

HTML

André Restivo

Index

Introduction

Resources

Content

Sections

Lists

Tables

Forms

Character Entities

Media

Metadata

Validation

Introduction

and some History

What is it?

- **Hyper Text Markup Language.**
- **Markup** language used to create **web pages**.
- Written using HTML **elements**.
- **Not** for design or presentation.
- All about **structure** and **semantics**.

History

- 1989-92: **HTML 1.0**, Tim Berners-Lee original **proposal**
- 1993: **HTML+**, Dave Raggett's **competing standard**
- 1994: **HTML 2.0**, tables, file upload, ... (IETF)
- 1995: Non-standard Netscape features
- 1996: Competing Netscape and Internet Explorer features
- 1996: **HTML 3.2**, W3C standard, the Browser Wars end
- 1997: **HTML 4.0**, stylesheets are introduced
- 1999: **HTML 4.01**, we have a winner!
- 2000: **XHTML 1.0**, an XML version of HTML 4.01
- 2001: **XHTML 1.1**, modularization
- 2008: **HTML 5**, reduces the need for proprietary plug-in based apps
- 2019: **W3C** and WHATWG reach an **agreement** about future HTML developments.

Learn more:

<http://en.wikipedia.org/wiki/HTML#History>

Browser Wars

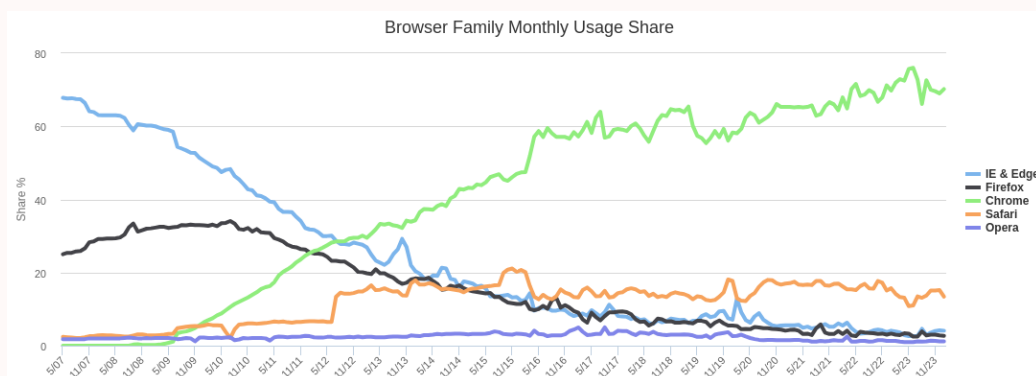
The First War (1995 – 2001):

- **Netscape** and **Internet Explorer** battle for WWW dominance.
- Web standards were still **not well established**.
- New **proprietary features** are introduced into HTML as browsers compete for market share.
- Developers were **forced** to have two versions of their websites.

Aftermath:

- Internet Explorer won the war and decided to **stale** any **new developments**.
- From the ashes of Netscape, **Firefox** starts to gain market share.
- Eventually, browsers decided to work together, and we now have a **much better** web landscape!

Browser Share (2007 – now)



Source: <http://www.w3counter.com/trends>

HTML Structure

- An **HTML file** has a **tree-like** structure where each node is an **HTML Element**.
- Elements can contain other elements and/or **text**.
- They are defined using **tags** and can have **attributes**.
- Browsers display each tag using a **predefined** style that can be changed using **CSS**.

HTML tells the browser how the document is **structured**.

CSS tells the browser how it should be **displayed**!

Tags

- Tags start with a < and end with a > and always contain a name.
- They are case insensitive, but **lowercase** is **recommended**.

```
<html>
```

- Most tags come in pairs: an opening tag and a closing tag.
- Closing tags have a / after the <.

```
<html> ... </html>
```

Tag Content

The content of a tag is everything between the opening and closing tags:

```
<p>Some content</p>
```

It can be text, but it can also be other tags:

```
<article>
  <p>Some content</p>
</article>
```

Some tags never have content and do not need to be closed:

```
<br>
```

Attributes

Tags can have attributes. Some are optional, and some are mandatory:

```

```

Quotes around attribute values are **not mandatory** in HTML 5, but they are **recommended**.

To set a **boolean attribute** to true, we can either **omit** its value or use the **name of the attribute** as its value.

```
<input type="checkbox" checked disabled="disabled
```

This checkbox is both *checked* and *disabled*.

Some Global Attributes

Some attributes (**global**) can be used on all HTML elements. These are some of them:

- **accesskey**: A guide for browsers to create a keyboard shortcut to the element.
- **autofocus**: Automatically focus the element, allowing the user to start typing right away.
- **hidden**: Indicates that the element is not relevant to the current state of the page (should be set using CSS most of the time).
- **style**: CSS rules to apply to the element. Only use in some very particular cases.
- **lang**: The primary language for the element's content; typically used in **html** tags if the document has only one language.
- **id**: An element *identification* so that CSS and JavaScript can identify the element.
- **class**: An element *class* so that CSS and JavaScript can identify the class of elements.

There are many more!

Id and Class

The **id** and **class** attributes are used to easily identify a tag for manipulation (using javascript) or styling (using CSS).

An HTML document **cannot** have two elements with the same **id**:

```

```

An HTML element can have more than one **class** (separated by whitespace).

```
<p class="first important">Some text</p>
```

You can think of the **id** as the name of the element and the **class** as its type.

The Most Basic Document

All HTML documents **must** have these elements:

- A document type declaration (DOCTYPE).
- A `<html>` root with two children: `<head>` and `<body>`.
- A non-empty `<title>` element inside the `<head>`.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
  </body>
</html>
```

- The `<head>` contains **metadata** about the document.
- The `<body>` contains the actual **structure** and **content**.

Semantics

During this presentation, we will talk about a lot about semantics:

Syntax describes the rules by which words can be combined into sentences, while semantics describes what they mean — The Cambridge Dictionary

But **why** is semantics **important** when describing a language to markup text?

Most HTML tags could be easily **replaced** by another (together with some CSS) and have the **same final result**.

At least for the end user.

End users are **not the only** readers of HTML; in fact, they don't read HTML at all:

- But bots do when they index websites.
- And developers when they fix other developers' mistakes.
- And specialized browsers for people with disabilities (*cf. accessibility*).

Whitespace

Except inside a few elements (e.g., **<pre>** and **<textarea>**), whitespace is collapsed into a single space.

So this *haiku*:

"The Old Pond" by Matsuo Bashō

An old silent pond
A frog jumps into the pond—
Splash! Silence again.

Renders as:

The Old Pond by Matsuo Bashō An old silent pond A
frog jumps into the pond— Splash! Silence again.

Resources

- References:
 - WHATWG Living Standard
 - Mozilla Developer Network (MDN) Reference
- Books:
 - Dive into HTML 5
- Tutorials:
 - <https://webplatform.github.io/docs/html/tutorials/>
 - <http://www.htmldog.com/guides/html/>

Content

Paragraphs and Line Breaks

- A paragraph is represented by the `<p>` tag.
- If we want to change lines (but not paragraphs, e.g., in a poem) we can use the `
` tag.

```
<p>"The Old Pond" by Matsuo Bashō</p>
```

```
<p>An old silent pond<br>  
A frog jumps into the pond—<br>  
Splash! Silence again.</p>
```

"The Old Pond" by Matsuo Bashō

An old silent pond
A frog jumps into the pond—
Splash! Silence again.

Text Semantics

Some elements that can be used for **text-level semantics**:

| | |
|---|---|
| <code>emphasized</code> | <code><!-- emphasized</code> |
| <code><small>small</small></code> | <code><!-- smaller</code> |
| <code>strong</code> | <code><!-- important</code> |
| <code><sub>subscripted</sub></code> | <code><!-- subscripted</code> |
| <code><sup>superscripted</sup></code> | <code><!-- superscripted</code> |
| <code><ins>inserted</ins></code> | <code><!-- inserted</code> |
| <code>deleted</code> | <code><!-- deleted</code> |
| <code><mark>highlighted</mark></code> | <code><!-- marked/highlighted</code> |

*emphasized*small**strong**_{subscripted}
superscriptedinserted~~deleted~~highlighted

Note: Although `` is represented by browsers as **bold**, that's not the semantic meaning of the element. The same can be said of all the other elements in this list.

Preformatted Text

Preformatted text (`<pre>`) can be useful to mark *ascii art*, or used together with other elements to mark, for example, computer code:

```
<pre>...</pre>    <!-- preformatted text    -->
<code>...</code>  <!-- computer code        -->
<kbd>...</kbd>    <!-- keyboard input        -->
<samp>...</samp>  <!-- sample computer code  -->
<var>...</var>    <!-- a variable            -->
```

```
<pre><code>
for (i = 0; i < 10; i++)
    print(i)
</code></pre>
```

```
for (i = 0; i < 10; i++)
    print(i)
```

More Semantics

Some **text-level semantics** elements are not even rendered differently by browsers, but they still have importance as they convey meaning to the text.

HTML is not only for humans.

| | |
|--|---|
| <code><abbr></abbr></code> | <code><!-- an abbreviation or acronym --></code> |
| <code><address></address></code> | <code><!-- contact information for someone --></code> |
| <code><time></time></code> | <code><!-- a time of the day --></code> |
| <code><progress></progress></code> | <code><!-- a progress of a task --></code> |
| <code><bdo></bdo></code> | <code><!-- the text direction --></code> |
| <code><blockquote></blockquote></code> | <code><!-- quoted from another source --></code> |
| <code><q></q></code> | <code><!-- an inline (short) quotation --></code> |
| <code><cite></cite></code> | <code><!-- the title of a work --></code> |
| <code><dfn></dfn></code> | <code><!-- a definition --></code> |

Wikipedia says `<abbr title="File Transfer Protocol">FTP</abbr>` is `<q cite="https://en.wikipedia.org/wiki/File_Transfer_Protocol">a standard communication protocol used for the transfer of computer files from a server to a client on a computer network</q>`.

Wikipedia says FTP is "a standard communication protocol used for the transfer of computer files from a server to a client on a computer network".

Span

The `` is an incredibly **useful** element for marking text, that means absolutely **nothing at all** (at least semantically).

But, together with the `id` and `class` attributes, we can convey this element *whatever meaning* we desire. And, with **CSS** and **JavaScript**, we can do whatever we want with it.

However, the `` element should only be used if no other, more *semantically correct*, element exists.

One of the boldest colors in the spectrum, `red` stands out in any work of art, hence its use to signal danger or warning.

Anchor

The `<a>` tag creates anchor elements that represent hyperlinks to other documents:

- The **href** attribute, if present, represents the URL of the other document.
- URLs can be **relative** (to the current document) or **absolute**.

```
<a href="anotherpage.html">Another Page</a>  
<a href="somewhere/deeper.html">Deeper</a>  
<a href=" ../start.html">Back</a>  
<a href="http://www.google.com">Search</a>
```

We can create an anchor to an element with a specific **id** within a page:

```
<a href="anotherpage.html#introduction">Another p
```


Images

- To represent an image we use the `` tag.
- The **src** attribute contains the address of the image.
A relative or absolute URL.
- The **alt** attribute is mandatory and represents an alternative image description for browsers incapable of showing images (cf. **accessibility**).

Setting this attribute to the empty string indicates that this image is not a key part of the content; non-visual browsers may omit it from rendering.

- The **width** and **height** indicate the width and height of the image in pixels.
- The main idea is for the browser to **allocate** space for the image before downloading it.
- Refrain from overusing this to resize images.

```
` represents **self-contained content**, potentially with an **optional caption**, which is specified using the `<figcaption>` element. The figure, its caption, and its contents are referenced as a *single unit*:

```
<figure>

 <figcaption>Fig 1: A dog playing in the garden<
</figure>
```

Note: It can be used with **other content** besides images.

# Sections

# Headings

- There are six levels of document headings, from `<h1>` to `<h6>`.
- A heading element briefly **describes** the topic of the **section** it introduces.

```
<h1>Title</h1> <!-- only one per document -->
<h2>Subtitle</h2>
<h3>Section</h3>
<h4>Sub-section</h4>
<h5>Each one less important...</h5>
<h6>...than the other</h6>
```

# Sectioning Content

- The `<article>`, `<section>`, `<nav>`, and `<aside>` elements are sectioning elements.
- Sectioning elements are those that are supposed to have **headings**.
- Each one of these has a different semantic meaning:
  - **article**: Represents a complete, or self-contained, element; one that can be independently distributable or reusable, for example, a blog post, a news article, or a comment.
  - **section**: A thematic grouping of content, generally with a heading.
  - **nav**: A section with navigation links.
  - **aside**: For content that is considered separate from the page's main content.

Sectioning content defines the scope of **headings**, **headers**, and **footers**.

# Header and Footer

- Sections usually have some **introductory** and **closing** content in the form of a `<header>` and a `<footer>`.
- Headers normally contain headings (`<h1>` – `<h6>`), but they can contain anything else.
- The **first** heading of a section represents the heading for that section.

```
<section id="posts">
 <h1>Posts</h1>
 <article>
 <header>
 <h2>Title of the Post</h2>
 <h3>And the subtitle</h3>
 </header>
 <p>The post content</p>
 <p>More content</p>
 <footer><p>Author of the post</p></footer>
 </article>
</section>
```

# Main

- The `<main>` element represents the dominant content of the document.
- A document must not have more than one main element (unless some are hidden).

```
<html>
 <head>...</head>
 <body>
 <header>
 <h1>Page Title</h1>
 </header>
 <nav>...navigation links...</nav>
 <main>
 <section id="posts">...</section>
 ...
 </main>
 </body>
</html>
```

# Div

The `<div>` is an incredibly **useful** element for grouping content, that means absolutely **nothing at all** (at least semantically).

But, together with the `id` and `class` attributes, we can convey this element *whatever meaning* we desire. And, with **CSS** and **JavaScript**, we can do whatever we want with it.

However, the `<div>` element should only be used if no other, more *semantically correct*, element exists.

```
<article class="post">
 <p>...</p>
 <div class="emphasis">
 <p>...</p>
 <p>...</p>
 </div>
 <p>...</p>
</article>
```



# Lists

# Ordered Lists

Ordered lists (`<ol>`) are lists of items (`<li>`) that have been **intentionally** ordered; so that if their order changes, it would change the meaning of the document.

```

 An item
 Another item
 And another one

```

1. An item
2. Another item
3. And another one

# Ordered List Attributes

The attributes **type** (1, a, A, i, I), **reversed**, and **start** allow us to change the default way the list is presented. The **value** attribute on a list item allows changing the value of that item.

```
<ol type="I" start="4" reversed>
 An item
 Another item
 <li value="10">And another one
 And another just for good measure

```

- IV. An item
- III. Another item
- X. And another one
- IX. And another just for good measure

These can also be set in CSS. You should **only** use them in HTML if they convey any meaning (e.g., they are referred to in the text by their values).

# Unordered Lists

Unordered lists (`<ul>`) are lists of items (`<li>`) where the order of the items is **not important**.

```

 An item
 Another item
 And another one

```

- An item
- Another item
- And another one

# Nested Lists

Lists can be nested inside other lists:

```

 A list:

 Something
 Something else

 Another item
 And another one

```

- A list:
  1. Something
  2. Something else
- Another item
- And another one

# Definition or Association Lists

- A *definition list* (`<dl>`) contains terms (`<dt>`) and definitions (`<dd>`).  
Definitions can also be called descriptions, or values.
- A term can have several descriptions, and a description can describe several terms.

```
<dl>
 <dt>A term</dt>
 <dd>And its definition</dd>
 <dt>This one</dt>
 <dd>Has a different definition</dd>
 <dd>And an alternative definition</dd>
</dl>
```

A term  
    And its definition  
This one  
    Has a different definition  
    And an alternative definition

# Tables

# Table

- Tables (`<table>`) represent **tabular data** (e.g., student grades) and should **not be used** for any kind of design layout.
- In its most simple form, tables are composed of rows (`<tr>`) of data cells (`<td>`):

```
<table>
 <tr><td>A</td><td>B</td><td>C</td></tr>
 <tr><td>D</td><td>E</td><td>F</td></tr>
</table>
```

|   |   |   |
|---|---|---|
| A | B | C |
| D | E | F |

Note: This table is styled using CSS, and it's not the default table design.



# Caption

A table can have an optional **caption**.

```
<table>
 <caption>Table 1: A table with letters</caption>
 <tr><td>A</td><td>B</td><td>C</td></tr>
 <tr><td>D</td><td>E</td><td>F</td></tr>
</table>
```

Table 1: A table with letters

|   |   |   |
|---|---|---|
| A | B | C |
| D | E | F |

# Headers

Some data cells can be headers (`<th>`):

- **Not** for making text bold (e.g., pointing out an important value).
- Headers can have an optional **scope** attribute that specifies which cells it applies to (*row*, *col*, *rowgroup*, and *colgroup*).

```
<table>
 <tr>
 <th scope="col">A</th><th scope="col">B</th><th scope="col">C</th>
 </tr>
 <tr>
 <td>D</td><td>E</td><td>F</td>
 </tr>
</table>
```

| A | B | C |
|---|---|---|
| D | E | F |

# Cell Merging

We can merge cells horizontally or vertically.

```
<table>
 <tr>
 <td>A</td><td colspan="2">B</td>
 </tr>
 <tr>
 <td rowspan="2">C</td><td>D</td><td>E</td>
 </tr>
 <tr>
 <td colspan="2">F</td>
 </tr>
 <tr>
 <td colspan="3">G</td>
 </tr>
</table>
```

|   |   |   |
|---|---|---|
| A | B |   |
| C | D | E |
|   | F |   |
| G |   |   |

# Sections

We can divide tables into three logical sections: **thead**, **tfoot**, and **tbody**:

- The order is not important.
- It allows, for example, a scrollable body with fixed header and footer.

```
<table>
 <thead>
 <tr><th>A</th><th>B</th><th>C</th></tr>
 </thead>
 <tfoot>
 <tr><td>100</td><td>200</td><td>300</td></tr>
 </tfoot>
 <tbody>
 <tr>
 <td>a</td><td>b</td><td>c</td>
 </tr>
 <tr>
 <td>d</td><td>e</td><td>f</td>
 </tr>
 </tbody>
</table>
```

# Column and Row Groups

So that we don't have to repeat the same information for each cell in a column, we can define column groups using the `<colgroup>` and `<col>` elements.

```
<table>
 <colgroup>
 <col span="2" class="firsttwo">
 <col class="middle">
 <col span="2" class="lasttwo">
 </colgroup>
 <tr>
 <td>A</td><td>B</td><td>C</td><td>E</td><td>F
 </tr>
</table>
```

They are very useful to set the *class* of each column without having to do it in each single `<td>`.

# Forms

# Form

A form (`<form>`) has form controls that allow users to provide data to be sent to a server for further processing (e.g. saving the data, return search results, or perform a calculation).

Forms have two main attributes:

- **action:** the URL of the service that will process the data.
- **method:** either **get** (values are sent in the URL) or **post** (values are sent inside the HTTP header)

More on HTTP methods later.

```
<form action="save.php" method="get">
 <!-- form controls go here -->
</form>
```

# Form Controls

Four main types of form controls:

- **input:** Several types of user-editable fields.
- **textarea:** A big editable text field.
- **select:** A dropdown list.
- **button:** A generic button.



# Input

An **<input>** field can vary in many ways, depending on the **type** attribute.

```
Date: <input type="date" name="date" value="2020-10-15">
Password: <input type="password" name="password" value="mysecretpasswo">
Number: <input type="number" name="number" value="123">
```

Date:  Password:  Number:

The **name** attribute is used to identify the field when processed in the server.

The **value** attribute contains the initial data in the field.

**Tip:** As always, dates are specified using **ISO 8601** (**obligatory xkcd**).

# Common Control Attributes

- **placeholder**: hint for the user shown only before text is entered.
- **autocomplete**: allow auto-completion by the browser (on/off).
- **readonly**: input value cannot be modified (boolean).
- **required**: input must be filled out (boolean).
- **disabled**: input is disabled (boolean).

```
Address: <input type="text" name="address"
 placeholder="your main address"
 required="required" disabled>
```

Address:

# Text Inputs

There are several different types of inputs just for normal text entry.

- **text**: text input with no constraints
- **password**: characters are not shown
- **tel**: input value is a telephone number
- **search**: input value is used to perform a search
- **url**: input value is an URL
- **email**: input value is an e-mail address

Some browsers may use slightly different controls for each type.

Search: `<input type="search" name="search" placeholder="Search" />`

Search:

# Number Inputs

```
<input name="n" type="number" value="5" min="0" m
```

There are two **types** for number inputs:

- **number**: a precise control for setting a number
- **range**: imprecise control for setting a number

Other attributes:

- **value**: the initial value
- **min**: the minimum value
- **max**: the maximum value
- **step**: limits the increments at which a value can be set

Number:

# Date/Time Inputs

There are many **types** for date-like inputs. They also have a **min**, **max**, **step** and **value** attributes.

- **date**: select a date
- **time**: control to select a time of the day
- **datetime-local**: select a time in a certain day
- **month**: select a month
- **week**: control to select a week

```
Date: <input name="date" type="date" value="2020-10-20" min="2020-10-01" max="2020-10-31" />
Time: <input name="time" type="time" value="10:00:30" />
Date and Time: <input name="datetime" type="datetime-local" value="2020-10-20T10:00:30" />
Month: <input name="month" type="month" value="2020-10" />
Week: <input name="week" type="week" value="2020-W09" />
```

Date:  Time:  Date and Time:  Month:  Week:

# Color Input

We can also select a color using the **color** type:

```
Color: <input name="color" type="color" value="#3
```

The **value** attribute contains the initial color in hexadecimal format.

Color:

# Checkbox

- A **checkbox** allows selecting **several** from a limited number of choices.
- If a checkbox is **selected**, its name/value pair is submitted to the server. If a checkbox is **not selected**, nothing is submitted.
- If two checkboxes have the **same name** and **both are selected**, both names/values are sent. In this case "vehicle=Bike&vehicle=Car".
- The boolean attribute **checked** sets the initial checked state of the checkbox.

```
<input type="checkbox" name="vehicle" value="Bike">Ride a bike
<input type="checkbox" name="vehicle" value="Car" checked>Drive a car
```

How do you get to school?

- ☐ Ride a bike  
☒ Drive a car

# Radio Button

- A **radio** allows selecting **one** from several choices.
- If a radio button is **selected**, then its name/value pair is submitted to the server.
- If a radio button is **not selected**, nothing is submitted.
- If two radio buttons have the **same name**, then only one can be selected; they form a selection group.

```
How do you get to school?

<input type="radio" name="gender" value="male" checked="checked">Male
<input type="radio" name="gender" value="female">Female
```

- ☒ Male  
☐ Female



# File Upload

The **file** type allows file uploading for storing or processing:

```
Upload:
<form action="upload_file.php" method="post" enctype="multipart/form-data">
 <input type="file" name="file" accept="image/png,image/jpeg" multiple="" />
</form>
```

**Important:** To use file uploading in a form, *method* must be **post** and *enctype* must be **multipart/form-data**.

The **accept** attribute can be used to **hint** the browser about what *mime-types* can be selected.

This is not enforced.

The **multiple** attribute allows the selection of **more than one** file.

Upload:

Escolher ficheiros nenhum ficheiro selecionado

# Hidden Input

Inputs with type **hidden** are not shown and are not meant to be changed by the user.

```
<input type="hidden" name="username" value="might"
```

**i** We will find what's their purpose later...

# Submit

The **submit** input type, allows the user to submit the form for processing.

The **value** contains the text to be used for the submit button. A multilingual default will be used if left blank.

```
<form action="save.php" method="get">
 <!-- Other form controls go here-->
 <input type="submit" value="Send">
</form>
```

The form will be submitted using the *method* and *action* defined in the **form** tag.



The button element (next slide) is a more modern way to achieve this behavior.

# Button

A different control is the **button** that can be used as:

- a generic button that has to be controlled with JavaScript (**type** button).
- an alternative way to submit a form (**type** submit, the default) using a different action (**formaction**) and method (**formmethod**).

```
<form>
 <button formaction="login.php" formmethod="post" type="submit">
 Login
 </button>
 <button formaction="register.php" formmethod="post" type="submit">
 Register
 </button>
</form>
```

This way, you can have different buttons with **different** actions and methods.



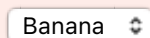
Login Register

# Select

The form control **select** allows the selection of one element (or several, with the **multiple** attribute) from a list of options.

```
<select name="fruit">
 <option value="orange">Orange</option>
 <option value="banana" selected>Banana</option>
 <option value="tomato">Tomato</option>
 <option value="apple">Apple</option>
</select>
```

For the **option** element, the **value** is what is sent to the server, the **content** is the value presented to the user, and the **selected** attribute allows to set the initially selected option.

A screenshot of a web browser showing a dropdown menu. The menu is open, and the word 'Banana' is visible in the selected state. The dropdown is set against a light orange background.

# Option Groups

Options in select controls can be grouped using the **optgroup** element; this makes selecting them in large lists more manageable.

```
<select name="food">
 <optgroup label="Fruits">
 <option value="orange">Orange</option>
 <option value="banana" selected>Banana</option>
 </optgroup>
 <optgroup label="Vegetables">
 <option value="lettuce">Lettuce</option>
 <option value="carrot">Carrot</option>
 </optgroup>
</select>
```

Banana

# DataList

The form control **datalist** is very similar to the **select** element.

The main difference is that it is connected to an **input** element (using the **list** and **id** attributes) and allows the user to write a value that does not exist in the list.

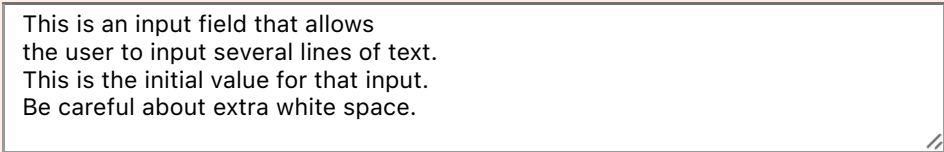
```
<input name="fruit" list="fruits" value="Banana">
<datalist id="fruits">
 <option>Orange</option>
 <option selected>Banana</option>
 <option>Tomato</option>
 <option>Apple</option>
</datalist>
```

# Text Area

The `<textarea>` element allows users to write larger, multiline texts.

```
<textarea name="description" rows="5" cols="60">
 This is an input field that allows
 the user to input several lines of text.
 This is the initial value for that input.
 Be careful about extra white space.
</textarea>
```

The initial value is a content of the tag and whitespace is significant. So be careful with it!



This is an input field that allows  
the user to input several lines of text.  
This is the initial value for that input.  
Be careful about extra white space.



# Label

The `<label>` element allows the association between text (the label) and its corresponding input:

- In most browsers, clicking the **label** focuses the **input**.
- This is of great importance in terms of **accessibility**.  
For example, it helps browsers for the visually impaired!

Two ways of creating the association:

```
<label for="id_name">Name:</label>
<input type="text" name="name" id="id_name">
```

```
<label>Name:
 <input type="text" name="name">
</label>
```

Name:

# Field Set

The `<fieldset>` element is useful to group controls in large forms.

The `<legend>` element contains the title of the group.

```
<form>
 <fieldset>
 <legend>Personal data:</legend>
 <label>Name: <input type="text"></label>
 <label>Email: <input type="text"></label>
 <label>Date of birth: <input type="text"></la
 </fieldset>
</form>
```

Personal data:

Name:

Email:

Date of birth:

# Character Entities

# Character Entities

A given **character encoding** may not be able to express all characters of the document character set.

Some characters might have some special meaning (<, >, " and &) and be confused by the browser as markup.

In HTML, character entity references may appear in two forms:

- Numeric character references (either decimal or hexadecimal).
- Named character entity references.

# Character Entities

Character entities always start with a **&** and end with a **;**

For example, the ampersand (&):

- Decimal character: `&#38;`;
- Hexadecimal character: `&#x26;`;
- Named character entity: `&amp;`;

Most important character entities:

- Less than sign (<): `&lt;`;
- Greater than sign (>): `&gt;`;
- Ampersand (&): `&amp;`;
- Double quote sign ("): `&quot;`;
- Non-breaking space ( ): `&nbsp;`;

[Other character entities](#) | [Character entity search](#)

Media

# Canvas

A **canvas** is an empty rectangle that can be used to draw on the fly using *JavaScript*.

```
<canvas width="400px" height="300px"></canvas>
```

Some cool **examples**.

# SVG

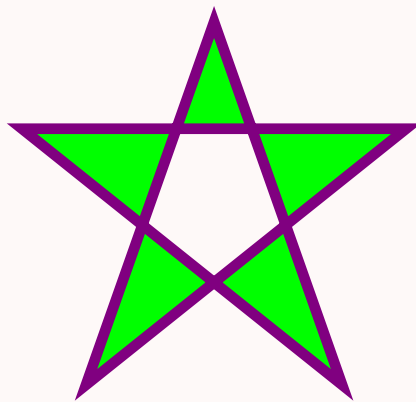
Scalable Vector Graphics (<svg>):

- SVG images can be created and edited with any text editor.
- SVG images can be searched, indexed, scripted, and **compressed**.
- SVG images are **scalable**.
- SVG images can be printed with high quality at **any resolution**.
- SVG images are **zoomable** without degradation.



# SVG Example

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="200" height="200">
 <polygon
 points="100,10 40,180 190,60 10,60 160,180"
 style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;"
 >
</svg>
```



# Other Media Tags

HTML 5 also includes specific tags for:

- **<audio>**: defines sound, such as music or other audio streams
- **<video>**: specifies video, such as a movie clip or other video streams
- **<source>**: specify multiple media resources for media elements (e.g., audio, video, and images).
- **<track>**: text tracks for video and audio elements

Learn more: [Using HTML5 Audio and Video](#)

# Metadata

You can define metadata for your document inside the head tag.

# Meta Content

The `<meta>` element is used to express metadata that cannot be expressed using other metadata elements (e.g., title).

```
<head>
 <meta name="?" content="?">
</head>
```

Possible values for the **name** attribute:

- **application-name**, defining the name of the web application running in the webpage.
- **author**, defining, in a free format, the name of the author of the document.
- **description**, containing a short and accurate summary of the content of the page.
- **generator**, containing, in a free format, the identifier to the software that generated the page.
- **keywords**, containing, as strings separated by commas, relevant words associated with the content of the page.

# Character Set

One of the uses of the **<meta>** element is to specify the **character encoding** of an HTML document.

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

Some encodings:

- **utf-8** Character encoding for Unicode (recommended).
- **iso-8859-1** Character encoding for the Latin alphabet.

Knowing about Unicode and charsets is **important** for every software developer.

# Validation

# Validation

- Browsers try to correct mistakes done by developers (e.g., missing closing tag).
- But we cannot rely on all browsers fixing our mistakes in the same way.
- Sometimes mistakes are not present in the rendered version of the page (e.g., using the wrong semantic element or missing a mandatory semantic attribute).
- These are some reasons why you should always validate your HTML code:  
<http://validator.w3.org/>