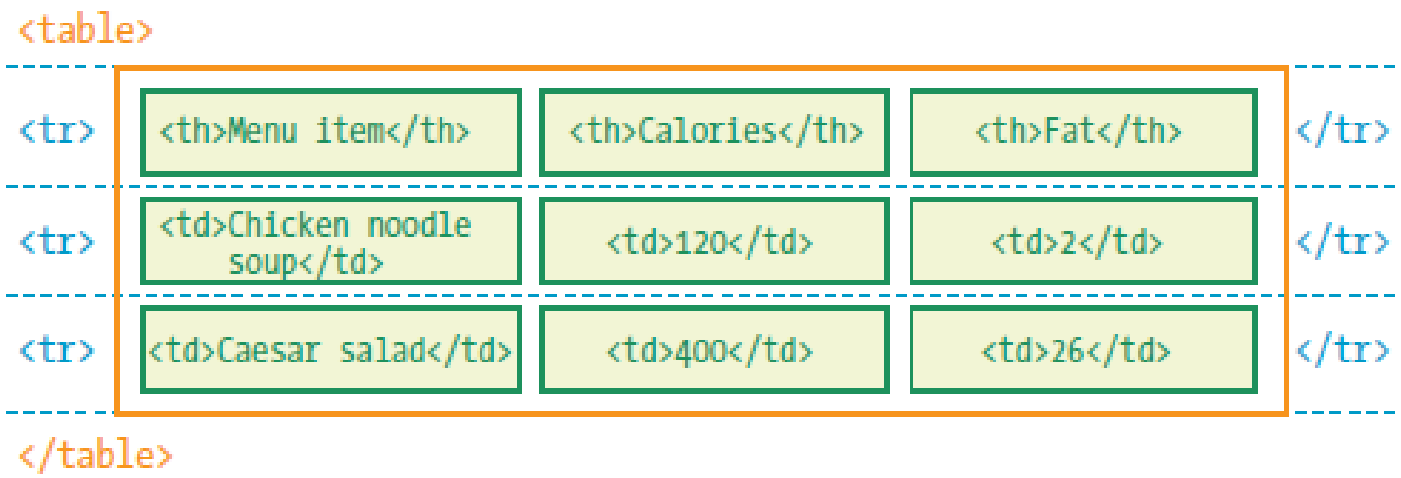


FIGURE 8-3. The elements that make up the basic structure of a table.

Table Row

<tr>...</tr>

Table row

```
<table>
    <tr>
        <th>Menu item</th>
        <th>Calories</th>
        <th>Fat (g)</th>
    </tr>
    <tr>
        <td>Chicken noodle soup</td>
        <td>120</td>
        <td>2</td>
    </tr>
    <tr>
        <td>Caesar salad</td>
        <td>400</td>
        <td>26</td>
    </tr>
</table>
```

Table Header and Data (Cell)

<th>…</th>
Table header
<td>…</td>
Table cell data

Table Structure

Code Structure

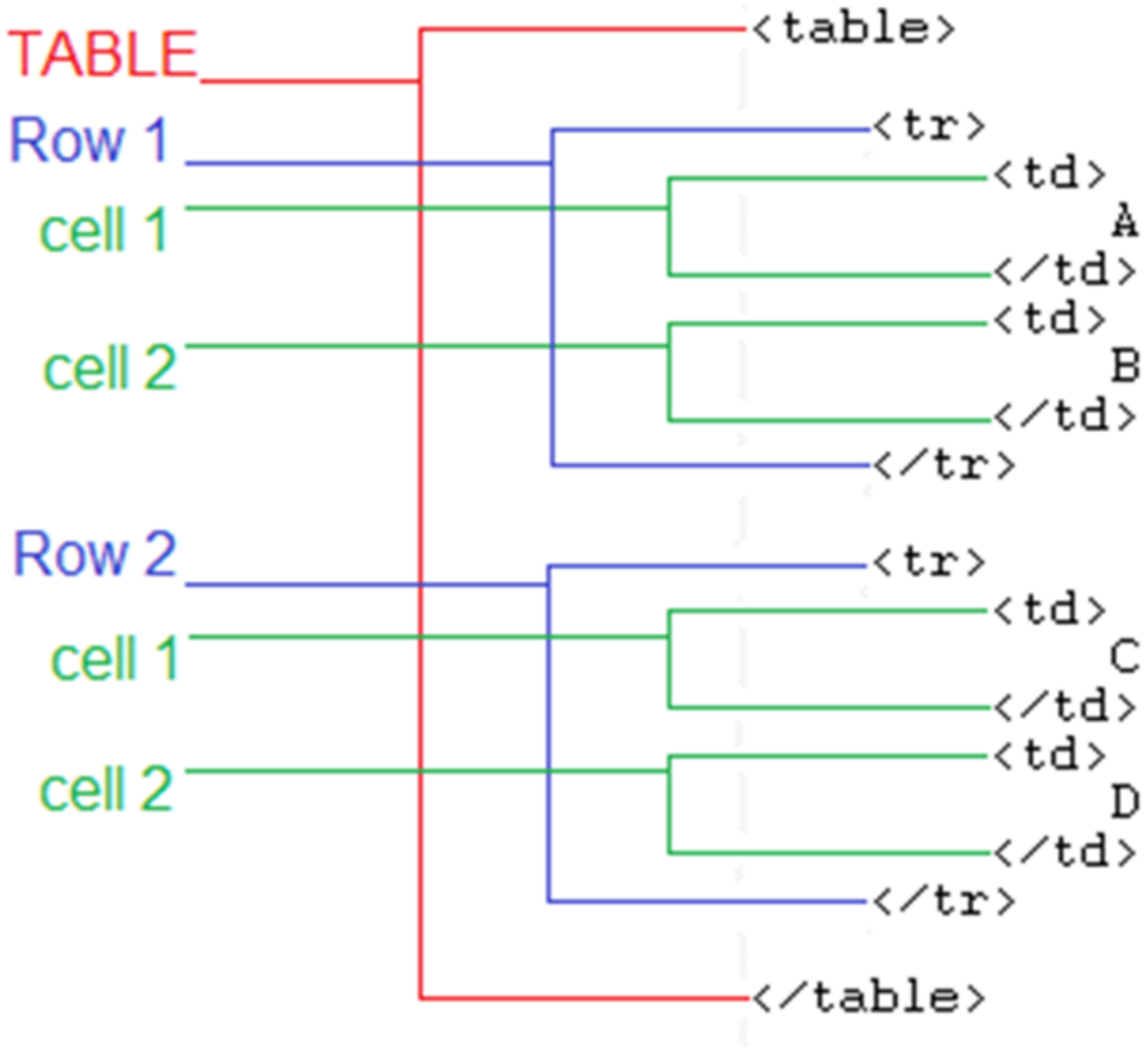


Table Structure
Structure

```
<table>
    <tr> <td></td> <td></td> <td></td> </tr>
    <tr> <td></td> <td></td> <td></td> </tr>
    <tr> <td></td> <td></td> <td></td> </tr>
    <tr> <td></td> <td></td> <td></td> </tr>
</table>
```

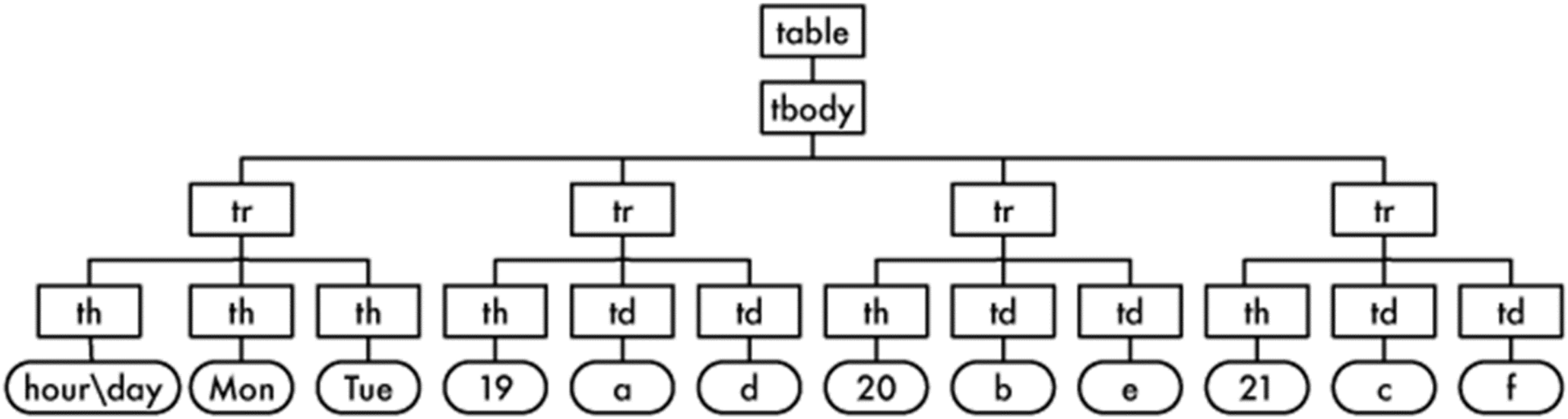


Table Structure
DOM model

Table Structure

Complete Code Structure

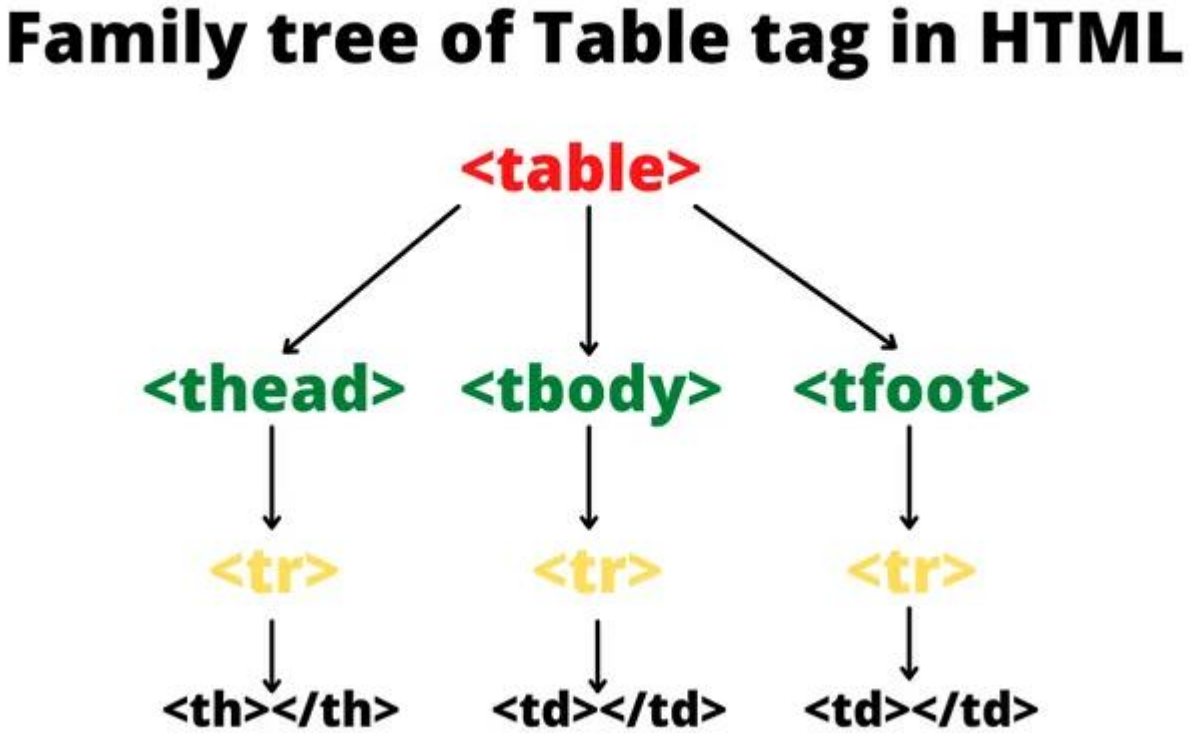


Table Structure

Complete Code Structure

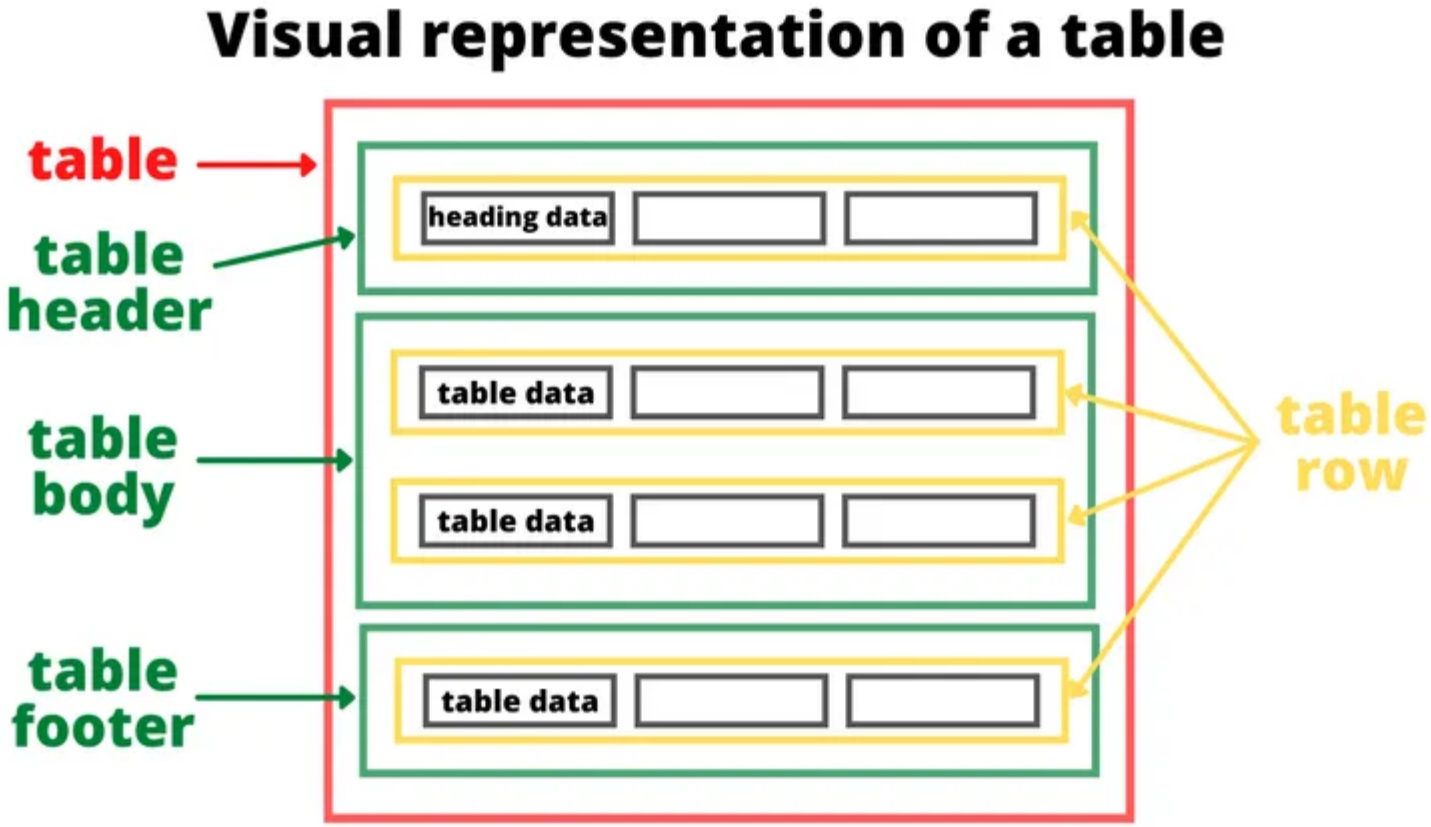


Table Structure

Complete Code Structure

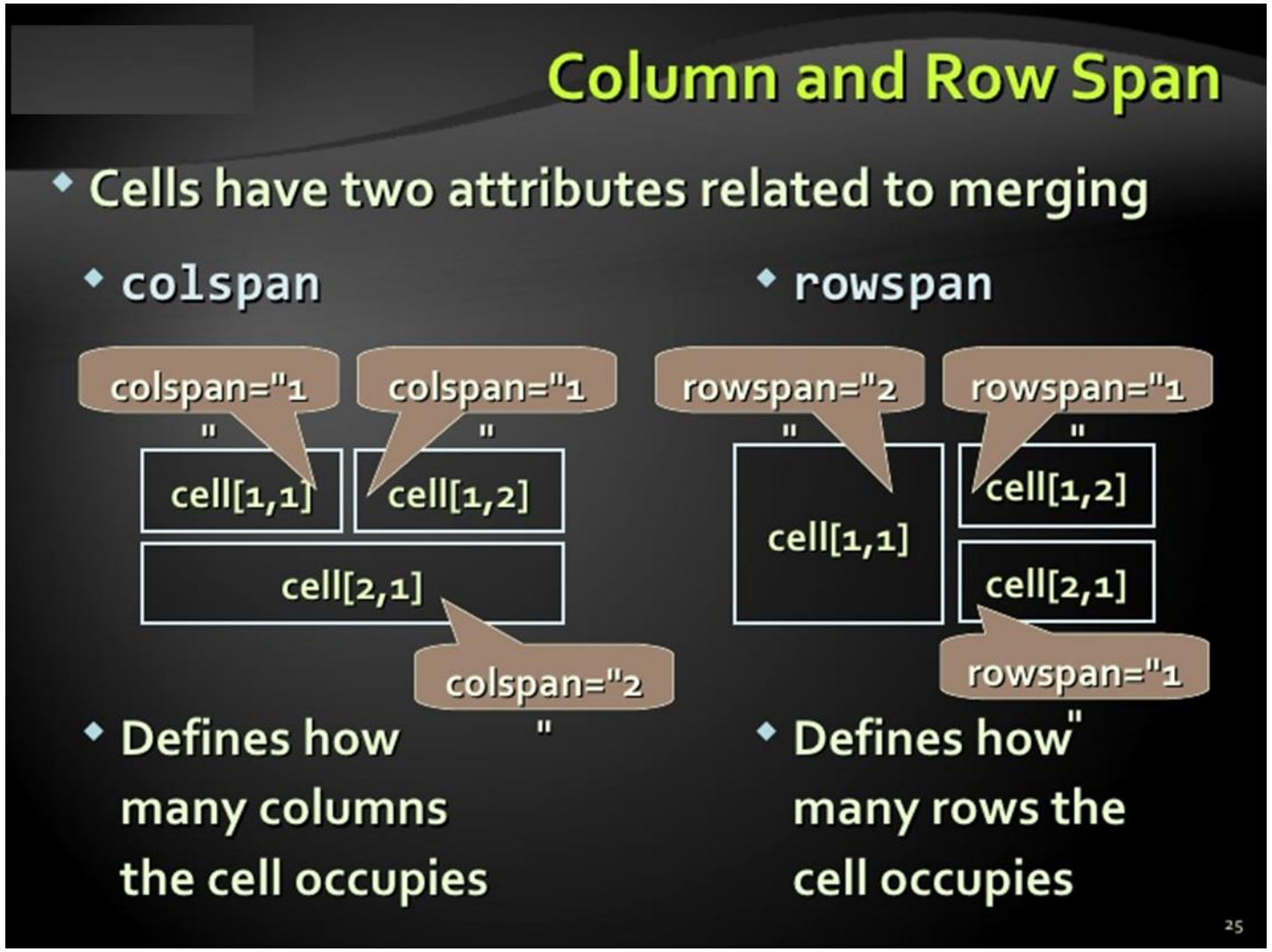
```
<TABLE>
        <CAPTION>CSS3 Browser Support*</CAPTION>

<THEAD>
        <TR>
                <TD>CSS Property</TD>
                <TD>Internet Explorer</TD>
                <TD>FireFox</TD>
                <TD> Chrome</TD>
                <TD>Safari</TD>
                <TD>Opera</TD>
        </TR>

<THEAD>

<TFOOT>
        <TD COLSPAN="6">*Latest browser versions</TD>
<TFOOT>

<TBODY>
        <TR>
                <TD class="LeftCol">Border Radius</TD>
                <TD>YES</TD>
                <TD>YES</TD>
                <TD>YES</TD>
                <TD>YES</TD>
                <TD>YES</TD>
        </TR>

        <TR>
                <TD class="LeftCol">Box Shadow</TD>
                <TD>YES</TD>
                <TD>YES</TD>
                <TD>YES</TD>
                <TD>YES</TD>
                <TD>YES</TD>
        </TR>

        <TR>
                <TD class="LeftCol">CSS Animations</TD>
                <TD>NO</TD>
                <TD>NO</TD>
                <TD>YES</TD>
                <TD>YES</TD>
                <TD>NO</TD>
        </TR>
<TBODY>

</TABLE>
```

Table Span

SECTION 2

Spanning Cells

- One fundamental feature of table structure is cell spanning, which is the stretching of a cell to cover several rows or columns. Spanning cells allows you to create complex table structures, but it has the side effect of making the markup a little more difficult to keep track of. It can also make it potentially more difficult for users with screen readers to follow.
- You make a header or data cell span by adding the **colspan** or **rowspan** attributes, as we'll discuss next.



Column span and Row span

colspan="number"
rowspan="number"

Column Spans

```
<table>
    <tr>
        <th colspan="2">Fat</th>
    </tr>
    <tr>
        <td>Saturated Fat (g)</td>
        <td>Unsaturated Fat (g)</td>
    </tr>
</table>
```

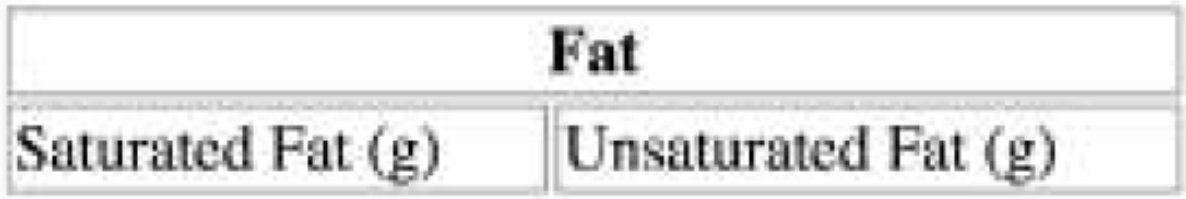
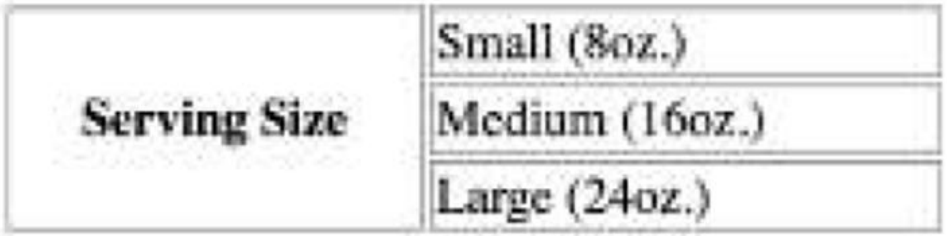


FIGURE 8-6. The **colspan** attribute stretches a cell to the right to span the specified number of columns.

Row Spans

```
<table>
    <tr>
        <th rowspan="3">Serving Size</th>
        <td>Small (8oz.)</td>
    </tr>
    <tr>
        <td>Medium (16oz.)</td>
    </tr>
    <tr>
        <td>Large (24oz.)</td>
    </tr>
</table>
```



Serving Size	Small (8oz.)
	Medium (16oz.)
	Large (24oz.)

FIGURE 8-8. The **rowspan** attribute stretches a cell downward to span the specified number of rows.

```
<table>
    <caption>Nutritional Information</caption>
    <tr>
        <th>Menu item</th>
        <th>Calories</th>
        <th>Fat (g)</th>
    </tr>
    <!-- table continues -->
</table>
```

Nutritional Information

Menu item	Calories	Fat (g)
Chicken noodle soup	120	2
Caesar salad	400	26

FIGURE 8-10. The table caption is displayed above the table by default.

Caption of a Table

<caption>

scope

The **scope** attribute associates a table header with the row, column, group of rows (such as **tbody**), or column group in which it appears by using the values **row**, **col**, **rowgroup**, or **colgroup**, respectively. This example uses the **scope** attribute to declare that a header cell applies to the current row:

```
<tr>
    <th scope="row">Mars</th>
    <td>.95</td>
    <td>.62</td>
    <td>0</td>
</tr>
```

Accessibility experts recommend that every **th** element contain a **scope** attribute to make its associated data explicitly clear.

Connecting Cells and Headers

scope

`headers`

For really complicated tables in which `scope` is not sufficient to associate a table data cell with its respective header (such as when the table contains multiple spanned cells), the `headers` attribute is used in the `td` element to explicitly tie it to a header's `id` value. In this example, the cell content ".38" is tied to the header "Diameter measured in earths":

```
<th id="diameter">Diameter measured in earths</th>
<!-- many other cells -->
<td headers="diameter">.38</td>
<!-- many other cells -->
```

Unfortunately, support of the `id`/`headers` feature is unreliable. The recommended best practice is to create tables in a way that a simple `scope` attribute will do the job.

Connecting Cells and Headers

headers

Column Groups and Row Groups

SECTION 3

```
<table>
…
<thead>
  <!-- headers in these rows-->
  <tr></tr>
  <tr></tr>
  <tr></tr>
<thead>
<tbody>
  <!-- data -->
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
</tbody>
<tfoot>
  <!-- footnote -->
  <tr></tr>
</tfoot>
</table>
```

Row Group Elements

<thead>…</thead>
Table header row group
<tbody>…</tbody>
Table body row group
<tfoot>…</tfoot>
Table footer row group

```
<table>
…
<thead>
  <!-- headers in these rows-->
  <tr></tr>
  <tr></tr>
  <tr></tr>
<thead>
<tbody>
  <!-- data -->
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
  <tr></tr>
</tbody>
<tfoot>
  <!-- footnote -->
  <tr></tr>
</tfoot>
</table>
```

Row Group Elements

<thead>…</thead>
Table header row group
<tbody>…</tbody>
Table body row group
<tfoot>…</tfoot>
Table footer row group

Multiple Col Elements

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

Column Group Elements

using <col span=“”>

<colgroup>…</colgroup>
A semantically related group of columns
<col>…</col>
One column in a column group

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
</style>
</head>
<body>
<h2>Multiple Col Elements</h2>
<table style="width: 100%;">
  <colgroup>
    <col span="2" style="background-color: #B6E8F8">
    <col span="3" style="background-color: #ACE4AC">
    <col span="2" style="background-color: #F9A409">
  </colgroup>
<tr>
<th>MON</th><th>TUE</th><th>WED</th><th>THU</th>
<th>FRI</th><th>SAT</th><th>SUN</th>
</tr>
```

Column Group Elements

multiple_column.html

<colgroup>…</colgroup>
A semantically related group of columns

<col>…</col>
One column in a column group

```
<tr>
<td>1</td><td>2</td><td>3</td><td>4</td>
<td>5</td><td>6</td><td>7</td>
</tr>
<tr>
<td>8</td><td>9</td><td>10</td><td>11</td>
<td>12</td><td>13</td><td>14</td>
</tr>
<tr>
<td>15</td><td>16</td><td>17</td><td>18</td>
<td>19</td><td>20</td><td>21</td>
</tr>
<tr>
<td>22</td><td>23</td><td>24</td><td>25</td>
<td>26</td><td>27</td><td>28</td>
</tr>
</table>
</body>
</html>
```

Column Group Elements

multiple_column.html

<colgroup>…</colgroup>

A semantically related group of columns

<col>…</col>

One column in a column group

Table colgroup with id Example

Group A	Group B		Group C	
1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20

Column Group Elements

using <colgroup id="A">

<colgroup>…</colgroup>
A semantically related group of columns
<col>…</col>
One column in a column group

using_id.html

```
<html>
    <head>
        <title>Column Group using id</title>
        <style>
            table{
              border: 1px solid black;
            }
            #groupA{
                background-color: pink;
                width: 5em;
            }
            #groupB{
                background-color:bisque;
                width: 10em;
            }
            #groupC{
                background-color:azure;
                width: 10em;
            }
        </style>
    </head>
```

using_id.html

```
<body>
        <table>
            <caption>Table colgroup with id Example</caption>
            <colgroup span="1" id="groupA"></colgroup>
            <colgroup span="2" id="groupB"></colgroup>
            <colgroup span="2" id="groupC"></colgroup>
            <thead>
              <tr>
              <th>Group A</th><th colspan="2">Group B</th><th colspan="2">Group C</th>
              </tr>
            </thead>
            <tbody>
                <tr><td>1</th><td>2</td></th><td>3</td><td>4</td><td>5</td></tr>
                <tr><td>6</th><td>7</td></th><td>8</td><td>9</td><td>10</td></tr>
                <tr><td>11</th><td>12</td></th><td>13</td><td>14</td><td>15</td></tr>
                <tr><td>16</th><td>17</td></th><td>18</td><td>19</td><td>20</td></tr>
            </tbody>
            </table>
    </body>
</html>
```

Normal Column

Hide Columns

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

Hidden Column

Hide Columns

THU	FRI	SAT	SUN
4	5	6	7
11	12	13	14
18	19	20	21
25	26	27	28

Hiding Columns

hiding_column.html

style="visibility: collapse"

Form

SECTION 4

How Forms work?

There are two parts to a working form:

- The first part is **the form that you see** on the page itself that is created using HTML markup. Forms are made up of buttons, input fields, and drop-down menus (collectively known as form controls) used to collect information from the user. Forms may also contain text and other elements.
- The other component of a web form is **an application or script on the server that processes the information** collected by the form and returns an appropriate response. It's what makes the form *work*. In other words, posting an HTML document with form elements isn't enough. Web applications and scripts require programming know-how that is beyond the scope of this book, but the "Getting Your Forms to Work" sidebar, later in this chapter, provides some options for getting the scripts you need.

From Data Entry to Response

1. Your visitor—let's call her Sally—opens the page with a web form in the browser window. The browser sees the form control elements in the markup and renders them with the appropriate form controls on the page, including two text-entry fields and a Submit button.
2. Sally would like to sign up for this mailing list, so she enters her name and email address into the fields and submits the form by hitting the Submit button.
3. The browser collects the information she entered, encodes, and sends it to the web application on the server.
4. The web application accepts the information and processes it. In this example, the name and email address are added to a mailing list database.

From Data Entry to Response

5. The web application also returns a response. The kind of response sent back depends on the content and purpose of the form. Here, the response is a simple web page saying thank you for signing up for the mailing list. Other applications might respond by reloading the form page with updated information, by moving the user on to another related form page, or by issuing an error message if the form is not filled out correctly, to name only a few examples.
6. The server sends the web application's response back to the browser, where it is displayed. Sally can see that the form worked and that she has been added to the mailing list.

From Data Entry to Response

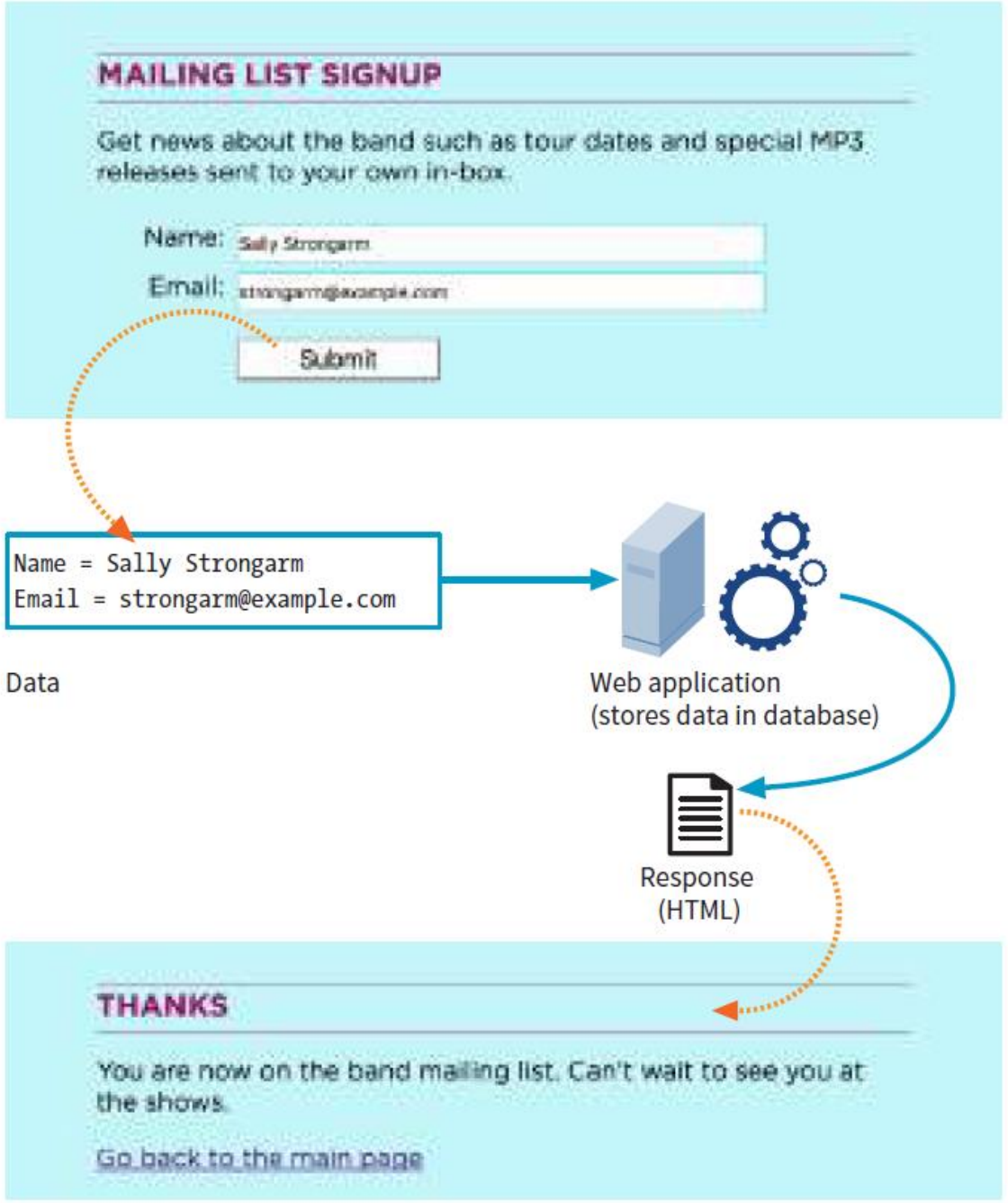
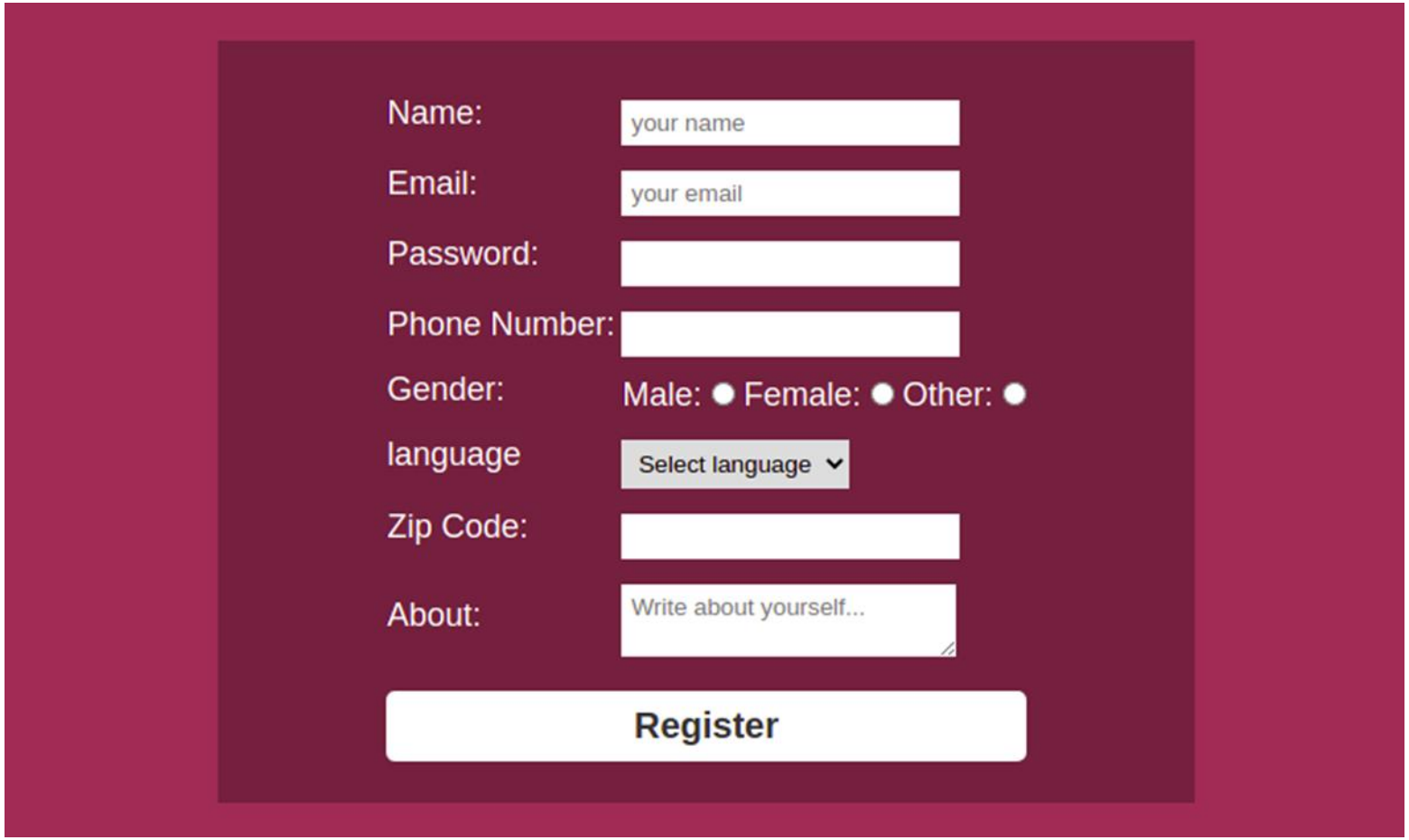
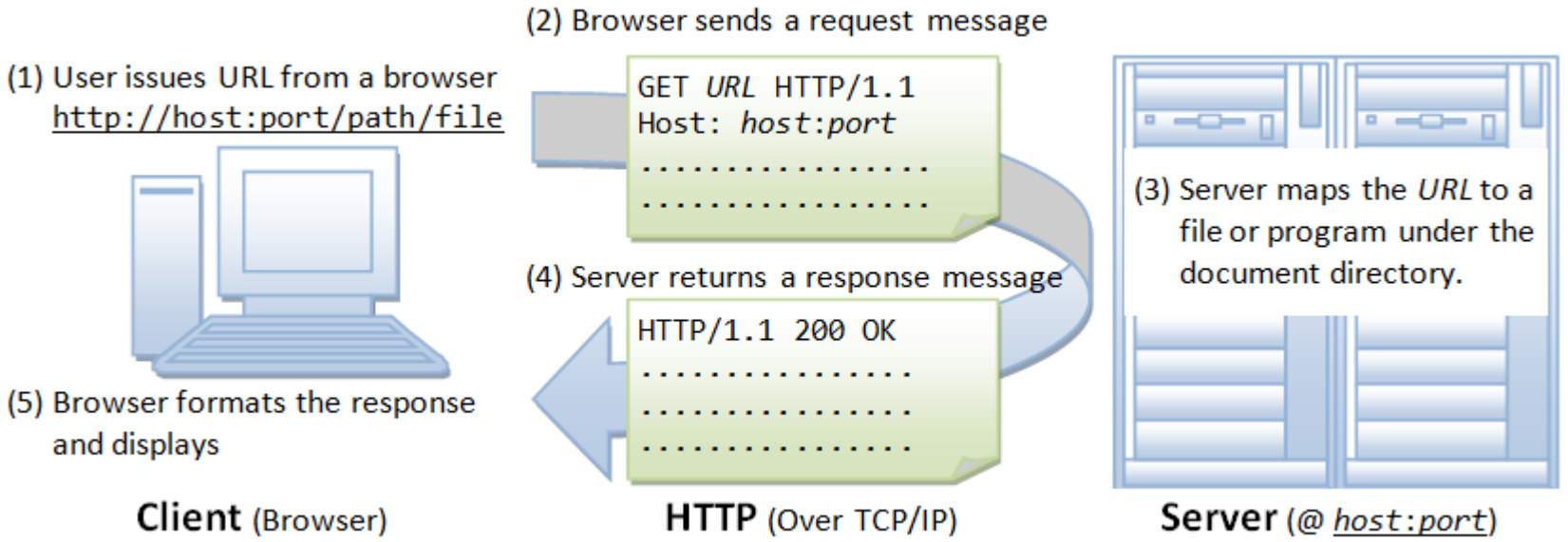


FIGURE 9-1. What happens behind the scenes when a web form is submitted.

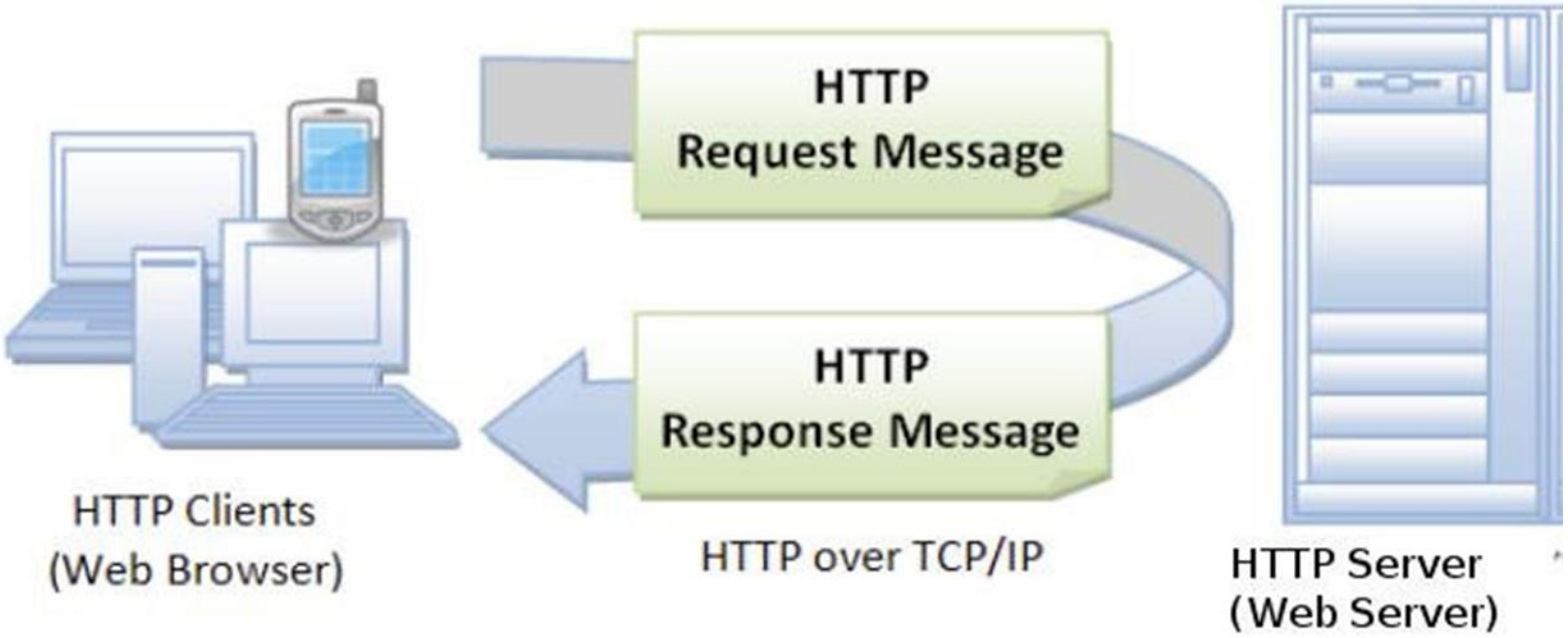
Part 1: From Preparation and Submission



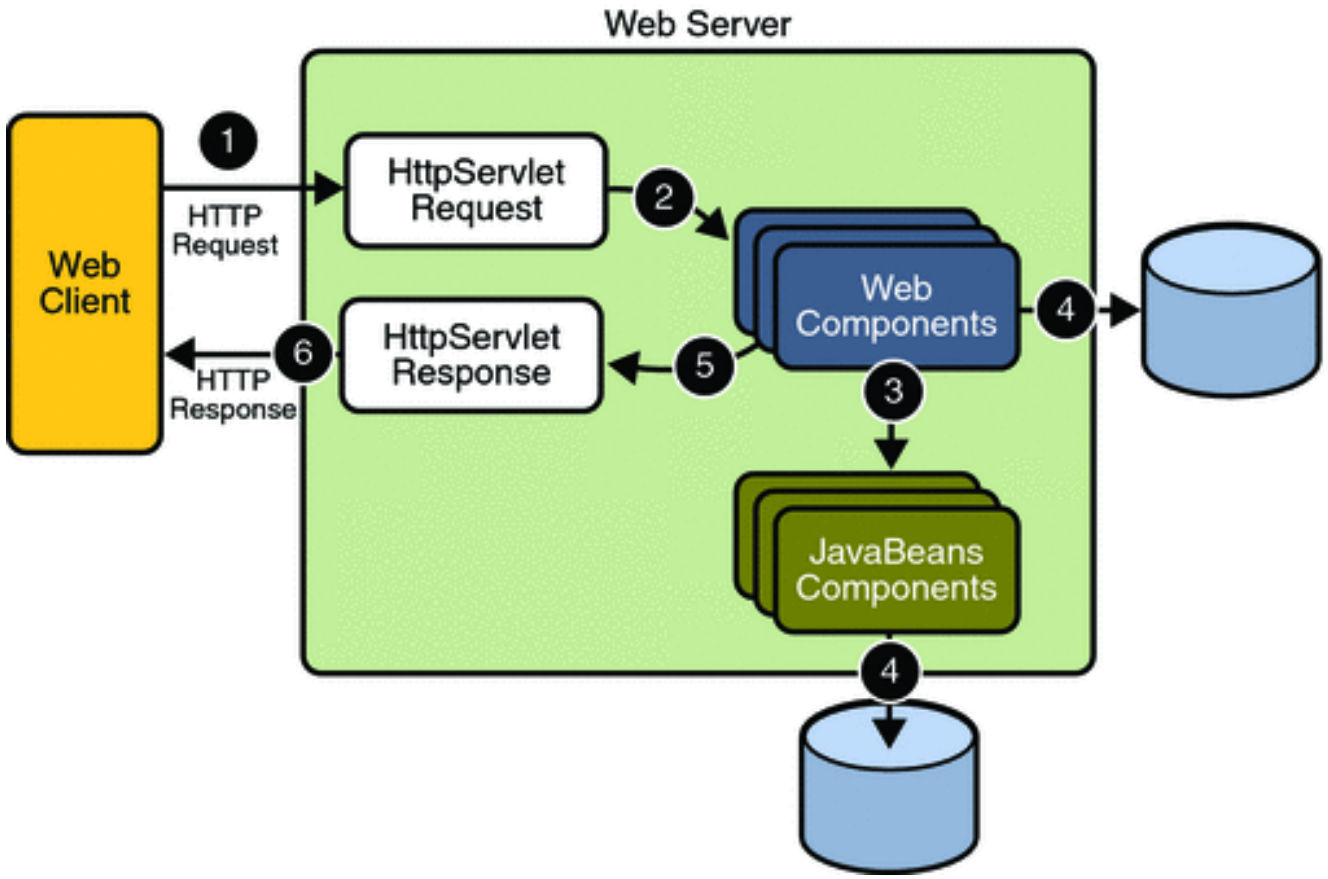
Part 1: From Preparation and Submission



Part 2: Processing the Form

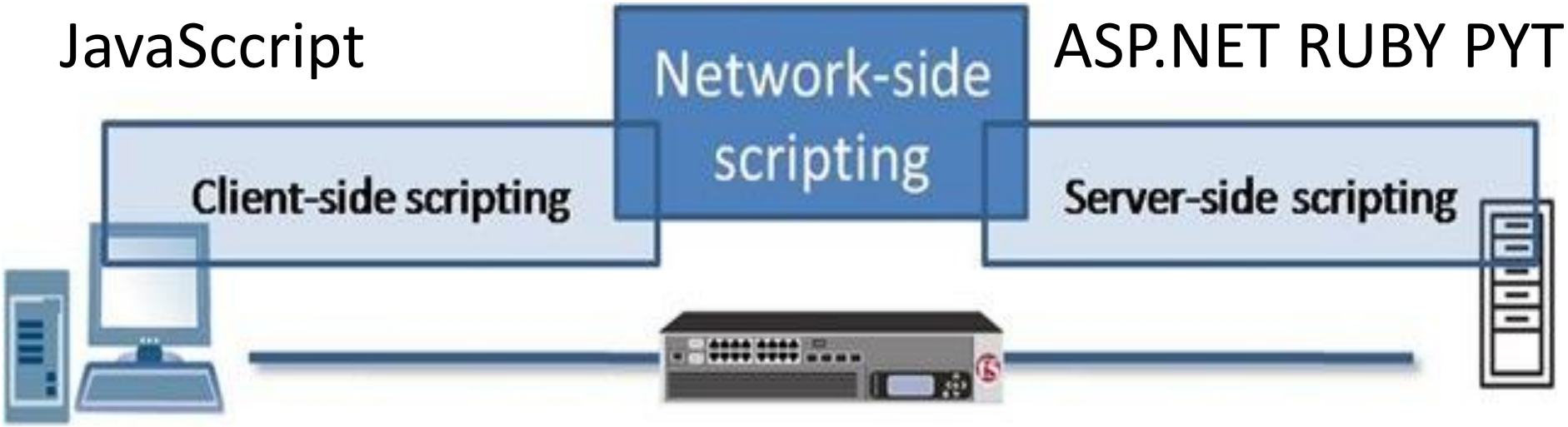


Part 2: Processing the Form



JavaSccript

ASP.NET RUBY PYTHON JSP PHP

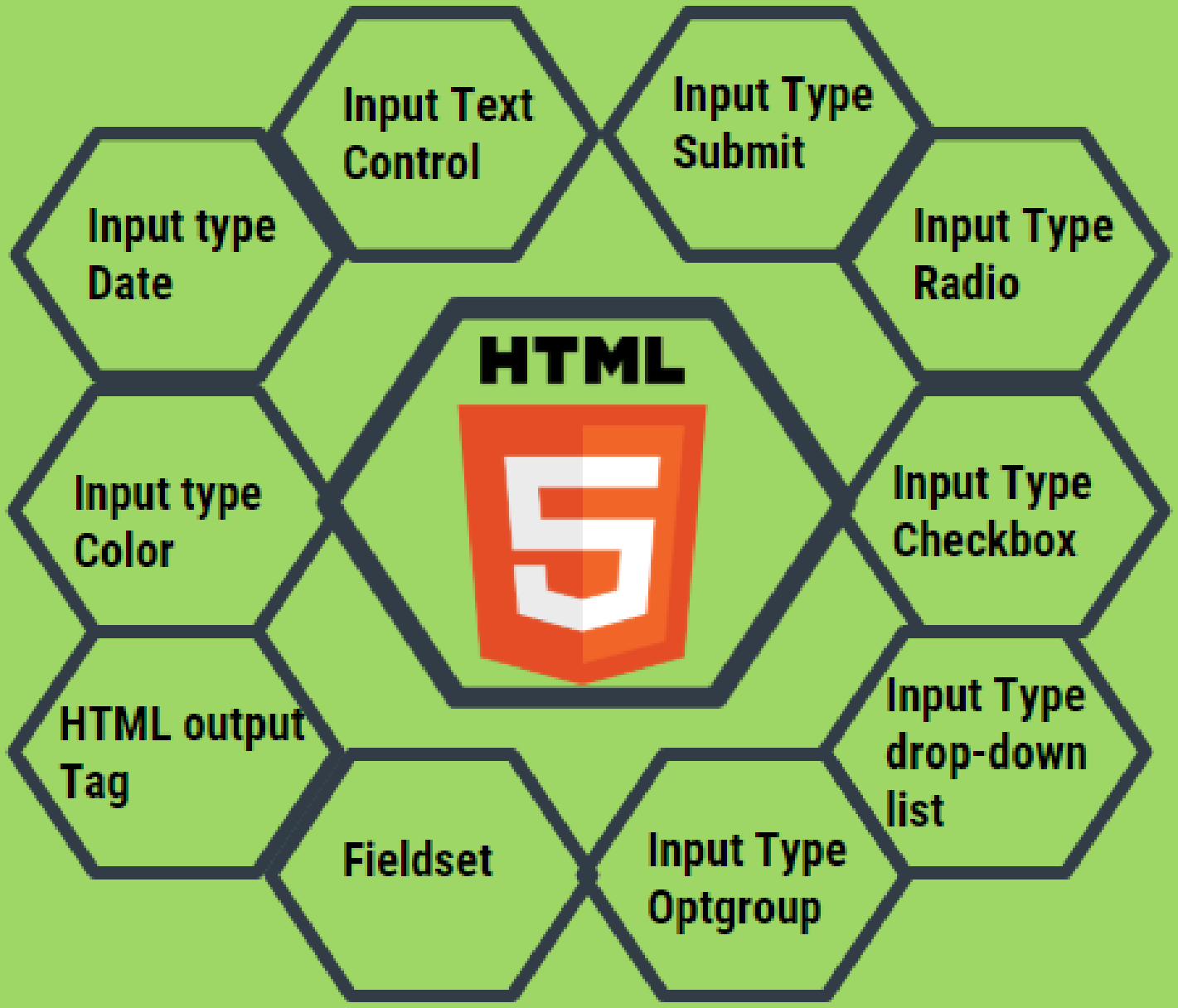


Web Input Form Request and Response

The Form Elements

SECTION 5


Click here for more details



HTML Form Controls

The Form Element

```
<!DOCTYPE html>
<html>
<head>
  <title>Mailing List Signup</title>
  <meta charset="utf-8">
</head>
<body>
  <h1>Mailing List Signup</h1>

  <form action="/mailinglist.php" method="POST">
    <fieldset>
      <legend>Join our email list</legend>
      <p>Get news about the band such as tour dates and special MP3
releases sent to your own in-box.</p>
      <ol>
       <li><label for="firstlast">Name:</label>
         <input type="text" name="fullname" id="firstlast"></li>
       <li><label for="email">Email:</label>
         <input type="text" name="email" id="email"></li>
      </ol>
      <input type="submit" value="Submit">
    </fieldset>
  </form>

</body>
</html>
```

The *.php* suffix indicates that this form is processed by a script written in the PHP scripting language, but web forms may be processed by any of the following technologies:

- **PHP (*.php*)** is an open source scripting language most commonly used with the Apache web server. It is the most popular and widely supported forms processing option.
- **Microsoft ASP (Active Server Pages; *.asp*)** is a programming environment for the Microsoft Internet Information Server (IIS).
- **Microsoft's ASP.NET** (Active Server Page; *.aspx*) is a newer Microsoft language that was designed to compete with PHP.
- **Ruby on Rails**. Ruby is the programming language that is used with the Rails platform. Many popular web applications are built with it.
- **JavaServer Pages (*.jsp*)** is a Java-based technology similar to ASP.
- **Python** is a popular scripting language for web and server applications.

The action Attribute

```
<form action="/mailinglist.php"
      method="POST">
...
</form>
```

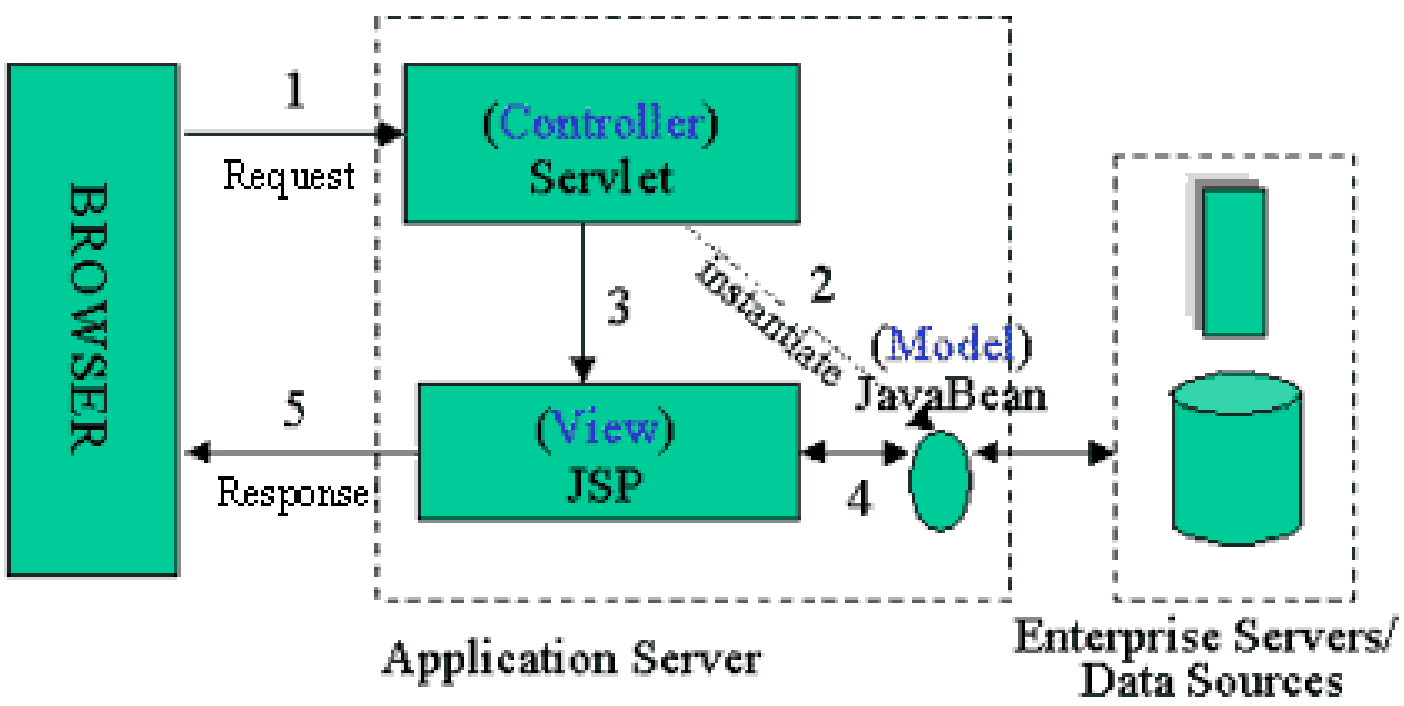
```
11  </head>
12  <body>
13    <h1>Demonstrating Forms</h1>
14    <form name="demo"
15          action="mailto:myClient@gmail.com"
16          method="get">
17
18        Your Name:
19        <input type="text"
20               name="txtFname"
21               size="20"
22               maxlength="25"
23               value="Sally" />
```

The action Attribute

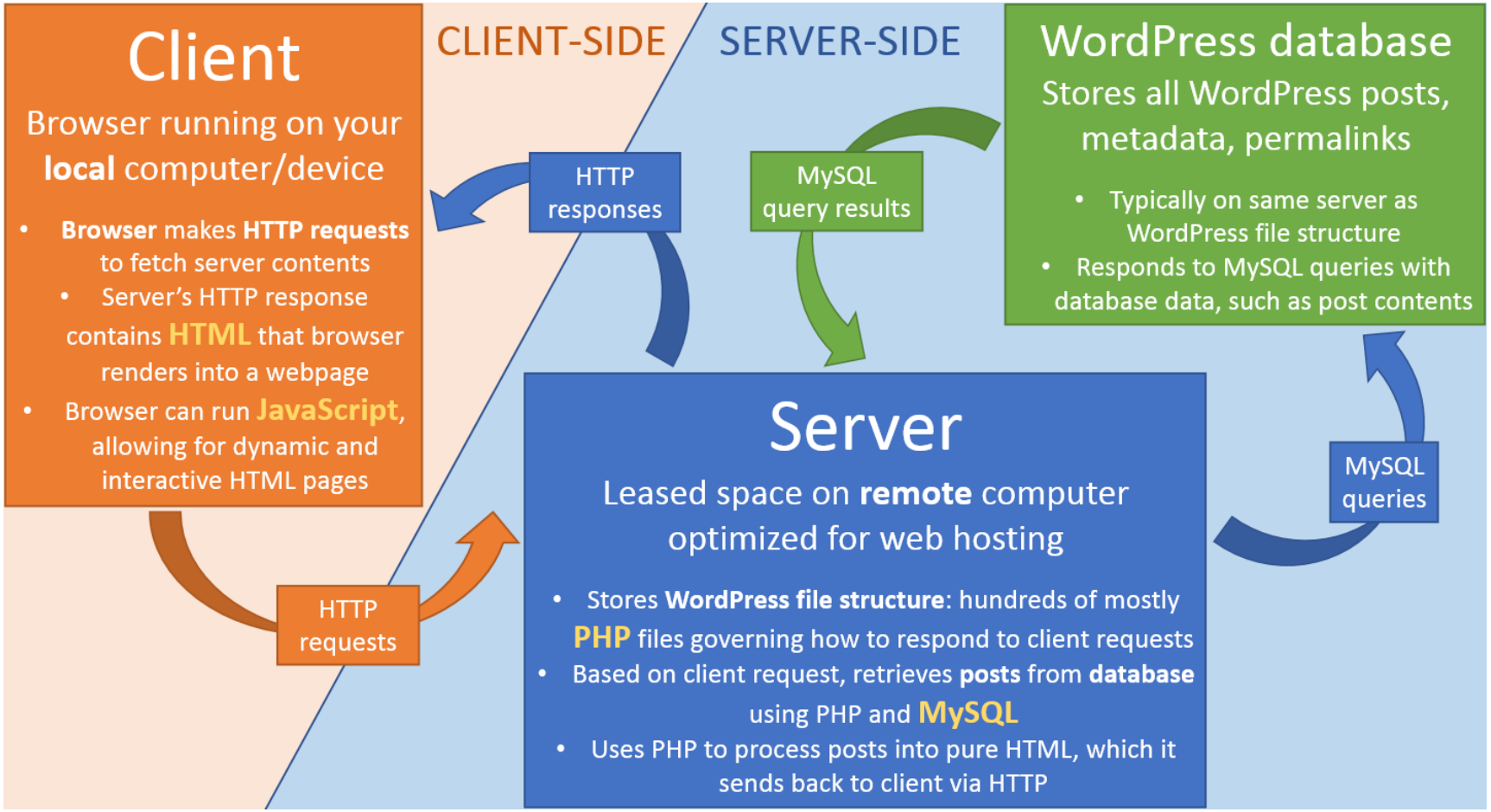
```
<form action="/mailinglist.php"
        method="POST">
...
</form>
```

Form Action Attribute

What Program will Work on The Form Variables



Server-side and Client-side Scripting



The *.php* suffix indicates that this form is processed by a script written in the PHP scripting language, but web forms may be processed by any of the following technologies:

- **PHP (*.php*)** is an open source scripting language most commonly used with the Apache web server. It is the most popular and widely supported forms processing option.
- **Microsoft ASP (Active Server Pages; *.asp*)** is a programming environment for the Microsoft Internet Information Server (IIS).
- **Microsoft's ASP.NET** (Active Server Page; *.aspx*) is a newer Microsoft language that was designed to compete with PHP.
- **Ruby on Rails**. Ruby is the programming language that is used with the Rails platform. Many popular web applications are built with it.
- **JavaServer Pages (*.jsp*)** is a Java-based technology similar to ASP.
- **Python** is a popular scripting language for web and server applications.

The method Attribute

<form action="/mailinglist.php" method="POST">...</form>

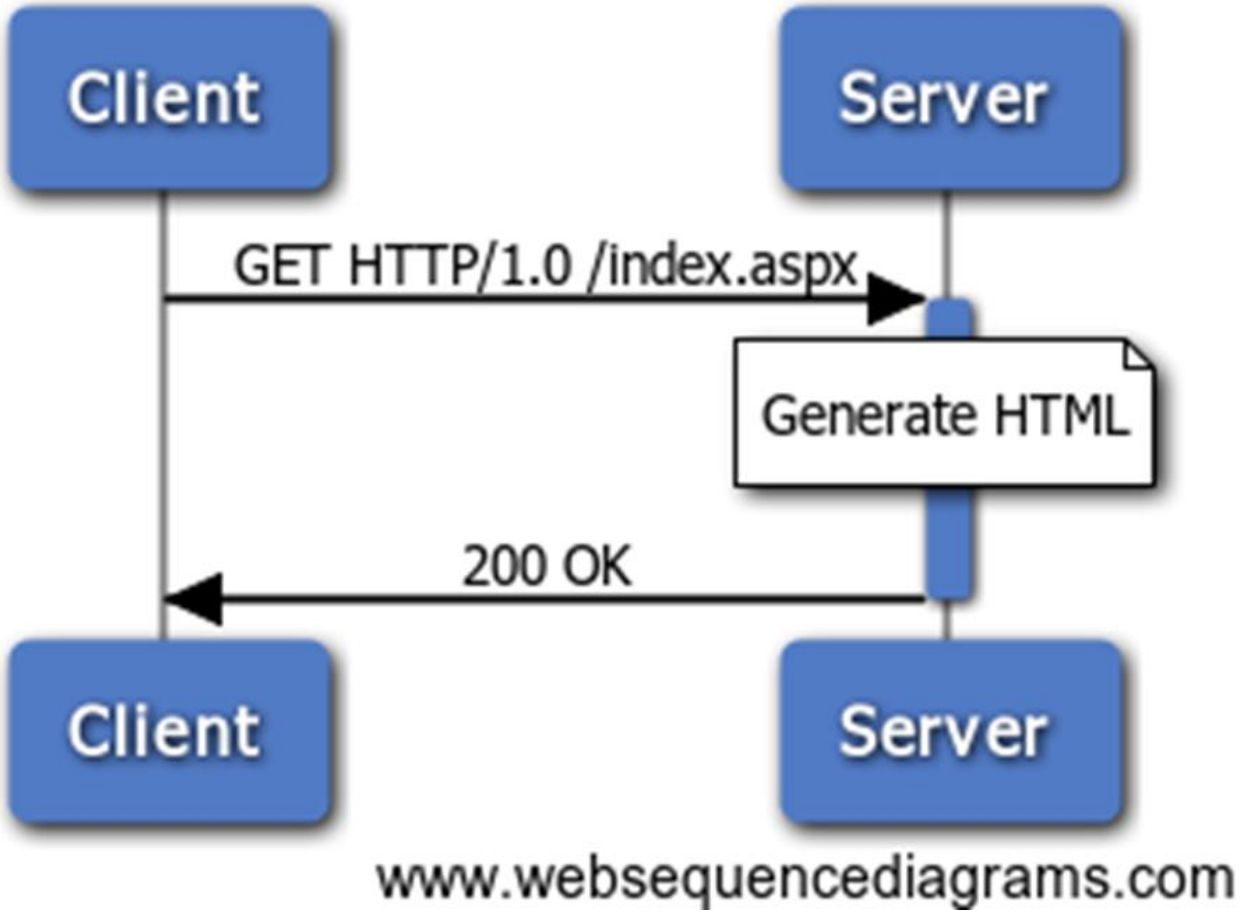
- With the GET method, the encoded form data gets tacked right onto the URL sent to the server. A question mark character separates the URL from the following data, as shown here:

  get http://www.bandname.com/mailinglist.php?name=Sally+Strongarm&email=strongarm%40example.com

- GET is inappropriate if the form submission performs an action, such as deleting something or adding data to a database, because if the user goes back, it gets submitted again.

GET Method

```
<form action="/mailinglist.php"
      method="POST">
...
</form>
```



GET Method

```
<form action="/mailinglist.php"
        method="POST">
...
</form>
```

- When the form’s method is set to POST, the browser sends a separate server request containing some special headers followed by the data. In theory, only the server sees the content of this request, and thus it is the best method for sending secure information such as a home address or other personal information.
- In practice, make sure HTTPS is enabled on your server so the user’s data is encrypted and inaccessible in transit.

Post Method

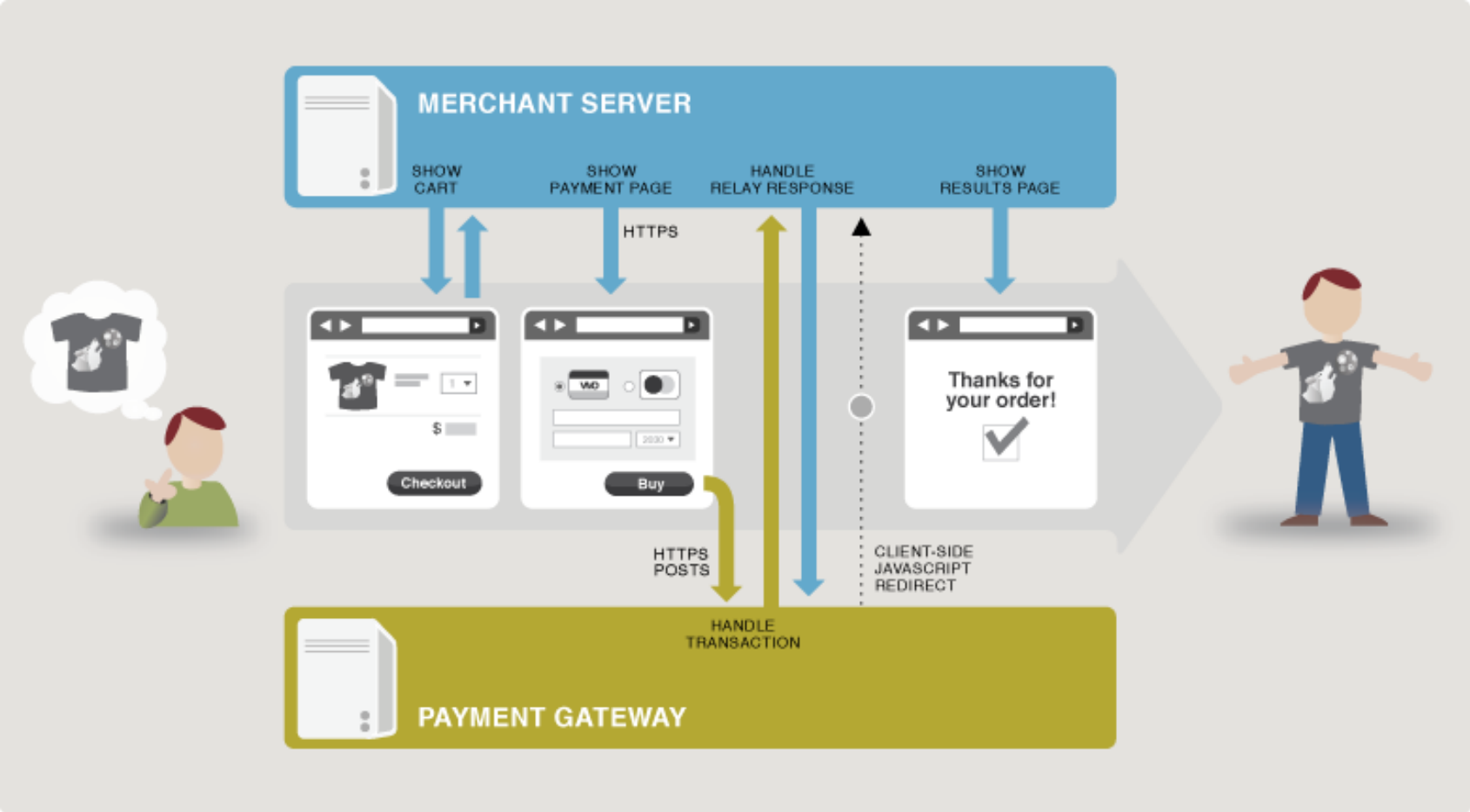
```
<form action="/mailinglist.php"
      method="POST">
...
</form>
```

- The **POST** method is also preferable for sending a lot of data, such as a lengthy text entry, because there is no character limit as there is for GET.
- The **GET** method is appropriate if you want users to be able to bookmark the results of a form submission (such as a list of search results). Because the content of the form is in plain sight, GET is not appropriate for forms with private personal or financial information. In addition, GET may not be used when the form is used to upload a file.

Post Method

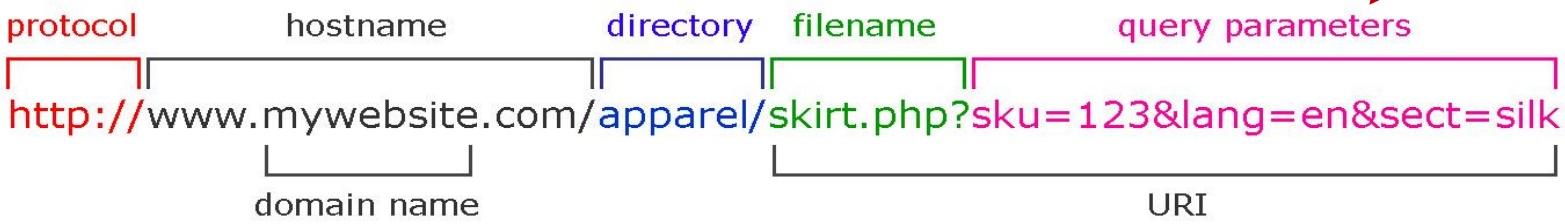
```
<form action="/mailinglist.php"
        method="POST">
...
</form>
```

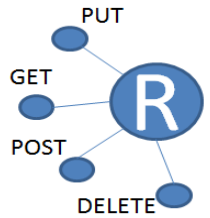
POST Method

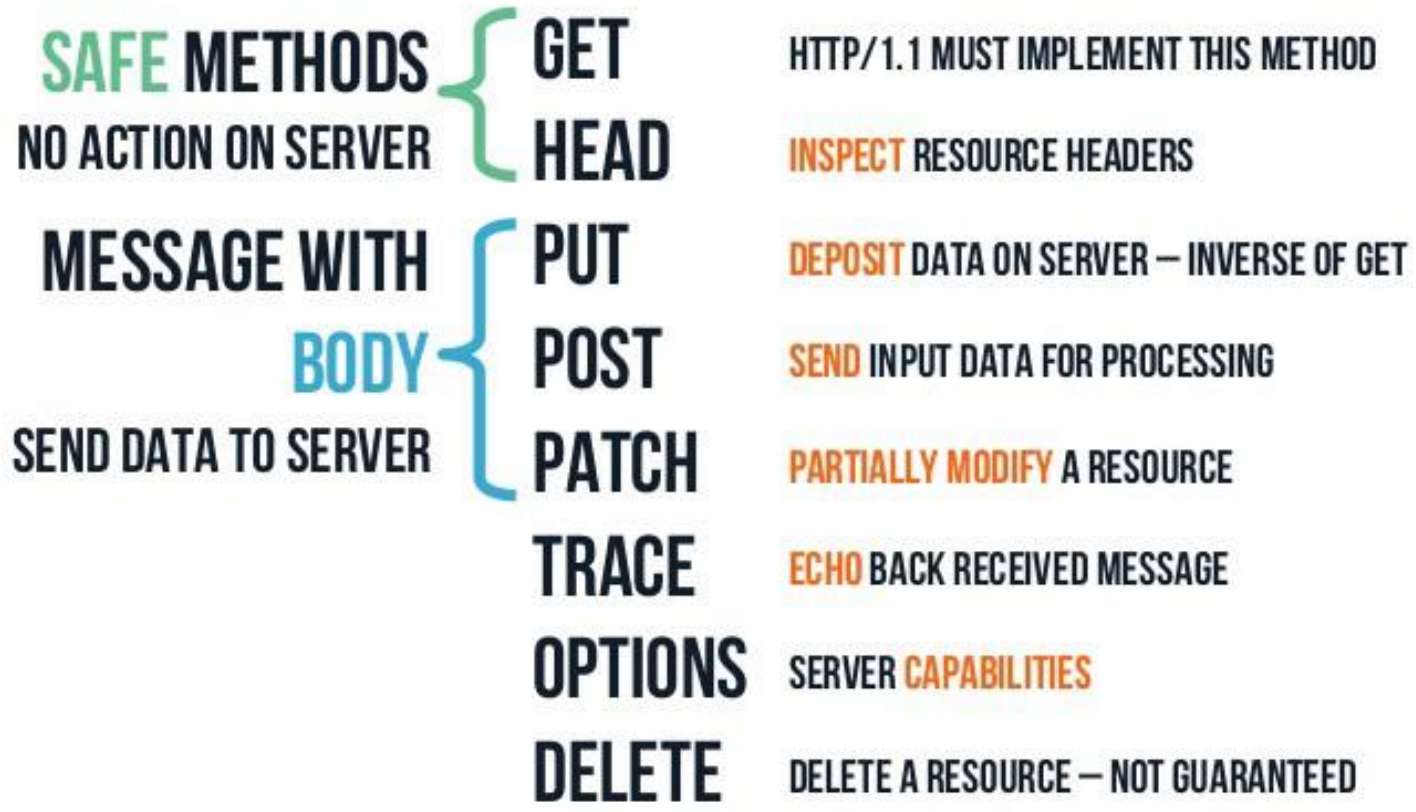

MERCHANT SERVER
SHOW CART
SHOW PAYMENT PAGE
HANDLE RELAY RESPONSE
SHOW RESULTS PAGE
HTTPS
Checkout
Buy
Thanks for your order!
HTTPS POSTS
CLIENT-SIDE JAVASCRIPT REDIRECT
HANDLE TRANSACTION
PAYMENT GATEWAY

HTTP Methods

- **GET** ····→ Request for a web page or an object from server
- **PUT** ····→ For sending a document to the server
- **POST** ····→ For sending data or information about client to the server
- **DELETE** ····→ Request to Delete an object on the server
- **HEAD** ····→ Request for information about a web page or a document
- **TRACE** ····→ Used to trace the proxies and tunnels in the path from client to server
- **OPTION** ····→ Used to determine server's capabilities






SAFE METHODS
NO ACTION ON SERVER
MESSAGE WITH
BODY
SEND DATA TO SERVER
GET
HEAD
PUT
POST
PATCH
TRACE
OPTIONS
DELETE
HTTP/1.1 MUST IMPLEMENT THIS METHOD
INSPECT RESOURCE HEADERS
DEPOSIT DATA ON SERVER – INVERSE OF GET
SEND INPUT DATA FOR PROCESSING
PARTIALLY MODIFY A RESOURCE
ECHO BACK RECEIVED MESSAGE
SERVER CAPABILITIES
DELETE A RESOURCE – NOT GUARANTEED

- When a user enters a comment in the field (“This is the best band ever!”), it would be passed to the server as a name/value (variable/content) pair like this:

  comment=This+is+the+best+band+ever%21

- All form control elements must include a name attribute so the form processing application can sort the information. You may include a name attribute for submit and reset button elements, but they are not required, because they have special functions (submitting or resetting the form) not related to data collection.

The name Attribute

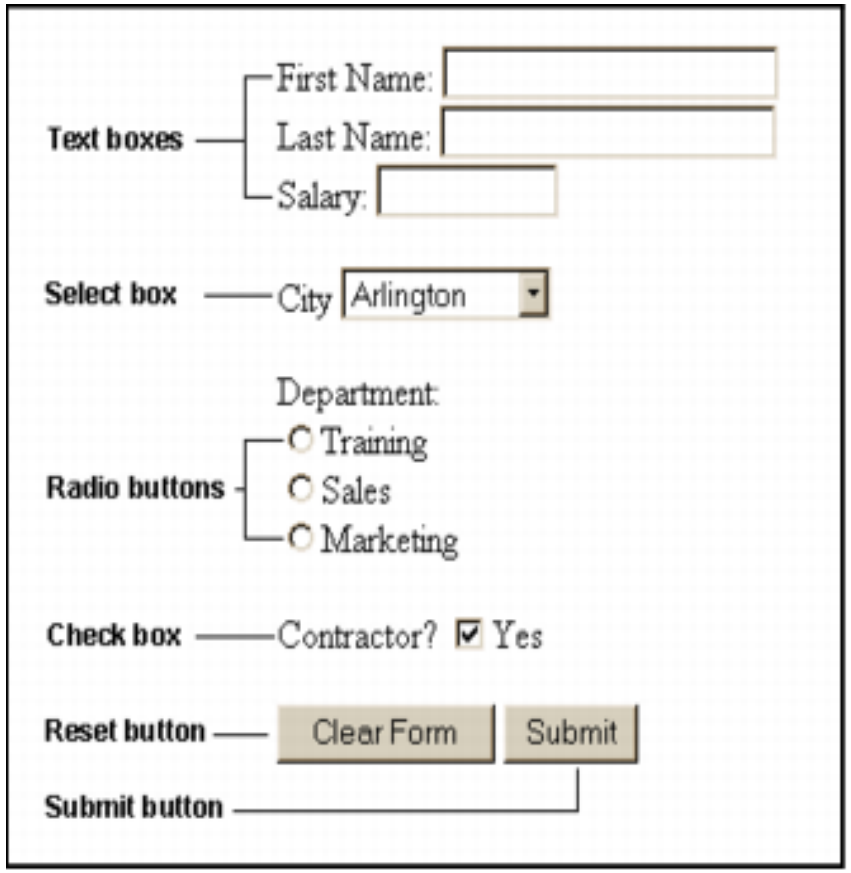
```
<textarea name="comment"
rows="4" cols="45"
placeholder="Leave us a
comment."></textarea>
```

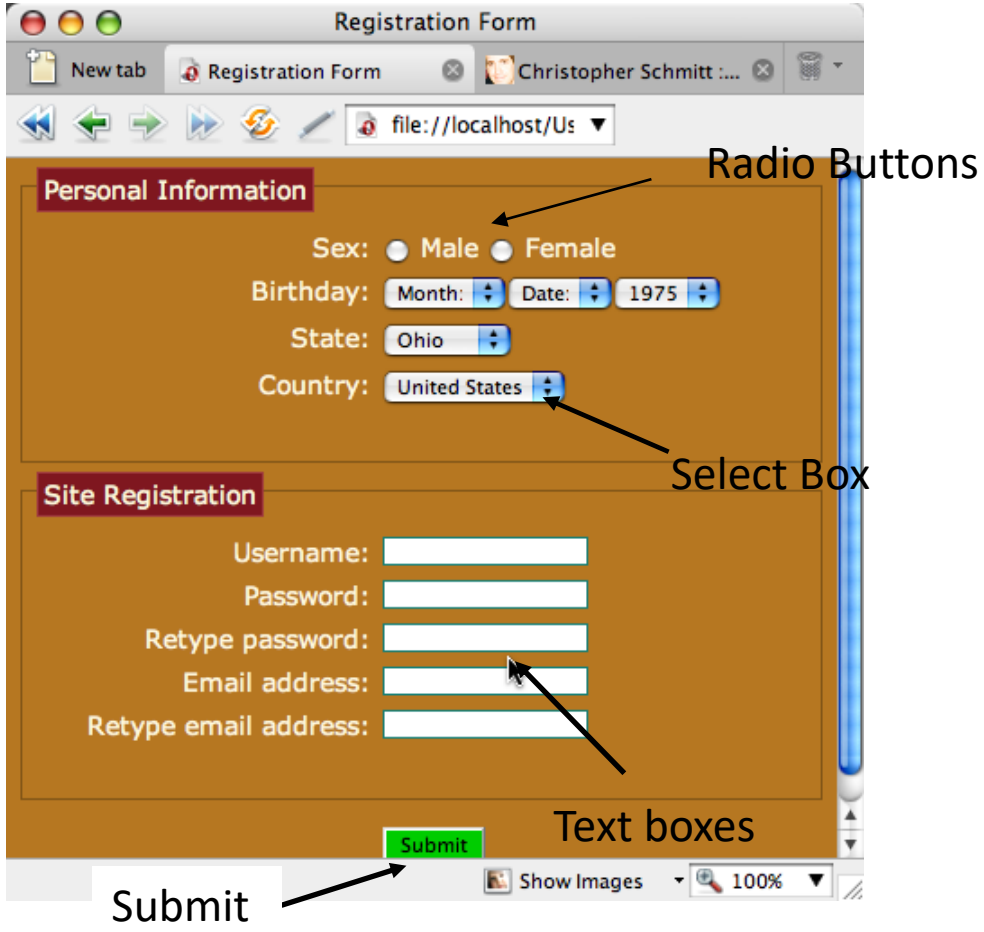
The Great Form of Control Roundup

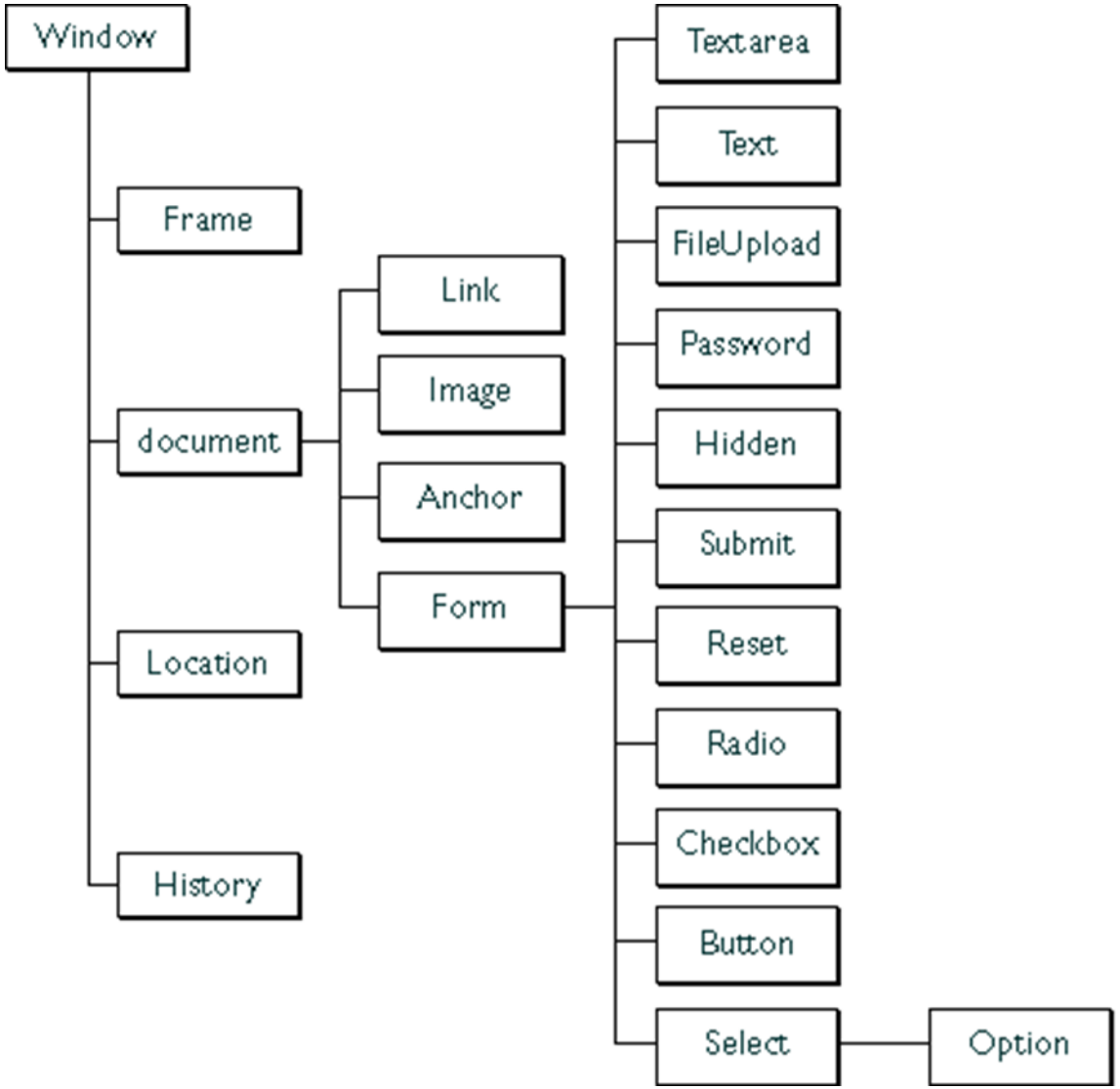
SECTION 6

Input Elements

Basic Form Elements







Input Elements

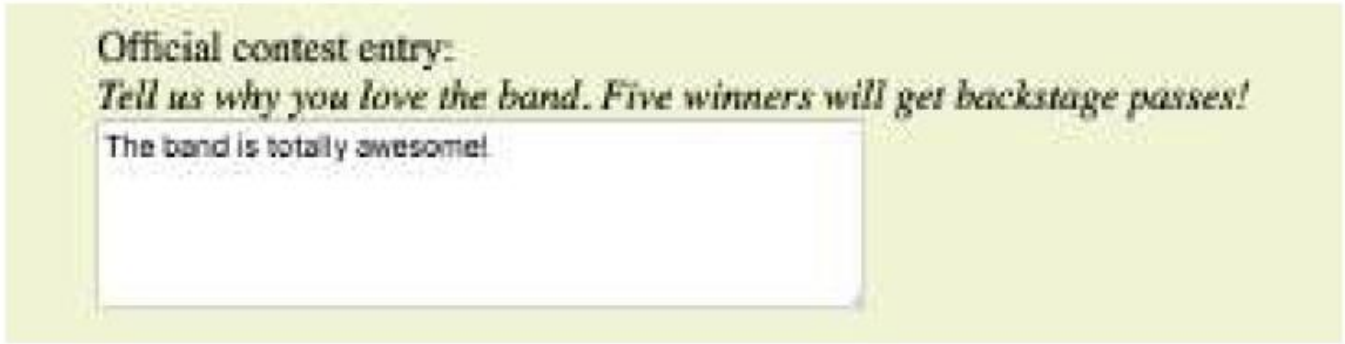
The Input Elements shown in Textbook

- Text Entry Controls
- Specialized text entry controls
- Submit and reset buttons
- Radio and checkbox buttons
- Pull-down and scrolling menus
- File selection and upload control
- Hidden controls
- Date and times (HTML5)
- Numerical controls (HTML5)
- Color picker control (HTML5)

```
<li><label>Favorite color: <input type="text" name="favcolor"
value="Red" maxlength="50"></label></li>
```

Text-entry field (`input type="text"`)

Multiline text-entry field with text content (`input type="textarea"`)



Multiline text-entry field with placeholder text (`input type="textarea"`)

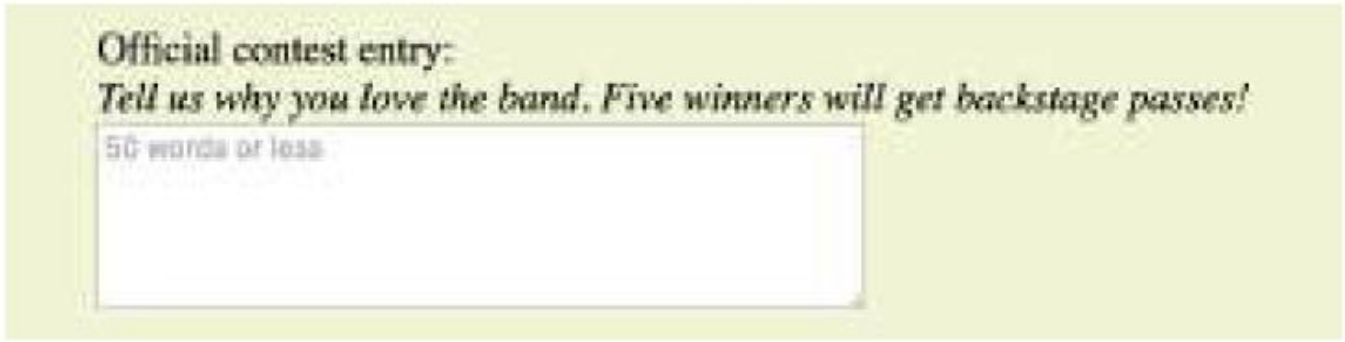


FIGURE 9-3. Examples of the text-entry control options for web forms.

Text-Entry Controls

Single-line text field

<input type="text">

Single-line text-entry control

- <label>City **<input type="text" name="city" id="form-city" value="Your Hometown" maxlength="50">**</label>
- specify a line item element
- <label>City **<input ….>**</label> associates the label City with the input text box.
- **name:** The name attribute is required for indicating the variable name.
- **value:** The value attribute specifies default text that appears in the field when the form is loaded. When you reset a form, it returns to this value.
- **maxlength:** limit your text string length.

Text-Entry Controls

Single-line text field

<input type="text">

Single-line text-entry control

- <label>Official contest entry:
  Tell us why you love the band. Five winners will get backstage passes!
  <textarea name="contest_entry" rows="5" cols="50">The band is totally awesome!</textarea></label>
- **title:** title of the textarea, also used as a hint
- **placeholder:** a hint of what it is in the text field
- **rows:** row number for showing the text area
- **cols:** col number for showing the text area
- **wrap:** specifies whether the text should keep its line breaks when submitted.
  soft means do not keep it. Hard means to keep it.
- **maxlength:** sets a limit on the number of characters that can be typed into the field

Text Entry Controls

Multiple lines text entry

<textarea>…</textarea>

Multiline text-entry control

```
<input type="password">
<input type="search">
<input type="email">
<input type="tel">
<input type="url">
<input type="number">
<input type="text"
    list="ideselect"
    name="idesl">
    <datalist id="ideselect">
    <option value="Visual Studio">
    <option value="VIM">
```

Specialized Text-Entry Fields

```
<li><label for="form-pswd">Password:</label><br>
 <input type="password" name="pswd" maxlength="12" id="form-pswd"></li>
```

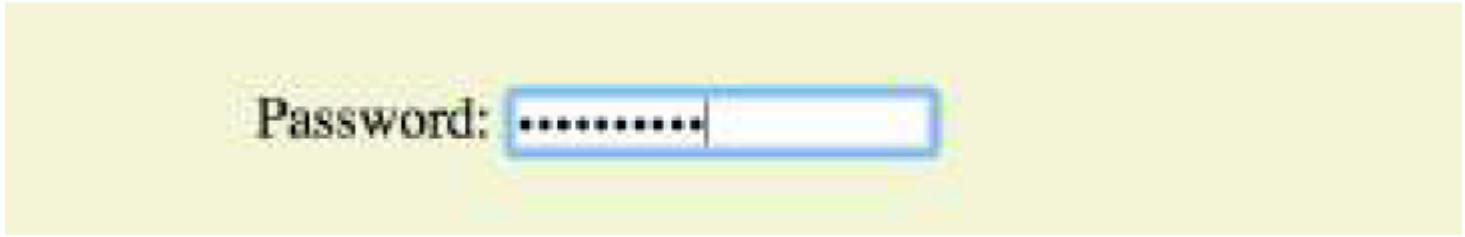


FIGURE 9-4. Passwords are converted to bullets in the browser display.

Specialized Text-Entry Fields

<input type="password">
Password text control

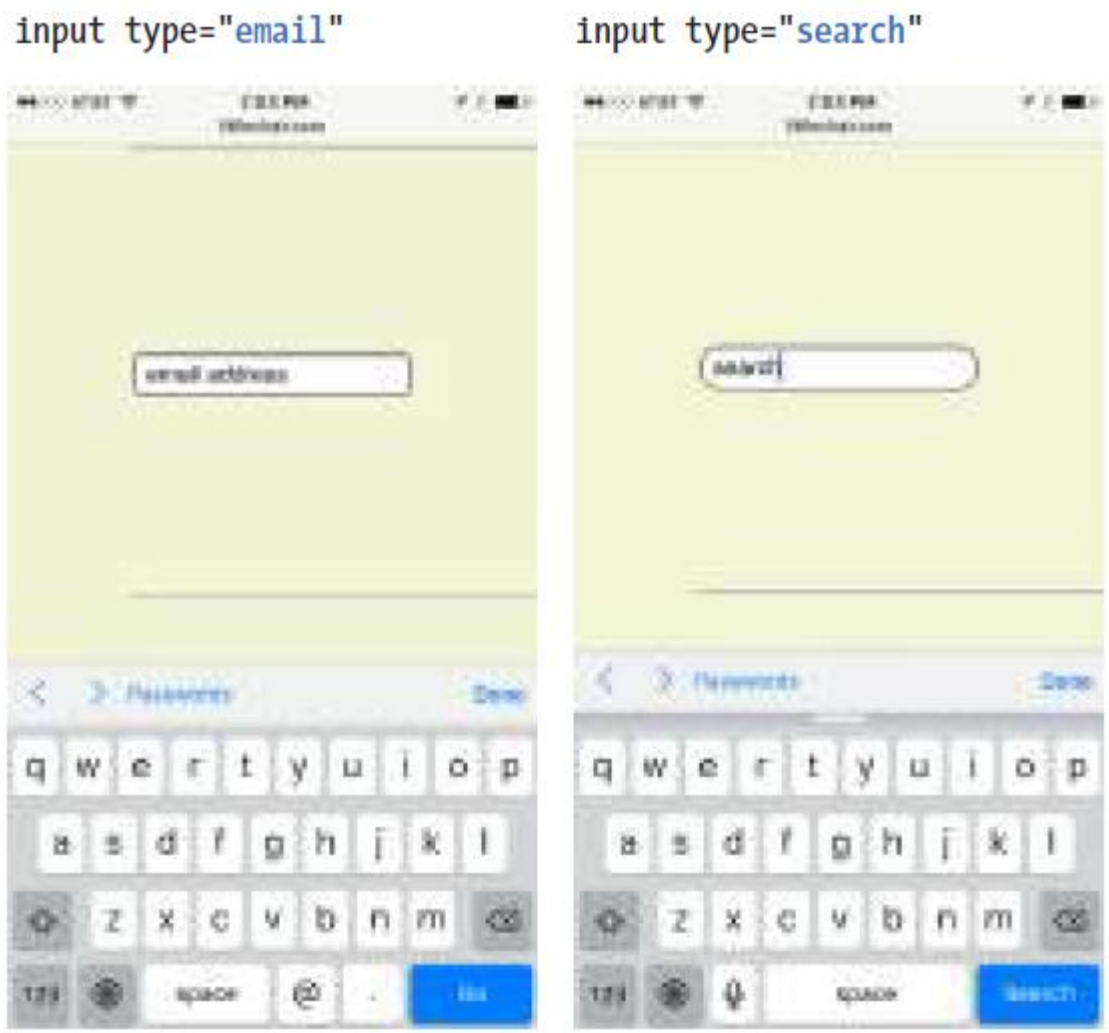


FIGURE 9-5. Safari on iOS provides custom keyboards based on the input type.

Specialized Text-Entry Fields

<input type="search">
Search field
<input type="email">
Email address
<input type="tel">
Telephone number
<input type="url">
Location (URL)

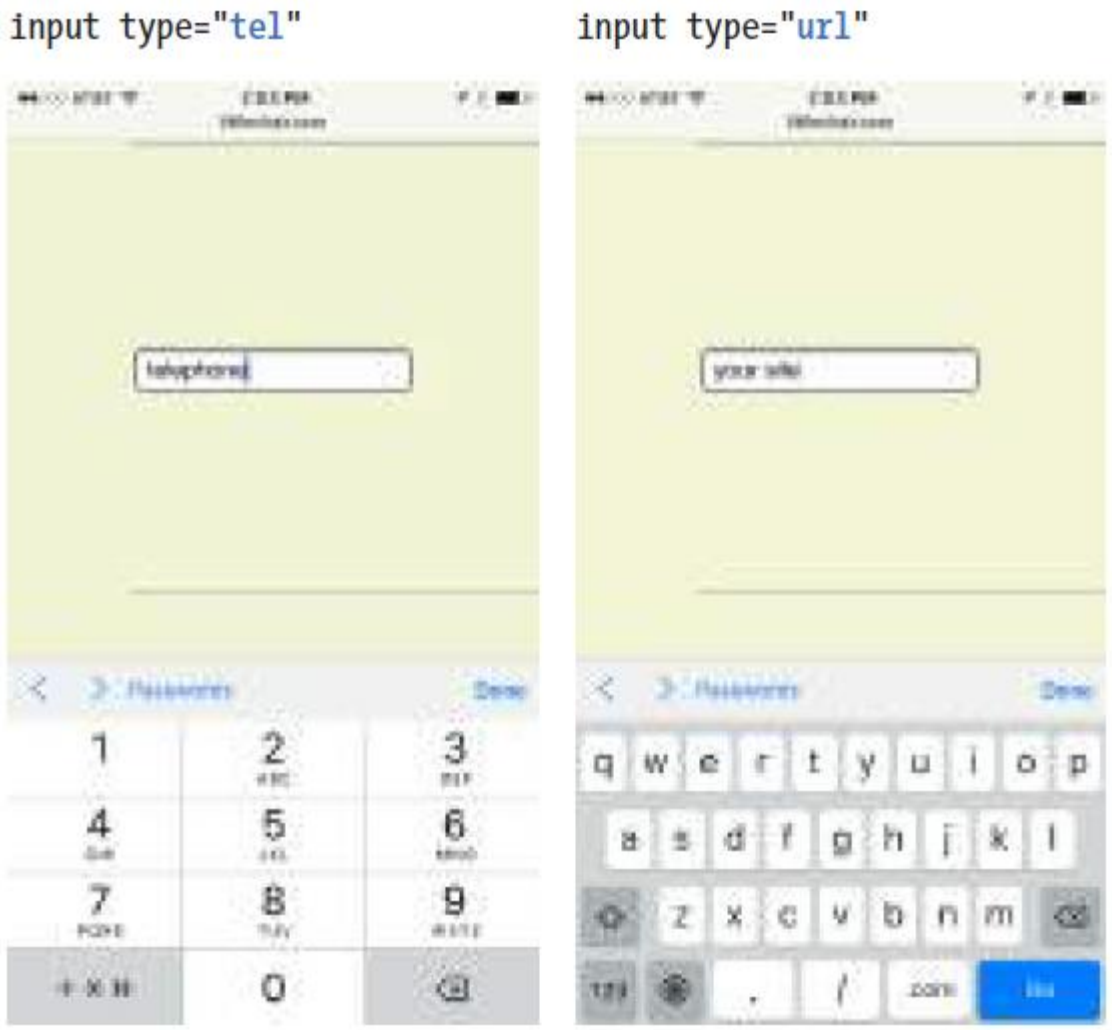


FIGURE 9-5. Safari on iOS provides custom keyboards based on the input type.

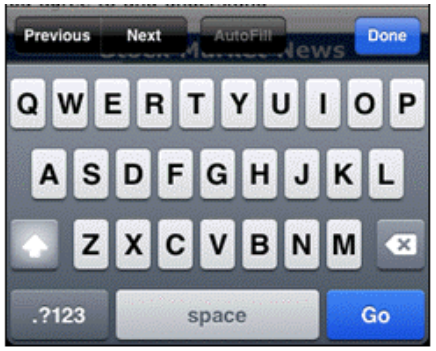
Specialized Text-Entry Fields

<input type="search">
Search field
<input type="email">
Email address
<input type="tel">
Telephone number
<input type="url">
Location (URL)


```
<input type="text"
    list="ideselect"
    name="idesl">
    <datalist id="ideselect">
    <option value="Visual Studio">
    <option value="VIM">
```

Keyboad when <input type="tel">

Keyboad when <input type="url">



Keyboard when <input type="text">

Keyboard when <input type="email">

```
<input type="password">
<input type="search">
<input type="email">
<input type="tel">
<input type="url">
<input type="number">
```

- The **datalist** element allows the author to provide a drop-down menu of suggested values for any type of text input. It gives the user some shortcuts to select from, but if none are selected, the user can still type in their own text.
- Within the **datalist** element, suggested values are marked up as option elements. Use the list attribute in the input element to associate it with the id of its respective **datalist**.

Drop-Down Suggestions

<datalist>…</datalist>
Drop-down menu input

```
<p>Education completed: <input type="text" list="edulevel"
name="education">

<datalist id="edulevel">
  <option value="High School">
  <option value="Bachelors Degree">
  <option value="Masters Degree">
  <option value="PhD">
</datalist>
```

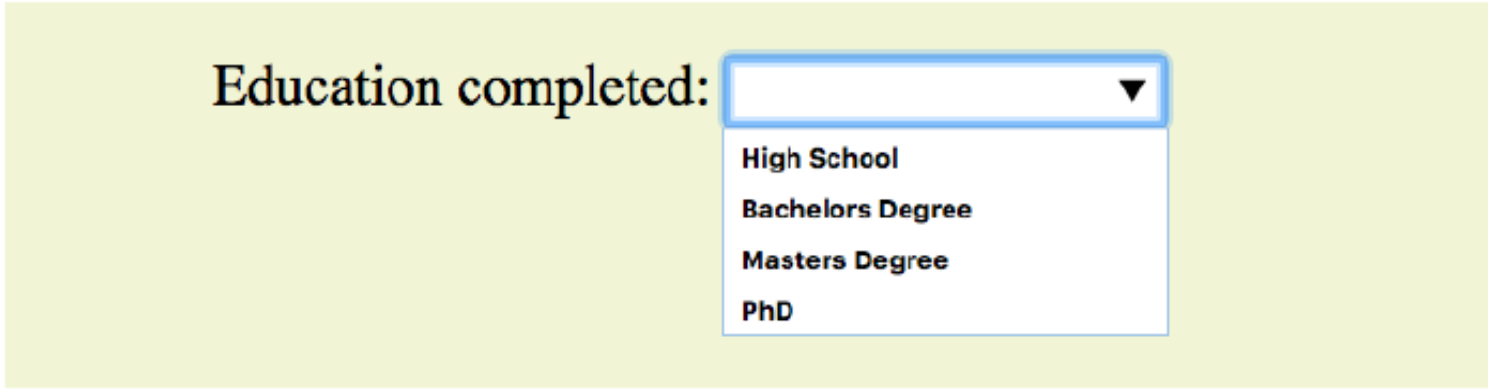


FIGURE 9-7. A **datalist** creates a pop-up menu of suggested values for a text-entry field.

Drop-Down Suggestions

<datalist>…</datalist>
Drop-down menu input

Buttons

SECTION 7

<input type="**submit**">: submit the form to take action (client side or server side)

<input type="**reset**">: clean up the form

<input type="**image**">: use external image as the button face

<input type="**button**">

<input type="**radio**">

<input type="**checkbox**">

<button></button>: type some text or show some image on the button face

Buttons

Submit, Reset, Radio and Checkbox, Image, Button, Button Element

```
<p><input type="submit"> <input type="reset" value="Start over"></p>
```

FIGURE 9-8. Submit and reset buttons.

Submit and Reset Buttons

<input type="submit">
Submits the form data to the server
<input type="reset">
Resets the form controls to their default settings

Image buttons

<input type="image">

- This type of **input** control allows you to replace the submit button with an image of your choice. The image will appear flat, not like a 3-D button.
- Unfortunately, this type of button has accessibility issues, so be sure to include a carefully chosen **alt** value.

- Setting the type of the **input** element to “button” creates a button that can be customized with JavaScript.
- It has no predefined function on its own, unlike submit and reset buttons.

Custom input button

```
<input type="button">
```

The button element

<button>…</button>

- The **button** element is a flexible element for creating custom buttons similar to those created with the **input** element. The content of the **button** element (text and/or images) is what gets displayed on the button.
- For more information on what you can do with the **button** element, read “Push My Button” by Aaron Gustafson at *digital-web.com/articles/push_my_button*. “When to Use the Button Element,” by Chris Coyier is another helpful read (*csstricks.com/use-button-element/*).

Radio Buttons and Checkboxes

SECTION 8

Radio and Checkbox Buttons

- Both checkbox and radio buttons make it simple for your visitors to choose from a number of provided options. They are similar in that they function like little on/off switches that can be toggled by the user and are added with the **input** element. They serve distinct functions, however. A form control made up of a collection of radio buttons is appropriate when only one option from the group is permitted—in other words, when the selections are mutually exclusive (such as “Yes or No,” or “Pick-up or Delivery”).
- When one radio button is “on,” all of the others must be “off,” sort of the way buttons used to work on old radios: press one button in, and the rest pop out.
- When checkboxes are grouped together, however, it is possible to select as many or as few from the group as desired. This makes them the right choice for lists in which more than one selection is OK.

Radio buttons

Radio buttons are added to a form via the **input** element with the **type** attribute set to “radio.” Here is the syntax for a minimal radio button:

<input type="radio" name="*variable*" value="*value*">

Default Value

```
<p>How old are you?</p>
<ol>
    <li><input type="radio" name="age" value="under24" checked>
    under
    24</li>
    <li><input type="radio" name="age" value="25-34"> 25 to 34</li>
    <li><input type="radio" name="age" value="35-44"> 35 to 44</li>
    <li><input type="radio" name="age" value="over45"> 45+</li>
</ol>
```

Same variable

Value candidates

Radio Buttons

<input type="radio">
Radio button

Checkbox buttons

Checkboxes are added via the input element with its type set to checkbox. As with radio buttons, you create groups of checkboxes by assigning them the same name value. The difference, as we've already noted, is that more than one checkbox may be checked at a time. The value of every checked button will be sent to the server when the form is submitted.

Default Value

```
<p>What type of music do you listen to?</p>
<ul>
    <li><input type="checkbox" name="genre" value="punk" checked> Punk rock</li>
    <li><input type="checkbox" name="genre" value="indie" checked> Indie rock</li>
    <li><input type="checkbox" name="genre" value="hiphop"> Hip Hop</li>
    <li><input type="checkbox" name="genre" value="rockabilly">
    Rockabilly</li>
</ul>
```

Same variable

Value candidates

Checkbox Buttons

<input type="checkbox">
Checkbox button

Radio and Checkbox Buttons

Radio buttons (`input type="radio"`)

How old are you?

- (•) under 24
- ( ) 25 to 34
- ( ) 35 to 44
- ( ) 45+

Checkboxes (`input type="checkbox"`)

What type of music do you listen to?

- [x] Punk rock
- [x] Indie rock
- [ ] Hip Hop
- [ ] Rockabilly

FIGURE 9-11. Radio buttons (left) are appropriate when only one selection is permitted. Checkboxes (right) are best when users may choose any number of choices, from none to all of them.

Menus

SECTION 9

Menus

<select>…</select>
Menu control
<option>…</option>
An option within a menu
<optgroup>…</optgroup>
A logical grouping of options within a menu

- Another way to provide a list of choices is to put them in a drop-down or scrolling menu. Menus tend to be more compact than groups of buttons and checkboxes.
- You add both drop-down and scrolling menus to a form with the **select** element. Whether the menu pulls down or scrolls is the result of how you specify its size and whether you allow more than one option to be selected.

Menu and Optional Selection

- Drop-down menus
- Scrolling menus
- Grouping menu options
- Text input datalist select (Option by user input, but Web-site provide options. See text input)

```
<p>What is your favorite 80s band?
    <select name="EightiesFave">
        <option>The Cure</option>
        <option>Cocteau Twins</option>
        <option>Tears for Fears</option>
        <option>Thompson Twins</option>
        <option value="EBTG">Everything But the Girl</option>
        <option>Depeche Mode</option>
        <option>The Smiths</option>
        <option>New Order</option>
    </select>
</p>
```

FIGURE 9-12. Pull-down menus pop open when the user clicks the arrow or bar.

Drop-down menus

```
<p>What 80s bands did you listen to?
    <select name="EightiesBands" size="6" multiple>
        <option>The Cure</option>
        <option>Cocteau Twins</option>
        <option selected>Tears for Fears</option>
        <option selected>Thompson Twins</option>
        <option value="EBTG">Everything But the Girl</option>
        <option>Depeche Mode</option>
        <option>The Smiths</option>
        <option>New Order</option>
    </select>
</p>
```

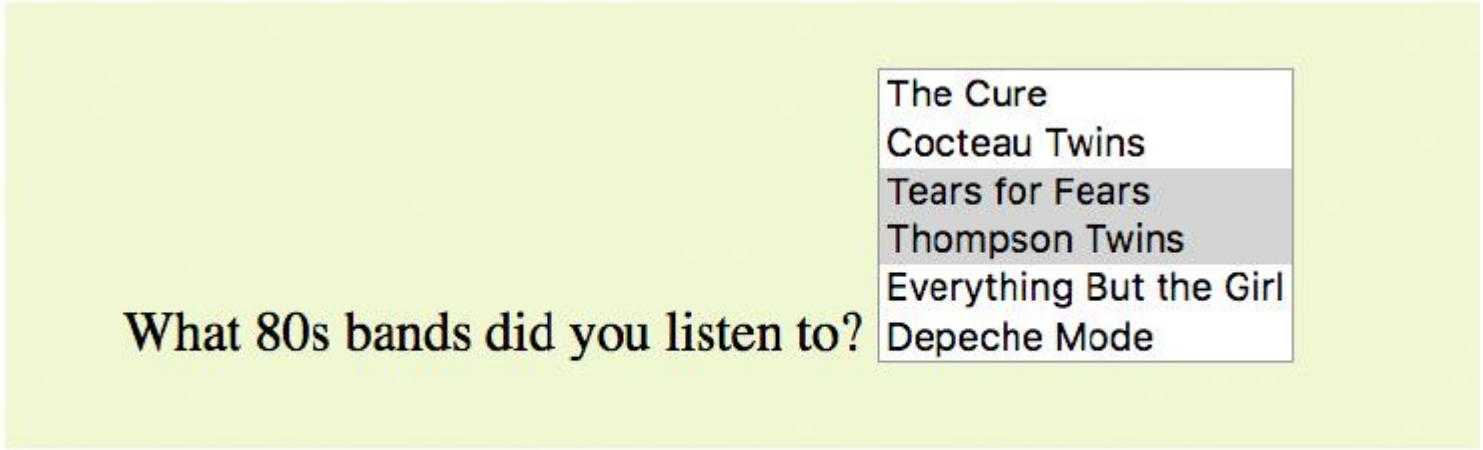


FIGURE 9-13. A scrolling menu with multiple options selected.

Scrolling menus

```
<select name="icecream" size="7" multiple>
    <optgroup label="traditional">
        <option>vanilla</option>
        <option>chocolate</option>
    </optgroup>
    <optgroup label="fancy">
        <option>Super praline</option>
        <option>Nut surprise</option>
        <option>Candy corn</option>
    </optgroup>
</select>
```

FIGURE 9-14. Option groups.

Grouping menu options

Window in a Window (iframe)

<iframe>…</iframe>

A nested browsing window

```
<h1>An Inline Frame</h1>

<iframe src="glossary.html" width="400" height="250" >
  Read the <a href="glossary.html">glossary</a>.
</iframe>
```

FIGURE 10-1. Inline frames (added with the **iframe** element) are like a browser window within the browser that displays external HTML documents and resources.

```
<form action="/client.php" method="POST"
        enctype="multipart/form-data">
    <label>Send a photo to be used as your online icon
    <em>(optional)</em><br>
    <input type="file" name="photo">
    </label>
</form>
```

File input (on Chrome browser)

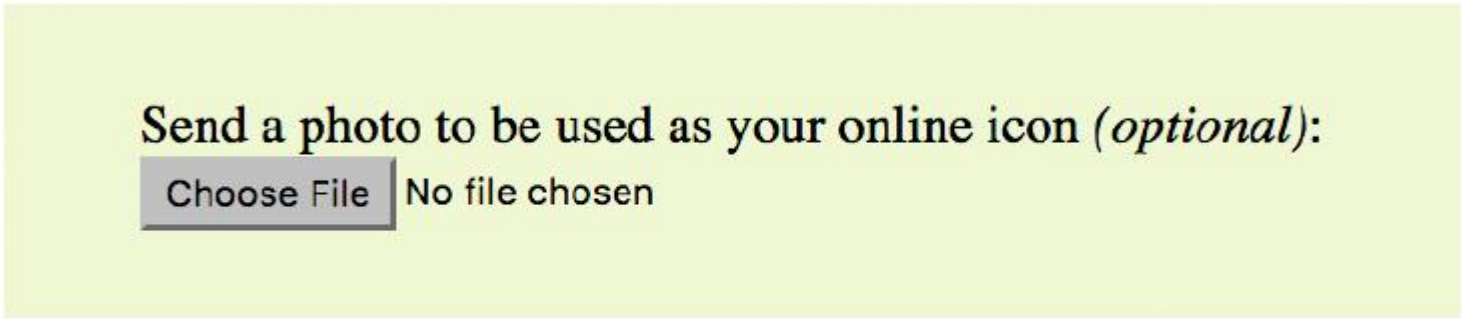


FIGURE 9-15. A file selection form field.

File Selection Control

<input type="file">
File selection field

- When the webpage want to initiate a form processing action which does not require user's input, the hidden control can be used.
- JavaScript script can generate a form format and then mark it as hidden and send to the server.
- Hidden controls are added via the **input** element with the **type** set to **hidden**.
- Its sole purpose is to pass a name/value pair to the server when the form is submitted.
- In this example, a hidden form element is used to provide the location of the appropriate thank-you document to display when the transaction is complete:

```
<input type="hidden" name="success-link"
value="http://www.example.com/
thankyou.html">
```

Hidden Controls

<input type="hidden">
Hidden control field

Other Input Controls

SECTION 10

input type="time"

input type="date"

input type="datetime-local"

input type="month"

input type="week"

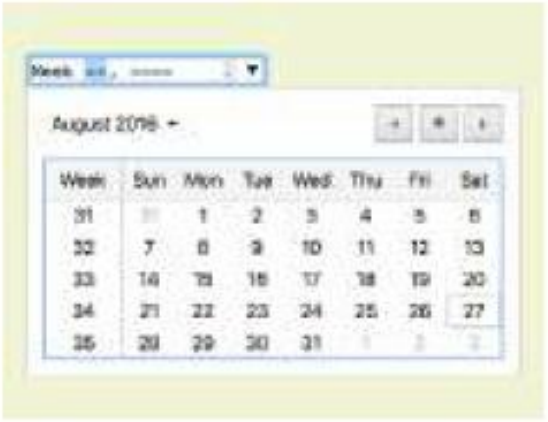


FIGURE 9-16. Date and time picker inputs (shown in Chrome on macOS).

Date and Time Controls

<input type="date">
Date input control
<input type="time">
Time input control
<input type="datetime-local">
Date/time control
<input type="month">
Specifies a month in a year
<input type="week">
Specifies a particular week in a year

The new date- and time-related input types are as follows:

<input type="date" name="name" value="2017-01-14">

Creates a date input control, such as a pop-up calendar, for specifying a date (year, month, day). The initial value must be provided in ISO date format (**YYYY-MM-DD**).

<input type="time" name="name" value="03:13:00">

Creates a time input control for specifying a time (hour, minute, seconds,
fractional sections) with no time zone indicated. The value is provided as **hh:mm:ss**.

<input type="datetime-local" name="name" value="2017-01-14T03:13:00">

Creates a combined date/time input control with no time zone information (**YYYY-MM-DDThh:mm:ss**).

Date and Time Controls

<input type="date">
Date input control
<input type="time">
Time input control
<input type="datetime-local">
Date/time control
<input type="month">
Specifies a month in a year
<input type="week">
Specifies a particular week in a year

<input type="month" name="name" value="2017-01">

Creates a date input control that specifies a particular month in a year (**YYYY-MM**).

<input type="week" name="name" value="2017-W2">

Creates a date input control for specifying a particular week in a year using an ISO week numbering format (**YYYY-W#**).

Date and Time Controls

<input type="date">
Date input control
<input type="time">
Time input control
<input type="datetime-local">
Date/time control
<input type="month">
Specifies a month in a year
<input type="week">
Specifies a particular week in a year

```
<label>Number of guests <input type="number" name="guests" min="1"
max="6"></label>

<label>Satisfaction (0 to 10) <input type="range" name="satisfaction"
min="0" max="10" step="1"></label>
```

input type="number"

input type="range"

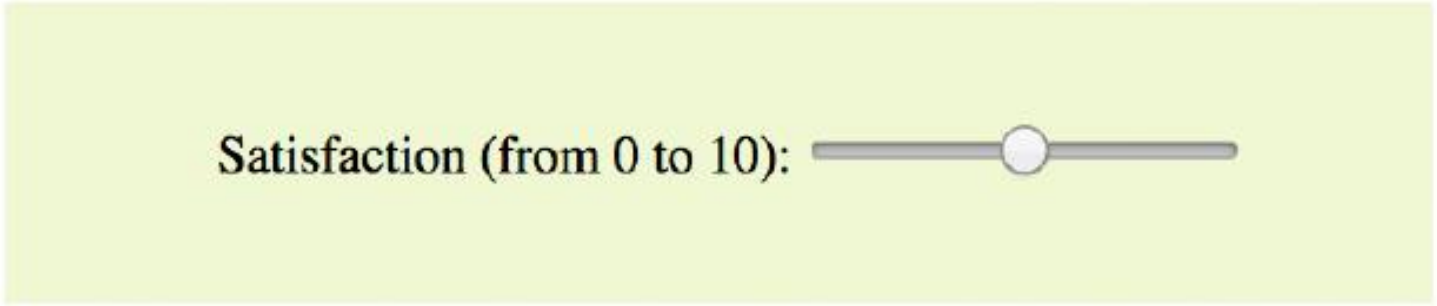


FIGURE 9-17. The **number** and **range** input types (shown in Chrome on macOS).

Numerical Inputs

<input type="number">
Number input
<input type="range">
Slider input

```
<label>Your favorite color: <input type="color" name="favorite">
</label>
```

FIGURE 9-18. The **color** input type (shown in Chrome on macOS).

Color Selector

<input type="color">
Color picker

- The **progress** element gives users feedback on the state of an ongoing process, such as a file download. It may indicate a specific percentage of completion (determinate), like a progress bar, or just indicate a “waiting” state (indeterminate), like a spinner.
- The **progress** element requires scripting to function.
- Percent downloaded:

```
<progress max="100" id="fave">0</progress>
```

progress

<progress>…</progress>
Indicates the state of an ongoing process

- **meter** represents a measurement within a known range of values (also known as a gauge). It has a number of attributes: **min** and **max** indicate the highest and lowest values for the range (they default to 0 and 100); **low** and **high** could be used to trigger warnings at undesirable levels; and **optimum** specifies a preferred value.

```
<meter min="0" max="100" name="volume">60%</meter>
```

meter

<meter>…</meter>
Represents a measurement within a range

- Simply put, the **output** element indicates the result of a calculation by a script or program. This example, taken from the HTML5.2 specification, uses the **output** element and JavaScript to display the sum of numbers entered into inputs a and b.

```
<form onsubmit="return false"
        oninput="o.value = a.valueAsNumber +
        b.valueAsNumber">
    <input name=a type=number step=any>
    + <input name=b type=number step=any> =
    <output name=o for="a b"></output>
</form>
```

output

<output>…</output>
Calculated output value

- This is typically called “**Ogg Theora**,” and the file should have an **.ogv** suffix. All of the codecs and the container in this option are open source and unencumbered by patents or royalty restrictions, but some say the quality is inferior to other options.
- In addition to new browsers, it is supported on some older versions of Chrome, Firefox, and Android that don’t support WebM or MP4, so including it ensures playback for more users.

Ogg container + Theora video codec + Vorbis audio codec.

Other Input Controls

SECTION 11

Form Accessibility Issues

- It is essential to consider how users without the benefit of visual browsers will be able to understand and navigate through your web forms.
- The **label**, **fieldset**, and **legend** form elements improve accessibility by making the semantic connections between the components of a form clear.
- Not only is the resulting markup more semantically rich, but there are also more elements available to act as “hooks” for style sheet rules.
- Everybody wins!

```
<ul>
  <li><label><input type="checkbox" name="genre" value="punk"> Punk
rock</label></li>
  <li><label><input type="checkbox" name="genre" value="indie"> Indie
rock</label></li>
  <li><label><input type="checkbox" name="genre" value="hiphop"> Hip
Hop</label></li>
  <li><label><input type="checkbox" name="genre" value="rockabilly">
Rockabilly</label></li>
</ul>
```

Labels

<label>…</label>

Attaches information to form controls

```
<fieldset>
  <legend>Mailing List Sign-up</legend>
  <ul>
    <li><label>Add me to your mailing list <input type="radio"
    name="list" value="yes" checked></label></li>
    <li><label>No thanks <input type="radio" name="list" value="no">
  </label></li>
  </ul>
</fieldset>

<fieldset>
  <legend>Customer Information</legend>
  <ul>
    <li><label>Full name: <input type="text" name="fullname"></label>
</li>
    <li><label>Email: <input type="text" name="email"></label></li>
    <li><label>State: <input type="text" name="state"></label></li>
  </ul>
</fieldset>
```

fieldset and legend

<fieldset>…</fieldset

Groups related controls and labels

<legend>…</legend>

Assigns a caption to a fieldset

FIGURE 9-19. The default rendering of fieldsets and legends.

fieldset and legend

<fieldset>…</fieldset

Groups related controls and labels

<legend>…</legend>

Assigns a caption to a fieldset

Form Layout and Design

- Usable Forms
- Avoid unnecessary questions
- Consider impact of label placement.
- Choose input types carefully.
- Group related inputs
- Clarify primary and secondary actions.

Embedded Media

SECTION 12

Embedded Media

- A window for viewing an external HTML source (**iframe**)
- Multipurpose embedding elements (**object** and **embed**)
- Video and audio players (**video** and **audio**)
- A scriptable drawing area that can be used for animations or game-like interactivity (**canvas**)

Window in a Window (iframe)

<iframe>…</iframe>

A nested browsing window

```
<h1>An Inline Frame</h1>

<iframe src="glossary.html" width="400" height="250" >
  Read the <a href="glossary.html">glossary</a>.
</iframe>
```

FIGURE 10-1. Inline frames (added with the **iframe** element) are like a browser window within the browser that displays external HTML documents and resources.

Window in a Window (iframe)

<iframe>…</iframe>
A nested browsing window

FIGURE 10-2. The edges of an `iframe` are usually not detectable, as shown in this embedded Google Map.

Multipurpose Embedder (Object)

<object>…</object>
Represents external resource
<param>
Parameters of an object

- To embed those media resources on the page, we used the **object** and **embed** elements. They have slightly different uses.
- The **object** element is a multipurpose object placer. It can be used to place an image, create a nested browsing context (like an iframe), or embed a resource that must be handled by a plugin.
- The **embed** element was for use with plug-ins only.

Multipurpose Embedder (Object)

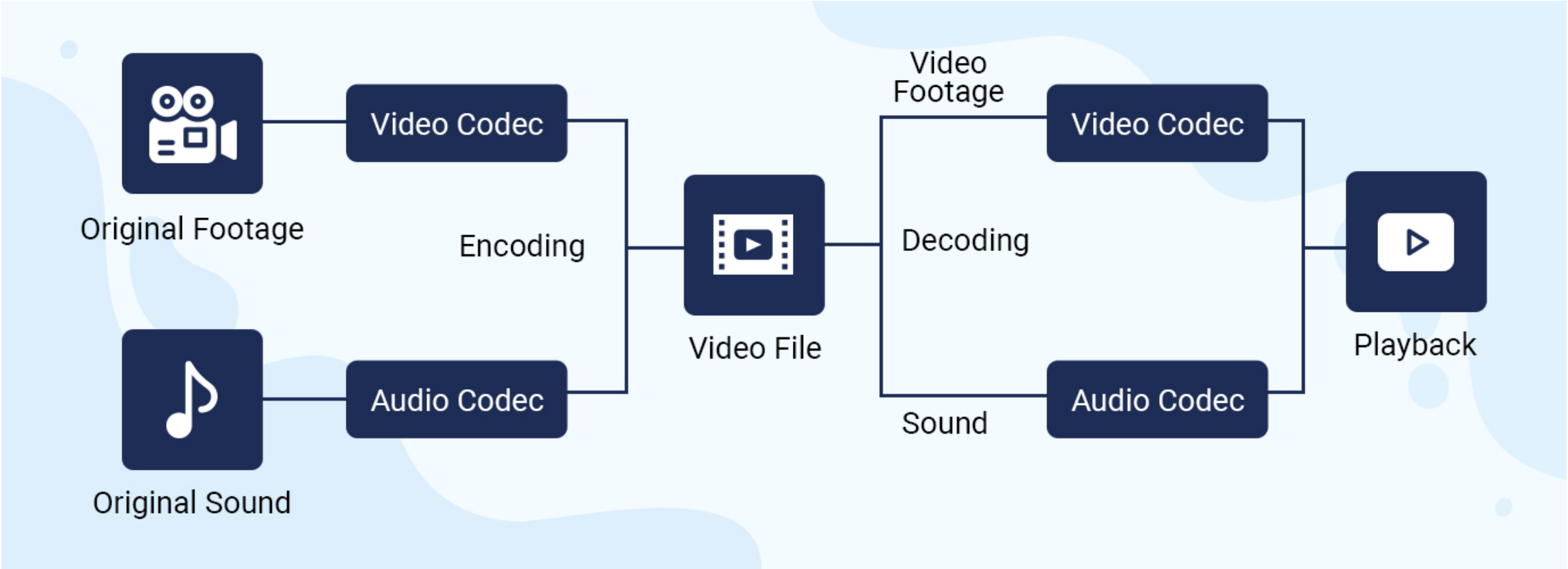
<object>…</object>
Represents external resource
<param>
Parameters of an object

Here is a simple **object** element that places an SVG image on the page and provides a PNG fallback:

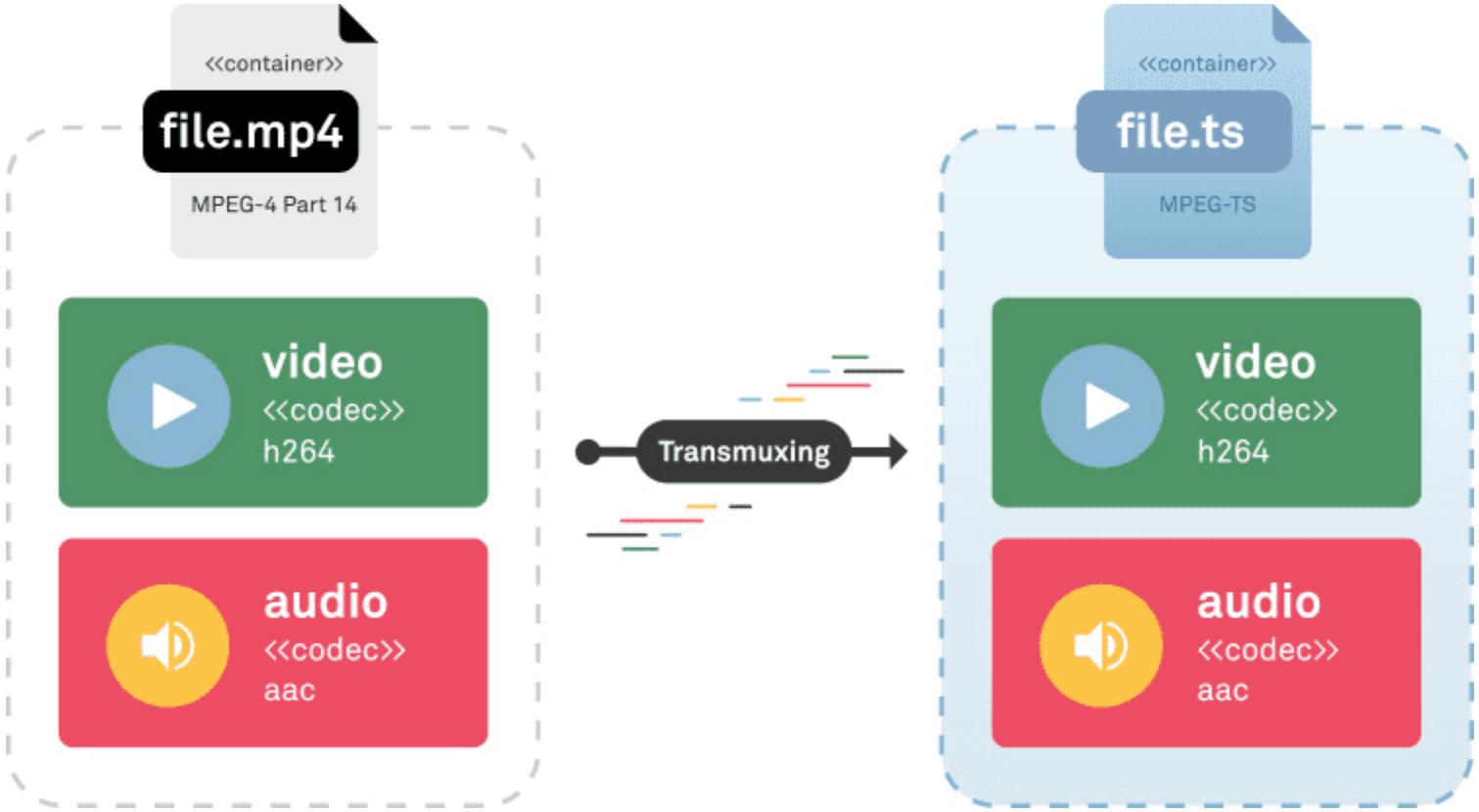
```
<object data="picture.svg"
        type="image/svg+xml">
  <img src="picture.png" alt="">
</object>
```

Video

SECTION 13


Original Footage
Video Codec
Encoding
Original Sound
Audio Codec
Video File
Video Footage
Video Codec
Decoding
Sound
Audio Codec
Playback

How Media Formats Work



- This combination is generally referred to as “**MPEG-4**,” and it takes the **.mp4** or **.m4v** file suffix. **H.264** is a high-quality and flexible video codec, but it is patented and must be licensed for a fee.
- All current browsers that support HTML5 video can play MPEG-4 files with the H.264 codec. The newer **H.265** codec (also known as **HEVC**, High Efficiency Video Coding) is in development and reduces the bitrate by half, but is not well supported as of this writing.

MPEG-4 container + H.264 video codec + AAC audio codec.

- “**WebM**” is a container format that has the advantage of being open source and royaltyfree.
- It uses the **.webm** file extension. It was originally designed to work with VP8 and Vorbis codecs.

WebM container + VP8 video codec + Vorbis audio codec.

- The **VP9** video codec from the **WebM** project offers the same video quality as VP8 and H.264 at half the bitrate. Because it is newer, it is not as well supported, but it is a great option for browsers that can play it.

WebM container +
VP9 video codec +
Opus audio codec.

- This is typically called “**Ogg Theora**,” and the file should have an **.ogv** suffix. All of the codecs and the container in this option are open source and unencumbered by patents or royalty restrictions, but some say the quality is inferior to other options.
- In addition to new browsers, it is supported on some older versions of Chrome, Firefox, and Android that don’t support WebM or MP4, so including it ensures playback for more users.

Ogg container + Theora video codec + Vorbis audio codec.

Video Codecs	H.264/AVC	H.265/HEVC	VP9	AV1	VVC
Hardware support	Yes	Yes	Yes	In development	In development
Browser support					In development
Mobile Platforms					In development
Coding Performance	Good	High	High	Very High	Very High
Royalties	Non-free	Non-free (unclear)	Free	Free	Non-Free

HTML Video - Browser Support

Browser	MP4	WebM	Ogg
Internet Explorer	Yes	No	No
Chrome	Yes	Yes	Yes
Firefox	Yes	Yes	Yes
Safari	Yes	No	No
Opera	Yes	Yes	Yes

Audio

SECTION 14

Meet the audio formats

MP3. The MP3 (short for MPEG-1 Audio Layer 3) format is a codec and container in one, with the file extension.*mp3*. It has become ubiquitous as a music download format.

WAV. The WAV format (.*wav*) is also a codec and container in one. This format is uncompressed so it is only good for very short clips, like sound effects.

Ogg container + Vorbis audio codec. This is usually referred to as “Ogg Vorbis” and is served with the *.ogg* or *.oga* file extension.

MPEG 4 container + AAC audio codec. “MPEG4 audio” (.*m4a*) is less common than MP3.

WebM container + Vorbis audio codec. The WebM (.*webm*) format can also contain audio only.

WebM container + Opus audio codec. Opus is a newer, more efficient audio codec that can be used with WebM.

TABLE 10-1. *Video support in desktop and mobile browsers (as of 2017)*

Format	Type	IE	MS Edge	Chrome	Firefox	Safari	Opera	Android	iOS Safari
MP4 (H.264)	`video/mp4 mp4 m4v`	9.0+	12+	4+	Yes*	3.2+	25+	4.4+	3.2+
WebM (VP8)	`video/webm webm webmv`	–	–	6+	4.0+	–	15+	2.3+	–
WebM (VP9)	`video/webm webm webmv`	–	14+	29+	28+	–	16+	4.4+	–
Ogg Theora	`video/ogg ogv`	–	–	3.0+	3.5+	–	13+	2.3+	–

* *Firefox version varies by operating system.*

TABLE 10-2. *Audio support in current browsers (as of 2017)*

Format	Type	IE	MS Edge	Chrome	Firefox	Opera	Safari	iOS Safari	Android
MP3	`audio/mpeg mp3`	9.0+	12+	3.0+	22+	15+	4+	4.1	2.3+
WAV	`audio/wav or audio/wave`	–	12+	8.0+	3.5+	11.5+	4+	3.2+	2.3+
Ogg Vorbis	`audio/ogg ogg oga`	–	–	4.0+	3.5+	11.5+	–	–	2.3+
MPEG-4/AAC	`audio/mp4 m4a`	11.0+	12+	12.0+	–	15+	4+	4.1+	3.0+
WebM/Vorbis	`audio/webm webm`	–	–	6.0+	4.0+	11.5+	–	–	2.3.3+
WebM/Opus	`audio/webm webm`	–	14+	33+	15+	20+	–	–	–

webm converter

both video and audio

Embedded Video and Audio

SECTION 15

FIGURE 10-3. An embedded movie using the **video** element (shown in Chrome on a Mac).

Adding a Video to a Page

<video>…</video>

Adds a video player to the page

Video conversion

- **Handbrake** (*handbrake.fr*) is a popular open source tool for converting to MPEG4 with H.264, H.265, VP8, and Theora. It is available for Windows, macOS, and Linux.
- **Firefogg** (*firefogg.org*) is an extension to Firefox for converting video to the WebM (VP8 and VP9) and Ogg Theora formats. Simply install the Firefogg extension to Firefox (crossplatform); then visit the Firefogg site and convert video by using its online interface.
- **FFmpeg** (*ffmpeg.org*)is an open source, command-line tool for converting just about any video format. If you are not comfortable with the command line, there are a number of software packages (some for pay, some free) that offer a user interface to FFmpeg to make it more user-friendly.
- **Freemake** (*freemake.com*) is a free video and audio conversion tool for Windows that supports over 500 media formats.

Audio conversion

- Audio Converter (*online-audio-converter.com*) is one of the free audio and video tools from *123Apps.com* that converts files to MP3, WAV, OGG, and more.
- Media.io (*media.io*) is a free web service that converts audio to MP3, WAV, and OGG.
- MediaHuman Audio Converter (*www.mediahuman.com/audio-converter/*) is free for Mac and Windows and can convert to all of the audio formats listed in this chapter and more. It has an easy drag-and-drop interface, but is pretty much no-frills.
- Max (*sbooth.org/Max/*) is an open source audio converter (Mac only).
- Audacity (*www.audacityteam.org*) is free, open source, cross-platform audio software for multitrack recording and editing. It can import and export files in many of the formats listed in this chapter.

Video element

- Here is a simple video element that embeds a movie and player on a web page:

```
<video src="highlight_reel.mp4" width="640" height="480"
        poster="highlight_still.jpg" controls autoplay>
       Your browser does not support HTML5 video. Get the
       <a href="highlight_reel.mp4">MP4 video</a>
</video>
```

- Browsers that do not support **video** display whatever content is provided within the **video** element. In this example, it provides a link to the movie that your visitor could download and play in another player.

Video element

width="*pixel measurement*"

height="*pixel measurement*"

Specifies the size of the box the embedded media player takes up on the screen. Generally, it is best to set the dimensions to exactly match the pixel dimensions of the movie. The movie will resize to match the dimensions set here.

poster="*url of image*"

Provides the location of an image that is shown in place of the video before it plays.

Video element

controls

Adding the **controls** attribute prompts the browser to display its builtin media controls, generally a play/pause button, a “seeker” that lets you move to a position within the video, and volume controls. It is possible to create your own custom player interface using CSS and JavaScript if you want more consistency across browsers.

autoplay

Makes the video start playing automatically after it has downloaded enough of the media file to play through without stopping. In general, use of **autoplay** should be avoided in favor of letting the user decide when the video should start. **autoplay** does not work on iOS Safari and some other mobile browsers in order to protect users from unnecessary data downloads.

Providing video format options

```
<video id="video" controls poster="img/poster.jpg">
    <source src="clip.webm" type="video/webm">
    <source src="clip.mp4" type="video/mp4">
    <source src="clip.ogg" type="video/ogg">
    <a href="clip.mp4">Download the MP4 of the clip.</a>
</video>
```

Adding Audio to a Page

```
<p>Play "Percussion Gun" by White Rabbits</p>
<audio id="whiterabbits" controls preload="auto">
    <source src="percussiongun.mp3" type="audio/mp3">
    <source src="percussiongun.ogg" type="audio/ogg">
    <source src="percussiongun.webm"
            type="audio/webm">
    <p>Download "Percussion Gun":</p>
    <ul>
        <li><a href="percussiongun.mp3">MP3</a></li>
        <li><a href="percussiongun.ogg">Ogg
        Vorbis</a></li>
    </ul>
</audio>
```

Adding Audio to a Page

FIGURE 10-4. Audio player as rendered in Firefox.

Text Tracks (Caption)

SECTION 16

Adding Text Tracks

<track>…</track> Adds synchronized text to embedded media

The **track** element provides a way to add text that is synchronized with the timeline of a video or audio track. Some uses include the following:

- **Subtitles** in alternative languages
- **Captions** for the hearing impaired
- **Descriptions** of what is happening in a video for the sight impaired
- **Chapter titles** to allow for navigation through the media
- **Metadata** that is not displayed but can be used by scripts

FIGURE 10-5. A video with captions.

Use the **track** element inside the **video** or **audio** element you wish to annotate. The **track** element must appear after all the **source** elements, if any, and
may include these attributes:

src

Points to the text file.

kind

Specifies the type of text annotation you are providing (**subtitles**, **captions**, **descriptions**, **chapters**, or **metadata**). If **kind** is set to **subtitle**, you must also specify the language (**srclang** attribute) by using a standardized IANA two-letter language tag (see Note).

label

Provides a name for the track that can be used in the interface for selecting a particular track.

default

Marks a particular track as the default and it may be used on only one track within a media element.

Adding Text Tracks

<track>…</track>

Adds synchronized text to embedded media

```
<video width="640" height="320" controls>
      <source src="japanese_movie.mp4"
      type="video/mp4">
      <source src="japanese_movie.webm“
                  type="video/webm">
      <track src="english_subtitles.vtt“
                  kind="subtitles“ srclang="en“
                  label="English subtitles“ default>
      <track src="french.vtt“ kind="subtitles“
                  srclang="fr“ label="Sous-titres en français">
</video>
```

Adding Text Tracks

<track>…</track>

Adds synchronized text to embedded media

WEBVTT

00:00:01.345 --> 00:00:03.456
Welcome to Artifact [applause]

00:00:06.289 --> 00:00:09.066
There is a lot of new mobile technology to discuss.

00:00:06.289 --> 00:00:13.049
We're glad you could all join us at the Alamo Drafthouse.

WebVTT

Canvas

SECTION 17

mahjong.frvr.com/

muro.deviantart.com/

www.effectgames.com/demos/canvascycle/

alteredqualia.com/canvasmol/

FIGURE 10-6. A few examples of the **canvas** element used for games, animations, and applications.

```
<canvas width="600" height="400"
        id="my_first_canvas">
Your browser does not support HTML5 canvas. Try
using Chrome, Firefox, Safari or MS Edge.
</canvas>
```

The canvas Element

<canvas>…</canvas>

Adds a 2-D dynamic drawing area

Drawing with JavaScript

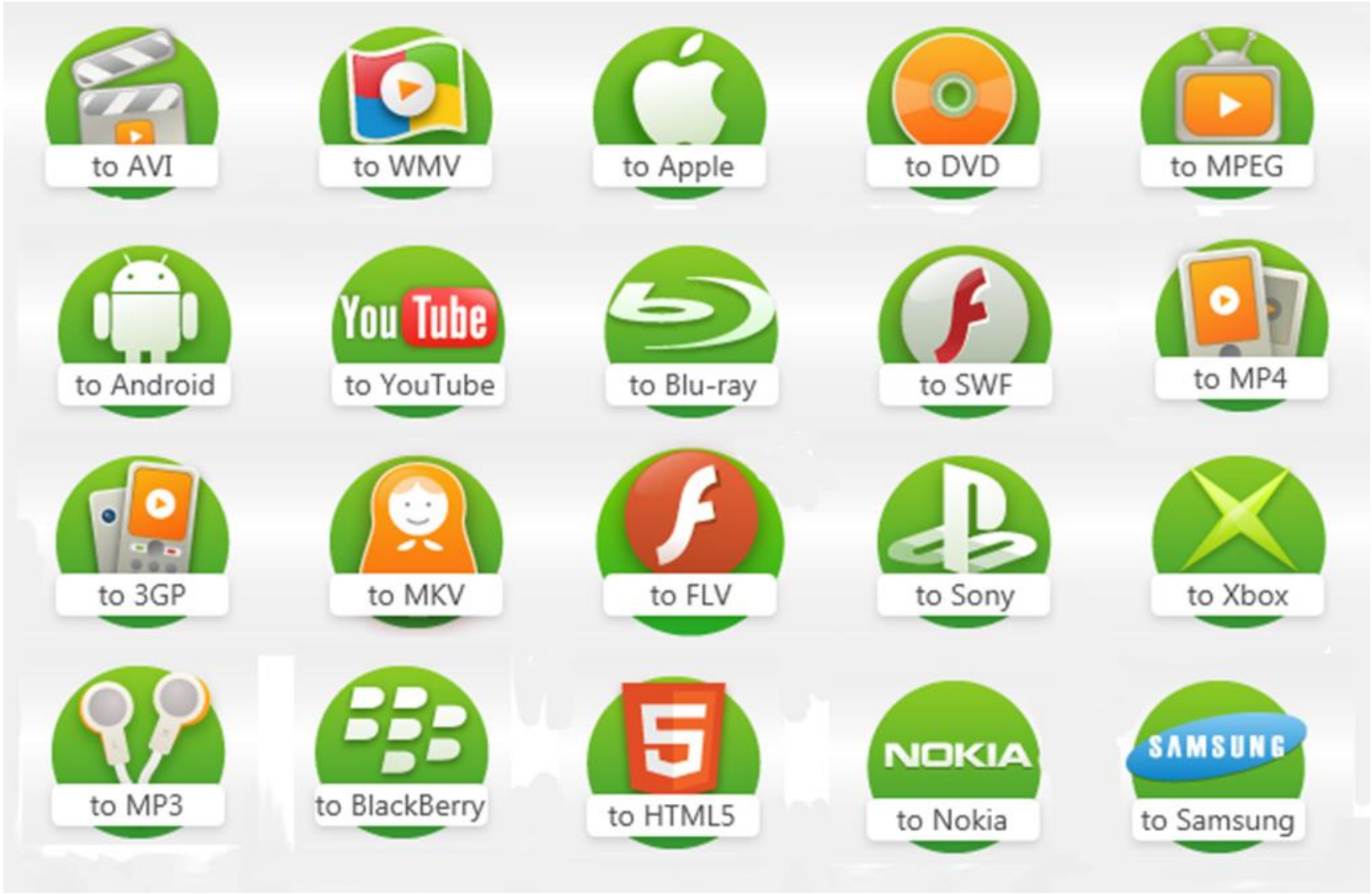
- The Canvas API includes functions for creating shapes, such as **strokeRect()** for drawing a rectangular outline and **beginPath()** for starting a line drawing.
- Some functions move things around, such as **rotate()** and **scale()**. It also includes attributes for applying styles (for example, **lineWidth**, **font**, **stroke-Style**, and **fillStyle**).
- Sanders Kleinfeld created the following code example for his book *HTML5 for Publishers* (O'Reilly). He was kind enough to allow me to use it in this book. FIGURE 10-7 shows the simple smiley face we'll be creating with the Canvas API.

FIGURE 10-7. The finished product of our "Hello Canvas" example. See the original at *examples.oreilly.com/0636920022473/my_first_canvas/my_first_canvas.html*.

API (Application Programming Interface)

A set of documented set of commands, data names, and so on, that let one software application communicate with another.

- Media Player API
- Session History API
- Offline Web Application API
- Editing API
- Drag and Drop API
- Canvas API
- Web Storage API
- Geolocation API
- Web Workers API
- Web Sockets API: sockets (connection between client and server)


to AVI
to WMV
to Apple
to DVD
to MPEG
to Android
YouTube
to YouTube
to Blu-ray
to SWF
to MP4
to 3GP
to MKV
to FLV
to Sony
to Xbox
to MP3
to BlackBerry
to HTML5
NOKIA
to Nokia
SAMSUNG
to Samsung