EDAF65
HTML

Per Andersson

Lund University
http://cs.lth.se/per_andersson/

January 31, 2018

Innehåll: *HTML, CSS, DOM, JavaScript*

Webben

Webben byggs upp av olika dokument:

- HTML - beskriver innehållet
- CSS - layout
- JavaScript - dynamiskt beteende

HTML

- många versioner
- resultat kan bero på webbläsare och dess version
- slarvig syntax
- kan även innehålla layout och JavaScript
- root-dokument, inkluderar CSS och JavaScript

HTML

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello World</title>
    <link rel="stylesheet" href="css/styles.css">
    <script src="my-awsome-code.js"></script>
    <base href="https://www.cs.lth.se/eda095/">
  </head>

  <body>
    <h1>Hello World</h1>
    <p id="my-blue-box">My awsome page.
  </body>
</html>
```

HTML - taggar

Semantiska taggar

`<h1>`, `<h2>`, `<p>`, `<abbr>`, `<code>`, `<samp>`, `<kbd>`, `<var>`, `<footer>`, `<header>`, `<details>`, `<nav>`...

Structur

`<table>`, `<ul>`, `<ol>`, `<div>`, `<span>`...

Funktionalitet

`<form>`, `<input>`, `<select>`, `<button>`, `<a>`...

För att lära dig mer om HTML-taggar se
https://www.w3schools.com/tags/default.asp

HTML - innehåll

Data till taggar:

- innuti, t.ex. `<h1>Min Rubrik</h1>`
  - visas på skärmen
  - text, och ofta andra element
  - kan innehålla flera element
- attribut, t.ex. `<a href="http://cs.lth.se">min länk</a>`
  - visas inte på skärmen
  - alltid text
  - alltid ett värde
  - `id` unikt för varje element
    används för att identifiera element

tag + innehåll $\approx$ element

Layout

Rendering av webbsidor

- HTML-element har speciella attribut som beskriver hur de renderas
- attributen kan ändras för att ge en sida ett annat utseende
- tilldela attributen genom:
  - `style` attributet i HTML-elementen
  - Cascading Style Sheets (CSS) är ett språk
    css + `class` attributet

exempel på vad css kan åstadkomma

CSS

Cascading Style Sheets

- samla layout information på ett ställe
- möjliggör responsiv design - anpassa efter skrämen
- en uppsättning regler
- varje regel består av:
  - urval
  - deklaration
- deklarationen appliceras på alla HTML-element som matchar urvalet

syntax:

```
urval :  { property1:  value1; property2:  value2;}
```

CSS - urval

CSS-urval:

- alla instanser ett element, t.ex. `<div>`
- alla element som har en klass, t.ex. `<div class="my-style">`
- ett element med ett givet `id`, t.ex. `<div id="my-tag">`
- `pseudo-klass`, t.ex. `focus, hover, visited, ...`
- `pseudo-element`, t.ex. `nth-child(2), only-child, ...`
- `attributvärden`, t.ex. `[title~="flower"], ...`

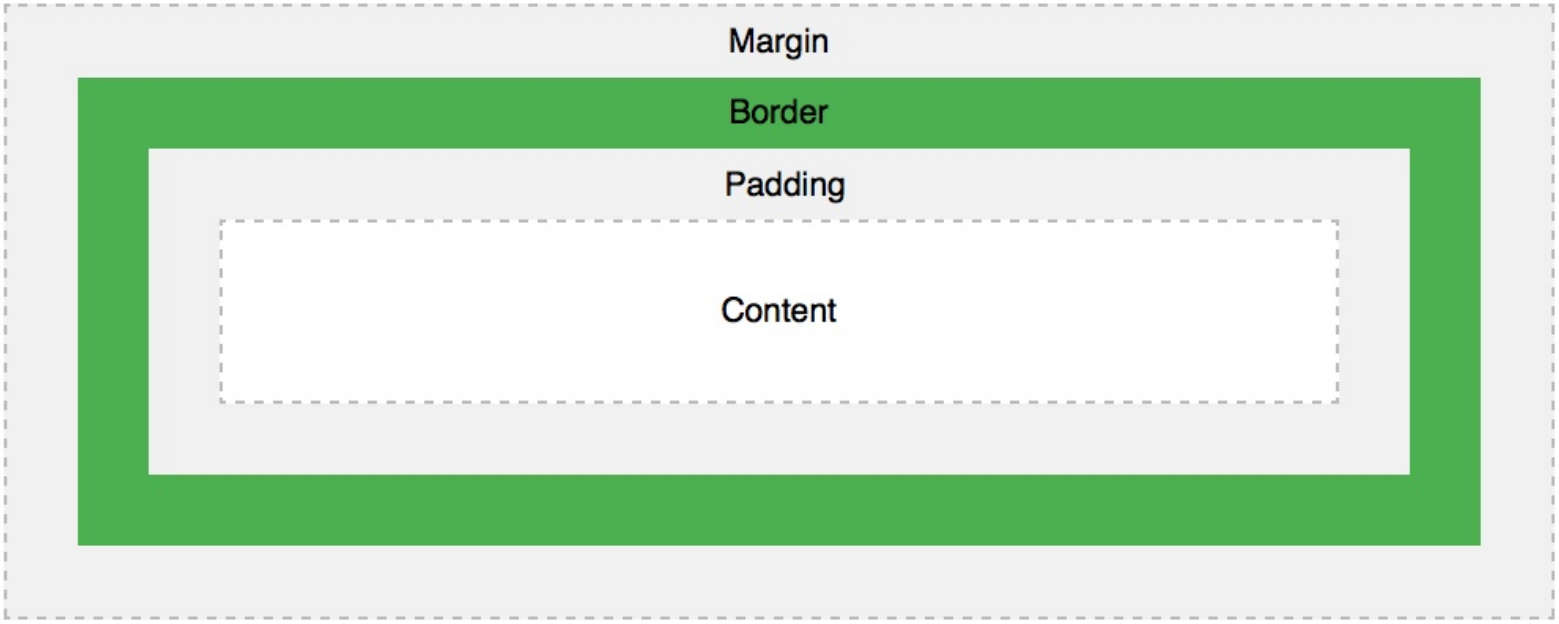
Urvalen kan kombineras.

CSS - exempel

Example:

```
// element <div>some text</div>
div { color: blue; border: 1px; }
// id, <p id="my-blue-box">
#my-blue-box { background-color: lightblue; }
// klass, <div class="center">
.center { text-align: center; color: red; }
// element och klass, <p class="center">
p.center { text-align: center; color: green; }
// inuti
div p { text-align: center; color: green; }
// alla p vars förälder är div
div > p { text-align: center; color: green; }
// direkt efter
div + p { text-align: center; color: green; }
// händelser
a:hover { background-color: lightblue; }
```

Box Modellen



Layout

CSS units

- `em`
- `vw`, `vh`, `vmin`, `vmax` - 1/100 of viewport
- `cm`, `mm`, `pt`
- `px`

Layout

CSS Properties for layout

- `display` - `block, inline, none, flex, ...`
- `visibility` - `visible, hidden, ...`
- `position` - `static, absolute, fixed, relative, ...`
- `overflow` - `visible, hidden, scroll, auto, ...`
- `z-index` - `auto,`*number*

Webkit Mozilla

Över tiden har samma attribut dykt upp med olika namn. av kompatibilitetsskäl bör flera attributnamn användas:

- `box-shadow`
- `-webkit-box-shadow`
- `-moz-box-shadow`

DOM

Document Object Model

- en webb-sida är ett träd
- varje HTML-element blir en nod i trädet
- HTML-attributen är attribut i noderna
- `<html>` är roten i trädet
- trädet nås via JavaScript-interfacet `Document`
- varje nod i trädet implementera JavaScript-interfacet `Element`

Händelse

- design patterns
  - observer-pattern
  - document-view
- DOM:en är modelllen
- attribut i elementen
- call-back-metoder
- t.ex. `<p id="demo" onclick='document.getElementById("demo").style.color = "red"'>`

server - klient

Servern

- implementerar HTTP-protokollet
- skickar filer (statisk hemsida)
- skapar dynamiska hemsidor
  - skapas ofta från databaser
  - php - apache
  - Java servlet - tomcat
  - JavaScript - node

Klienten - webbläsaren

- hämtar HTML, CSS m.m .från servern
- bygger en DOM
- renderar sidor från DOM:en
- JavaScript run-time för dynamiskt beteende

HTML och Java

Bra

- Java har bra stöd för HTML via bibliotek
- Java används på server-sidan
- Java EE
- servlets - `main()` för att skapa ett HTML-dokument

Problem

- Java stöds inte i webbläsare (java applet)
- begränskningar att rendera HTML från Java

jsoup

- `jsoup.org` är en mycket populär HTML parser.
- externt bibliotek.
- bygger en DOM från `String`, `URL` eller `File`
- navigera DOM, CSS urval

```
Document doc1 = Jsoup.parse(myHtmlString);
Document doc2 = Jsoup.connect("http://example.com/").get();
Document doc3 = Jsoup.parse(myHtmlFile, "UTF-8",
    "http://example.com/");
```

jsoup

```
public class First {
    void parse() {
        String html = "<html><head><title>First
            parse</title></head>"
                + "<body><p>Parsed HTML into a
                    doc.</p></body></html>";
        Document doc = Jsoup.parse(html);
        System.out.println(doc);
    }

    public static void main(String[] args) {
        First first = new First();
        first.parse();
    }
}
```

Extracting links with jsoup

```
void parse() throws IOException {
    URL url = new URL("http://cs.lth.se/eda095/");
    InputStream is = url.openStream();
    Document doc = Jsoup.parse(is, "UTF-8", "http://cs.lth.se/");
    Elements base = doc.getElementsByTag("base");
    System.out.println("Base : " + base);
    Elements links = doc.getElementsByTag("a");
    for (Element link : links) {
        String linkHref = link.attr("href");
        String linkAbsHref = link.attr("abs:href");
        String linkText = link.text();
        System.out.println("href: " + linkHref + "abshref: "
            + linkAbsHref + " text: " + linkText);
    }
    is.close();
}
```

Extracting Text with jsoup

```
public String html2text(String html) {
    return Jsoup.parse(html).text();
 }
```

Extracting Headers with jsoup

```
public void outline(String html) {
    Document doc = Jsoup.parse(html);
    Elements els = doc.select("h1, h2, h3, h4, h5, h6");
    for (Element el : els) {
        //Last char
        Integer last = new Integer((el.tagName()).
            substring(el.tagName().length() - 1));
        String prefix = "";
        for (int i = 0; i < last; i++) {
            prefix += '\t';
        }
        System.out.println(prefix + el.text());
    }
```