# Lecture notes

## Lecture 1 - History n stuff

### Web Main Components

- Client with Web Browser
- Internet
- Web Server

### Internet

- Transmission Control Protocol/Internet Protocol (TCP/IP)
  - Same for all computers
  - Allows for data exchange between computers
  - Tolerates unreliable sub-networks
  - Guarantees proper data transmission, since the phisical network can't
  - Layers
    - Application (HTTP, telnet, ftp, email, VoIP)
      - Ability to access services of other layers
      - Defines protocols for data exchange
    - Transport (TCP, UDP)
      - Responsible for end-to-end message delivery
    - Internet/Network (IP)
      - Routing messages
      - All routers run on the IP protocol
    - Physical (Ethernet, WiFi, ATM, x.25, Frame Relay)
      - Translates message into physical representation
      - Puts messages on the wire/wireless network/fiber-optic wire
- Addressing Schemes (where to go)
  - Unique address required
  - Internet Protocol (IP) address
  - Ex.: 205.46.117.104 (each between 0 and 255)
- Domain names (human understandable)
  - DNS: The phonebook of the internet
  - Naming system
  - Associates domain names (human readable) with IP addresses
  - Between client and server
  - 1985
- Routing Traffic Across the internet (how to go)
- The World Wide Web (WWW)
  - Geneva, 1989
  - System of interconnected hypertext documents
  - A distributed hypertext system
  - Set of common communication protocols
  - Multi-platform
  - Hypertext: method of organizing information
    - Gives the reader control over the order in which the information is presented
- HTML
  - Markup: structure
  - Publishing language of the World Wide Web
  - HTML 5 newest
  - `<body>` Everything inside is shown in browser window
  - `<head>`
    - Information about the page
    - `<title>`, charset, etc.
  - `<title>`
    - Shown in the top of the browser
  - Attributes
    - Additional information to elements (name=value)

## Lecture 2 - Tables, URLs, etc.

### HTML structure

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Title of my webpage</title>
  </head>
  <body>
    <p>Hello there</p>
  </body>
</html>
```

- `<!DOCTYPE html>`: Document Type Definition (DTD)
- `<meta charset="utf-8">`
  - Character encoding
  - Can also have:
    - `<meta name="description" ... >`
    - `<meta name="keywords" ... >`
    - `<meta name="author" ... >`
    - Etc.
- Tags can be one-sided or two-sided
- `<!- comment -->`: The comment tag
- HTML treats white-space as a single blank space
  - Exception: the `<pre>` tag
- Inline elements
  - Displayed without starting a new line
  - `<b>`, `<td>`, `<a>`, `<img>`, etc.
- Block elements
  - Start and ends with new line
  - Ex.: `<h1>`, `<p>`, `<ul>`, `<table>`, etc.
- Grouping tags
  - `<span>` (inline)
  - `<div>` (block)
- Semantic Markup (logical elements)
  - `<q>`/`<blockquote>`
  - `<abbr>`
  - `<address>`
- Lists
  - ordered `<ol>`
  - unordered `<ul>`
  - definition `<dl>`
    - `<dt>` for the term
    - `<dd>` for the definition of the term (below, not inside)
  - Can be nested
  - Styled list: `<ol style="list-style-type: upper-roman>`
- Links
  - `<a href="http://exmaple.com>`
  - When linking to page within the same site, you can use a relative URL
  - Can link to elements with id: http://www.sundance.org/#jumbotron
- Images
  - _src_: path to image
  - _alt_: text description, for accessibility and indexing purposes
  - Specifying height/width nice so that the browser can render the rest of the page while leaving room
    - Either as attributes or CSS
    - Measure images in pixels
- Figure captions
  ```html
  <figure>
    <img src=".." alt="..." />
    <br />
    <figcaption>some text</figcaption>
  </figure>
  ```
- JPEGs (Joint Photographic Experts Group) or GIFs (Graphical Interchange Format) best
- Tables
  ```html
  <table>
    <tr>
      <th scope="col">Saturay</th>
      <th scope="col">Sunday</th>
    </tr>
    <tr>
      <td>1</td>
      <td>1</td>
    </tr>
    <tr>
      <th scope="row">Total sales</th>
      <td>1</td>
      <td>1</td>
    </tr>
  </table>
  ```
  - Use colspan="2" on row to span two columns
  - Use rowspan="2" on row to span two rows
  - Tags for accessibility, indexing and desgn purposes:
    - `<thead>`
    - `<tbody>`
    - `<tfoot>`
- Site maps
  - Website guide
    Should not require further user interaction
  - When to use
    - Large site
    - Large archive, isolated pages, etc.
    - New site and few external links
    - Rich media content
  - When not to use
    - Small site
    - Comprehensive linking
    - Few media files

### URLs

- Protocol
  - http://www.example.com/path/to/myfile.html?key1=value1&key2=value2…
  - http:// indicates protocol (http default)
  - www.exmaple.com is the domain name
  - /path/to/myfile.html is path to resource on server
  - ?key1=value1&key2=value2 are extra query parameters (key/value pairs separated with &)
- GET method
  - Request data from a specified resource
  - Query strings
  - Can be cached
  - Remain in browser history
  - Can be bookmarked
  - Can be distributed and shared
  - Have lenght restrictions (max URL length)
  - Can be hacked
- POST requests
  - Sent as a separate message
  - Never cached
  - Used for sensitive data
  - No max length
- URL anchor
  - …to/myfile.html?key1=value1&key2=value2<b>#SomewhereInTheDocument</b>
  - Anchor to part of th resource itself

## Lecture 3 - CSS

- XHTML is a stricter version of HTML
  - Recommended for accuracy reasons
- The Problems with HTML
  - No consistency
  - Can omit attributes, etc. (poorly written code)
  - Coding design
    - `<font>`
  - Compatibility
    - Cross-platform compatibility has been degraded (different browsers, etc.)
- Solution: Separate Structure from Appearance
  - CSS (Cascading Style Sheets)
  - CSS1 in December 1996
  - Types
    - External style sheet (included in `<head>`)
    - Internal style sheet
    - Inline style
- How does CSS work?
  - Web browsers assume a default set of values for each property
  - CSS defines an extensive set of presentation properties
  - A CSS style sheet changes that default rendering
  - Use !important to override
  - Rules with same origin conflict --> more specific applies
  - Equally specific -> last is chosen
  - Style prioritization: inline > embedded > external
- Inheritence
  - `<h1>The headline <em>is</em> important!</h1>`
  - If h1 has color blue, em gets color blue
  - Only some styles are inherites (font-family, text-alignment, etc.)
  - Others are not (padding, etc.)
- Multiple declarations
  - Several `h1` declarations will be merged
- Grouping
  - Comma-separated selectors will give all the same declaration
- Selecting elements
  - `*` - universal
  - `h2` - type selector
  - `.element` - class selector
  - `#elementID` - ID selector
  - `p a` - descendant selector
  - `div[style]`, `input[type="text"]` - attribute selector
  - `one > two` - immediate child selector
  - `h2+p` - adjacent sibling selector, same parent
  - `h2~p` - general sibling selector (does not have to be adjacent)
  - `a:visited`, `a:hover` - pseudo-class
    - Selects an element based on a state the element is in
  - `p::first-letter`, `p::before` - pseudo-element
    - Used to add special effects to some selectors
    - Represents elements that are not really part of the rendered HTML
    - Creates new virtual elements
- Specificity Value
  - Style attribute > ID > Class, pseudo-class, attribute > Elements

## Lecture 4 - CSS properties

- Colors
  - Properties
    - color
    - background-color (color or "transparent")
    - background-image (URI)
  - Color Values
    - Named
    - RGB
    - Hex
    - HSL with optional opacity value
- Fonts
  - Properties
    - font-family (specify order of preferenct)
    - font-style (normal, italic or oblique)
    - font-variant (normal, small-caps)
    - font-weight (normal, bold, bolder, lighter)
    - font-size
      - Absolute
        - mm, cm, in, pt, pc
      - Relative
        - Percentage
        - em (1em == 100% == font-size of parent element)
        - rem
      - Can use pixels or keywords
- Text
  - text-indent (amount of indentation using absolute length or percentage)
  - text-align (left, center, right, justify)
  - text-decoration (non, underline, overline, line-through)
  - etc.
- The Box Model
  - Margin
  - Border
  - Padding
  - Content
- Overflowing Content
  - What to do if the content is largen than the box itself?
  - hidden, overflow, scroll, etc.
- Borders
  - border-width: 1px 4px 12px 4px (clockwise starting from the top)
  - Border styles:
    - solid, dotted, dashed, double, groove, ridge, inset, outset
    - Ex.: border-style: dotted solid double (top inline bottom)
- Padding
  - Space between the content and its border
- Margin
  - Gap between boxes
- Change inline/block
  - `display: inline` - turn a block element into an inline one
- Hiding
  - `visibility: hidden` hides the element but it leaves a space where it would have been
- Border Images
  1. The url of the image
  2. Where to slice the image
  3. What to do with the straight edges
     1. Stretch
     2. Round
- Box Shadows
  - Horizontal offset
  - Vertical offset
  - Blur distance
  - Spread of shadow

  - `p.one { box-shadow: -5px -5px #777777; }`
- Position
  - position: static
    - Default
    - Each block element sits on top of the next one
  - position: relative
    - Moves an element in relation to where it would have been in normal flow
    - top, left, right, bottom, etc.
  - position: absolute
    - Box is taken out of normal flow
  - position: fixed
    - Position in relation to the browser window
- Overlapping
  - relative, fixed, absolute --> boxes can overlap
  - Later element sits on top of older
  - Specify order with z-index
- Floating elements
  - Allows you to take an element in normal flow and place it as far to the left/right as it goes
- Table properties
  - width
  - padding
  - letter-spacing, font-size
  - border-top/bottom
  - :hover
- Display styles
  - none
  - block
  - inline
  - inline-block
  - inline-table
  - inherit
  - list-item
  - run-in
  - table
  - table-caption
  - table-cell
  - table-column
  - table-column-group
  - table-footer-group
  - table-header-group
  - table-row
  - table-row-group

## Lecture 5 - Flex and Grid

### Flexbox

- Properties for the Parent (Flex Container)
  - `display: flex` enables flex context for all direct children
  - `flex-direction: row (default) | row-reverse | column | column-reverse;`
  - `flex-wrap: nowrap (default) | wrap | wrap-reverse;`
  - `flex-flow: column wrap;` - shorthand for `flex-direction` and `flex-wrap`
  - `justify-content: flex-start (default) | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;` - alignment along the main axis
  - `align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... safe | unsafe;` - alignment along the cross axis
  - `align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;`
    - Align lines within
    - Only takes effect on multi-line flexible containers (flex-wrap)
  - gap, row-gap, column-gap controls the space between flex items
- Properties for the Children (Flex Items)
  - order (default is 0)
  - flex-grow
    - Ability for a flex item to grow if necessary
    - Unitless proportion value
  - flex-shrink (default is 1)
    - Ability for a flex item to shrink if necessary
    - Higher -> more shrink (?)
  - flex-basis
    - Default size of an element before the remaining space is distributed
    - Length or keyword (auto/content)
    - auto -> "look at my width and height property"
    - content -> "size it based on the item's content
  - flex
    - Shorthand for flex-grow, flex-shrink, and -flex-basis combined
    - Default: 0 1 auto
  - `align-self: auto | flex-start | flex-end | center | baseline | stretch;`
    - Overrides default alignment for individaul flex items

### Grid

- Properties for the Parent (Grid Container)

  - display: grid | inline-grid;
    - grid – generates a block-level grid
    - inline-grid – generates an inline-level grid
  - grid-template-columns, grid-template-rows
    ```css
    .container {
      grid-template-columns: ... ...;
      /* e.g.
          1fr 1fr
          minmax(10px, 1fr) 3fr
          repeat(5, 1fr)
          50px auto 100px 1fr
      */
      grid-template-rows: ... ...;
      /* e.g.
          min-content 1fr min-content
          100px 1fr max-content
      */
    }
    ```
    - Defines the columns and rows of the grid with a space-separated list of values
    - Explicitt namining of grid lines:
      ```css
      .container {
        grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
        grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];
      }
      ```
    - `fr` sets the size of a track as a fraction of the free spaceof the grid container
      ```css
      .container {
        grid-template-columns: 1fr 1fr 1fr;
      }
      ```
      - Caldulates free space after any non-flexible items
  - grid-template-areas

    - Defines a grid template by referencing the names of the grid areas which are spacified with the `grid-area` property
    - Values:

      - `<grid-area-name>` – the name of a grid area specified with grid-area
      - . – a period signifies an empty grid cell
      - none – no grid areas are defined

      ```css
      .item-a {
        grid-area: header;
      }
      .item-b {
        grid-area: main;
      }
      .item-c {
        grid-area: sidebar;
      }
      .item-d {
        grid-area: footer;
      }

      .container {
        display: grid;
        grid-template-columns: 50px 50px 50px 50px;
        grid-template-rows: auto;
        grid-template-areas:
          'header header header header'
          'main main . sidebar'
          'footer footer footer footer';
      }
      ```

  - grid-template
    - A shorthand for setting grid-template-rows, grid-template-columns, and grid-template-areas in a single declaration.
    ```css
    .container {
      grid-template-rows: [row1-start] 25px [row1-end row2-start] 25px [row2-end];
      grid-template-columns: auto 50px auto;
      grid-template-areas:
        'header header header'
        'footer footer footer';
    }
    ```

- Properties for the Children (Grid Items)
  - grid-column-start, grid-column-end, grid-row-start, grid-row-end
    - Determines a grid item’s location within the grid by referring to specific grid lines
    - Values:
      - `<line>` – can be a number to refer to a numbered grid line, or a name to refer to a named grid line
      - span `<number>` – the item will span across the provided number of grid tracks
      - span `<name>` – the item will span across until it hits the next line with the provided name
      - auto – indicates auto-placement, an automatic span, or a default span of one
    ```css
    .item {
      grid-column-start: <number> | <name> | span <number> | span <name> | auto;
      grid-column-end: <number> | <name> | span <number> | span <name> | auto;
      grid-row-start: <number> | <name> | span <number> | span <name> | auto;
      grid-row-end: <number> | <name> | span <number> | span <name> | auto;
    }
    ```
    ```css
    .item-b {
      grid-column-start: 1;
      grid-column-end: span col4-start;
      grid-row-start: 2;
      grid-row-end: span 2;
    }
    ```
  - grid-column, grid-row
    - Shorthand for grid-column-start + grid-column-end, and grid-row-start + grid-row-end, respectively
    ```css
    .item-c {
      grid-column: 3 / span 2;
      grid-row: third-line / 4;
    }
    ```
  - grid-area
    - Gives an item a name so that it can be referenced by a template created with the grid-template-areas property
    ```css
    .item {
      grid-area: <name> | <row-start> / <column-start> / <row-end> /
        <column-end>;
    }
    ```

## Lecture 6 - Multimedia

- Image Formats
  - GIS (Graphic Interchange Format) GIF87 and GIF89A
    - Patente LZW (Lempel-Ziv-Welch) compression and supports:
      - Multiple images in a single file
      - Interlacing (GIF89A)
      - Transparency (GIF89A)
      - Animation (GIF89A)
    - 8 bit graphics
    - Maximum 256 colors
    - Used for icons and images with solid regions of color
    - Non-interlaced Graphic
      - GIF appears as it is slowly retrieved by the Web browser
      - Top-down
    - Interlaced Graphic
      - Starts out blurry
      - Gradually comes into focus
    - Transparent GIFs
      - Color not displayed when image is viewed
      - Browser displays whatever is on the page background
    - Animated GIFs
      - Composed of several images displayed in rapid succession
      - Larger than static GIF images
      - Early browser versions may not support
  - JPEG (Joint Photographic Experts Group format)
    - Designed for storing full-color or greyscale images of real-world scenes
    - Smooth variations in color represented more faithfully
    - 24-bit color
    - Default lossy compression
    - Compression ratio of ~10:1 or 20:1
  - PNG (Portable Network Graphics)
    - Improved version of GIF
    - Uses a free algorithm for compression ("PING")
    - Greater color depth
    - 8-bit transparency
    - Lossless compression (like GIF)
    - Millions of colors
    - Larger than JPGs and GIFs
  - SVG (Scalable Vector Graphics)
    - Vector graphics format
    - Scaling independent of resolution
    - Editable through code
    - Markup enables CSS
    - Wide support
- Creating images
  1. Save images in the right format
  2. Save images at the right size
  3. Mesure images in pixels
- Figure captions
  - `<figure>` can contain and associate multiple images and their (common) caption
  ```html
  <figure>
    <img src”images/image1.jpg” alt=“the first image” />
    <figcaption>This is the caption of the first image</figcaption>
  </figure>
  ```
- Background image
  - `background-image: url(url);`
  - < 20kB
- Video
  - Reducing the frame rate reduces the size of your file
  - Formats in HTML5 standard
    - MP4
    - WebM
    - Ogg
  - New `<video>` element (and/or flash video)
    - preload (none, auto, metadata)
    - src
    - poster (loading image)
    - width, height
    - controls
    - Multiple sources/formats with `<source>`
    ```html
    <video poster="images/puppy.jpg" width="400“ height="320"
      preload controls loop>
      <source src="video/puppy.mp4“ type='video/
        mp4;codecs="avc1.42E01E, mp4a.40.2"' />
    ```

      <source src="video/puppy.webm" type='video/
      webm;codecs="vp8, vorbis"' />
      <p>A video of a puppy playing in the snow</p>
    </video>

- Audio
  - Sample Resolution
    - The precision in measuring the amplitude for each sample (8, 16, 32 bit)
  - Source File Formats
    - WAV (Waveform Audio File Format)
      - Uncompressed
      - High quality, large file size
    - MP3
      - Compresses audio files, min source quality impact
    - WMA (Windows Media Audio) uses Microsoft proriertary compression algorithm
    - Ogg
    - MIDI (Musical Instrument Digital Interface)
      - Limited to instrumental music and not speech/general sounds
        Only MP3, WAV, and Ogg are supported by HTML5
  - Non-streaming media must be completely downloaded before being played
  - Streaming media continuously streams as they are downloaded
  - Adding audio to web pages
    1. Use a host service
    2. Use flash
    3. Use HTML5 with `<audio>` element (recommended)
    ```html
    <audio controls autoplay preload loop>
      <source src="audio/test-audio.ogg" />
      <source src="audio/test-audio.mp3" />
      <p>This browser does not support our audio format.</p>
    </audio>
    ```
  - Embedding
    `<embed src="b9.wma" width="275" height="40" autostart="0"/>`
- Image Maps
  - Different part of an image can contain hyperlinks
    - Hotspots
      `<area shape=“shape” coords=“coordinates” href=“url” alt=“text” />`
      - Shape values
        - rect, circle, polygon
  - Types: server-side image maps and client-side image maps
  - `usemap` references a map element which contain list of hotspots
    ```html
    <img src="“supersite.gif”" usemap="“#map1”" />
    <map name="“map1”">
      <area coords="“0,0,50,50”" href="“topleft.html”" />
      <area coords="“50,50,80,80”" href="“bottomright.html”" />
    </map>
    ```

## Lecture 7 - HTML forms and Responsive Design

### Forms

- How forms work

  - User fills out, press submit, send info to server
    - Name of each for control is sent along with value
      - key/value pairs
    - method - indicates how the data collected by the form should be transmitted to the server using HTTP
      - GET - form data is appended to the URL
      - POST - form data is sent as a separate message
  - Adding text
    - Text input
    - Password Input
    - Text area
    ```html
    <form action="http://www.example.com/login.php">
      <p>
        Username:
        <input type="text" name="username" size="15" maxlength="30" />
      </p>
      <p>
        Password:
        <input type="password" name="password" size="15" maxlength="30" />
      </p>
      <p>What did you think of this gig?</p>
      <textarea name="comments" cols="20" rows="4">
    Enter
    your comments...</textarea
      >
    </form>
    ```
  - Makin choices - Radio buttons - Checkboxes - Drop-down boxes

    ```html
    <form action="http://www.example.com/profile.php">
      <p>Please select your favorite genre:
      <br />
      <input type="radio" name="genre" value="rock" checked="checked" />
      Rock
      <input type="radio" name="genre" value="pop" /> Pop
      <input type="radio" name="genre" value="jazz" /> Jazz
      </p>
      <input type="checkbox" name="service" value="itunes"
      checked="checked" /> iTunes
      <input type="checkbox" name="service" value="lastfm" /> Last.fm
      <input type="checkbox" name="service" value="spotify" /> Spotify
      </p>
      <select name="devices" multiple="multiple">
        <option value="ipod">iPod</option>
        <option value="radio">Radio</option>
        <option value="computer">Computer</option>
      </select>
      <select>
        <optgroup label="Fruit">
          <option value="apple">Apple</option>
          <option value="pear">Pear</option>
        </optgroup>
          <optgroup label="Vegetable">
          <option value="carrot">Carrot</option>
          <option value="turnip">Turnip</option>
        </optgroup>
      </select>
    </form>
    ```

  - Submitting forms
    - Submit buttons
    - Image buttons
  - File upload
  - Creating input boxes
    - type="image"
    - type="password"
    - type="button"
    - type="file"
    - type="hidden"
  - Labelling form controls
    ```html
    <input id="female" type="radio" name="gender" value="f" />
    <label for="female">Female</label>
    ```
  - Grouping form elements
    ```html
    <fieldset>
      <legend>Contact details</legend>
      <label>Email:<br /><input type="text" name="email" /></label><br />
      <label>Mobile:<br /><input type="text" name="mobile" /></label><br />
      <label>Telephone:<br /><input type="text" name="telephone" /></label>
    </fieldset>
    ```
  - Form validation
    - `required="required"`
  - Date input
    - `type="date"`
  - Email, URL, and search
    - `type="url"`
    - `type="email"`
    - `type="search"`

### Responsive Web Design

- Multitude of different devices
- Guidelines for smaller screens
  - Resize your content to fit on the screen
  - Remove non-essential content
  - Increase font size for legibility
  - Make your links and buttons recognizable and clickable
  - Whitespace is still important
- How to adapt for responsiveness
  - Media queries
    ```css
    @media screen and (max-width: 700px) {
      .colorblock {
        background-color: blue;
      }
    }
    ```
    - Media queries comes after the original CSS definition (overrides)
      - Only the rule in question is changed
  - Other responsive fices
    - A div itself can responsive
      - 100%, max-width, etc.
    - display: none
    - Set inner elements to be percentage widths of outer elements
    - Load lower image resolutions for narrower screen widths
    - Arrange things vertically instead of horizontally
      - display: inline or display: inline-block -> display: block
      - inline-block
        - Allows elements to act as block elements in terms of size and containing other elements, but is treated like an inline element for display purposes, allowing it to line up side-by-side with other inline elements

## Lecture 8 - Usability and Inspecting Elements

### Usability

- Strive for Consistency
- Cater to Universal Usability
- Offer informative feedback
- Design dialogs to yield closure
- Prevent errors
- Permit easy reversal of actions
- Support internal locus of control
  - Make sure users always know where on your website they are located
    - Breadcrumbs
- Reduce short term memory
  - Don't make your users remember information from page to page
- No broken links
- Most important information easiest to find
- Provide a hierarchy of information
- Use white space
- Test for usability
- Eye tracking studies
  - Users read in an F-shape
  - First two paragraphs are the most improtant
  - Sub-headings and lists stand out
- People ignore ads (banner blindness)

## Lecture 9 - Intro to JavaScript

- Lightweight programming language
- Works in major browsers
- Instruction == statement
  - Blocks
- Dynamically gneerate content
- React to user events
- Validate form data
- Variables
  - `var` is function scoped
  - `let` works ass you would expect `var` does
    - Block scoped
  - `const` - block scoped and a constant
- `for`/`while` loop

  ```javascript
  for (initialization; condition; final - expression) {}

  for (let i = 0; i < 10; i++) {
    console.log(i);
  }

  let j = 0;
  while (j < 10) {
    j++;
  }
  ```

- Objects

  - Standalone entity with properties
  - Using constructor functions:

    ```javascript
    const person = function (name, age) {
      this.name = name;
      this.age = age;
    };

    const Peder = new Person('Peder', 12);
    ```

## Lecture 10 - DOM introduction

- Document Object Model (DOM)
  - The programming interface for HTML, XML, and SVG documents
  - Tree structure
  - Can access the DOM using the global `document` interface
- Noes have event handlers attached to them

```javascript
const el = document.getElementById('myElement');
el.innerHTML = 'New content';
el.addEventListener('click', () => {});
el.addEventListener('keydown', (e) => {
  e.target.value = e.target.value.toUpperCase();
});
el.addEventListener('change', () => {});
```

- Change event
  - When element is activated (radio buttons, etc.)
  - When user commits the change explicitly (select, datepicker)
  - When the element loses focus after its value was changed, but not commited (textare, input)

## Lecture 11 - DOM Manipulation and Event handling

- DOM selectors
  - getElementById (retrieve a single HTML element by its unique ID attribute)
  - getElementByName (retrieve a collection (array-like) of HTML elements with a specific name attribute)
  - getElementByClassName (retrieve a collection of HTML elements that have a specified class name)
  - getElementByTagName (retrieve a collection of HTML elements with a specific tag name (e.g., "div," "p," "a"))
- DOM traversal
  - childNodes
  - nextSibling
  - parentNode
- Manipulation
  - innerHTML
  - textContent
  - createElement
  - insertBefore

~~~markdown
  - removeChild
- DOM objects
  - `window` - the browser window
  - `document` - the web document displayed in the window
  - `document.body` - the body of the web document
  - `event` - events or actions occuring within the browser window
  - `history` - list of previously visited sites within the browser window
  - `location` - URL of the document currently being displayed
  - `navigator` - the browser itself
  - `screen` - the screen displaying the document
- Element styling
  ```javascript
  const el = document.getElementById('myDiv');
  el.style.display = 'none';
  ```
- Event propagation phases
  1. Event capturing
  2. Target
  3. Event bubbling
- Event delegation
  - Pattern to handle events efficiently
  - Can add an event listener to a parent element and call an event on a particular target using the `.arget` property of the event object
  ```html
  <form name="myForm" id="myFormId">
    <input type="text" name="nameOfInput" />
  </form>
  ```
  ```javascript
  document.fname // using the form name
  document.forms[index] // using the index
  document.forms[‘fid’] // using the ID
  document.getElementById(‘fid’) // select by id
  ```
- Form validation
  - Client-side
    - Check not blank
    - Check for dates, email, number, formats and more
    - Derived information
    - Guide
  - Server-side
    - Final validation
    - Never trust user input
  - Real-time
    - Event handlers like `onChange`
  - Batch validation
    - Validates the whole form when submitting
  - `onSubmit`
    - Fires when a form is submitted
  - `onReset`
    - Fires when a user chooses to reset a form
  - `Event.preventDefault`
    - Prevent browser from firing default behaviour
- Separating behaviour and structure
  - No JS inside markup
- `fetch`
  - The HTML5 way of making requests

## Lecture 12 - XML, JSON

- XML (eXtensible Markup Language)

  - Subset of the Standard Generalized Markup Language (SGML)
  - HTML is an application of SGML (newer HTML does not follow the SML standard)
  - XML vocabularies
    - Mathematical Markup Language (MathML)
    - XHTML
    - Musical Markup Language (MML)
    - Real Simple Syndication (RSS)
    - Vector Markup Language (VML)
    - Scalable Vector Graphics (SVG)
    - Office Open XML (docx)
  - XML structure
    - The prolog (optional) - info about the document
    - The body - content of the document
    - The epilog (optional) - final comments or processing instructions
  - XML elements
    - Opening and closing tag
    - Content betwen tags
    - May contain attributes
  - XML namespacing

    ```xml
    <model
      xmlns:a="http://w3.org/TR/html4/"
      xmlns:b="http://idi.ntnu.no/furniture">

      <!— html table —>
      <a:table>
      <a:tr>
      <a:td>This is content</a:td>
      </a:tr>
      </a:table>

      <!— a furniture —>
      <b:table>
      <b:material>Jarrah</b:material>
      <b:cost>$100</b:cost>
      </b:table>
    </model>
    ```

- JSON (JavaScript Object Notation)
  - Human and machine readable
  - Language independent
  - Two structures
    - a collection of name/value pairs
    - an ordered list of values
      - string, number, object, array, booleans, and null
- XML vs. JSON
  - Both are self-describing, hierarchical, widely used
  - JSON
    - no end tags
    - shorter
    - quick to read/write
    - can use arrays
  - XML has to be parsed with an XML parser
  - JSON can be parsed by a standard JS function
- Parsing JSON
  - `.stringify(...)` - converts a JS object into a JSON string
  - `parse(...)` - converts a JSON string into a JS object
~~~