

Web Applications 2017

Week 1, Slides 1

Content

- HTTP,
- Internet Media Types
- Uniform Resource Locators
- Platforms and frameworks
- NodeJS
- Java Enterprise Edition
- HTML
- CSS
- Frontend user interface frameworks

HTTP

HTTP Method ↕	RFC ↕	Request Has Body ↕	Response Has Body ↕	Safe ↕	Idempotent ↕	Cacheable ↕
GET	RFC 7231	No	Yes	Yes	Yes	Yes
HEAD	RFC 7231	No	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	No	Yes	No	Yes	No
CONNECT	RFC 7231	Yes	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	Yes

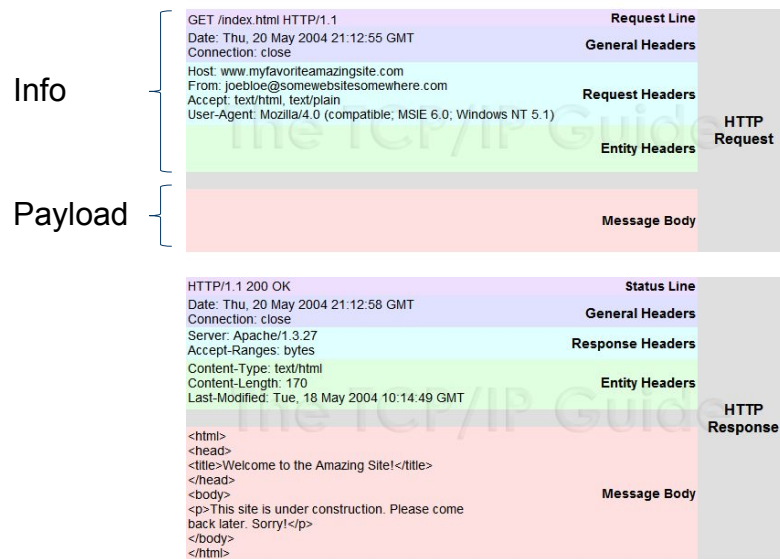
The "Verbs"

The basic protocol for web applications.

Readings

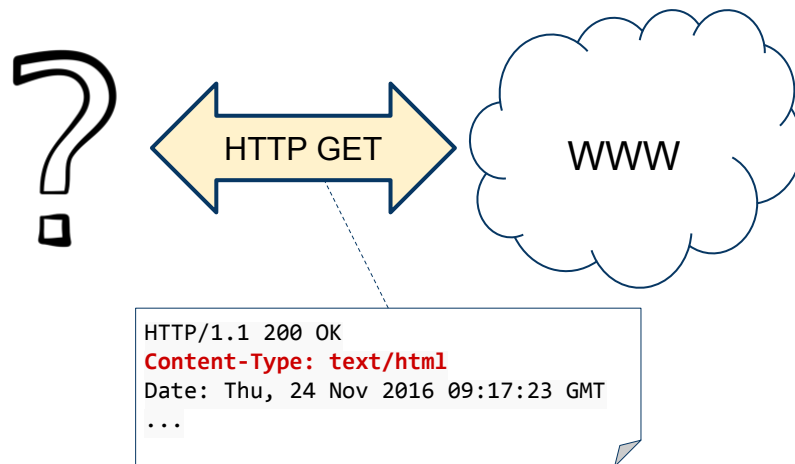
- [HTTP Wikipedia](#)
- [HTTP for web developers](#)
- [HTTP-in depth](#)
- [HTTPS Wikipedia](#)
- [Cacheable](#)

HTTP Messages



List of [status codes](#)

Internet Media Types

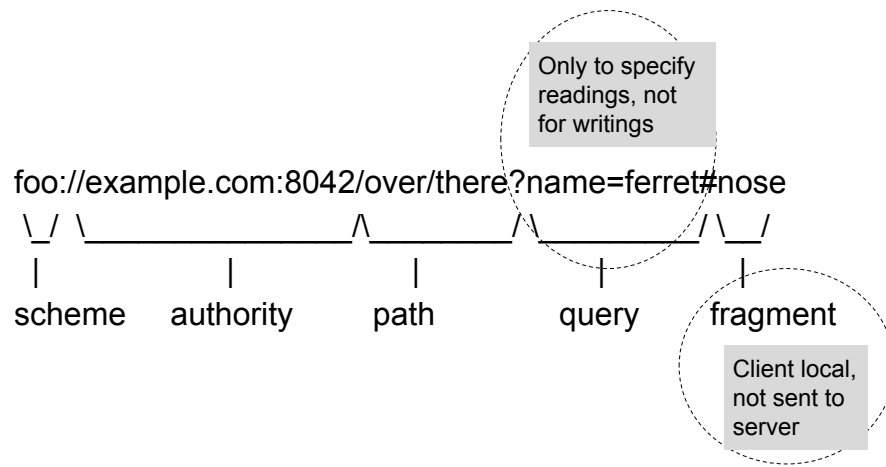


HTTP messages are capable of carrying arbitrary "labeled" content. The mechanism used to label such content is a media type.

Readings:

- [Media types Wikipedia](#)
- Full list of med [media types](#).

Uniform Resource Locator

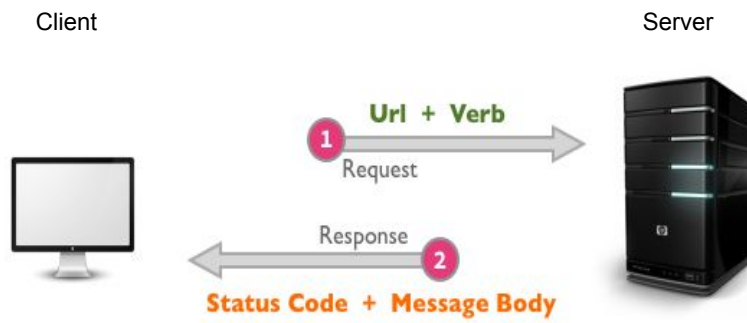


Resources specified by URLs

Readings:

- [URLs Wikipedia](#)
- [Percent Encoding](#) of URLs

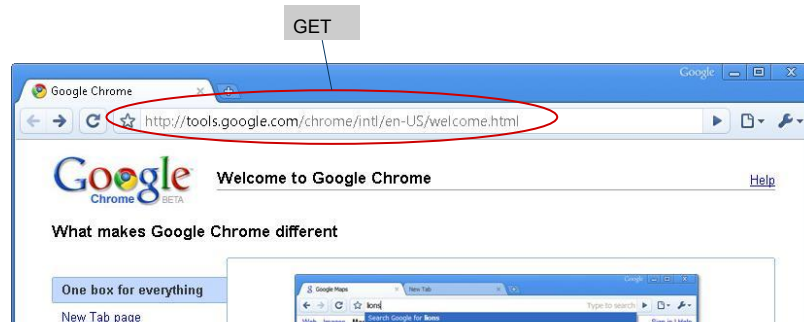
Request and Response



The basic interaction!

Sending Requests

```
$ curl -X POST -TODO
```



Sending request

- Using command line tool [curl](#) or browser

Platforms and Frameworks



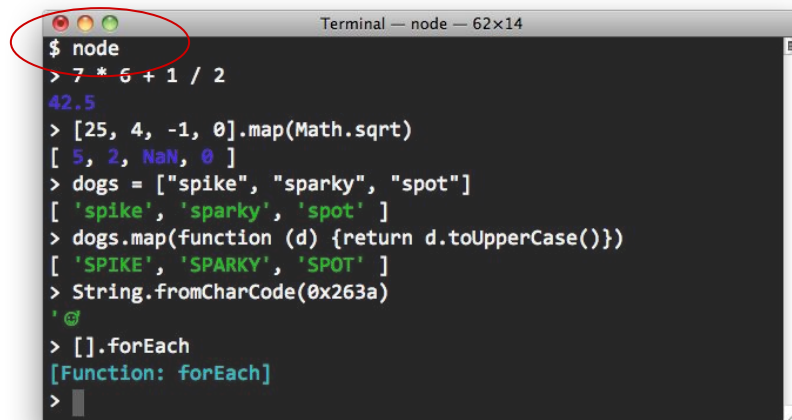
Major [platforms](#) and [frameworks](#)

- I would say: A platform > a framework
- Most (all?) cover both client and server side

Readings:

- [Web frameworks Wikipedia](#)
- [Comparison of web frameworks](#)
- [.NET](#)
- [JEE](#)
- [LAMP](#)
- [MEAN](#)

NodeJS

A screenshot of a macOS Terminal window titled "Terminal — node — 62x14". The window shows a Node.js REPL session. The prompt is "\$ node". The first command is "> 7 * 6 + 1 / 2", which outputs "42.5". The second command is "> [25, 4, -1, 0].map(Math.sqrt)", which outputs "[5, 2, NaN, 0]". The third command is "> dogs = [\"spike\", \"sparky\", \"spot\"]", which outputs "['spike', 'sparky', 'spot']". The fourth command is "> dogs.map(function (d) {return d.toUpperCase()})", which outputs "['SPIKE', 'SPARKY', 'SPOT']". The fifth command is "> String.fromCharCode(0x263a)", which outputs "'😊'". The sixth command is "> [].forEach", which outputs "[Function: forEach]". The prompt is now ">".

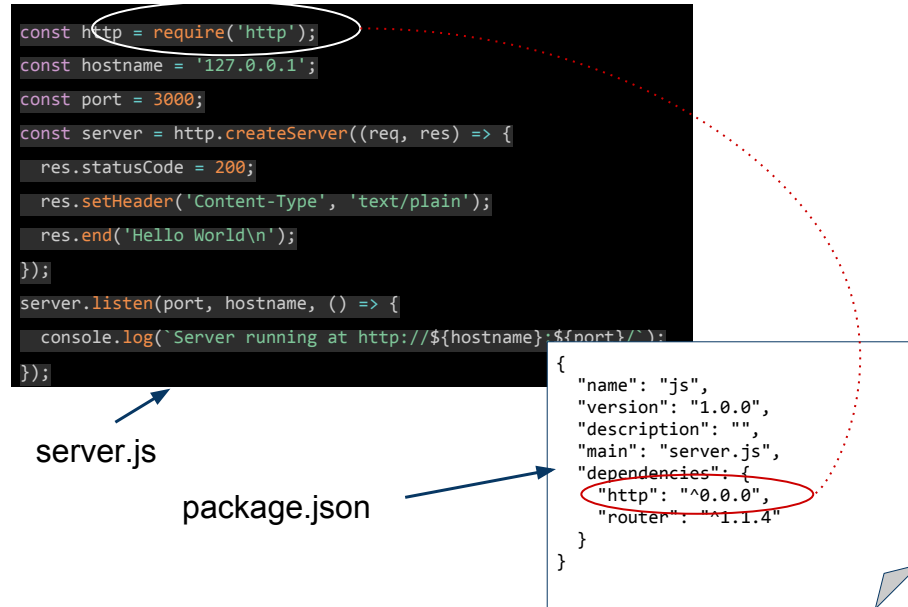
```
$ node
> 7 * 6 + 1 / 2
42.5
> [25, 4, -1, 0].map(Math.sqrt)
[ 5, 2, NaN, 0 ]
> dogs = ["spike", "sparky", "spot"]
[ 'spike', 'sparky', 'spot' ]
> dogs.map(function (d) {return d.toUpperCase()})
[ 'SPIKE', 'SPARKY', 'SPOT' ]
> String.fromCharCode(0x263a)
'😊'
> [].forEach
[Function: forEach]
>
```

NodeJS is a an asynchronous (more later) event driven JavaScript runtime (server side)

Readings

- [NodeJS Home](#)
- [About NodeJS](#)
- [NodeJS full API](#)
- [Node Hero](#) (tutorial)

Node Package Manager (npm)

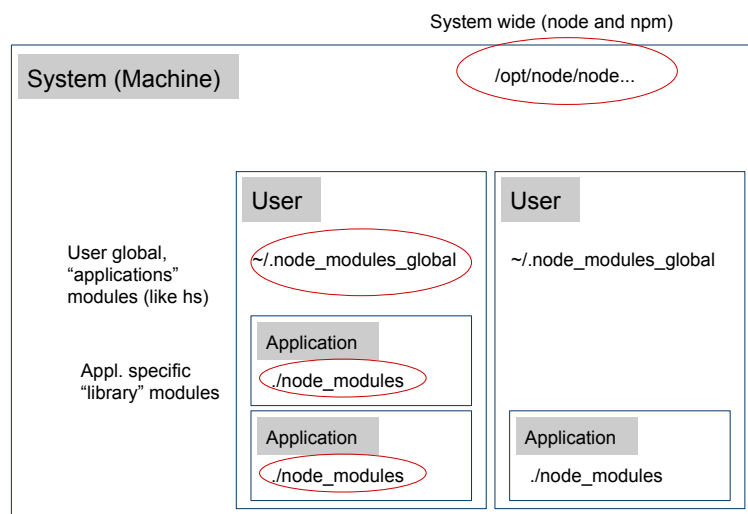


npm is a module (dependencies) system for NodeJS (compare Maven for Java)

Readings

- [NPM Home](#)
- [Installing](#) (tutorial)

Npm Environment



npm is a module (dependencies) system for NodeJS (compare Maven for Java)

Readings

- [NPM Home](#)
- [Installing](#) (tutorial)

Http-server

```
$ npm install http-server -g
```

User global

```
$ hs
```

```
Starting up http-server, serving
```

```
./public
```

```
Available on:
```

```
http://127.0.0.1:8080
```

```
http://192.168.1.12:8080
```

```
Hit CTRL-C to stop the server
```

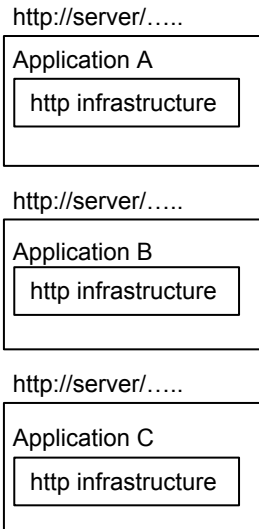
A development server built on top of NodeJs.

Readings:

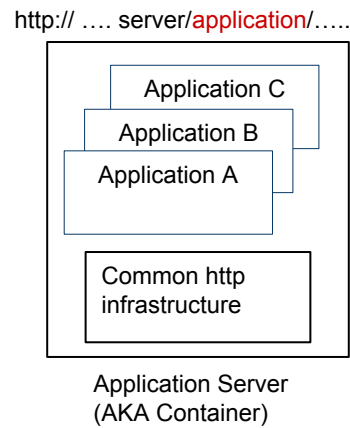
- [http-server](#)

Architectures

NodeJS architecture



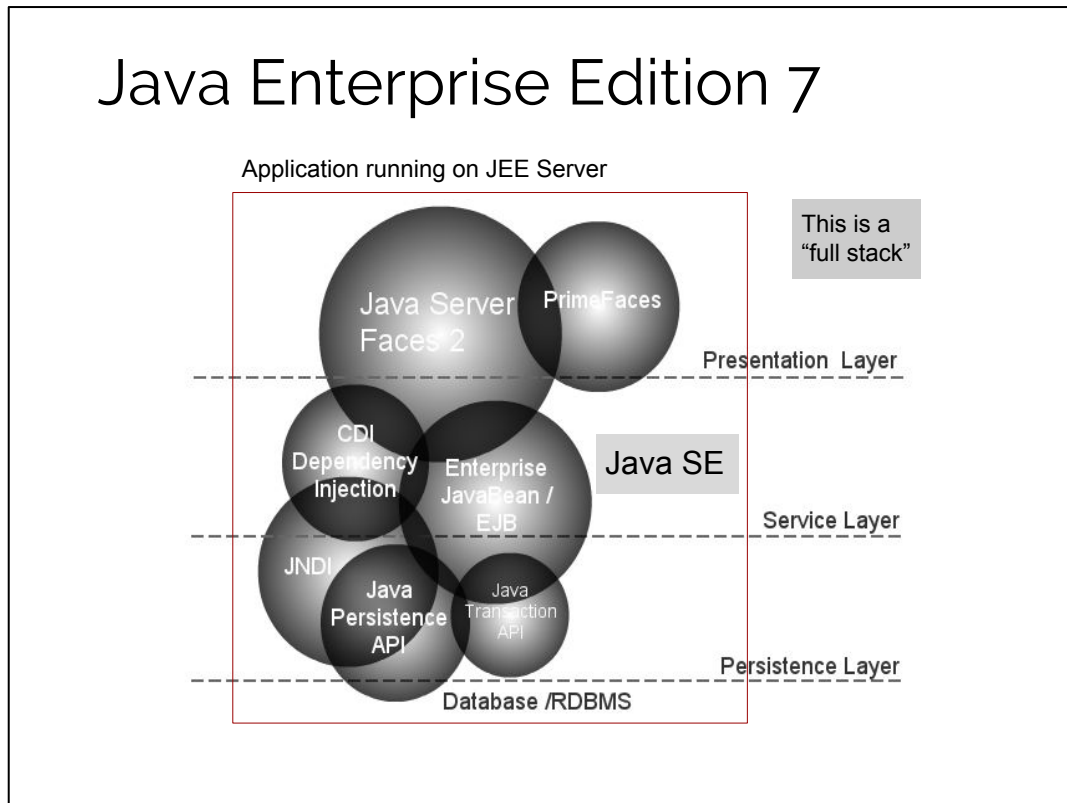
.NET, Java Enterprise architecture



Architecture

- (Server) process and application the same (process to application is 1:1) ...
- ...or applications in server process (server to application is 1:M)?
- If more applications on same server have to distinguish them (path)

Java Enterprise Edition 7



Java Enterprise Edition 7 (JEE 7)

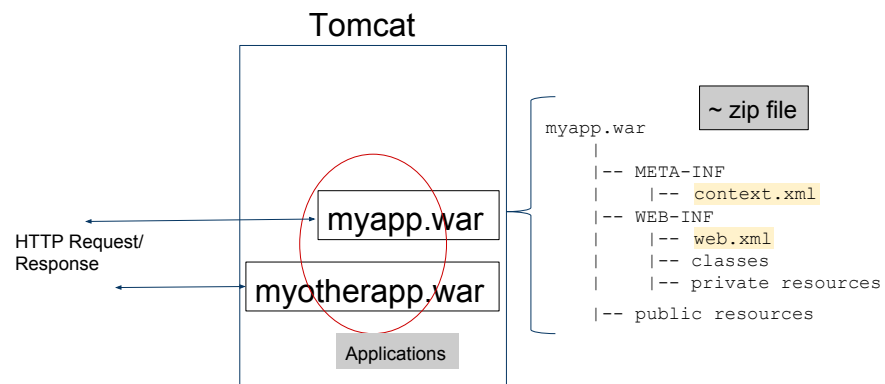
Platform for developing and running web/enterprise software, including network and web services, and other large-scale, multi-tiered, scalable,

- Java SE is a subset of Java EE

Readings

- [Specifications](#)
- [JEE7 is an umbrella specification](#) including a specific subset of JSRs.
- Java Specification Requests, [JSRs](#),

Apache Tomcat (JEE Server)



Tomcat is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies.

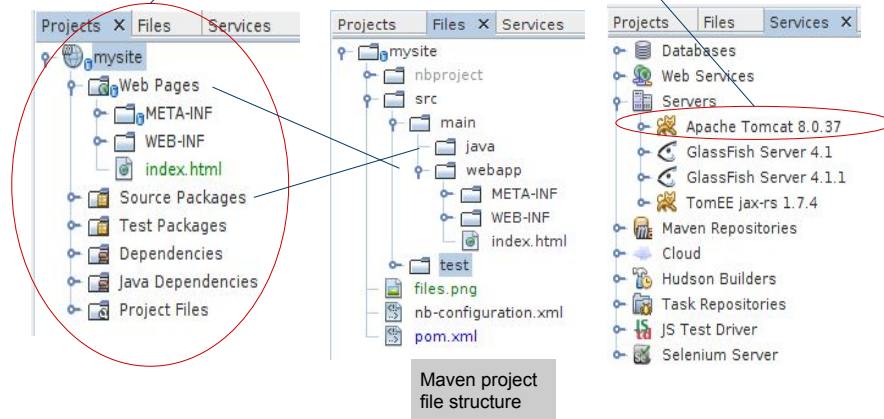
- More on technologies later
- Application deployed as *.war files (like *.zip).
- Server also serve static content like HTML.

Readings

- [Apache Tomcat](#)

NetBeans and Tomcat

Maven Web Application Project running on Tomcat



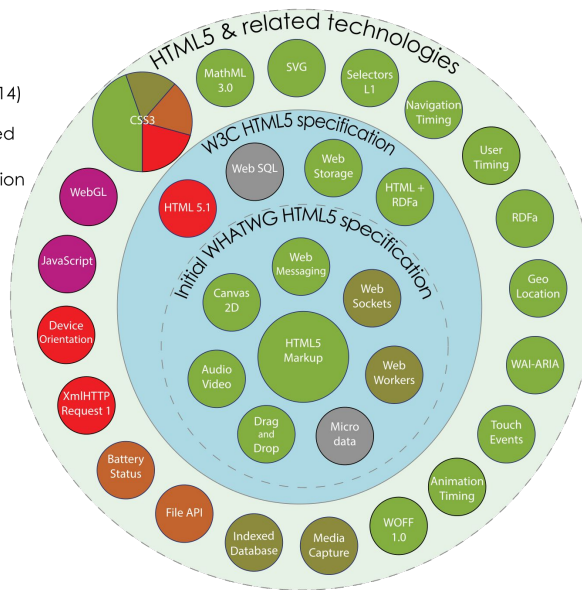
Tomcat bundled with NetBeans

- Will pack, deploy and run application on Tomcat (in place) from inside IDE

HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive

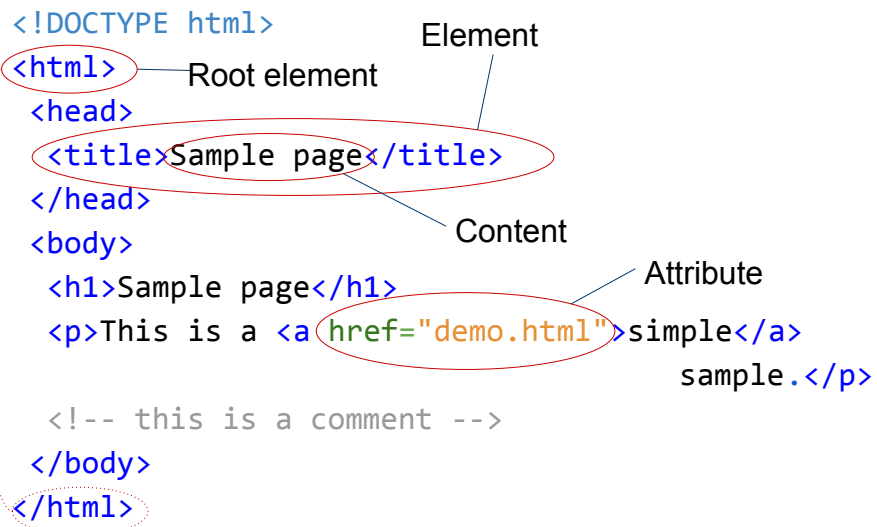


HTML ("HyperText Markup Language") is a language to describe the contents of web documents

Readings

- [About w3c](#)
- [W3C HTML5 recommendation](#)
- [HTML5 Quick introduction](#)
- [HTML5 is not same as HTML4](#)
- [WHATWG](#)

HTML Syntax



The diagram illustrates the structure of an HTML document with the following code and annotations:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <h1>Sample page</h1>
    <p>This is a <a href="demo.html">simple</a>
      sample.</p>
    <!-- this is a comment -->
  </body>
</html>
```

Annotations:

- Element**: Points to the opening tag of an element (e.g., `<html>`).
- Root element**: Points to the `<html>` tag.
- Content**: Points to the text between the opening and closing tags of an element (e.g., `Sample page` in `<title>Sample page</title>`).
- Attribute**: Points to the `href="demo.html"` part of the `<a>` tag.

`<head>` non-visible metadata, `<body>` visible content.

Readings:

- [Basics of HTML](#)
- [HTML data types](#)
- [Tutorials at W3C](#)
- [Tutorials at MDN](#)
- [Tutorial at w3schools](#)
- [HTML Validator](#)

HTML Semantics

4.9.1 The `table` element

Categories:

[Flow content](#).
[Palpable content](#).

Contexts in which this element can be used:

Where [flow content](#) is expected.

Content model:

In this order: optionally a [caption](#) element, followed by zero or more [colgroup](#) elements, followed optionally by a [thead](#) element, followed optionally by a [tbody](#) element, followed by either zero or more [tbody](#) elements or one or more [tr](#) elements, followed optionally by a [tfoot](#) element (but there can only be one [tfoot](#) element child in total), optionally intermixed with one or more [script](#) supporting elements.

Content attributes:

[Global attributes](#)
[border](#)
[sortable](#) - Enables a sorting interface for the table

Tag omission in text/html:

Neither tag is omissible

Allowed ARIA role attribute values:

[Any role value](#).

Allowed ARIA state and property attributes:

[Global aria-* attributes](#)
Any [aria-*](#) attributes applicable to the allowed roles.

DOM interface:

```
IDL
interface HTMLTableElement : HTMLElement {
    attribute HTMLTableCaptionElement? caption;
    HTMLTableCaptionElement? createCaption();
    void deleteCaption();
    attribute HTMLTableSectionElement? tHead;
    HTMLTableSectionElement? createHead();
    void deleteHead();
    attribute HTMLTableSectionElement? tFoot;
    HTMLTableSectionElement? createFoot();
    void deleteFoot();
    readonly attribute HTMLCollection tBodies;
    HTMLCollection createBodies();
    readonly attribute HTMLCollection rows;
    HTMLTableElement insertRow(optional long index = -1);
    void deleteRow(long index);
    attribute DOMString border;
};
```

Element
definition

Elements, attributes, and attribute values in HTML are defined to have certain meanings ([semantics](#)).

- Authors must not use elements, attributes, or attribute values for purposes other than their appropriate intended semantic purpose, as doing so prevents software from correctly processing the page (browsers, search engines).

Readings:

- [Browsing context](#)

Periodic Table of the Elements

joshduck.com/periodic-table.html

The periodic table displays HTML elements grouped by color and function. The elements are arranged in rows and columns, with some elements spanning multiple rows or columns. The colors represent different categories: Root element (green), Metadata and scripting (purple), Embedded content (pink), Text-level semantics (yellow), Grouping content (orange), Forms (light green), Document sections (blue), Tabular data (light orange), and Interactive elements (dark blue).

html																	col	table											
head	span															div	fieldset	form	body	h1	nav	colgroup	tr						
title	a															pre	meter	select	aside	h2	main	caption	td						
meta	rt	dfn	em	i	small	ins	s	br	p	blockquote	legend	optgroup	address	h3	section	menu	th												
base	rp	abbr	time	b	strong	del	kbd	hr	ol	li	dl	label	option	datalist	h4	header	command	tbody											
link	noscript	q	var	sub	mark	bdi	wbr	figcaption	ul	dt	input	output	keygen	h5	article	summary	thead												
style	script	cite	samp	sup	ruby	bdo	code	figure	li	dd	textarea	button	progress	h6	footer	details	tfoot												
																		img	area	map	embed	object	param	source	iframe	canvas	track	audio	video

Legend:

- Root element
- Metadata and scripting
- Embedded content
- Text-level semantics
- Grouping content
- Forms
- Document sections
- Tabular data
- Interactive elements

Readings

- [The elements of HTML](#)
- [MDNs HTML element reference](#)
- W3schools [HTML Reference](#)

Some Global Attributes

Id


```
<section id="content"> ... </section>
```

Class

```
<article class="important"> ... </article>
```

Custom Attribute

```
<li data-length="2m11s">Beyond The Sea</li>
```



No attribute to describe
duration time, so we use **data-***

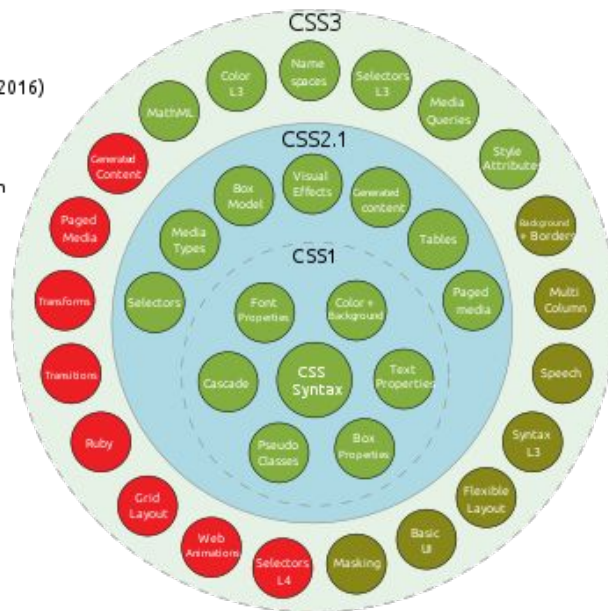
Readings:

- [Global attributes](#)

CSS3

Taxonomy & Status (Dezember 2016)

- W3C Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Obsolete or inactive



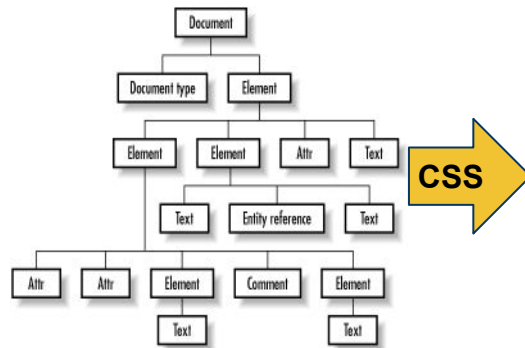
Cascading Style Sheets (CSS) is a language for describing the presentation semantics of markup documents
CSS allows authors to move style information to a separate style sheet resulting in considerably simpler HTML markup

Readings:

- [CSS Roadmap](#) (see selectors far down)
- [CSS at MDN](#)
- [CSS at w3Schools](#)
- [Cascade](#)
-

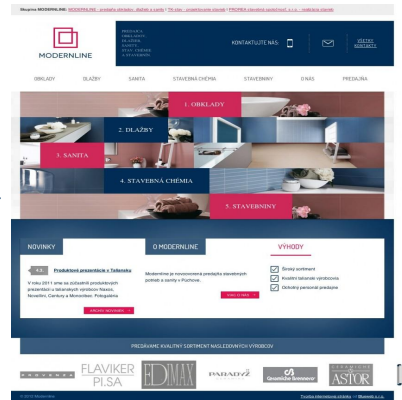
Rendering HTML

The HTML structure



CSS

The “look”



In HTML page

```
<link href="resources/css/default.css" rel="stylesheet" />
```


CSS Selectors and Rules

Element rule

```
h1 { /* A comment */  
    font-size: 34px;  
    font-weight: bold;  
}
```

Id rule (hash)

```
#content {  
    background: #ebe8d9;  
}
```

Class Rule (dot)

```
.important {  
    background: #fdc86c;  
}
```

Selector(s)

CSS: Case insensitive, comments begin with the characters "/*" and end with the characters "*/"

- A [rule set](#) (also called "rule") consists of a selector followed by a declaration block.
- [Selectors](#) are patterns used to determine which style rules apply to elements in the document tree.

Reading

- [CSS Syntax](#)

@ Rules (at rules)

At-rules are instructions or directives to the CSS parser. Not targeting specific HTML elements or attributes.

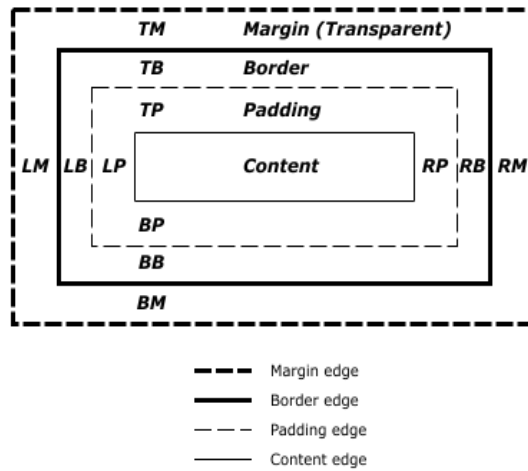
Some are:

- @import: Import one style sheet into another. All @import at-rules must appear before any rules.
- @media: Target media types we specify.
- @font-face: Custom fonts

Readings

- ["At rules"](#)

The Box Model

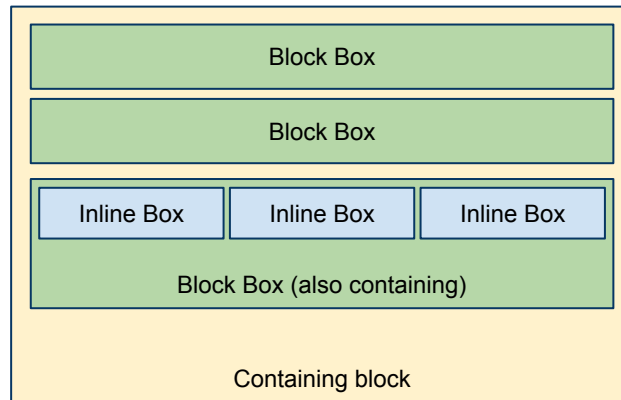


The CSS box model describes the rectangular boxes that are generated for HTML elements in the document tree and laid out according to the visual formatting model.

Readings:

- [Intro to box model](#)
- [CSS units](#)

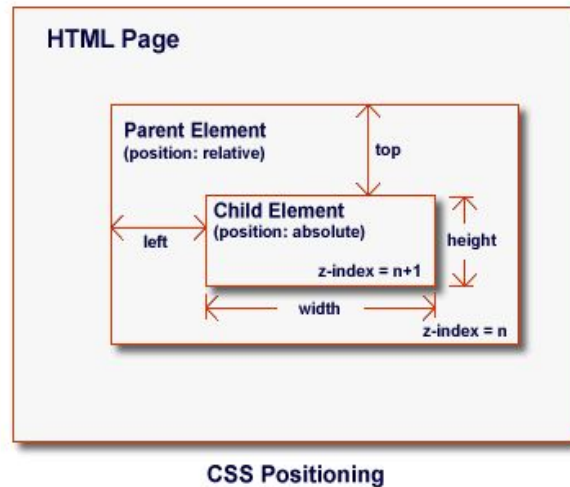
Visual Formatting



Readings

- [Visual Formatting](#)
- [MDN CSS Layout](#)

Positioning Schemas



In CSS 2.2, a box may be laid out according positioning schemes

Readings

- [Positioning CSS3](#)
- [Advanced HTML and CSS](#)

CSS Preprocessors

CSS designed for non programmers

Preprocessors add features for programmers

- Variables
- Nesting
- Mixins
- Extends
- Selection/Iteration
- Math
- Imports

Readings

- [Intro to CSS preprocessors](#)
- [Sass](#)
- [node-sass](#)

Front End User Interface Frameworks

```
<body class="container">  
  
<nav class="navbar navbar-default" role="navigation">  
  <a class="navbar-brand" href="/home">Home</a>  
  ...  
</nav>
```

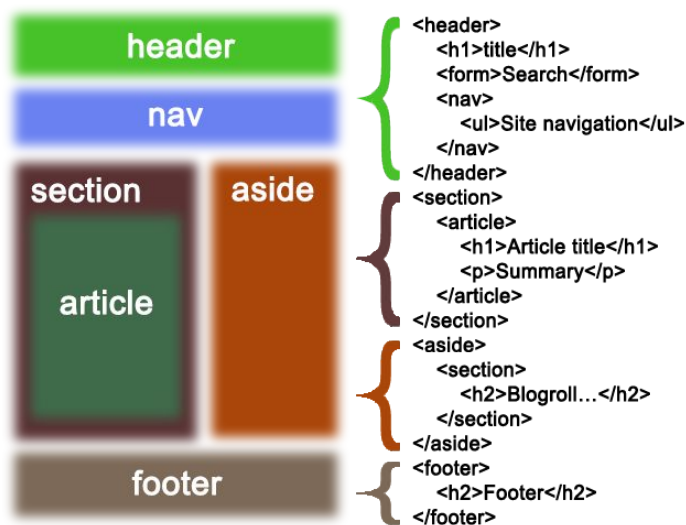


We're not graphical designers ...

- CSS very tedious ...
- Better use some [client side framework](#)
- Possibly Bootstrap
 - Bootstrap [components](#) ...
 - ...mostly using the class-attribute

NOTE: Bootstrap uses some JavaScript so it's not a pure CSS framework

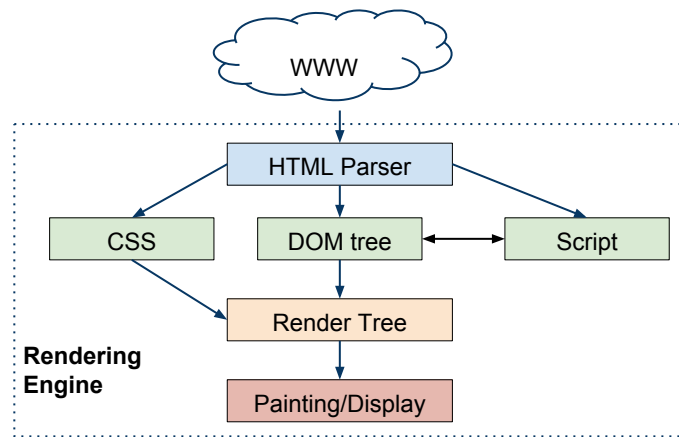
Basic Page Structure with HTML/CSS



Creating a basic page style (layout and styles)

- HTML document using [section tags](#)
- CSS style sheet to normalize the [default browser style](#) ([normalize.css](#) or similar recommended)
- CSS stylesheet (default.css) containing positioning rules for the section tags ...
- ... and colors, fonts, etc. for remaining tags

Page Loading



Readings:

- [How Browsers Work](#)