L2: Browser/HTML/Accessibility

Web Engineering
188.951 2VU SS20

Jürgen Cito
L2: Browser/HTML/Accessibility

- Browser Overview
- Semantic HTML
- Accessibility for the Web
  Web Accessibility Initiative (WAI)
Learning Goals

• Understand the browser as a model for frontends and its limitations
• Create basic documents for the web with semantically correct HTML
• Explain how forms in HTML documents translate to HTTP requests
• Understand the relation of accessibility and semantic markup structure
Browser: A model for frontend applications

Powerful abstractions

- Powerful Declarative Language for defining user interfaces
  - Rapid prototyping for interfaces
  - Ability to include various forms of media (images, video, audio)
  - Cross-platform frontends (across devices, operating systems)
  - Adapts to different window sizes
- Shared understanding of interaction models
  - URLs as standard entry-point for resources
  - Links to internal and external resources
  - Back/forward/refresh buttons ubiquitously understood
- Underlying complex processes abstracted (rendering, networking, threading, etc.)

Limitations

- Interface possibilities limited (compared to native interfaces)
  - Document-centric structure may limit (or make it harder) to implement certain interaction models
  - Limited communication capabilities
  - Limited access to user machine
  - Mostly pull communication
    - Limited set of protocols (HTTP(S))
    - Restricted local access for security and privacy reasons (cookies, local storage)
    - Exception: WebSockets (nonetheless, most web communication is pull-based)
Browser Internals - Overview

- **Standard UI elements in the browser** (address bar, menus, back button, etc.)
- **Translates between UI and rendering engine**
- **Responsible for visually displaying requested resource in main window**
- **Handles incoming and outgoing network requests (HTTP)**
- **Parses and executes JavaScript in the browser process**
- **Renders basic widgets (forms) and translates to underlying operating system UI rendering methods**
- **Local persistent data** (cookies, local storage, etc.)

Source: https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/
Browser Internals - Rendering Engine

(1) Construct the Document Object Model (DOM) from parsing the HTML document

(2) Construct Render Tree from styling information (CSS) together with visual instructions in HTML

Gradual process:
One process will not wait for the previous one to finish completely, but rather the rendering engine will try to display contents as soon as possible

(3) The Layout Process is a recursive process that attaches coordinates to each node

(3) In Painting each node from the render tree will be painted using the UI backend

Continuous Resource Fetching:
The document can contain links to external resources (stylesheets, images, scripts, etc.) that are continuously fetched in this gradual process

Source: https://dbaron.org/talks/2008-11-12-faster-html-and-css/slide-6.xhtml
(Semantic) HTML

The structure of web documents and applications
HTML Overview

• Hyper Text Markup Language
• Standardized by the W3C
• Describes structure and content of a document
• Human and non-human users
  • Browser parses the content and presents it to the end user
  • Crawler indexes the parsed content (machine-readability)

```html
<tagname attribute="value">content</tagname>
```
HTML5 Overview

• Goal
  • Web Documents → Web Applications
  • Updating the HTML specification
  • Consider low-powered devices (e.g., smartphones)
  • Reduce the need for external plug-ins (e.g., Flash)
  • More built-in markup to replace scripting

• Features
  • One language
  • Form validation
  • Web storage
  • Offline support
  • Multimedia support
  • …
HTML Structure

Basic Structure

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <meta name="author" content="WE"/>
    <title>Title</title>
  </head>
  <body>
    <h1>First order header</h1>
    <p>Paragraph content</p>
  </body>
</html>
```
HTML Structure - Document Type

**XML declaration**
- Only necessary for XHTML
- Version of XML being used

```
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.1//EN"
   http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<?xml version="1.0" encoding="UTF-8"?>
```

**Document Type**
- Distinguishes versions
- "Quirks mode"
  Layout mimics non-standard behavior
  (i.e., to support web sites built before widespread adoption of web standards)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
 http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

**Document element**
- Single root element
- For XHTML add namespace

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xml:lang="de" lang="de">
  ...
</html>
```
HTML Structure - Global Structure

Head with meta data
- Title
- Data from meta element
  - Author, Keywords, Date, …
- Linking to other resources
  - CSS, JavaScript, …

Body containing content

Global attributes (excerpt)
- id: Unique identifier
- class: Assigned class for CSS
- title: Description of an element
- style: Element specific layout information
- data-*: Invisible attached data
  (Custom data accessible through JavaScript)

Example Code:
```html
<head>
  <meta name="author" content="JC"/>
  <title>Title</title>
</head>

<link rel="stylesheet" type="text/css" href="/path/to/my/style.css">

<div
  id="someID"
  class="someClass"
  title="Text displayed as tooltip"
  lang="en"
  data-loaded="false"
  style="display:block;"
>
  Content
</div>
```
HTML Structure - Element Semantics

Syntax

<tagname attribute="value">content</tagname>

Semantics

- Not given by standard visual representation!
  - `<h1>` is a first order header != the thickest printed text
  - `<b>` prints text bold != `<em>` emphasizes the text
  - `<table>` represents tabular data != layout mechanism

Why use syntactically and semantically correct elements?

- Browser compatibility, accessibility (later)
- Easier processing for tools, e.g., transformations, indexing for search engines
- More efficient browsing (no interpretation of wrong HTML necessary)

Shift towards better use of semantics enables

- Ability for better interpretation for accessibility
- Easier code understanding and maintainability
HTML Structure - Content Structure

- `<header>`
  - defines header of document or section
- `<nav>`
  - defines navigation region of page or section
- `<main>`
  - main content of the page
- `<section>`
  - thematic grouping of content
- `<h1>-<h6>`
  - Heading from most to least important
  - Reflects structural depth, e.g. in sections.
  - Exactly one `<h1>` per page
- `<article>`
  - specifies complete, self-contained content
- `<aside>`
  - defines content aside from main content
- `<footer>`
  - defines footer of document or section

Many of these elements can be nested and it's not always straightforward which element should be used!
HTML - Block vs. Inline

**Block elements** take up full width and force a line break before and after
<h1>, <p>, <div>, <section>, ...

**Inline elements** take up as much width as necessary
<span>, <a>, <em>, <strong>, ...
HTML - Generic Elements

<div>
Generic block element
</div>

<span>
Generic inline element
</span>

Use these when no other element with more appropriate semantics is left.
HTML - Selected Grouping Elements

• Paragraphs
• Contact Information
• Pre-formatted content
• Figure (self-contained flow content) with caption
• Blockquote
• Cite: Citation

• Lists with list Elements
  • Unordered Lists (ul)
  • Ordered Lists (ol)

```xml
<p>Lorem ipsum dolorem sit amet…</p>
Contact: <address>Name: Jane Doe</address>
<pre>public static void main(){}</pre>
<figure>
  <blockquote>Any idiot can put up a website.</blockquote>
  <footer>
    <figcaption>Some quote</figcaption>
    <cite>Patricia Briggs</cite>
  </footer>
</figure>
<ul>
  <li>Some element</li>
  <li>Another element</li>
</ul>
<ol>
  <li>First element</li>
  <li>Second element</li>
</ol>
```
Links and Anchors

- Links refer to (other) documents or elements within (other) documents
- Anchors define bookmarks within a document, which can be used by links

```html
<a href="http://www.w3.org/html" target="_blank">HTML Standard</a>
<a href="index.html#registration" target="_parent">Registration</a>
<a href="#timetable" target="_self">Timetable/Lectures</a>
```
HTML - Basic Forms

- **Buttons**
  ```html```
  <input type="submit" value="Submit" />
  ```html```

- **Checkboxes**
  ```html```
  <input type="checkbox" name="..." value="..." />
  ```html```

- **Radio Buttons**
  ```html```
  <input type="radio" name="..." value="..." />
  ```html```

- **Menus**
  ```html```
  <select>
    <option value="EWA">EWA</option>
    ... 
  </select>
  ```html```

- **Text Input**
  ```html```
  <input type="text" />
  <input type="password" />
  <textarea type="text" rows="2" cols="50">
  </textarea>
  ```html```

- **File Select**
  ```html```
  <input type="file" />
  ```html```

- **Hidden Controls**
  ```html```
  <input type="hidden" name="..." value="..." />
  ```html```

*Only for storing values between different sites*

*Not for sensitive data!*

---

*TU Informatics*
Fieldset
Grouping for part of an HTML form

Legend
Caption for fieldset

Label
Caption for elements in an HTML form

```html
<fieldset>
  <legend>Choose your favourite course</legend>
  <p>
    <input type="radio" id="we" name="course">
    <label for="we">Web Engineering</label>
  </p>
  <p>
    <input type="radio" id="ssd" name="course">
    <label for="ssd">Semi-structured Data</label>
  </p>
</fieldset>
```
HTML5 - Newer Form Elements

New form elements
- `<datalist>` defines a list of pre-defined options
- `<keygen>` specifies a key-pair generator
- `<output>` represents the result of a calculation

New form attributes
- `autocomplete`: use previous values
- `novalidate`: disable form validation

New input types
- color, date, number, time, url, ...

New input attributes (excerpt)
- `pattern`: regexp for allowed values
- `required`: field must not be empty
- `placeholder`: suggest value for field
What happens when I send a form?

<!DOCTYPE html>
<html>
<head>
  <title>A HTML5 Document</title>
</head>
<body>
  <p>This is a sample <a href="...">HTML 5</a> document.</p>
  <form action="/processForm" method="post">
    <p>
      <label for="userName">Your name:</label>
      <input type="text" id="userName" name="userName" />
    </p>
    <p><input type="submit" value="Submit the form" name="action" /></p>
  </form>
</body>
</html>

HTML forms only allow GET and POST requests
What happens when I send a form?

1. submit.html

2. Hi Jürgen,
   thanks for using our sample web application...

3. POST Request
   Message Body
   - userName=“Jürgen”

4. Web Server
   - Program
   - userName=“Jürgen”
   - response.html

5. response.html

6. response.html

7. Client

HTTP

Server
Accessibility

Designing and building web experiences for universal access
Universal Accessibility

Who are we designing for?
Everyone (as much as possible)
• People with physical disabilities
• People with mental disabilities
• People with (temporary) injuries
• Non-native speakers (internationalization/localization)

Who is our references point?
Ourselves
• It’s hard to design and build for other people (intrinsically)

Common Disabilities
Vision Problems
• Blindness, low-vision, color-blindness, etc.
Hearing Problems
• Deafness, high-frequency loss, etc.
Movement Problems
• Paraplegic, wrist problems, broken arms/hands, etc.
Reading Difficulty
• Dyslexia, illiteracy, non-native speakers

Standardized Guidelines can help design and build universally accessible web experiences
Web Accessibility Guidelines

Components
Authoring Tool Accessibility Guidelines (ATAG), 1.0
Guidelines for Web authoring tools (software that creates web sites)

User Agent Accessibility Guidelines (UAAG), 1.0
Guidelines for user agents (web browsers, media players, etc.)

Web Content Accessibility Guidelines (WCAG), 2.0
Guidelines for information in a web site (text images, forms, etc.)

Accessible Rich Internet Applications (WAI-ARIA)
How to develop dynamic web content and web applications

Independent User Interface (Indie UI)
How user actions are communicated to web applications

Evaluation and Report Language (EARL)
A machine-readable language for expressing test results

Focus in this lecture on WCAG
Web Accessibility Guidelines - WCAG

Web Content Accessibility Guidelines
- Web Content Accessibility Guidelines 1.0 (WCAG 1.0)
  - W3C Recommendation since 05 May 1999
  - 14 Guidelines

- Web Content Accessibility Guidelines 2.0 (WCAG 2.0)
  - W3C Recommendation since 11 December 2008
  - Four Principles of Accessibility
  - 12 Guidelines beneath these principles
  - Three Conformance Levels
    - Conformance Level A (Priority 1 checkpoints)
    - Conformance Level Double-A (Priority 1 and 2 checkpoints)
    - Conformance Level Triple-A (Priority 1, 2, and 3 checkpoints)
  - Main documents
    - W3C standard, http://www.w3.org/TR/WCAG20/
    - How to Meet WCAG 2.0, http://www.w3.org/WAI/WCAG20/quickref/
    - Understanding, http://www.w3.org/TR/UNDERSTANDING-WCAG20/
    - Techniques, http://www.w3.org/TR/WCAG20-TECHS/

Accessibility Checker
- FAE: http://fae20.cita.illinois.edu
- AChecker: http://achecker.ca/checker/
- WAVE: http://wave.webaim.org/

Screen reader
- Orca (for Linux): http://live.gnome.org/Orca
- Jaws (for Windows only): http://www.freedomsci.de/serv01.htm
- Webformator (for Windows only): http://www.webformator.com/
Web Accessibility - Principles

Perceivable

Information and user interface components must be presentable to users in ways they can perceive

- **Text Alternatives**
  - Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language

- **Time-Based Media**
  - Provide alternatives for time-based media

- **Adaptable**
  - Create content that can be presented in different ways (for example simpler layout) without losing information or structure

- **Distinguishable**
  - Make it easier for users to see and hear content including separating foreground from background

Is there any content on the site that people with any (audio-visual) impairment could not perceive?

alt Attribute

```html
<img alt="Man walking dog down the street" …>
```

Don’t rely only on color. Use proper semantics (`<em>` to emphasize)
Web Accessibility - Principles

Operable

User interface components and navigation must be operable

- Keyboard Accessible
  - Make all functionality available from a keyboard

- Enough Time
  - Provide users with disabilities enough time to read and use content

- Seizures
  - Do not design content in a way that is known to cause seizures

- Navigable
  - Provide ways to help users with disabilities navigate, find content and determine where they are

Can all functions be performed with a keyboard? Is completing tasks easy?

Use Proxies (Lynx, Fangs) to determine issues with keyboard/navigation accessibility

Ensure user control of time-sensitive content changes
(Ensure that moving, blinking, scrolling objects may be stopped by the user)
Web Accessibility - Principles
Understandable

Information and the operation of user interface must be understandable

- **Readable**
  - Make text content readable and understandable

- **Predictable**
  - Make web documents and apps appear and operate in predictable ways

- **Input Assistance**
  - Help users avoid and correct mistakes
Web Accessibility - Principles

Robust

Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies

- Compatible
  - Maximize compatibility with current and future user agents, including assistive technologies

Do I maximize the use of semantic markup to support different technologies? Is my app backwards-compatible (or has some form of graceful degradation)?

Test on different devices, operating systems, and modes of operations

Design for performance
- Not everyone has 3G/LTE
- Not everyone has powerful devices
Web Accessibility - Examples

- Using CSS to hide (a portion of the link) text (C7)

```
.accessibility {
  position: absolute;
  left: -10000px; /* off screen */
  width: 1px;
  height: 1px;
  overflow: hidden;
}
```

- Supplementing link text with the title attribute (H33)

```
<a href="http://example.com/WORLD/africa/kenya.elephants.ap/index.html"
  title="Read more about failed elephant evacuation">
  Evacuation Crumbles Under Jumbo load
</a>
```

- Using alt attributes on img elements (H37)

```
<img src="ipa-logo.jpg" alt="Interactive Programming and Analysis Lab" />
```

- Using semantic markup to mark emphasized or special text (H49)

```
What she <em>Evacuation</em> meant to say was...
This is an excerpt from <cite>The story of my life</cite>: ...
```
Web Accessibility - Bad Practice Form Scenario

```html
...<form action="/login">
  <p>
    <font color="#29672D"><b>Login</b> form
    <table>
      <td>User name<br>Password</td>
      <td><input name="1"><br>
          <input name="2">
          <input type="submit" value="submit">
    </table>
    <span>not registered? click <a href="register">here</a></span>
  </p>
</form>...
```
Web Accessibility - Bad Practice Form Scenario

Screen Reader Linearization

Imagine 15 fields!
Just visual (no logical) binding of label and field
→ Where do I fill in what?

Color blindness and/or myopia

What does here mean?
When jumping through links (tab key) the link title is just „here“.  

No semantics of „Login form“
Just visually bigger, but no real heading

First indication: Is not valid HTML