Hypertext Markup Language

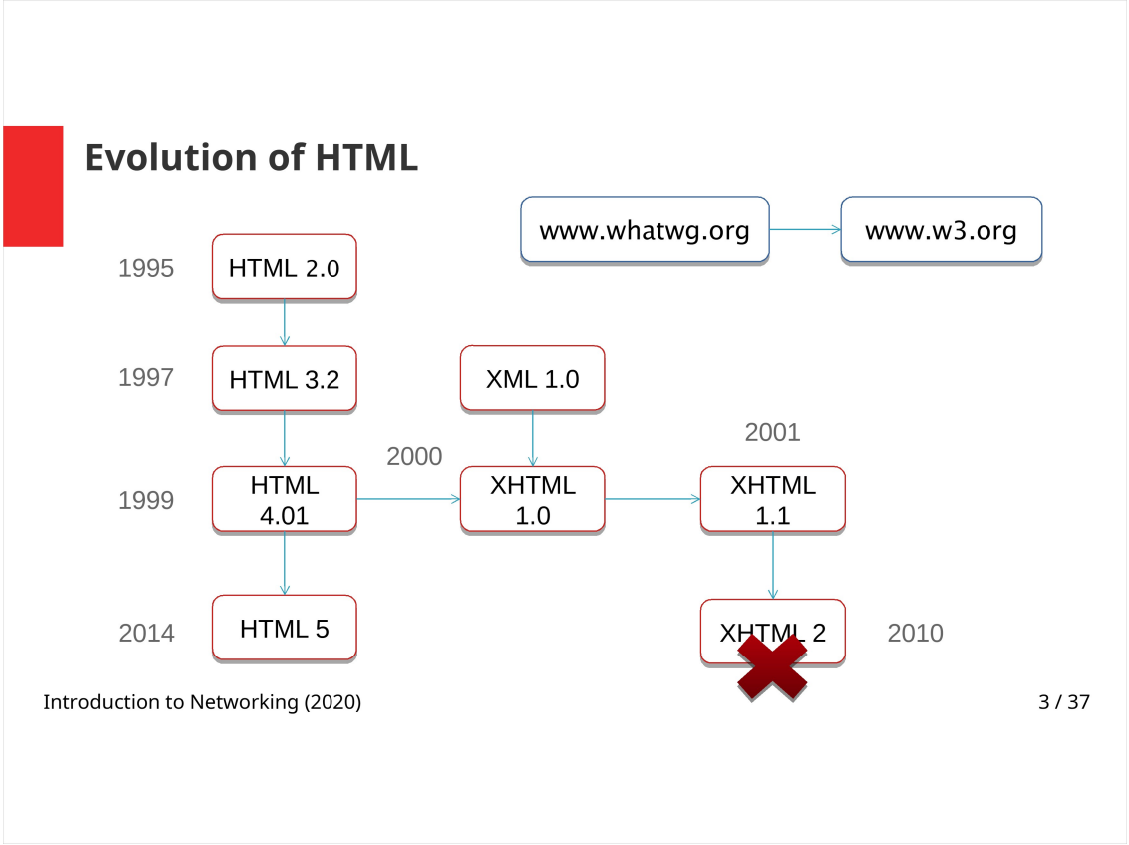
Introduction to networking

Dr. Klára Pešková, Klara.Peskova@mff.cuni.cz
Department of Software and Computer Science Education

HTML – Hypertext Markup Language

- World Wide Web's markup language
- Language for describing the structure of Web pages

- Web page
  - A structured document
  - Plain text marked by HTML tags that create the document structure



HTML was created by Tim Berners-Lee in 1990. Until 1993 it did not support images.

HTML 2.0 – supports forms and images.

HTML 3.2 was published as a W3C Recommendation. It was the first version developed and standardized exclusively by the W3C. Tables, text alignment, and other visual markup tags were added. HTML 3.0 was never accepted as a standard as it was too complicated.

HTML 4.0 – new tables and forms elements, frames.

HTML 4.01 – minor edits; it was supposed to be the last version before switching to XHTML – a successor of HTML that uses XML language.

XHTML – evolutionary dead end; did not continue after 2010.

WHATWG community started to create a new HTML specification; from 2007 HTML Working Group within W3C.

HTML5 – major corrections, semantic elements added.

Evolution of HTML

- WhatWG
  - http://whatwg.org/html
  - "Living Standard" (see Last Updated)
- W3C
  - http://www.w3.org/TR/html5/
  - Formal standardization process

The Web Hypertext Application Technology Working Group (WHATWG) is a community of people interested in evolving HTML and related technologies. The WHATWG was founded by individuals from Apple Inc., the Mozilla Foundation and Opera Software, leading Web browser vendors, while World Wide Web Consortium (W3C) is a non-commercial consortium.

The WHATWG was formed in response to the slow development of W3C Web standards and W3C's decision to abandon HTML in favour of XML-based technologies. The aim of the community however is to create specifications that will be approved by W3C.

HTML source code

```
<!DOCTYPE html>
<html>
    <head>
        <title>Iris</title>
    </head>

    <body>
        <h1>Iris versicolor (Blue flag)</h1>

        <p>
            Iris takes its name from the Greek word for a rainbow.
        </p>

        <img src="iris.jpg" alt="Blue flag">
    </body>
</html>
```


Example of the source code:

The document begins with `<!DOCTYPE>` that defines the type of the document, `html` stands for HTML5.

HTML itself consists of two parts
- `<head>` with document metadata
- and document `<body>` – this part is displayed on the page.

In the example above, head contains a title that is shown in the browser's title bar or in the page's tab.

Body contains
- A heading
- A paragraph
- An image

HTML source code

```
<!DOCTYPE html>
<html>
    <head>
        <title>Iris</title>
    </head>

    <body>
        <h1>Iris versicolor ... </h1>

        <p>
            Iris takes its name ...
        </p>

        <img src="iris.jpg" alt="Blue flag">
    </body>
</html>
```

Iris versicolor (Blue flag)

Iris takes its name from the Greek word for a rainbow.

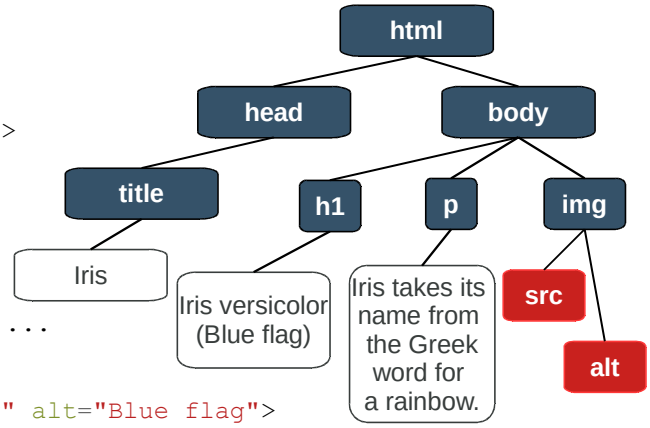
This is how the page is displayed in a browser.

HTML syntax

- HTML document is structured as a tree
- Various types of nodes
  - Elements, text, attributes, comments, …
- HTML syntax is an infix serialization of the tree
- Tree is represented as Document Object Model (DOM) inside the browser
  - The DOM can be manipulated with JavaScript and plays role in CSS selectors evaluation

DOM (Document Object Model) tree

```
<!DOCTYPE html>
<html>
    <head>
        <title>Iris</title>
    </head>

    <body>
        <h1>Iris ... </h1>

        <p>
            Iris takes its ...
        </p>

        <img src="iris.jpg" alt="Blue flag">
    </body>
</html>
```



Example of a hierarchical tree structure of a HTML Document Object Model.

HTML elements properties can be set by attributes

- `<img>` element has `src` and `alt` attributes

HTML Body markup

- Inside element `<body>`
- Text-level semantics elements
  - Denote parts of the text in a HTML document with a specific semantics
- Sectioning content
- Grouping content
- Tables, forms
- External sources (images)
- Hyperlinks
- ...

First we will go over the HTML `<body>` contents.

The document body consists of HTML elements (or more precisely by their content). Elements add semantics and formatting to parts of document.

Semantic elements are elements with a meaning – not only they are for example containers for grouping their content, they also describe a meaning of its content within a web page. E.g. `<nav>` semantic element groups its nested elements and it also defines its content as a set of navigation links.

HTML syntax – HTML element

- A HTML element represents a fragment of a web page
  - Gives semantic meaning to its content
- Opening and closing tags work as parentheses for the content (i.e. boundaries of the element)
  - Content-less elements may omit closing tag → paired / singular or empty tags
  - Elements may not cross-overlap
- Attributes
  - Name-value pairs specified in an opening tag
  - Values are optional and they are optionally enclosed in single or double quotes (recommended)
    - Quotes are mandatory if the value contains ", ', <, or >

```
<p>Iris takes its name from the Greek word for a rainbow.</p>
```


A HTML element is defined by a start tag, some content, and an end tag.

The properties of HTML elements are set as tag attributes – an `attribute="value"` pairs. If more attributes are set, they are separated by a space. Setting attributes is optional (if not default values are used).

Elements are nested (thus the hierarchical structure of HTML document is created), but they may not cross-overlap. E.g. the following overlapping is not allowed:

```
<form>
<p>
</form>
</p>
```

HTML syntax

- Tags
  - Tag names in angle brackets **<>**
  - Closing tag name starts with **/**
- Whitespace characters do not play any role
- Comments
  - Enclosed in **<!--** and **-->**
  - Not displayed when page is rendered
- HTML entities: **&entity-name;**
  - Provide a way to encode special characters

```
<h1>Heading</h1>
```

```
<!-- comment -->
```

```
< … &lt;
> … &gt;
& … &amp;
" … &qout;
© … &copy;
```

```
non-breaking space …  
numerically represented characters ü … &#252;
```


Whitespace characters (space, tab, new line) are not displayed, more whitespace characters in a row are reduced to a single space.

This means you can use indentation and separate the parts of you code vertically with new lines.

(To add spacing to the elements, use CSS properties.)

HTML element types

- Semantic elements
  - describe the meaning of the HTML contents – to web browsers and web developers
  - e.g. `<nav>`, `<header>`, `<form>`, `<table>`, ...
- Non-semantic elements
  - e.g. `<div>`, `<span>`...
  - Tell nothing about their contents

A semantic element clearly describes its meaning to both the browser and the developer. Browser can adjust the web page to this semantics, and the source code is more readable.

The semantic meaning of elements is not defined strictly. A page is written by a human and it is assumed that it will be also read by a human. The same semantic tag can be used differently according to a specific situation. It is always up to a web developer to decide, what semantic elements to use.

Text-level semantic elements

`<em>` represents stress emphasis of its content. The level of stress is given by the level of nesting of particular `em` elements

`<strong>` represents strong importance of its content

`<small>` represents a side comment

`<cite>` represents a title of a work (book, game, software, song, ...)

`<code>` defines a piece of computer code

`<abbr>` an abbreviation or acronym, optionally with its expansion in `title` attribute

`<i>` indicates a different quality of text

`<b>` a content to which attention is being drawn

`<sub>` `<sup>` subscript and superscript

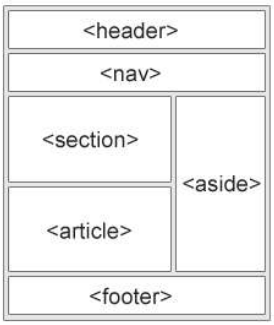
`<br>` a line break

Text-level elements are used inline, i.e. for a part of a paragraph or a sentence. With these elements, no extra vertically separated block is created, they are part of a line of text.

Note:
`title` attribute can be used for any HTML element. Its value is displayed on mouse hover.

Sectioning content – Semantic HTML5 elements

`<section>` defines a section in a document (with a heading)
`<article>` independent, self-contained content; can be shared independently (newspaper article, blog post)
`<header>` a header for a document or section (intro content, set of navigational links, logo, ...)
`<footer>` a footer for a document or section, copyright, contact info, back to top links, ...
`<nav>` defines navigation links
`<aside>` content aside from the page content; its contents should relate to adjacent content
`<figure>` specifies self-contained content (illustrations, diagrams, photos, code listings, etc.)
`<figcaption>` a caption for a `<figure>` element
`<details>` additional details that the user can view or hide
`<summary>` a visible heading for a `<details>` element
`<main>` the main content of a document
`<mark>` marked/highlighted text
`<time>` defines a date/time



Semantic elements that describe the page structure were introduced in HTML5. Formally neutral `<div>` elements that only group the contents were used for this purpose.

The figure on the right shows a typical structure of a web page described by semantic elements. The way of using these elements is left up to a web developer. For example it is not specified, whether `<section>` should be higher in a hierarchy than `<article>` or vice versa. Both elements may be (repeatedly) nested into each other. Thus `<section>` can contain `<article>`s and the `<article>` can consist of `<section>` elements.

Grouping

`<p>` a paragraph

`<pre>` a block of preformatted text

`<div>` block element with no special meaning (generic container)

`<span>` inline element with no special meaning (generic container)

`<main>` represents a block with a dominant content

`<ul>` an unordered list

`<ol>` an ordered list

`<li>` a list item

`<dl>` a definition list

`<dt>`, `<dd>` definition term, definition description

Grouping elements can also define the structure of the grouped elements, lists are an example of defining such a structure.

Heading elements – <h1> – <h6>

- Before HTML5
  - Six levels of importance (rank)
  - <h1> most important (highest), <h6> least important
- Headings in HTML5
  - Combined with sectioning <section>, <article>, ...
  - Attempt to keep some backwards compatibility
  - Quite difficult to do though
  - Each section has its own heading hierarchy
  - First heading element in a section is the main heading of that section (no matter its rank)

Tags <h1> – <h6> are paired, their content is the actual text of the heading.

In HTML5 you can use only <h1> headings.

```
<h1>Favourite flowers</h1>

<section>
  <h1>Iris</h1>
  <p>There are many Iris species.</p>

  <h1>Wild Rose</h1>
  <p>I like roses because...</p>
</section>
```

From HTML5 specification:

Sectioning content is content that defines the scope of headings and footers… Each sectioning content element potentially has a heading and an outline.

HTML lists – `<list>` element

- Ordered or un-ordered
- Can be nested

- day
- month
    1. January
    2. February
    3. March
    4. ...
- year

```
<ul>
   <li>day</li>
   <li>month
      <ol>
         <li>January</li>
         <li>February</li>
         <li>March</li>
         <li>...</li>
      </ol>
   </li>
   <li>year</li>
</ul>
```

HTML tables – <table> element

<table> defines a table
<caption> defines a table caption (must be inserted immediately after the <table> tag)
<thead> groups the header content in a table (optional)
<tbody> groups the body content in a table (optional)
<tfoot> groups the footer content in a table (optional)

<tr> table row
<td> table cell

- a cell can span more than one row/column (attributes colspan, rowspan)

<th> header cell

<table> element should only be used to describe tabularized (2D) data. Formerly tables were often use to define the page layout.

HTML table – example

Users

E-mail	Name
smith@gmail.com	Adam Smith
joe@black.ml	Joe Black
me@black.ml	

```
<table>
  <caption>Users</caption>
  <thead>
    <tr>
      <th>E-mail</th><th>Name</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>smith@gmail.com</td>
      <td>Adam Smith</td>
    </tr>
    <tr>
      <td>joe@black.ml</td>
      <td rowspan="2">Joe Black</td>
    </tr>
    <tr>
      <td>me@black.ml</td>
    </tr>
  </tbody>
</table>
```

HTML images – <img> element

- Singular element (has no content → closing tag can be omitted)
- Attributes
  - `src` – the URL of an image to be loaded
  - `alt` – alternative textual representation
    - Describes, what the image shows
    - Used, when the image loading fails, or for “non-visual” outputs
  - `title` – shows on hover

```
<img src="iris.jpg"
 alt="Blue flag"
 title="Iris versicolor">
```

Image description in an alternative text (`alt`) attribute is very important for accessibility of the web page (e.g. if a visually impaired person is using a screen reader, the alternative text is read.)

HTML hyperlinks – <a> element

- Links to external resources
- Attributes
  - `href` – specifies URL of linked resource
  - `target = "_blank"` – opens the link in a new window

```
Go to <a href="http://www.google.com">Google</a>
```

Go to Google

HTML hyperlinks continued...

- Link to a web page fragment

```
<a href="#net">Go to Networking section</a>
<section id="net">Networking ...</section>
```

- Special links

```
<a href="tel:+420603123456">603 123 456</a>
<a href="mailto:klara@pisecko.cz">send e-mail</a>
```

- e.g. clickable image (put image into a link)

```
<a href="http://www.seznam.cz">
   <img src="logo.gif">
</a>
```


Linking to page fragments:

Hyperlinks can target a specific point in a document that has been created with a “fragment” identifier – a unique ID set in elements `id` attribute. Browser “scrolls” to this element.

If the specified fragment is on the same web page as the link `href` attribute contains `#fragment_id`, as shown in the example on the above slide. Page fragment can also be used when linking to external pages – in this case `#fragment_id` is the last part of the URL.

Linking to a page fragment can be used to navigate within a long document content.

Special hyperlinks:

- Link with `mailto` opens the default e-mail client
- Link with `tel` dials the phone number

HTML forms

- A component composed of UI controls

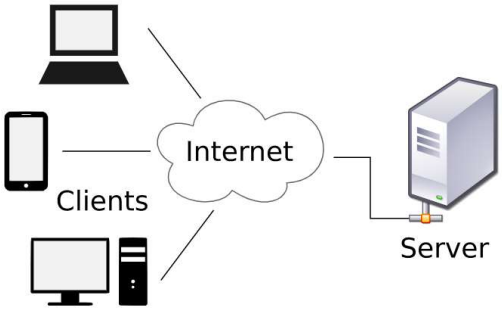
```
<form action="script.php">
  <label for="name">Name:</label>
  <input type="text" id="name">
  <label for="pwd">Password:</label>
  <input type="password" id="pwd">
  <input type="submit" value="OK">
</form>
```

Forms represent a way to get user input to our application. Output is then presented as a dynamically generated HTML.

Single user controls are nested inside `<form>` element.

Submitting forms

- User fills in the data
- Form data are validated (optional)
- Data are sent to a server to be processes
- Data are pre-processed and validated on the server
- Server processes the data
- Response is sent to a client (web browser) as a HTML file



Form controls

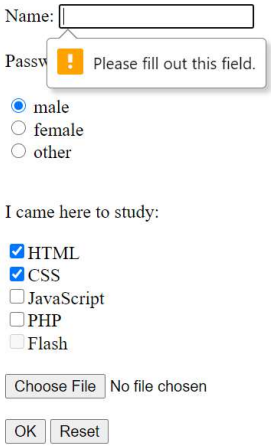
- `<input>`
  - Basic input controls
  - Various types based on input attributes
- `<textarea>`
  - Input for longer (multi row) texts
- `<select>`
  - Selection list with `<options>`
- `<button>`
  - Submit or reset button

`<input>` is a singular tag (it does not have an end tag). The other elements mentioned above do have a content.

We will go over these form controls in the following slides.

Basic form components – <input>

```
<form>

  <input type="text">
  <input type="radio" name="group" checked>
  <input type="checkbox" checked>
  <input type="submit" value="OK">

  <input type="hidden">
  <input type="file">

  <input type="password">

  <input type="reset">
  <input type="button">

</form>

checked, disabled, required, maxwidth, size="num of chars"
```



`<input>` is a singular (empty) tag. Its type is set by `type` attribute.

Input types (possible `type` attribute values):

- `text` – a simple text field
- `radio` – radiobuttons form a group ("connected" by `name` attribute); only one of them can be selected
- `checkbox` – it is possible to check more options
- `hidden` – a hidden (not displayed) form element can be used to send extra data to a script that processes the form on a server
- `file` – a button that can be clicked to select a file
- `password` – when user enters a value to this field, * are displayed instead of characters

- `submit` – a button that submits the form; `value` attribute value is used as the button caption
- `reset` – a button that sets default values to all form controls
- `button` – a general button; for example a JavaScript code can be assigned to the button and executed when the button is clicked

Attributes

- `checked` – boolean attribute (no value is set); can be set for `radio` or `checkbox`
- `disabled` – element that is visible but can not be filled in
- `required` – HTML5 validates the presence of data on form submit
- `size` – displayed width of the field (set as a number of chars); it does not set the number of chars that can be entered into a field
- `maxlength` – max number of chars that can be entered into a field

HTML5 inputs

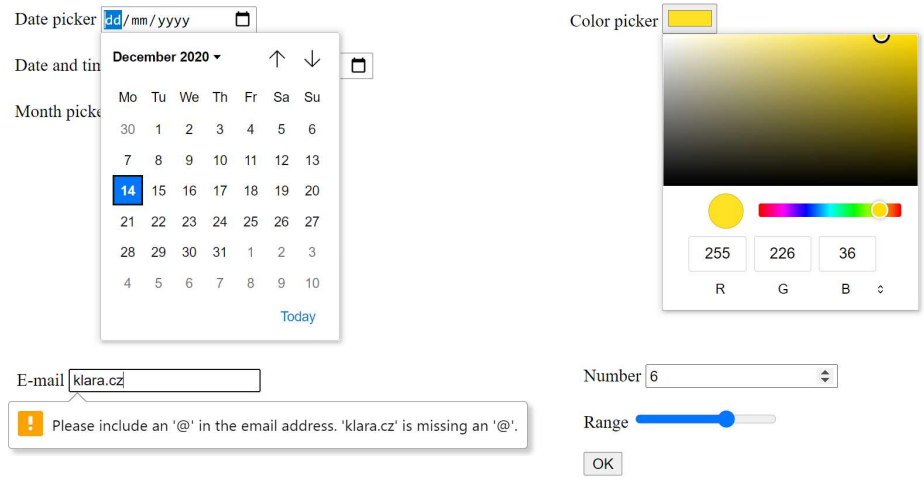
```
<form>
  <input type="color">
  <input type="date">
  <input type="datetime-local">
  <input type="month">
  <input type="email">
  <input type="number">
  <input type="range">
</form>
```


New `<input>` types have been introduced in HTML5.

Unsupported types are treated as `text` inputs.

On the following slide, there are examples of these user controls.

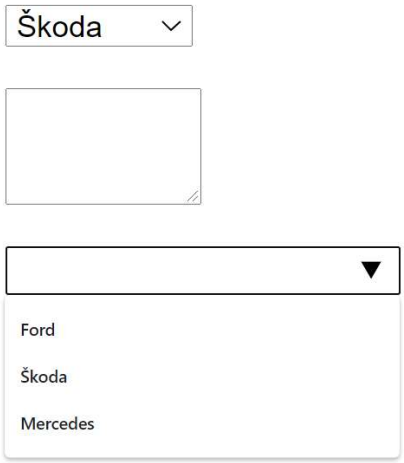
HTML5 inputs – examples



OK

More form components

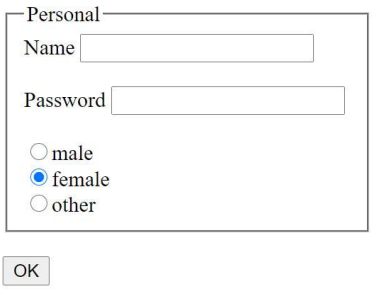
```
<select>
  <option>Ford</option>
  <option selected>Škoda</option>
  <option>Mercedes</option>
<select>

<textarea rows="3" cols="10">
</textarea>

<input list="cars">
<datalist id="cars">
  <option>Ford</option>
  <option>Škoda</option>
  <option>Mercedes</option>
</datalist>
```



- `<select>` – value can be chosen from more options; options are defined by `<option>` elements
  - `multiple` – use this attribute to enable a user to choose more options at a time; attribute has no value
  - `selected` –is an `<option>` attribute; it defines the pre-selected option; attribute has no value
  - `<option>` elements can be groupped in `<optgroup>` element
- `<textarea>` - paired tag; its size can be set using `rows` and `cols` attributes (values are set as numbers of characters); content of this element is displayed inside the text area
- `<datalist>` - is connected with `<input>` element using `id` and `list` attributes. Datalist provides a drop-down list of pre-defined options; user can input a custom value

More on forms

```
<label for="element_id">
    Element description
</label>
<fieldset>
  <legend>Fieldset heading</legend>
</fieldset>
```



- Form elements attributes:

`autocomplete="on"/"off"` **– can be set for the form or for single form controls**

`placeholder`

`title` - **a hint for the field (“bubble” with text)**

`<label>` – a label for an element inside the form. Label and the element are connected by `id` and `for` attributes.

When a user clicks the text within the `<label>`, it toggles the connected user input element.
Screen readers read the description in `<label>`.

`<fieldset>` – is used to group related elements in a form

`autocomplete` – autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values (e.g. saved username and password); default value is `on`

`placeholder` – the placeholder attribute specifies a short hint that describes the expected value of an `<input>` field. The short hint is displayed in the input field before the user enters a value.

`title` – this attribute may be used with any HTML element

HTML forms – `<form>` element

- Attributes
  - `method` – HTTP method used for data transfer (GET/POST)
    - `post` – data are transferred in HTTP request body
    - `get` – data are encoded in URL (query part) – visible to everyone
  - `action` – URL where the data are sent to

```
<form action="script.php" method="get">
   ...
</form>
```


`<form>` attributes `action` and `method` set the parameters of the HTTP request that is created on form submit.

In the above example, the form is processed by `script.php` file, HTTP GET method is used to send the data to a server.

Submitting forms

- When a form is submitted, data are encoded into HTTP request and sent to the selected URL using the given method (GET/POST)
- Submit button works as a link to a new URL created based on filled form data

```
<form action="script.php" method="get">
   <input type="text" name="name">
   <input type="password" name="pwd">
</form>

=> URL: script.php?name=John&pwd=45ak!
```

When the form is submitted, special characters, e.g. space, are encoded into URL.

When GET method is used, user data (including a password) are visible in the URL.

Note:
Instead of using `<input>` with type `submit` as a form submit button, paired `<button>` element can be used. The content of the element is displayed as a button caption. The advantage of using `<button>` element is that the `value` of the element and the displayed text are separated.

In HTML5 `<button>` element can be placed outside the form, it is connected to a form using `form` attribute.

HTML Head – document metadata

- Additional information about the document
- Inside element `<head>`
  - `<title>` - document title or name (browser window caption)
  - `<link>` - links document to other resources, e.g. CSS files
  - `<style>` - embeds style (CSS) information inside the document
    - Contents is written in CSS stylesheet syntax
  - `<meta>` - additional metadata and HTTP supplements
    - Description, keywords, document author, ...

Medatata from `<head>` element are not directly displayed to a user.

HTML Head – metadata - example

```
<html>
  <head>
    <title>Iris species</title>

    <link rel="stylesheet" href="styles.css">

    <style type="text/css">
        body { font-size: 12pt; }
    </style>

    <meta name="author" content="Joe Black">
    <meta charset="utf-8">

  </head>
</html>
```


In the above example, following meta-information is set:

- Page title, displayed in the browser's window/tab
- a .css file is linked to the document by `<link>` element
- Inside of the `<style>` element more CSS styles are defined
- Finally the following metadata are set
  - Author of the web page
  - Text encoding (set to `utf-8`)

More on links <a>, <link>

- `href` attribute
  - URL of resource linked by relationship
- `rel` attribute
  - Type of the relationship
- `media` attribute
  - Specified the media the linked resource applies to
  - E.g., `print`, `screen`, `all`
- `type` attribute
  - MIME type of linked resource
  - `text/html`, `application/xhtml+xml`, `text/css`, `application/pdf`

A link definition according to HTML5:

Link represents relationship of particular type between current document and other web resource

Syntactically links can be defined by elements `<link>` or `<a>`.

Two kinds of links (according to HTML5 spec.):
- Links to external resources – augment/further specify current document
- Hyperlinks – exposed to the user to navigate between resources

The kind depends on the element used and on the relationship type (attribute `rel`)

More on links – examples

```
<html>
  <head>
    <title>NSWI142 – Materials</title>
    <link rel="stylesheet" href="default.css"
          type="text/css" media="screen">
    <link rel="stylesheet" href="default-print.css"
          type="text/css" media="print">
  </head>
  <body>
    <footer>Author:
      <a href="http://www.ksvi.mff.cuni.cz/~peskova"
         rel="author">Klára Pešková</a>
    </footer>
  </body>
</html>
```


There are two CSS files linked to the document, the first one (`default.css`) is used when the page is displayed on a screen, the other (`print.css`) is used for printing the page on a printer.

An example of setting a relation for a `<a>` element follows.

Sources

- **HTML markup – 4.01 vs 5**
  - For differences see http://www.w3.org/TR/html5-diff
- **Tutorials**
  - https://www.w3schools.com/