Week 2: Webpages written in HTML and CSS

L2.1: *Information Representation in a machine*

Information Representation

- Computers work only with "bits" (0s and 1s)
- Numbers:
  - Place value: binary numbers: (6 = 0110)
  - Two's complement: negative numbers (-6 = 1010)

Representing Text

ASCII

- American Standard Code for Information Interchange.
- $7$7 bits: $128$128 different entities.
  - `'a'..'z'`
  - `'A'..'Z'`
  - `'0'..'9'`
  - Special characters: `!@#$%^&*()_+{}|:"<>?~`[]\;',./...`
  - Control characters: newline, tab, backspace, etc.

Unicode

- Allow codes for more scripts, characters
- Universal Character Set encoding - UCS
  - UCS-1 (UTF-8): $1$1 byte per character - max $256 = 2^8$256 = 28 characters
  - UCS-2 (UTF-16): $2$2 bytes per character - max $65,536 = 2^{16}$65, 536 = 216 characters
  - UCS-4 (UTF-32): $4$4 bytes per character - max $4$4 Billion+ ($4,294,967,296$4, 294, 967, 296 = 2^{32}232) characters

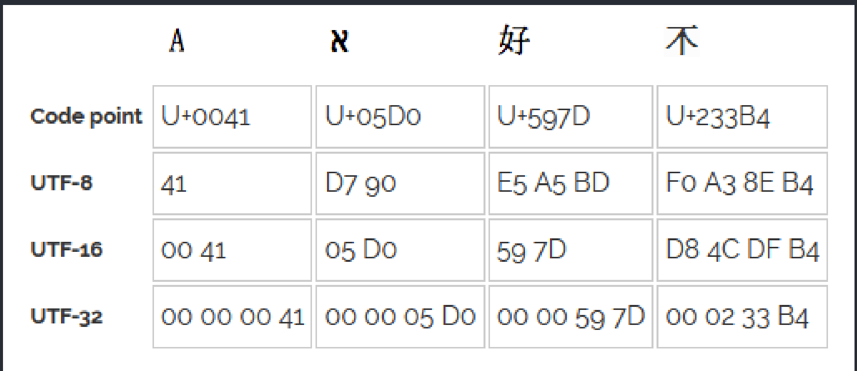
> $$\text{Bits required} = \text{Number of characters} * \text{Bits per character}$$
> Bits required = Number of characters * Bits per character

L2.2: *Efficiency of Encoding*

Prefix Coding

$1st$1st Byte	$2nd$2nd Byte	$3rd$3rd Byte	$4th$4th Byte	Free Bits	Max. Expressible Unicode Value
$0xxxxxxx$ 0xxxxxxx	-	-	-	77	$2^7 - 1 = 127 = 007F$ 27 – 1 = 127 = 007F (hex)
$110xxxxx$ 110xxxxx	$10xxxxxx$ 10xxxxxx	-	-	$5 + 6 = 11$5 + 6 = 11	$2^{11} - 1 = 2047 = 07FF$ 211 – 1 = 2047 = 07FF (hex)
$1110xxxx$ 1110xxxx	$10xxxxxx$ 10xxxxxx	$10xxxxxx$ 10xxxxxx	-	$4 + 6 + 6 = 16$ 4 + 6 + 6 = 16	$2^{16} - 1 = 65535 = FFFF$ 216 – 1 = 65535 = FFFF (hex)
$11110xxx$ 11110xxx	$10xxxxxx$ 10xxxxxx	$10xxxxxx$ 10xxxxxx	$10xxxxxx$ 10xxxxxx	$3 + 6 + 6 + 6 = 21$ 3 + 6 + 6 + 6 = 21	$2^{21} - 1 = 2097151 = 1FFFF$ 221 – 1 = 2097151 = 1FFFF (hex)

- So, only with 11 byte, we can represent 128128 characters.
- With 22 bytes, we can represent 20482048 characters.
- With 33 bytes, we can represent 6553665536 characters.
- With 44 bytes, we can represent 2, 097, 1522, 097, 152 characters.



	A	א	好	不
Code point	U+0041	U+05D0	U+597D	U+233B4
UTF-8	41	D7 90	E5 A5 BD	F0 A3 8E B4
UTF-16	00 41	05 D0	59 7D	D8 4C DF B4
UTF-32	00 00 00 41	00 00 05 D0	00 00 59 7D	00 02 33 B4

- Here we can see, how we can represent characters in different encoding schemes.

UTF-8

- Use 88 bits for most common characters: ASCII subset
- All other characters can be encoded based on prefix string
- Most common encoding in use today

L2.3: *What is Markup?*

- Markup is a way of using cues or codes in the regular flow of text to indicate how text should be displayed.
- Markup is very useful to make the display of text clear and easy to understand.

Types of Markup

Presentational

- **WYSIWYG** (What You See Is What You Get): directly format output and display
- Embed codes not part of regular text, specific to the editor.
- MS Word, Google Docs etc...

Procedural

- **LaTeX** is a procedural markup language.
- Details on how to display
  - chnage font to large, bold, italic, etc.
  - skip 2 lines, indent 3 inches, etc.

Descriptive

- This is a `<title>`, this is a `<section>`, etc...
- **HTML** is a descriptive markup language.

Semantic Markup

- **Content** vs **Presentation**
- *Semantics*
  - Meaning of the text
  - structure or logic of the document
- Example: using semantic tags like `<header>`, `<main>`, `<footer>` instead of `<div>`.

L2.4: *Introduction to HTML*

HTML (HyperText Markup Language)

- HTML first used by Tim Berners-Lee in 1989
- Considered an application of **SGML** (Standard Generalized Markup Language)
- HTML meant for browser interpretation
  - Very forgiving: loose validity checks
  - Best effort to display

HTML Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>My First Webpage</title>
    </head>
    <body>
        <h1>My First Webpage</h1>
        <p>This is my first webpage.</p>
    </body>
</html>
```

Timelines

- SGML based
  - 19891989: HTML 1.01.0
  - 19951995: HTML 2.02.0
  - 19971997: HTML 33, 44
- XML based
  - XHTML - 19971997 to mid 2010*s*2010s
- HTML5
  - first release 2008
  - W3C recommendation - 20142014

HTML5

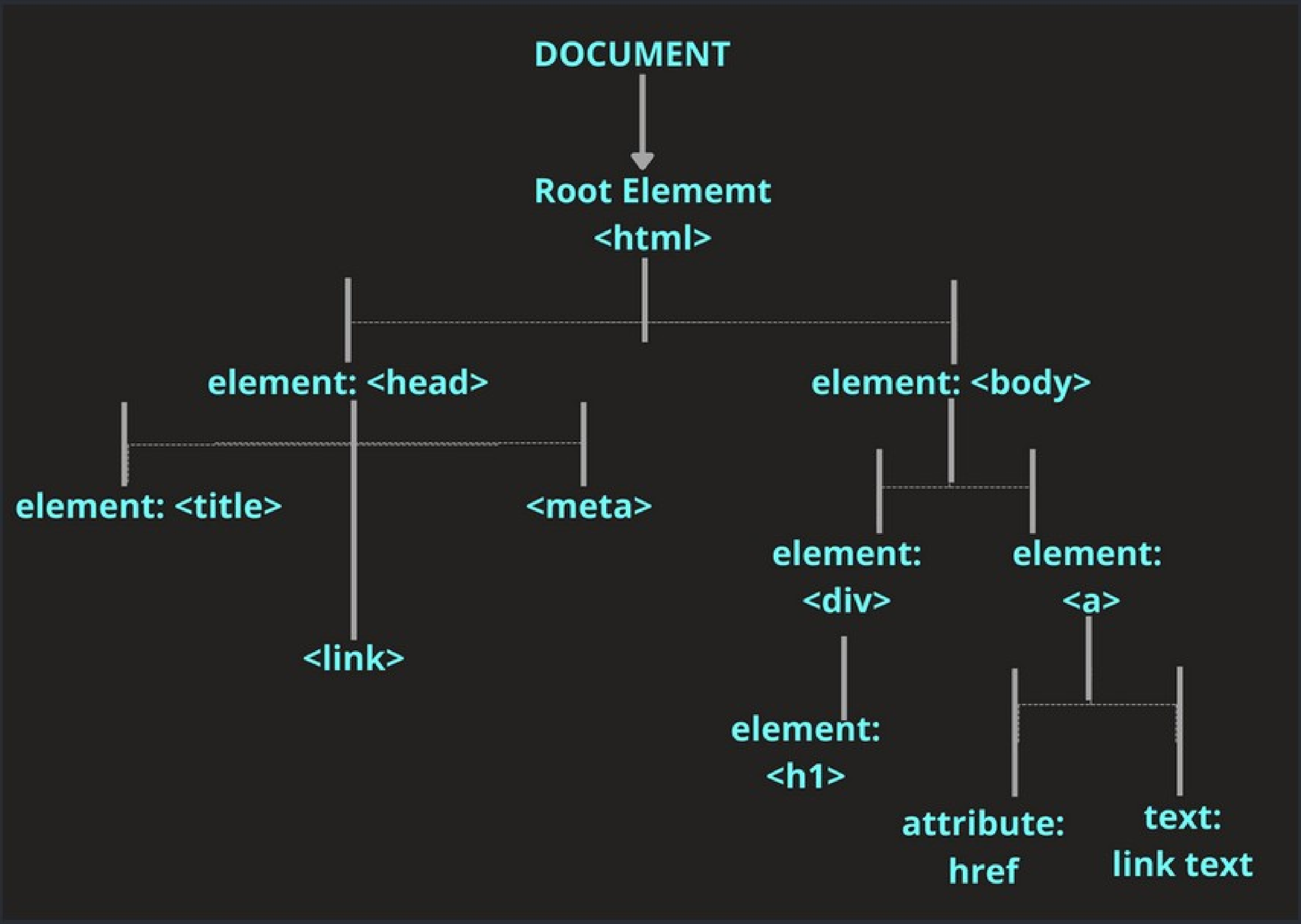
- **Block elements**: `<div>`
- **Inline elements**: `<span>`
- **Logical elements**: `<header>`, `<footer>`, `<nav>`,...
- **Media elements**: `<img>`, `<audio>`, `<video>`,...

Remove "presentation only" tags:

- `<center>`
- `<font>`

Document Object Model (DOM)

```
<!DOCTYPE html>
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
        <h1>A heading</h1>
        <p>A paragraph with <a href="link">link</a>.</p>
    </body>
</html>
```



DOM

- Tree structure representing logical layout of document
- *Direct manipulation of tree possible*
- **Application Programming Interface** (API)
  - *Canvas*
  - *Offline*
  - *Web Storage*
  - *Drag and Drop* & ...
- JavaScript primary means of manipulating
- CSS used for styling

Documentation

> Refer to **MDN Web Docs** 🔗 for HTML documentation.
> Refer to **W3Schools** 🔗 for learning HTML
> Refer to **MDN Front-end Web Developer Learning Path** 🔗 for learning front-end web development.

L2.5 & 2.6: *Introduction to Styling & Types of CSS styling and Responsive Websties*

CSS (Cascading Style Sheets)

- CSS is a powerful language used for styling and visually enhancing HTML documents.
- With CSS, you can control the layout, color, fonts, and many other aspects of your web pages.
- There are multiple ways to apply CSS styles to HTML elements, giving you flexibility and control over the appearance of your website.

1. Inline CSS

- Inline CSS is the most basic and straightforward way to apply styles directly to individual HTML elements.
- By using the `<style>` attribute within an HTML tag, you can specify CSS properties and values.
- Example:

  ```
  <p style="color: orange; font-size: 24px;">
      This is a paragraph with inline styles applied.
  </p>
  ```

  Rendered HTML:

  This is a paragraph with inline styles applied.

2. Internal CSS

- Internal CSS allows you to define styles within the HTML document itself, typically within the `<head>` section.
- To apply internal CSS, you enclose the styles within the `<style>` tags.
- Example:

  ```
  <head>
      <style>
          p {
              background-color: blue;
              color: white;
              font-size: 16px;
          }
      </style>
  </head>
  <body>
      <p>This is a paragraph with internal styles applied.</p>
  </body>
  ```

  Rendered HTML:

  This is a paragraph with internal styles applied.

3. External CSS

- External CSS is the recommended and most scalable approach for styling HTML documents.
- With this method, you create a separate CSS file with the `.css` extension and link it to your HTML document using the `<link>` tag.
- Example:
  - HTML file: `index.html`

```html
<head>
    <link rel="stylesheet" href="styles.css" />
</head>
<body>
    <p>This is a paragraph with external styles applied.</p>
</body>
```

  - CSS file: `styles.css`

```css
p {
    background-color: #c678dd;
    color: #282c34;
    font-size: 28px;
    font-weight: bold;
    font-style: italic;
    text-decoration: line-through;
}
```

Rendered HTML:

~~***This is a paragraph with external styles applied.***~~

CSS Selectors

1. Element Selector

- Targets elements based on their tag name.
- Example:

```css
p {
    color: blue;
}
```

2. Class Selector

- Targets elements with a specific class attribute.
- Example:

```html
<p class="highlight">This is a paragraph with class selector applied.</p>
```

```css
.highlight {
    background-color: yellow;
}
```

Rendered HTML:

This is a paragraph with class selector applied.

3. ID Selector

- Targets a specific element with a unique ID attribute.
- Example:

```
<p id="top">Some para...</p>
```

```
#top {
    color: red;
    font-size: 24px;
}
```

Rendered HTML:

Some para...

4. Universal Selector

- Targets all elements on the page.
- Example:

```
* {
    color: cyan;
}
```

Rendered HTML:
This is a paragraph with universal selector applied.

Priority of CSS Selectors

Selector	Priority
Inline styles	**Most**
ID selectors	**Second most**
Class selectors	**Third most**
Tag selectors	**Least**
!important	**Overrides all others**

Documentation

> Refer to **MDN Web Docs** 🔗 for CSS documentation.
> Refer to **W3Schools** 🔗 for learning CSS.

Responsive Design

- Responsive design is an approach to web design that aims to create websites that adapt and respond to different screen sizes and devices.
- It involves designing and developing websites in a flexible and fluid manner, using CSS media queries and other techniques to ensure optimal user experience across various devices.
- Responsive design allows content and layouts to automatically adjust, resize, and reposition based on the user's screen size, providing a consistent and user-friendly experience on desktops, tablets, and mobile devices.

Bootstrap Framework

- Bootstrap is a popular front-end framework developed by Twitter that simplifies the process of building responsive and mobile-first websites.
- It provides a collection of pre-designed HTML, CSS, and JavaScript components, such as navigation bars, buttons, forms, and grid systems, which can be easily customized and integrated into web projects.

Rapid Development

- Bootstrap offers ready-to-use components and a responsive grid system, enabling developers to quickly prototype and build websites or web applications.

Consistency

- By utilizing Bootstrap's predefined styles and components, websites built with Bootstrap maintain a consistent and professional look across different pages and sections.

Device Compatibility

- Bootstrap's responsive features ensure that the website layout and components automatically adjust to different devices, providing an optimal viewing experience for users on desktops, tablets, and mobile devices.
