University of Tissemsilt
Faculty of Science & Technology
Departement of Math and Computer Science

Application Web Development
HTML

25 février 2024

Lecturer

Dr. HAMDANI M

Speciality : Computer Science (ISIL)
Semester : S4

Plan

Introduction

Web Application Development refers to the process of creating and maintaining software applications that are accessed over the internet through web browsers.

Types of web developers

- **Frontend developers** : Frontend developers implement web page designs using HTML and CSS. They make sure the website looks pretty on different devices, and that the forms and buttons work.

Types of web developers

- **Frontend developers** : Frontend developers implement web page designs using HTML and CSS. They make sure the website looks pretty on different devices, and that the forms and buttons work.
- **Backend developers** : Backend developers create the backbone of the web application. They write code logic that handles a user's input (for example, what should happen when you click the signup button after filling in a form).

Types of web developers

- **Frontend developers** : Frontend developers implement web page designs using HTML and CSS. They make sure the website looks pretty on different devices, and that the forms and buttons work.
- **Backend developers** : Backend developers create the backbone of the web application. They write code logic that handles a user's input (for example, what should happen when you click the signup button after filling in a form).
- **Full stack developers** : Full stack developers do bits of both backend and frontend.

Internet vs. WWW

Most people use the two terms interchangeably but they are in fact different.

- **The Internet** is a vast, international network, made up of computers and the physical connections (wires, routers, etc.) allowing them to communicate.
- **The World Wide Web** (WWW or just the Web) is a collection of software that spans the Internet and enables the interlinking of documents and resources.

Provides a way of accessing information on the Internet.

Web Apps Compared to Desktop Apps

Advantages of web apps :

- Accessible from any internet-enabled computer.
- Usable with different operating systems and browser platforms.
- Easier to roll out program updates since only need to update software on server and not on every desktop in organization.
- Centralized storage on the server means fewer concerns about local storage (which is important for sensitive information such as health care data).

Web Apps Compared to Desktop Apps

Disadvantages of web apps :

- Internet is not always available everywhere at all time).
- Security concerns about sensitive private data being transmitted over the internet.
- Concerns over the storage, licensing and use of uploaded data.
- Problems with certain websites on certain browsers not looking quite right.
- Limited access to the operating system can prevent software and hardware from being installed or accessed (like Adobe Flash on iOS).

Internet

- Global public network.
- Accessible by anyone with an internet connection.
- Emphasis on external security.
- Open for public information sharing.
- Users : Worldwide public.

Intranet

- Private network within an organization.
- Access limited to organization members.
- Emphasis on internal security.
- Primarily for internal collaboration.
- Users : Employees or members.

Extranet

- Extends to external parties.
- Controlled access for trusted stakeholders.
- Emphasis on both internal and external security.
- Facilitates external collaboration.
- Users : External parties (e.g., clients, partners).

Static Websites

- Fixed Content
- HTML, CSS, and Limited JavaScript
- Manual Updates
- Limited Interactivity
- Examples : Personal Blogs, Brochure Sites

Dynamic Web Sites

- Dynamic Content
- Server-Side Scripting (PHP, Python, etc.)
- Easy Updates (Content Management Systems)
- High Interactivity (User Logins, E-commerce)
- Examples : Social Media, E-commerce, Web Applications

Server-side and client-side

- Server-side processes are executed on the web server.
- Client-side processes are executed on the user's device.

Server-Side

- **Execution Location** : Code runs on the web server.
- **Languages** : Commonly uses server-side languages like PHP, Python, Ruby, Node.js, etc.
- **Responsibilities** : Handles server operations, database interactions, and business logic.
- **Data Processing** : Data processing occurs on the server.
- **Security** : Secure for sensitive data and logic.
- **Examples** : Content management systems (CMS), e-commerce platforms, web applications.

Client-Side

- **Execution Location** : Code runs in the user's web browser.
- **Languages** : Primarily JavaScript.
- **Responsibilities** : Enhances user interface, interactivity, and user experience.
- **Data Processing** : Limited data processing ; relies on server for critical operations.
- **Security** : Limited security for sensitive data and logic.
- **Examples** : Interactive websites, single-page applications (SPAs), browser games.

Server-side and client-side : Key take-aways

- Server-side and client-side refer to the location where certain tasks or processes are carried out in a web application.
- Server-side processes are executed on the web server before the web application is delivered to the user's device.
- Client-side processes are executed on the user's device after the web application is delivered.
- Server-side processes have more access to resources and are more secure, while client-side processes have less access to resources and are potentially less secure.

What is HTML ?

- HTML(Hypertext Markup Language), is the standard markup language used to create and structure content on the World Wide Web.
- It is the foundation of web pages and is used to define the structure and layout of web documents.
- HTML documents are interpreted by web browsers to render text, images, links, forms, and other elements on a web page.

HTML Skeleton

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
      initial-scale=1.0">
<title>Document</title>
</head>
<body>

</body>
</html>
```

Basic structure

- `<!DOCTYPE html>` : document type and version of HTML. "HTML5" is the recommended standard.
- `<html>` : The root element that encapsulates the entire HTML document.
- `<head>` : Contains metadata about the document, such as character encoding, document title, and links to external resources like CSS and JavaScript files.
- `<meta charset="UTF-8">` : Declares the character encoding for the document. UTF-8 is a widely used encoding that supports a wide range of characters from various languages.
- `<body>` : Contains the visible content of the web page, including text, images, links, and other elements.

TITLE

Best Practice

The text in your **TITLE** should be as descriptive as possible because this is what many search engines, on the internet, use for indexing your site.

Tags

- Enclosed in angle brackets (< and >)
- Usually paired
- The opening tag indicates the beginning of an element, while the closing tag is used to mark the end of that element
- Not case sensitive

```
<TITLE> My Web Page </TITLE>
```

Attributes

- Attributes are additional information or properties provided within the opening tag of an HTML element
- Used to specify various properties, behaviors, or settings for the element.

```
<img src="image.jpg" alt="An example image" />

<input type="text" name="username" disabled />

<a href="https://www.example.com">Visit Example.com</a>
```

HTML elements

- HTML documents consist of a series of elements that define the structure and content of a web page.
- Each HTML element has a specific purpose and meaning, and they can be combined to create the visual and interactive components of a webpage.
- These elements are represented by tags, and each tag has a specific purpose and meaning.

Common Tags

- `<h1>, <h2>, <h3>, ...` : Headings of various levels.
- `<p>`: Defines a paragraph of text.
- `<a>`: Creates hyperlinks to other web pages or resources.
- `<img>`: Embeds images in the document.
- `<ul>`: Defines an unordered (bulleted) list.
- `<ol>`: Defines an ordered (numbered) list.
- `<li>`: Represents individual items within a list.
- `<table>`: Defines a table.
- `<tr>, <td>`: to define the structure and content of tabular data.
- `<div>`: A generic container element used to group and structure content for styling or scripting purposes.

Semantic Markup

- Semantic markup involves using HTML tags to reinforce the meaning and structure of web content.
- HTML5 introduces a set of semantic elements designed to describe the content's purpose.

Semantic Elements in HTML5

- <**header**> : Defines the header of a section or page.
- <**footer**> : Specifies the footer of a section or page.
- <**nav**> : Represents a navigation menu.
- <**article**> : Defines independent, self-contained content.
- <**section**> : Represents a generic section of a document.

Advantages of Semantic Markup

- Accessibility : Semantic elements improve accessibility for users of assistive technologies by providing clearer structure.
- SEO (Search Engine Optimization) : Search engines can better understand the content and context of a webpage, leading to improved search rankings.
- Consistency : Semantic markup promotes consistency in web development practices and encourages better organization of content.

Example of Semantic Markup

New Semantic Elements

- <**header**> : Defines a header for a document or section.
- <**footer**> : Specifies a footer for a document or section.
- <**article**> : Defines independent, self-contained content.
- <**section**> : Represents a generic document or application section.
- <**nav**> : Defines navigation links.

These elements help structure web pages more semantically and improve SEO and accessibility.

Form Enhancements

- New form types for improved user input : email, date, time, url, search, etc.
- New attributes like placeholder, autocomplete, required, and pattern for better form validation and user experience.

Multimedia Support

- <**video**> and <**audio**> elements for embedding video and audio content natively without requiring third-party plugins.
- Support for multiple source files to ensure compatibility across different browsers.

Graphics and Animation

- Canvas API : Allows for dynamic, scriptable rendering of 2D shapes and bitmap images.
- SVG (Scalable Vector Graphics) : Supports vector graphics embedding directly in HTML documents.
- CSS3 animations and transitions : Enhance web pages with visual effects.

Enhanced Connectivity

- New technologies for communication such as WebSockets for real-time bidirectional communication between client and server.
- Offline storage capabilities with Application Cache, Web Storage, and IndexedDB for creating web applications that work offline.

Accessibility Improvements

- HTML5 places a strong emphasis on making content accessible to all users, including those with disabilities.
- ARIA (Accessible Rich Internet Applications) roles and properties can be used with HTML5 to make web applications more accessible to people with disabilities.

Headings

- Used to define the hierarchical structure and titles of sections or content on a web page.
- Improve readability, accessibility, and SEO.
- Represented by the <h1>, <h2>, <h3>, <h4>, <h5>, and <h6> elements, each indicating a different level of importance and hierarchy.
- You should not skip heading levels : e.g., an H3 should not appear after an H1, unless there is an H2 between them.

Heading Levels

- <h1> : Represents the highest level of importance and is typically used for the main heading or title of the page. There should be only one <h1> per page.
- <h2> : Represents a second-level heading, often used to subdivide the content under the main heading.
- <h3> to <h6> : Represent subsequent lower levels of headings, with h3 being less important than h2 , and so on. They are used to further structure the content within sections.

SEO Benefits (1)

- Hierarchy and Content Organization : search engines use this hierarchy to determine the importance and relationship of different sections of your page
- Keyword Usage : Search engines consider the text within headings as important clues about the page's topic.
- User Experience : Visitors can quickly skim through the headings to get an idea of the page's content.

SEO Benefits (2)

- Semantic Markup : using semantic elements like <header>, <nav>, and <section> alongside appropriate headings helps search engines understand the meaning and relationships between different parts of your page.
- Accessibility : Well-structured headings also improve web accessibility, which is a crucial factor for SEO.
- Featured Snippets : Headings are often used as the basis for featured snippets in search results.

Example Usage

```
<body>
<h1>Main Heading</h1>
<p>This is some introductory content.</p>

<h2>Section 1</h2>
<p>Content for section 1 goes here.</p>

<h3>Subsection 1.1</h3>
<p>Content for subsection 1.1 goes here.</p>

<h2>Section 2</h2>
<p>Content for section 2 goes here.</p>
</body>
```

Common Errors in Heading Usage

- Skipping Levels : Using heading levels non-sequentially (e.g., jumping from h2 to h4) can confuse both users and search engines.
- Styling for Appearance : Applying heading tags solely for styling (e.g., making text larger) rather than for semantic meaning.
- Overuse of <h1> : Using <h1> excessively throughout a page, which can lead to a lack of content hierarchy.
- Empty Headings : Creating headings with no content or using them solely as decorative elements.
- Using Headings for Links : Assigning headings to links (e.g., <a> with <h2>), which can disrupt the page's structure.

Random text generator

Text Generator

1. Microsoft Word :
   - =rand(), or
   - =lorem(4,5) : for 4 paragraphs of 5 sentences
2. VSCode : **lorem***Number_of_words*
   Example : lorem10
3. https://www.lipsum.com/

Paragraphs, <P> </P>

- used to format and present textual content on web pages, such as articles, blog posts, and informational content.
- Defined using the and tags

```
<p>
This is a paragraph of text.
It can contain multiple sentences and line breaks.
</p>

<p>This is the first paragraph.</p>
<p>This is the second paragraph.</p>
<p>This is the third paragraph.</p>
```


-
 : is used to insert a line break or line break element within the content
-
 element does not have a closing tag

```
<p>
   This is some text. <br> This text is on a new line.
</p>
```

<pre> Preformatted Text

- The tag will be displayed exactly as it is written in the HTML source code, including spaces, line breaks, and indentation.

```
<pre>
  This is an example of preformatted text.
  Here are some spaces:     and some tabs:   \t
  This text will be displayed exactly as written,
  including line breaks.
</pre>
```

Text Formatting Tags

```
<B> Bold Face </B>
<I> Italics </I>
<U> Underline </U>
<BR> Next Line
```

Add Space in HTML :

- ** ;** (Non-Breaking Space)
- ** ;**(En Space)
- ** ;**(Em Space)

<HR>

- <HR> : display a horizontal line (rule) within the content
- <HR> does not use a closing tag

```
<p>This is some text.</p>
<hr>
<p>This is more text below the horizontal rule.</p>
```

- <HR> attributes :

```
<hr size="2" width="50%" noshade align="center">
```

Property	Description	Example
`color`	Sets the text color	`color: #333;`
`font-family`	Specifies the font	`font-family: Arial, sans-serif;`
`font-size`	Sets the font size	`font-size: 16px;`
`font-weight`	Controls the font weight	`font-weight: bold;`
`font-style`	Applies font style (italic, etc.)	`font-style: italic;`
`text-align`	Aligns text horizontally	`text-align: center;`
`text-decoration`	Adds text decoration	`text-decoration: underline;`
`text-transform`	Controls text casing	`text-transform: uppercase;`
`line-height`	Sets the line height	`line-height: 1.5;`
`letter-spacing`	Adjusts character spacing	`letter-spacing: 2px;`
`text-shadow`	Applies shadow to text	`text-shadow: 2px 2px #000;`
`text-overflow`	Specifies text overflow behavior	`text-overflow: ellipsis;`
`white-space`	Specifies how white space is handled	`white-space: nowrap;`
`overflow-wrap`	Controls word wrapping	`overflow-wrap: break-word;`

Key CSS Text Formatting Properties

a element

Is used to define hyperlinks, which allow users to jump from one location to another

- **href** : (Hypertext REFerence) : URL of the page the link goes to. This attribute is what makes an <***a***> element a hyperlink.
- **link text** : The clickable text that is displayed to the user.

<a> examples

```
<a href="https://www.example.com">Visit Example.com</a>
```

Open in a New Tab :

```
<a href="https://www.example.com" target="_blank">Visit
      Example.com</a>
```

Download Link :

```
<a href="/path/to/file" download="filename">Download
      File</a>
```

Email Link :

```
<a href="mailto:example@example.com">Send Email</a>
```

<img> tag

- **src** : path to the image file
- **alt** : provides alternative text for the image. It's important for accessibility and is displayed if the image fails to load.

```
<img src="image.jpg" alt="Description of the image">
```

Ordered list

Ordered list (numbered list) :

```
<ol>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
</ol>
```

Result

1. First item
2. Second item
3. Third item

Unordered list

Unordered list (bulleted list) :

```
<ul>
    <li>First item</li>
    <li>Second item</li>
    <li>Third item</li>
</ul>
```

Result

- First item
- Second item
- Third item

Table

- <**table**> : defines the table.
- <**caption**> : Sets the caption displayed above the table
- <**thead**> : contains the table header row(s).
- <**th**> : defines a header cell in the table.
- <**tbody**> : contains the table body rows.
- <**tr**> : defines a row in the table.
- <**td**> : defines a cell in the table.

```
<table>
  <thead>
    <tr>
      <th>Column 1 Heading</th>
      <th>Column 2 Heading</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Data cell 1, row 1</td>
      <td>Data cell 2, row 1</td>
    </tr>
    <tr>
      <td>Data cell 1, row 2</td>
      <td>Data cell 2, row 2</td>
    </tr>
  </tbody>
</table>
```

Result

Column 1 Heading	Column 2 Heading
Data cell 1, row 1	Data cell 2, row 1
Data cell 1, row 2	Data cell 2, row 2

About FORMs

- Forms are used to collect information from people viewing your web site.
- For example, you can use forms to find out details about your visitors through surveys and feedback, or engage in e-commerce by selling your goods and services to people.
- Forms are defined by the <FORM> </FORM> tags and are made up of different elements to collect data.
- Once the user inputs all of the information, they submit the form by using the "submit" button that you create.
- What happens with the data is a decision you will need to make.
- You can use a script to manage the data, sent the data to database, or even receive data via e-mail.

FORMs content

Forms can contain :

- Text boxes
- Password boxes
- Check boxes
- Radio buttons
- Buttons
- Select lists
- Text areas
- Labels
- Fieldsets
- Legends
- ...

Types of Form elements

Element	Description
<input type="text" >	Allows users to input single-line text.
<input type="password" >	Entered characters, typically used for passwords.
<textarea ></textarea >	Allows users to input multiple lines of text.
<input type="checkbox" >	Allows users to select one or more options from a list.
<input type="radio" >	Allows users to select one option from a list.
<select ></select >with <option ></option >	Presents a dropdown list of options to the user.
<input type="file" >	Allows users to select and upload files from their device.
<input type="submit" >	Submits the form data to the server for processing.
<input type="reset" >	Resets all form fields to their initial values.
<input type="hidden" >	Hidden from the user, used to pass data that should not be visible.
<input type="number" >	Allows users to input numeric values.
<input type="date" >, <input type="time" >, etc.	Allows users to input specific types of data (date, time, etc.).

<form> Tag

- **action** : Specifies where to send the form-data when a form is submitted.
- **method** : Defines the HTTP method for sending data (usually "GET" or "POST").
- **enctype** : Specifies how the form-data should be encoded when submitting it to the server (important for forms with file uploads).
- **autocomplete** : Indicates whether inputs can have their values automatically completed by the browser.
- **novalidate** : Tells the browser not to validate the form before submitting.
- **target** : Defines where to display the response received after submitting the form

METHOD Get

The METHOD attribute specifies the HTTP method to be used when submitting the form data :

GET :

- The default method when submitting form data
- Submitted form data will be visible in the page address field
- The length of a URL is limited (about 3000 characters)
- Never used to send sensitive data ! Better for non-secure data
- Useful for form submissions where a user want to bookmark the result

METHOD POST

- The POST method does not display the submitted form data in the page address field.
- Used for sensitive or personal information.
- Has no size limitations, and can be used to send large amounts of data.

ACTION

- The ACTION attribute defines the action to be performed when the form is submitted.
- Normally, the form data is sent to a web page on the server when the user clicks on the **submit** button.
- In the example below, the form data is sent to a page on the server called "action_page.php". This page contains a server-side script that handles the form data :

```
<form action="action_page.php">
```

Input Elements

- **type** : Specifies the type of input (e.g., text, password, submit).
- **name** : Defines the name of the input.
- **id** : provides a unique identifier for the input element
- **value** : Sets the default value of the input.
- **placeholder** : Provides a hint to the user about what to enter in the input.
- **required** : an input field must be filled out before submitting the form.
- **disabled** : Disables the input field.
- **readonly** : Makes the input field read-only.
- **autocomplete** : Specifies if the browser should autocomplete the form
- **autofocus** : Automatically focuses the input when the page loads.
- **min** and **max** : Define the minimum and maximum values for input types like "number" or "date".
- **maxlength** and **minlength** : maximum and minimum lengths of the input.
- **pattern** : Defines a regular expression against which the input's value will be checked.

Other Attributes

- **multiple** (for <input type="file"> and <select>) : Allows multiple file selections or multiple option selections.
- **selected** (for <option> in <select>) : Specifies that an option should be pre-selected when the page loads.
- **checked** (for <input type="checkbox"> and <input type="radio"> Indicates that a checkbox or radio button is selected by default.

```
<form action="/submit-form" method="post">
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username"
         placeholder="Enter your username" required><br><br>
  <label for="password">Password:</label><br>
  <input type="password" id="password" name="password"
         required><br><br>
  <label for="email">Email:</label><br>
  <input type="email" id="email" name="email" required><br><br>

  <label for="birthdate">Birthdate:</label><br>
  <input type="date" id="birthdate" name="birthdate"
         required><br><br>

  <label for="country">Country:</label><br>
  <select id="country" name="country">
    <option value="algeria">Algeria</option>
    <option value="canada">Canada</option>
    <option value="uk">UK</option>
  </select><br><br>
```

```
<label for="gender">Gender:</label><br>
<input type="radio" id="male" name="gender" value="male">
<label for="male">Male</label>
<input type="radio" id="female" name="gender"
       value="female">
<label for="female">Female</label><br><br>

<label for="color">Favorite Color:</label><br>
<input type="color" id="color" name="color"><br><br>

<label for="avatar">Profile Picture:</label><br>
<input type="file" id="avatar" name="avatar"><br><br>

<label for="bio">Bio:</label><br>
<textarea id="bio" name="bio" rows="4"
       cols="50"></textarea><br><br>

<input type="submit" value="Submit">
</form>
```

legend

- Resides within the <**fieldset**> element
- Acts as a descriptive title for the fieldset
- Improves accessibility for screen readers and other assistive technologies
- Enhances clarity and navigation for users

Fieldsets

- Fieldsets are a powerful tool for structuring and organizing forms in HTML
- They help group related input elements together,
- <**fieldset**> opening tag
- Optional <legend> element for the title
- Content : form controls, labels, and other elements
- </**fieldset**> closing tag

```
<form>
  <fieldset>
   <legend>Personal Information</legend>
      <label for="fname">First Name:</label>
      <input type="text" id="fname" name="fname" /><br>
      <label for="lname">Last Name:</label>
      <input type="text" id="lname" name="lname" /><br>
   </fieldset>
   <fieldset>
     <legend>Contact Information</legend>
       <label for="email">Email:</label>
       <input type="email" id="email" name="email" /><br>
       <label for="phone">Phone:</label>
       <input type="tel" id="phone" name="phone" /><br>
       <label for="address">Address:</label>
       <textarea id="address" name="address"></textarea><br>
     </fieldset>

   <input type="submit" value="Submit" />
</form>
```

div

The element is one of the most fundamental building blocks in HTML, serving as a generic container for any type of content

- Groups related content together semantically, even if it has no inherent meaning itself.
- Creates visual sections on a webpage for styling and layout purposes.
- Acts as a placeholder for applying CSS styles to specific sections.

div : Common Use Cases

- Creating sections like headers, footers, main content, sidebars.
- Grouping related form elements.
- Building layouts using CSS grid or flexbox.
- Highlighting specific content with unique styles

```
<div class="container">
  <h2>This is a heading</h2>
  <p>
     This is some content wrapped in a `<div>` element with
     the class "container".
  </p>
</div>
```

Article and Section

article :

- Represents a self-contained, independent piece of content
- Provides semantic meaning for both users and search engines
- Improves accessibility by helping screen readers identify and announce distinct content units

section :

- Defines a thematic section within a document
- Used to organize and structure content within an article or larger page
- Offers a way to visually and semantically divide content for better understanding

An <article> can contain multiple <section>

```
<article class="blog-post">

 <header>
   <h1>This is an Article Title</h1>
 </header>

 <section class="introduction">
   <p>This is the introduction of the article, providing a brief
           overview.</p>
 </section>

 <section class="main-body">
    <h3>Headline 1</h3>
    <p>This is the main content of the article, with detailed information and
           explanations.</p>
   <h3>Headline 2</h3>
    <p>Here's another section with additional information related to the main
           topic.</p>
  </section>

  <section class="conclusion">
   <p>This is the conclusion of the article, summarizing the key points and
           leaving a final thought.</p>
  </section>

</article>
```

Questions ?