1 Basic Technologies

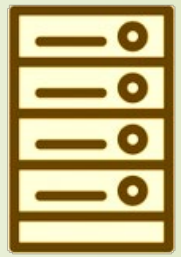
HTML, CSS and their enhancements

II.1

Fundamental Concepts

Main areas in web development

- front end
  - focused on creating interfaces, i.e. the collection of elements on the web page that the user is able to see and interact with
- back end
  - opposite of front end, i.e. performing processing of information



Front End

- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation

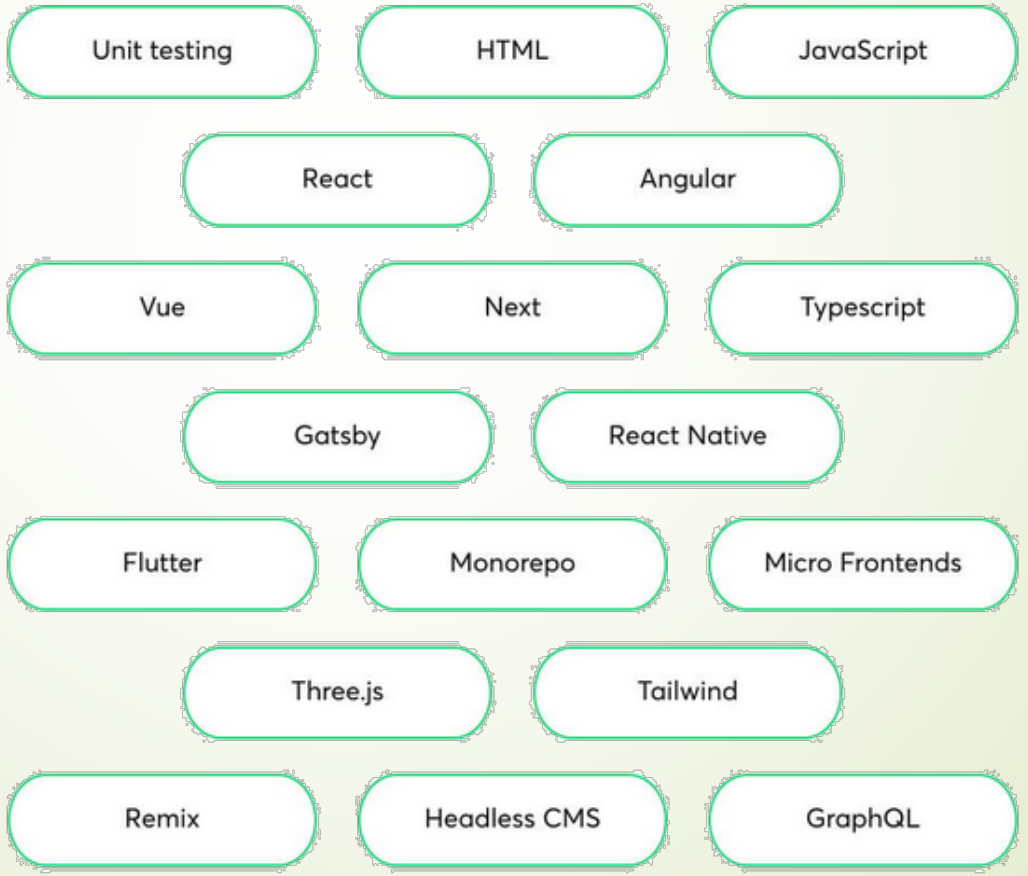
Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

II.1

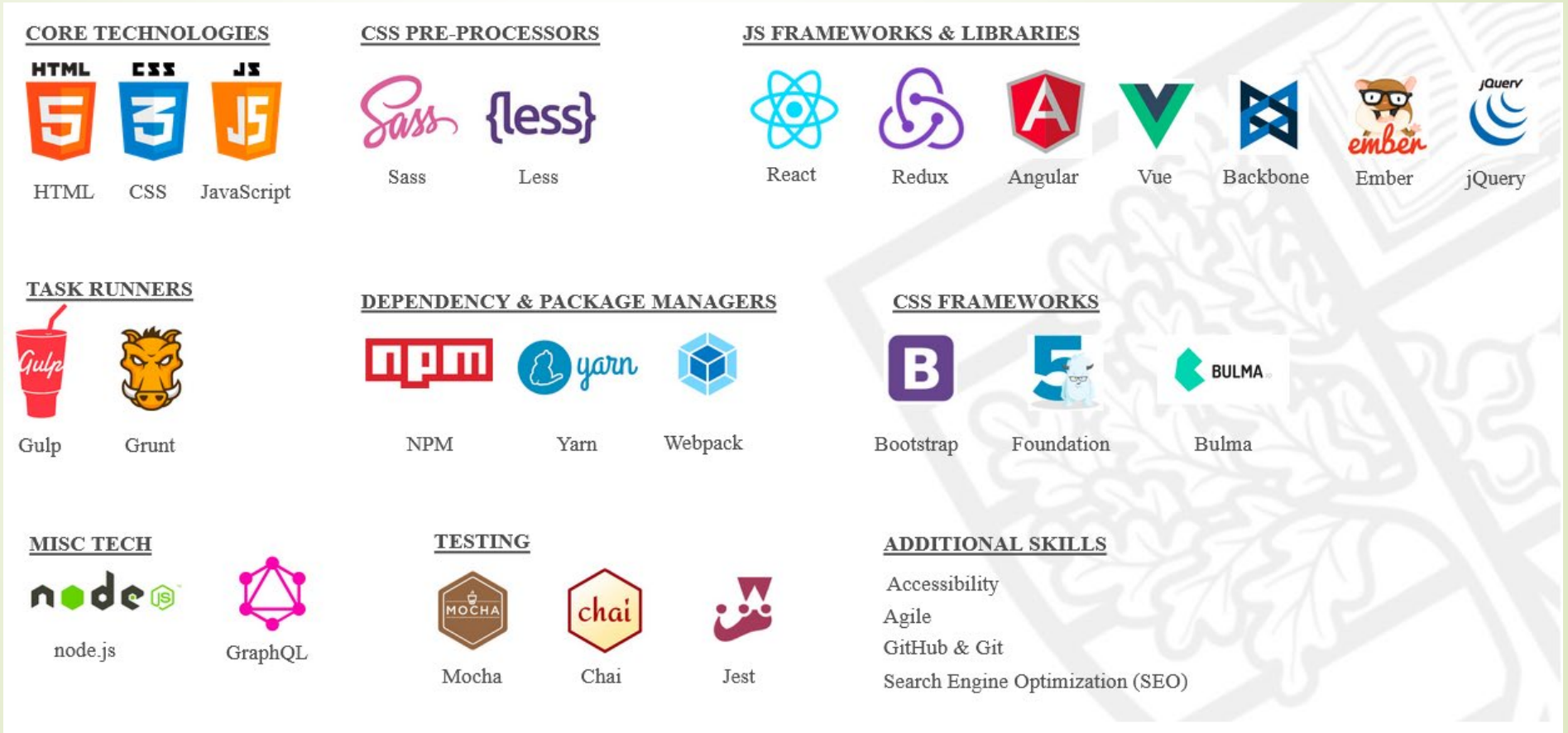
Fundamental Concepts

Top Front End Technologies



Fundamental Concepts

Top Front End Technologies



II.1

Hypertext Markup Language

HTML

- most widely used language to write web pages
- hypertext refers to the way in which Web pages are linked together
- describes the content and structure of information on a web page
  - not the same as the presentation (appearance on screen)
- surrounds text content with opening and closing tags
- each tag's name is called an element
  - syntax: <element> content </element>
  - example: This is a paragraph
- stricter, more standard version XHTML

II.1

Hypertext Markup Language

XHTML

- EXtensible HyperText Markup Language
- a stricter, more XML-based version of HTML
- developed to make HTML more extensible and flexible to work with other data formats (such as XML)
- important differences:
  - elements must always be properly nested
  - elements must always be closed
  - elements must always be in lowercase
  - attribute names must always be in lowercase
  - attribute values must always be quoted

Hypertext Markup Language

XHTML – Structure of a page

- DOCTYPE
  - used to declare the DTD (Document Type Definition)
- xmlns attribute
  - specifies the xml namespace
  - provide a method to avoid element name conflicts

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Title of document</title>
</head>
<body>

  some content here...

</body>
</html>
```

II.1

Hypertext Markup Language

XHTML – Structure of a page

- head
  - contains machine-readable information (metadata) about the document, like its title, scripts, and style sheets
- body
  - contains the page's content

Hypertext Markup Language

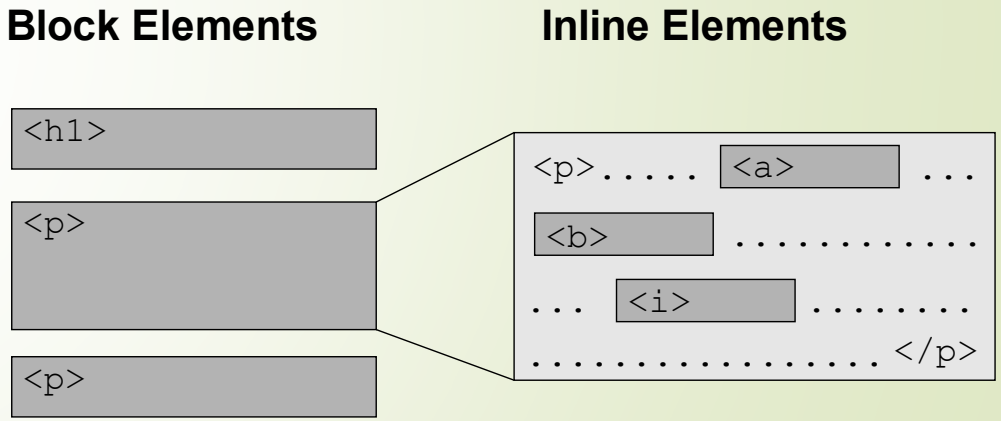
XHTML – Inside head

- meta-tag
  used to provide such additional information
  - Name
    Name for the property. Can be anything. Examples include, keywords, description, author, revised, generator etc.
  - content
    Specifies the property's value
- style- resp. link-tag
  used to specify embedded CSS resp. external CSS
- script-tag
  used to load JavaScript code

Hypertext Markup Language

XHTML – Block and inline elements

- block elements
  meant to structure the main parts
  of your page, by dividing your content
  in coherent blocks
  start with a new line and capture
  the available full width
- Inline elements
  meant to differentiate part of a text,
  to give it a particular function or
  meaning
  Inline elements usually comprise a single or few words



II.1

Hypertext Markup Language

XHTML – Basic Tags

- heading tags <h1>...</h1>
- paragraph tag ...
- line break tag
- centering content ...
- preserve formatting <pre> ... </pre>
- computer <code> ... </code>
- list tags <ul>, <ol>,

Hypertext Markup Language

XHTML – Basic Tags

- a-element
  specifies a link

```
<a href="http://www.thi.de" target="_blank">BGU</a>
```

- types of links
  - links from one website to another website
  - links from one page to another page on the same website
  - links from one part of a web page to another part of the same page
  - other types of links, such as those that start up your email program and compose a new email to someone

Hypertext Markup Language

XHTML – Basic Tags

- `href`-attribute can be:
  - an absolute URL which points to another website, such as `href="http://www.thi.de"`
  - a relative URL which points to a file within a website
  - a link to an element with a specified id within the current web page

    ```
    href="#top"
    ```

  - a combination of a URL and location within the page

    ```
    href="index.html#top"
    ```

Hypertext Markup Language

XHTML – Grouping

- `div`-element
  - allows to group a set of elements together in one block-level box
  - used to indicate a logical section or area of a page
  - use case of the `div`-element: to create an empty container
- `span`-element
  - acts like an inline equivalent of the `div`-element
    - contain a section of text where there is no other suitable element to differentiate it from its surrounding text
    - contain a number of inline elements
  - can control the appearance of the content of inner elements, using CSS

Hypertext Markup Language

XHTML – Attributes

- defines the characteristics of an HTML element and is placed inside the element's opening tag
- are made up of two parts: a name and a value

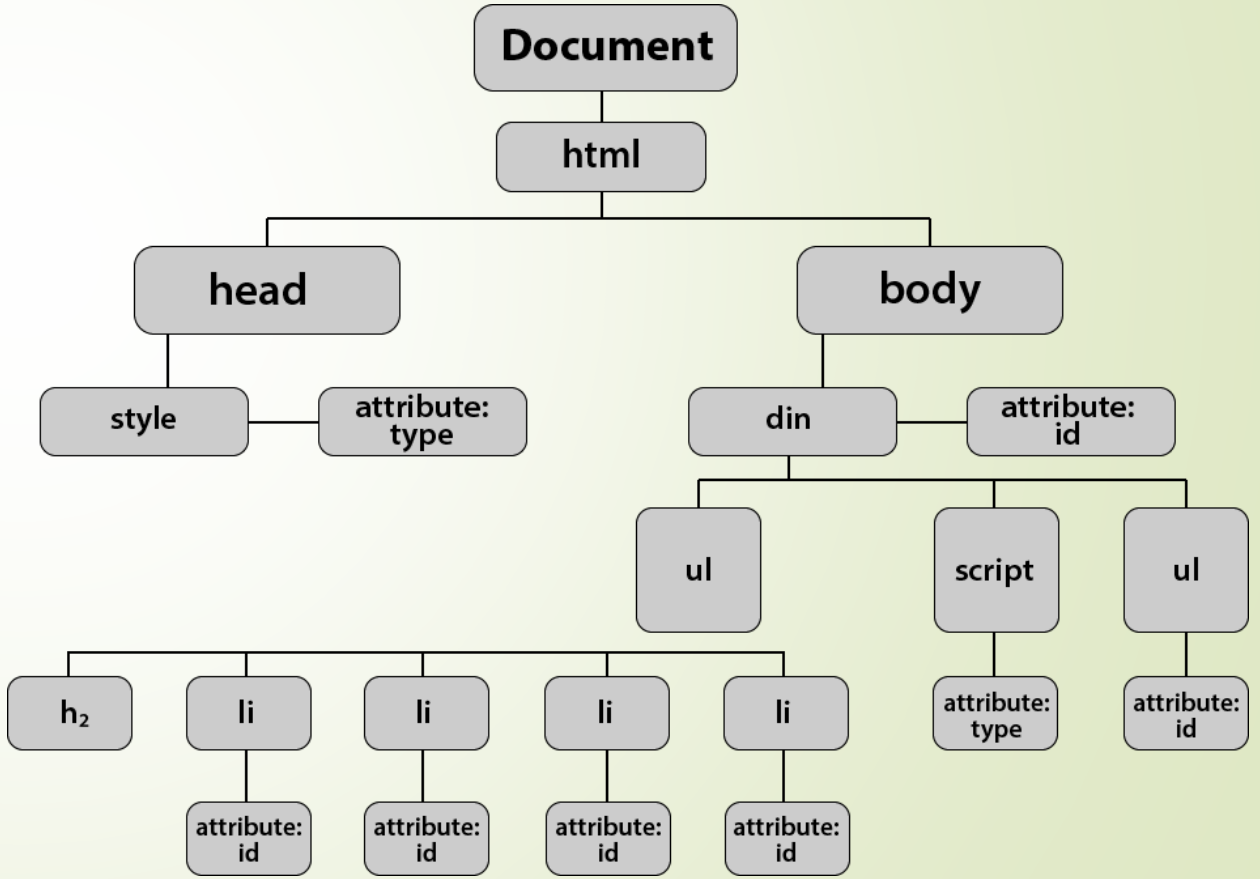
```
<p disabled="disabled"> ... </p>
```

- core attributes
  - id — unique identifier
  - title — the element's title
  - class — specifies the style sheet category
  - style — specifies Cascading Style Sheet (CSS) rules within the element

Parsing HTML

DOM - Document Object Model

- standard operations
  for traversing and modifying
  document elements arranged
  in a hierarchy of objects



Parsing HTML

xml.dom — The DOM API

- Objects

Interface	Section	Purpose
DOMImplementation	*DOMImplementation Objects*	Interface to the underlying implementation.
Node	*Node Objects*	Base interface for most objects in a document.
NodeList	*NodeList Objects*	Interface for a sequence of nodes.
DocumentType	*DocumentType Objects*	Information about the declarations needed to process a document.
Document	*Document Objects*	Object which represents an entire document.
Element	*Element Objects*	Element nodes in the document hierarchy.
Attr	*Attr Objects*	Attribute value nodes on element nodes.
Comment	*Comment Objects*	Representation of comments in the source document.
Text	*Text and CDATASection Objects*	Nodes containing textual content from the document.
ProcessingInstruction	*ProcessingInstruction Objects*	Processing instruction representation.

II.1

Parsing HTML

xml.dom — The DOM API

- Methods (link to Python Standard Library)
  - traversing the tree
  - manipulating the tree
  - finding an element

II.1

HTML Forms

Formulas

- forms contain special control elements like input text box, check boxes, radio-buttons and submit buttons
- `<form>` tag is used to defines an HTML form

Attributes	Description	Syntax & Example
action	defines URL of the program or page where action will be performed on form data.	`action="page2.php"`
method	specify the HTTP method to send form data.	`method="get"`
target	specify the target window or frame	`target="_blank"`
enctype	specify the encoding of form data (for media files)	`enctype="multipart/form-data"`

II.1

HTML Forms

Formulas

- form elements are defined inside the <form>-tag
  - <input>-tag
    - type attribute specifies different input elements
  - button
    - simple button
    - submit button
    - reset button

Attribute	Description	Syntax & Example
Text Box	used to define text box that allow user to enter some text.	input type="text"
Radio Button	defines radio button that allow users to select any one option or choice	input type="radio"
Checkbox	defines checkbox that allow users to select multiple option or choices	input type="checkbox"
Button	defines normal buttons for users for some action	input type="button"
Reset Button	defines button that reset form data when user click on it.	input type="submit"
Submit Button	defines Form submission button that submit form data when user clicks on it	input type="reset"

II.1

Realizing a Website

Design Steps

- Plan
  - collection of all possible information (requirements) regarding the website
  - define goals and objectives in order to achieve the objectives
- Design
  - determine the structure of the web site
- Develop
  - write all the code that brings the content to life and makes the site a reality
- Validate
- Release

II.1

Realizing a Website

Design Steps

- How to structure of a Web Site
  - when planning web pages, first list all the information (as headings) to put on them
  - bring all the headings that belong together into related groups: each group will require a separate page, or group of related pages, which will be individual parts of your web site
  - if there are to be a lot of web pages, it is advisable to have each group of pages kept in a separate folder
  - remember that any individual pages, which contain a lot of data, should be broken down into smaller pages, linked together
- Often
  - need to organize information, spreading it over several pages that are organized and linked via an initial "home/index page".

Realizing a Website

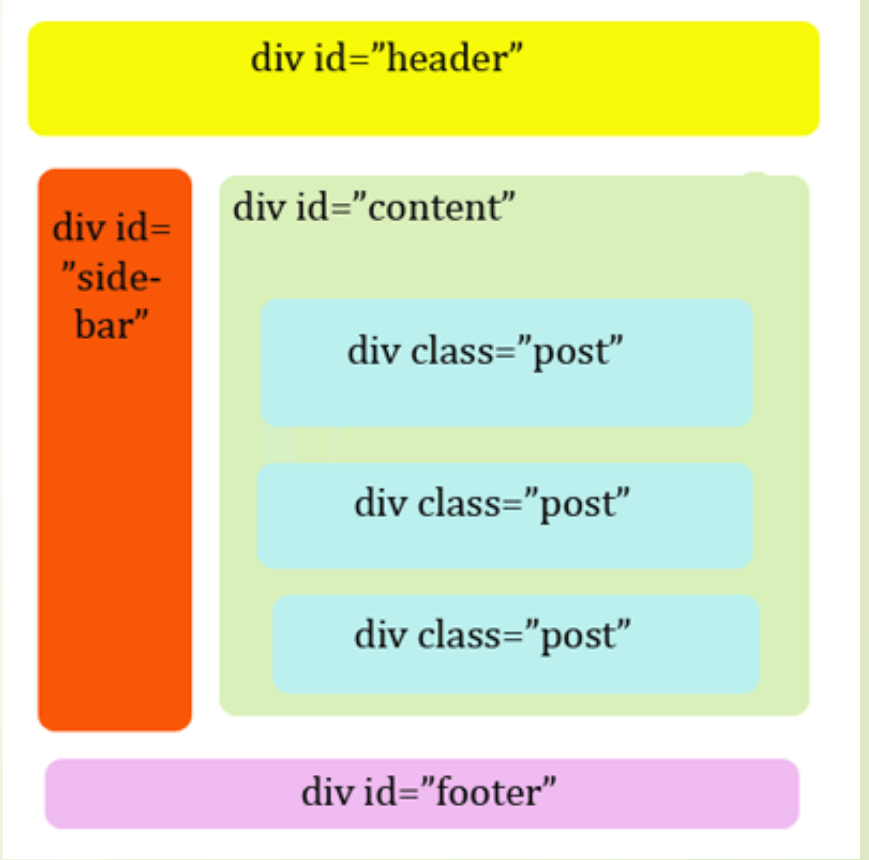
Design Steps - A typical page structure

- Only very few sites that don't at least loosely follow this pattern:
  - header at the top of the page, usually containing the top level heading of the page, and/or a company logo
  - main content column, which holds the main content of functionality
  - one or more sidebars, holding peripheral content that is either related to the page's main content and changes as new pages are loaded, or is always relevant and persists across the whole site
  - navigation menu going across or down the page. This is often put in a sidebar, or may form part of the header.
  - a footer that goes across the bottom of the page and contains secondary information such as copyright information and contact details

Realizing a Website

Design Steps - A typical page structure

- structuring mechanisms
  - `<div>` is a block container
  - `<span>` is an inline container
- characteristics
  - have no inherent content or dimensions, only that of their contents, until they are styled via CSS or manipulated by JavaScript
  - they also have no semantics, and the only way you can identify them is by giving them a class or id attribute



II.1

Realizing a Website

Intro to semantic HTML

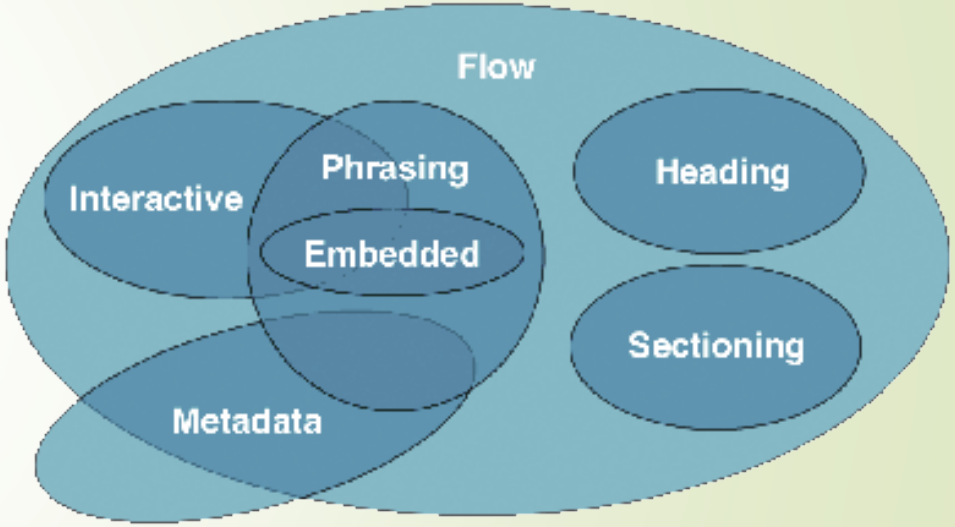
- disadvantages of div's
  - accessibility
  - readability
  - consistency and standards

- Now:
  - HTML5 was introducing a standardized set of semantic elements
  - a semantic element clearly describes its meaning to both the browser and the developer

HTML5

HTML5 – Content model

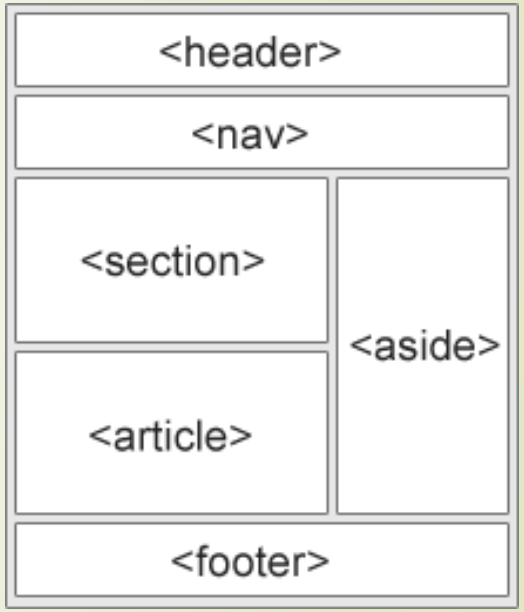
- Metadata content, e.g. link, script
- Flow content, e.g. span, div, text
- Sectioning content, e.g. aside, section
- Heading content, e.g. h1
- Phrasing content, e.g. span, img, text
- Embedded content, e.g. img, iframe, svg
- Interactive content, e.g. a, button, label
  Interactive content is not allowed to be nested



HTML5

HTML5 – Semantic elements

- `<header>` defines a header for the document or a section
- `<footer>` defines a footer for the document or a section
- `<nav>` defines navigation links in the document
- `<main>` defines the main content of a document
- `<section>` defines a section in the document
  the spec defines this as "a thematic grouping of content, typically with a heading
- `<article>` defines an article in the document
- `<aside>` defines content aside from the page content



HTML5

HTML5 – Form improvements

- Web Forms 2.0
  - extension to the forms features
  - form elements and attributes in HTML5 provide a greater degree of semantic mark-up

Type	Description
datetime	A date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601 with the time zone set to UTC.
datetime-local	A date and time (year, month, day, hour, minute, second, fractions of a second) encoded according to ISO 8601, with no time zone information.
date	A date (year, month, day) encoded according to ISO 8601.
month	A date consisting of a year and a month encoded according to ISO 8601.
week	A date consisting of a year and a week number encoded according to ISO 8601.
time	A time (hour, minute, seconds, fractional seconds) encoded according to ISO 8601.
number	This accepts only numerical value. The step attribute specifies the precision, defaulting to 1.
range	The range type is used for input fields that should contain a value from a range of numbers.
email	This accepts only email value. This type is used for input fields that should contain an e-mail address. If you try to submit a simple text, it forces to enter only email address in email@example.com format.
url	This accepts only URL value. This type is used for input fields that should contain a URL address. If you try to submit a simple text, it forces to enter only URL address either in http://www.example.com format or in http://example.com format.

HTML5

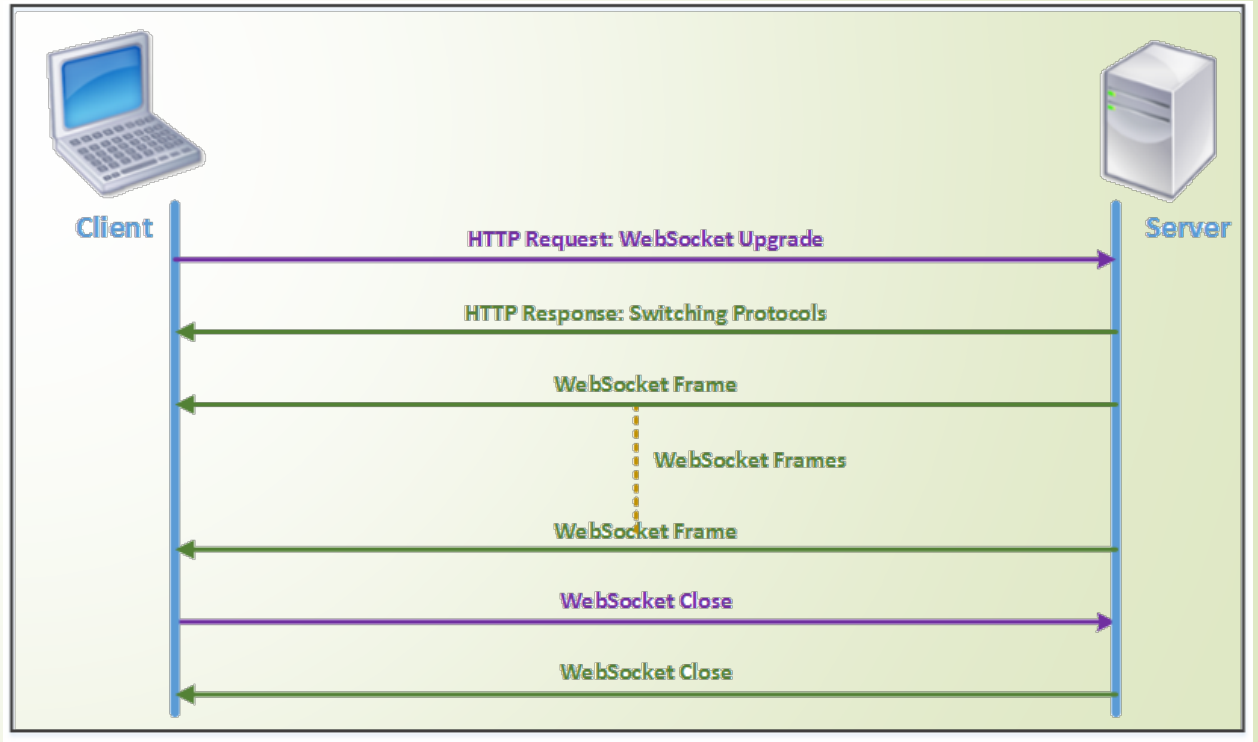
HTML5 – Websocket

- allows the browser to create a socket client (inside a web page)
- defines a full-duplex single socket connection over which messages can be sent between client and server
- two-way communication without expensive/annoying client-side polling
- ideal for low-latency persistent connections, e.g. for real-time Web applications
- requires server-side support and client-side support (JavaScript)

HTML5

HTML5 – Websocket

- starts off as a HTTP request, which indicates that it wants to "upgrade" to the WebSocket protocol
- if the server can understand it, then the http connection is switched into a WebSocket connection
- once connected, data is transmitted (bidirectionally) via frames



HTML5

HTML5 – Web Storage

- store some information locally on the user's computer, similar to cookies, but it is faster and much better than cookies and is no more secure than cookies
- information stored in the web storage isn't sent to the web server as opposed to the cookies where data sent to the server with every request

HTML5

HTML5 – Web Storage

- two types of web storage, which differ in scope and lifetime
  - local storage
    to store data for your entire website on a permanent basis
  - session storage
    to store data on a temporary basis

Example

```
<script>
// Check if the localStorage object exists
if(localStorage) {
    // Store data
    localStorage.setItem("first_name", "Peter");

    // Retrieve data
    alert("Hi, " + localStorage.getItem("first_name"));
} else {
    alert("Sorry, your browser do not support local storage.");
}
</script>
```