Internet Applications Design and Implementation

2015 - 2016 - 1st edition

(4 - Client applications)

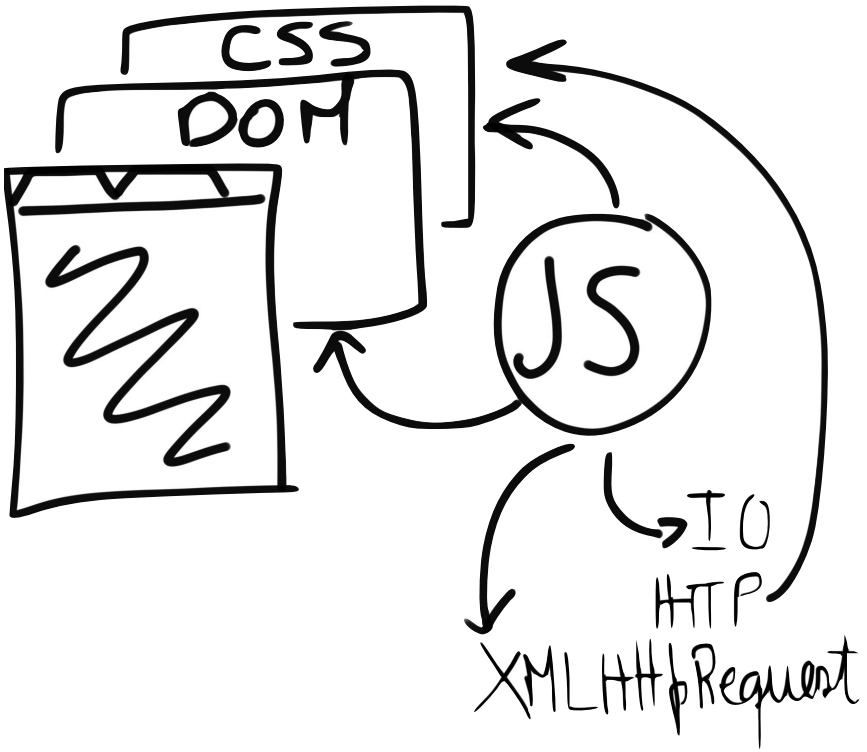
MIEI - Integrated Master in Computer Science and Informatics
Specialization block

João Costa Seco (joao.seco@fct.unl.pt)
Jácome Cunha (jacome@fct.unl.pt)

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

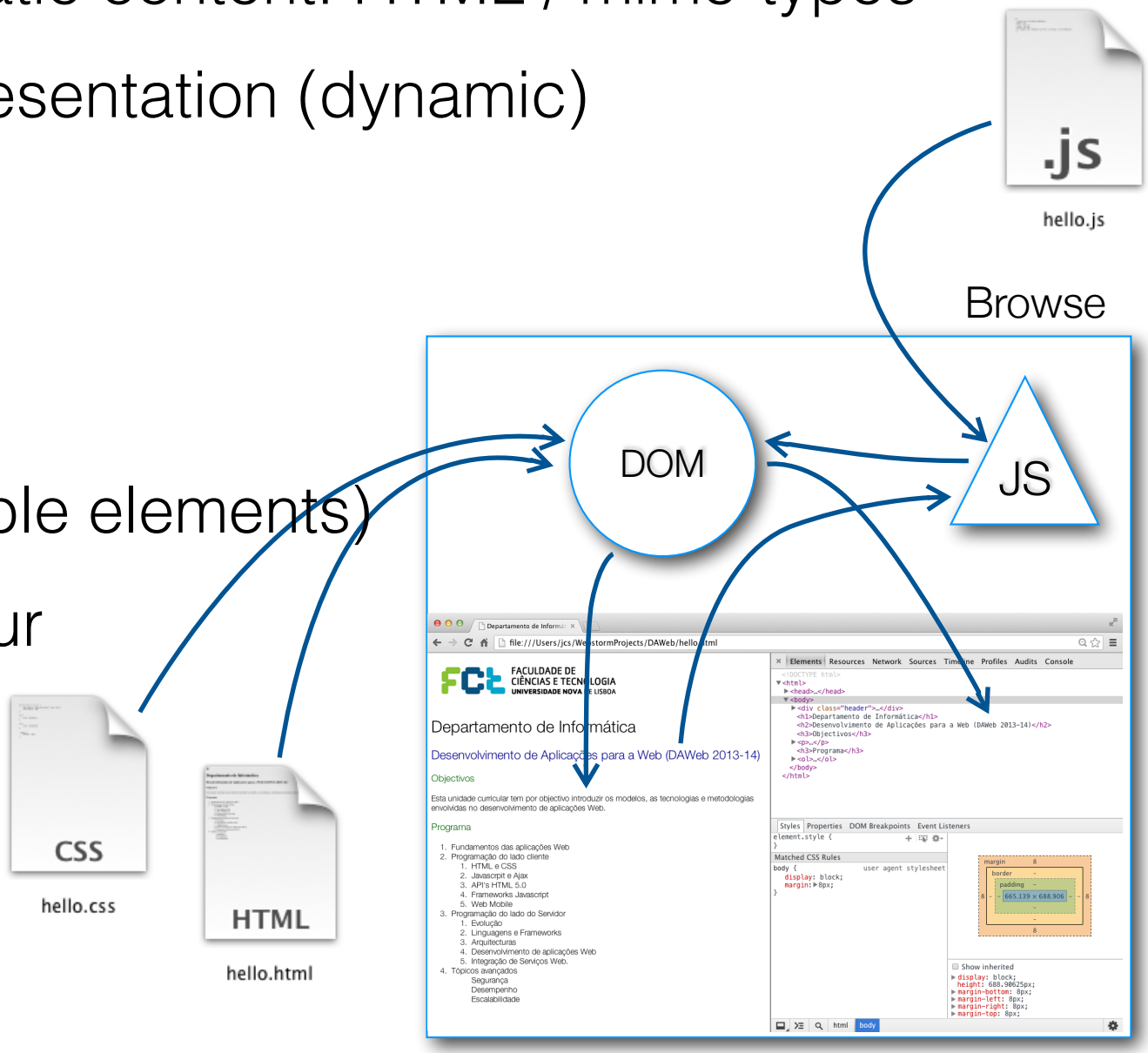
Web client architecture - Virtual Machine

- Browser (HTML5)
  - HTML (structure and semantics)
  - CSS (style and UI behaviour)
  - JS + AJAX,
    Socket interfaces (behaviour)
  - DOM
    (the supporting data structure)
  - UI Events & callbacks
    (mechanism for dynamic structuring of behaviour)



Browser logic architecture

- Source files with static content: HTML / mime-types
- DOM abstract representation (dynamic)
  - Elements
  - Style annotations
  - Event handlers
- User-interface (visible elements)
- Javascript behaviour



Client Web (mobile) Applications

- Running on a Browser
  - Cross-platform compatibility (HTML, CSS, Javascript)
  - Slow interpreted code
  - Limited capabilities (no camera, no gyro, no resources, sandboxed)

- Native code
  - All device capabilities available
  - Fast and efficient applications
  - Proprietary languages and APIs (Java/Android ADT, Swift, C#/C++ UWP)

- Running on a shell (PhoneGap, etc)
  - Cross-platform compatibility (HTML, CSS, Javascript)
  - All device capabilities available
  - Precompiled and packaged code (fast)

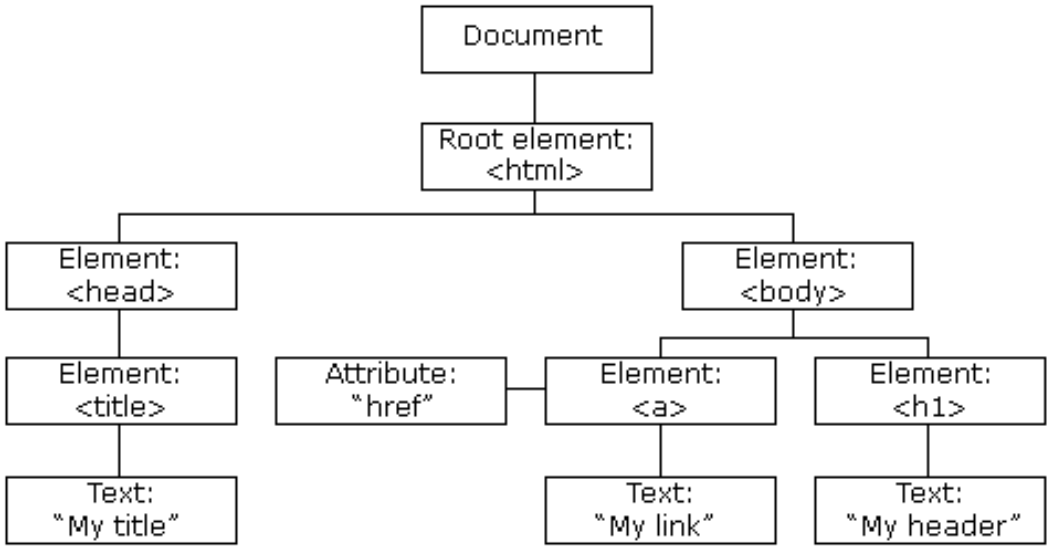
HTML5

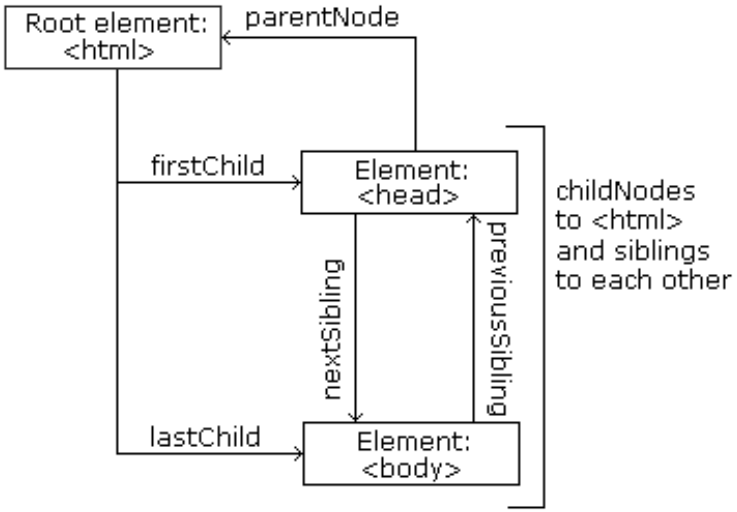
HTML5 comes as a package

- HTML5 is divided into:
  - HTML code to define structure and basic content to web pages. (new tags like `<video>` `<audio>` `<svg>`)
  - CSS (Cascading style sheets) to define the appearance, layout, and some behaviour of DOM elements
  - Javascript is used to define behaviour associated to the webpage elements. Behaviour includes dynamic construction of pages.
- The supporting structure for all these elements is the DOM.

DOM - Document Object Model

- Standard (W3C) specification of the API of the data structure representing a structured document (HTML, XML, etc.).
- Defines objects and properties for all elements of a page and methods to access and modify them.
- Allows the dynamic retrieval, constructions, and modification of HTML elements in a web page.





http://www.w3schools.com/htmldom

DOM - Document Object Model

- DOM is a convention to represent HTML (XML) documents in a tree of objects. It's the dynamic supporting structure of a web page.
- Each node of the DOM has a particular structure (attributes) and associated behaviour (methods).
- DOM objects can be created, accessed, and modified using:
  - HTML: the static structure of DOM objects
  - CSS: using (rich) object selectors
  - Javascript: by traversing the DOM tree using the DOM API.

HTML

- Each HTML element has a tag
  - <body>, …
- Content
  - This is a paragraph
- and a set of attributes:
  - Generic: set on any HTML element. (ex: id, class, hidden…)
  - Particular: img/src, option/value, etc.
  - Events: dependent on the kind of element
    body / onload, button / onclick, etc.

HTML essentials in one slide

- <!DOCTYPE html>
- <html></html> root element
- <head></head> section to declare meta-information, stylesheets and scripts
- <body></body> section to define content
- <p> <h1>… block elements
-   <i> inline elements
- < > é escaped characters

General structure - <html>


```
<!DOCTYPE html>
<html>
  <head>
      ...
  </head>

  <body>
      ...
  </body>
</html>
```

Invisible info - <head>


```
<head>
   <!-- head defines invisible information about the page -->
   <!-- Meta information (machine parseable) can also be defined -->
   <meta charset="UTF-8">
   <meta name="description" content="My Home Library Web Site">
   <meta name="keywords" content="HTML,CSS,XML,JavaScript">
   <meta name="author" content="João Costa Seco">
   <meta http-equiv="refresh" content="30">

   <!-- The title of the browser window (mandatory) -->
   <title>My home library</title>


   <!-- To define the base URL for all relative links -->
   <base href="ctp.di.fct.unl.pt/~jcs/daweb14-15">

   <!-- To define the style of the elements of the page -->
   <style>
       * { font-family: sans-serif;}
   </style>
   <!-- Or import an external file with the stylesheet -->
   <link href="screen.css" type="text/css" rel="stylesheet" media="screen"/>

   <!-- Custom behaviour can be defined using Javascript -->
   <!-- Directly in the HTML code -->
   <script>
       function Hello() { alert("Hello World."); }
   </script>
   <!-- Or by importing a script file -->
   <script src="books.js"></script>
</head>
```

Visible info - <body>


```
<body>
   <article>
      <h1>My home library in a mouse click.</h2>
      This is the summary of the books in my bookshelf. I divide it in cat
      and then present the sinopsis of each one.

      <h2>Fiction</h2>

      <h3>The War of Thrones</h3>
      <ul>
      <li><b>Volume I:</b> A Game of Thrones (1996)</li>
      <p>Summers span decades. Winter can last a lifetime. And the struggle for
      the Iron Throne has begun.
      <p> It will stretch from the south, where heat breeds plot, lusts and
      intrigues; to the vast and savage eastern lands; all the way to the frozen
      north, where an 800-foot wall of ice protects the kingdom from the dark
      forces that lie beyond. Kings and queens, knights and renegades, liars,
      lords and honest men… all will play the Game of Thrones.</p>
      <p> Winter is coming…</p>
      <li><b>Volume II:</b> A Clash of Kings (1998)</li>
      <p>Throughout Westeros, the cold winds are rising.</p>
      <p>From the ancient citadel of Dragonstone to the forbidding lands of
      Winterfell, chaos reigns as pretenders to the Iron Throne of the Seven
      Kingdoms stake their claims through tempest, turmoil and war.</p>
      <p>As a prophecy of doom cuts across the sky – a comet the colour of blood
      and flame – five factions struggle for control of a divided land. Brother
      plots against brother and the dead rise to walk in the night.</p>
      <p> Against a backdrop of incest, fratricide, alchemy and murder, the price
      of glory is measured in blood.</p>
```

Textual block elements


```
<html>
    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
        <p>This is another paragraph.</p>
    </body>
</html>
```

Textual inline elements


```
<html>
    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph. <span>This is a span</span> <b>This is a strong tag</b>
        <a>This is an anchor</a> </p>
        <p>This is another paragraph.</p>
    </body>
</html>
```

Structural block elements


```
<html>
    <body>
        <nav>This is a navigation section</nav>
        <article>This is a document section.
            <section>This is another section. </section>
            <section>This is another section. </section>
        </article>
        <article>This is a document section.
            <section>This is another section. </section>
            <section>This is another section. </section>
        </article>
    </body>
```

Block elements

This is a block element.

This is a block element.

This is a block element.

Inline elements


```
<span>This is </span> <span> an inline </span> <span> element.
</span> <span>This is </span> <span> an inline </span> <span>
element. </span> <span>This is </span> <span> an inline </span>
<span> element. </span>
```

Attributes

- HTML elements can be modified through attributes

```
<a href="http://www.w3schools.com">This is a link</a>

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

<div style="border:1px">This is a section.</div>
```

HTML <button> Tag

« Previous Complete HTML Reference Next »

Example

A clickable button is marked up as follows:

```
<button type="button">Click Me!</button>
```

Try it Yourself »

Attributes

= New in HTML5.

Attribute	Value	Description
autofocus	autofocus	Specifies that a button should automatically get focus when the page loads
disabled	disabled	Specifies that a button should be disabled
form	*form_id*	Specifies one or more forms the button belongs to

HTML - Basic Interaction

- Links
  - Produce a GET request to the given URL and replace the entire content of the DOM
- Forms
  - Pre-defined controls (buttons)
  - Produce a request and expects a response document (HTML)
  - replaces the entire content of the DOM (depends on _target)
  - method attribute defines the request type (GET/POST)
  - called URL format depends on the used method (body/query string)

HTML - FORMS


```
<form action="/my-handling-form-page" method="post">
    <div>
        <label for="name">Name:</label>
        <input type="text" id="name" />
    </div>
    <div>
        <label for="mail">E-mail:</label>
        <input type="email" id="mail" />
    </div>
    <div>
        <label for="msg">Message:</label>
        <textarea id="msg"></textarea>
    </div>

    <div class="button">
        <button type="submit">Send your message</button>
    </div>
</form>
```

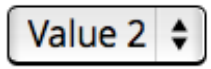
HTML

- Inputs

```
<input type="text" name="input" value="Type here">
```



```
<button name="button">Click me</button>
```




```
<select name="select">
  <option value="value1">Value 1</option>
  <option value="value2" selected>Value 2</option>
  <option value="value3">Value 3</option>
</select>
```

- Other interface elements (output)

```
<p>Heat the oven to <meter min="200" max="500" value="350">350 degrees</meter>.</p>
```

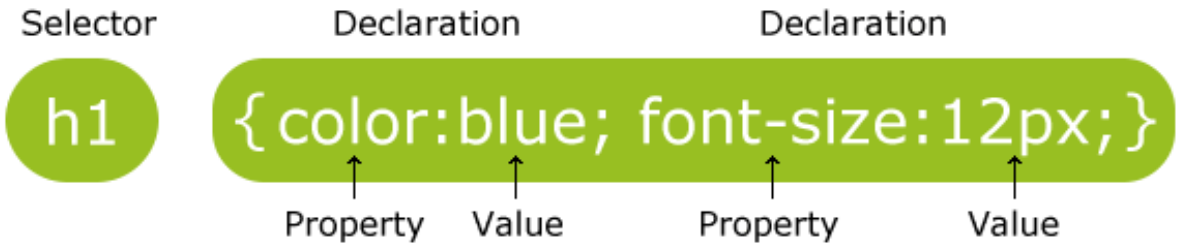
```
<progress value="70" max="100">70 %</progress>
```

CSS - Cascading Style Sheets

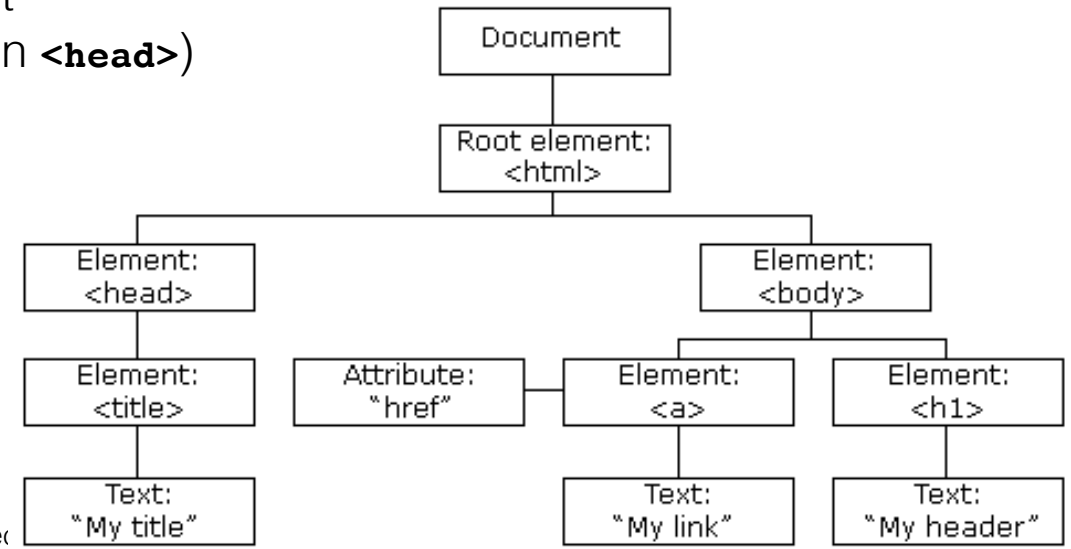
- Describes the presentation of a document defined using a markup language (HTML, XML)
- Decouples presentation from the document structure and behaviour of a web application.
- Based on declarative object descriptors and object attributes (and values) - Rules
- Based on general application priorities on conflicting cases
- Includes basic presentation behaviour (animations)

Decorating the DOM

- A set of rules is applied to the DOM



- rule = selector + style definitions with the form `property:value`
- Rules are applied by the following order
  - Browser default
  - External and Internal Style Sheet (following the order in the section **<head>**)
  - Inline Style (HTML element)
- Propagate hierarchically through the DOM (it depends)...



CSS object selectors

- all elements:
- by HTML tag
- by id
- by class
- composed selector
- composed selector
- pseudo-classes

```
* { font-family: sans-serif; }

h1, h2 { color: #f0f0f0; }

#form1 { border: solid 1px; }

.listitem { list-style: none; }

#booklist li { background-color: "rbg(100,80,10)"

ul.menu > li { margin: 10px; padding: 5px;}

.menuitem:hover { background-color: blue; }
```

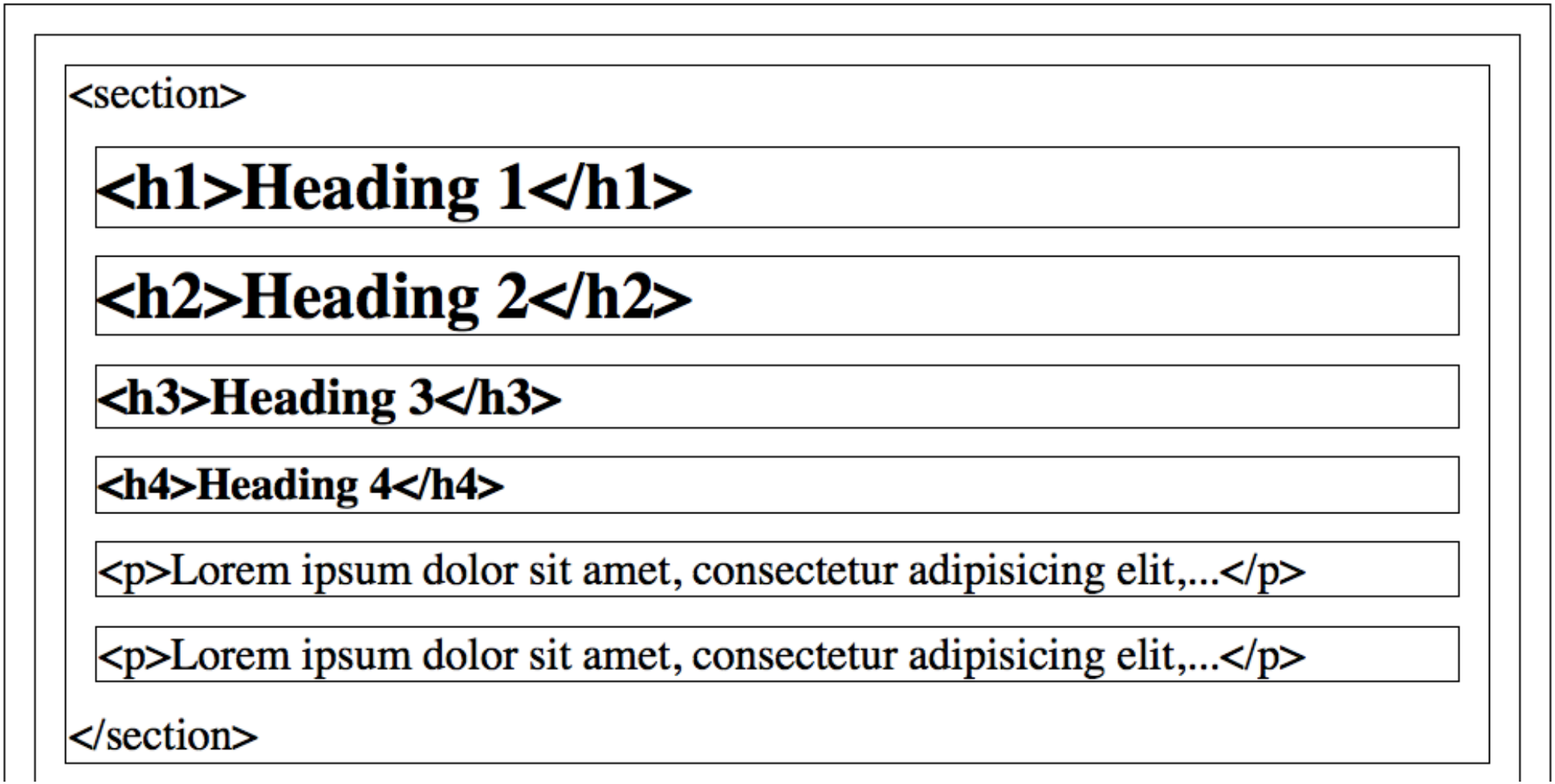
CSS Selectors

from http://www.w3schools.com/cssref/css_selectors.asp

Selector	Example	Example description	CSS
.class	.intro	Selects all elements with class="intro"	1
#id	#firstname	Selects the element with id="firstname"	1
*	*	Selects all elements	2
element	p	Selects all <p> elements	1
element,element	div, p	Selects all elements and all <p> elements	1
element element	div p	Selects all <p> elements inside elements	1
element>element	div > p	Selects all <p> elements where the parent is a element	2
element+element	div + p	Selects all <p> elements that are placed immediately after elements	2
element1~element2	p ~ ul	Selects every element that are preceded by a <p> element	3
[*attribute*]	[target]	Selects all elements with a target attribute	2
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"	2
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"	2
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value starting with "en"	2
[*attribute^=value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"	3

CSS Selectors

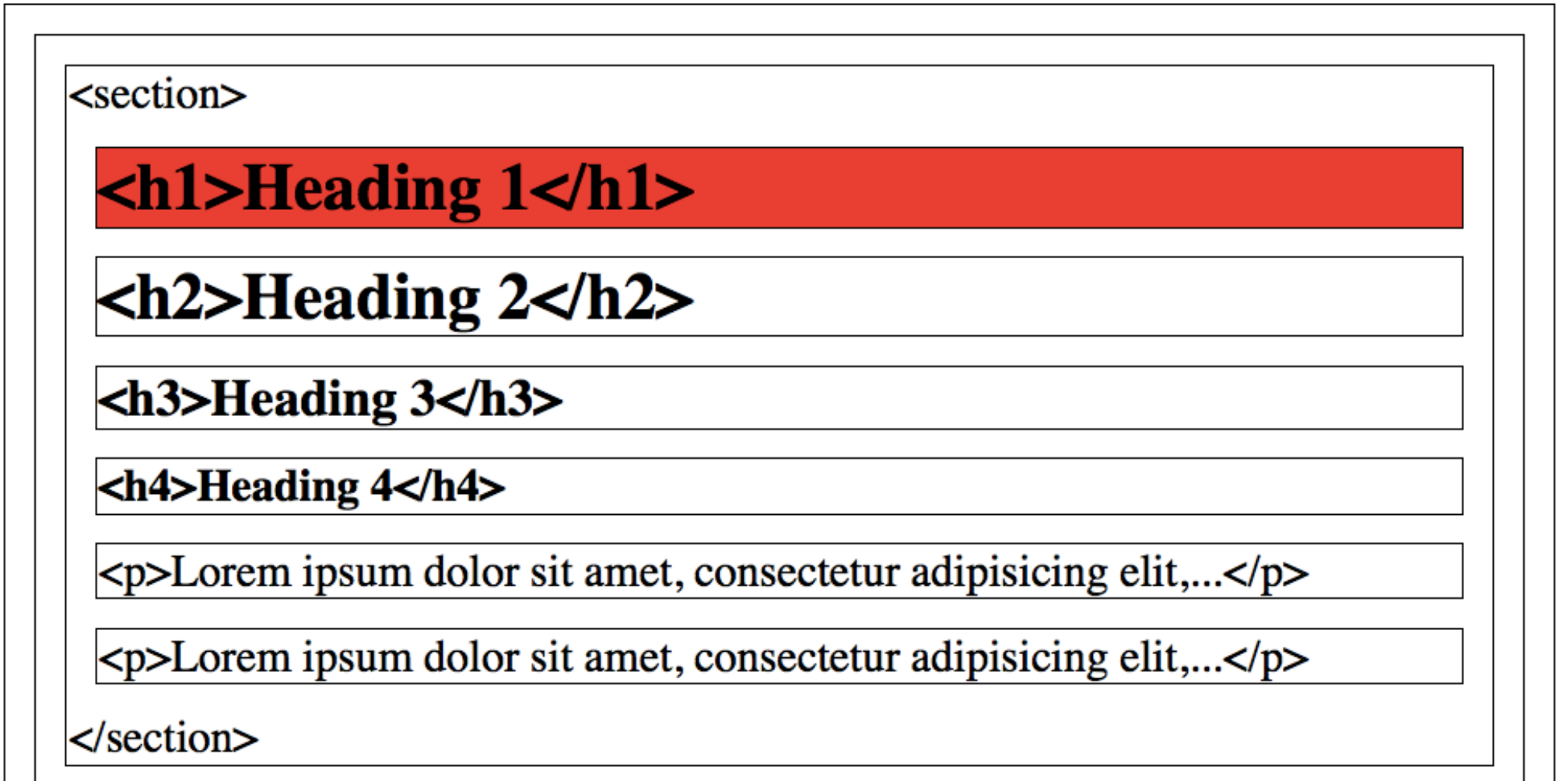
- h1 {background-color: red;}



CSS Selectors

- h1 {background-color: red;}

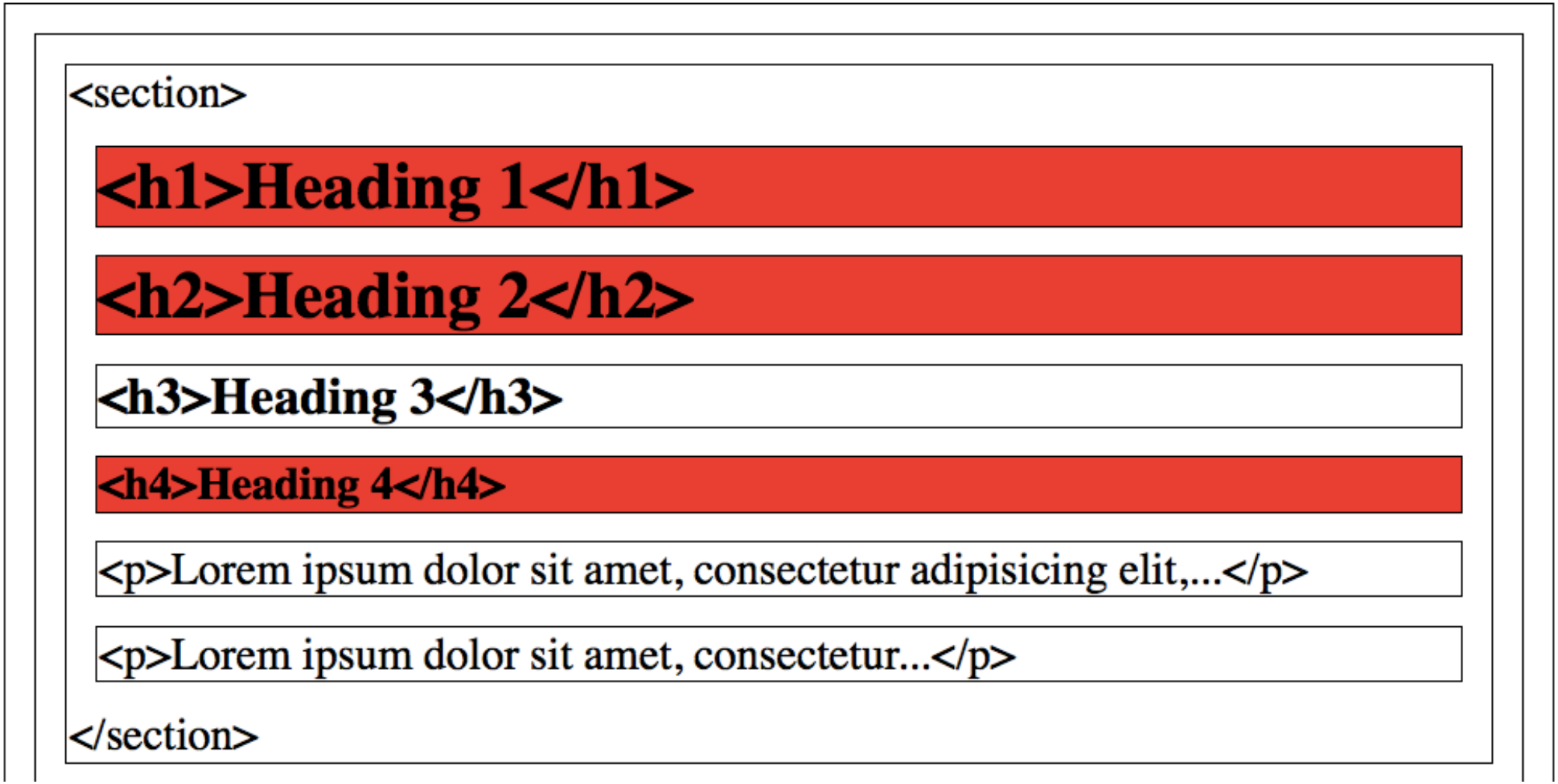
```
h1 {background-color: red;}
```



CSS Selectors

- h1 {background-color: red;}

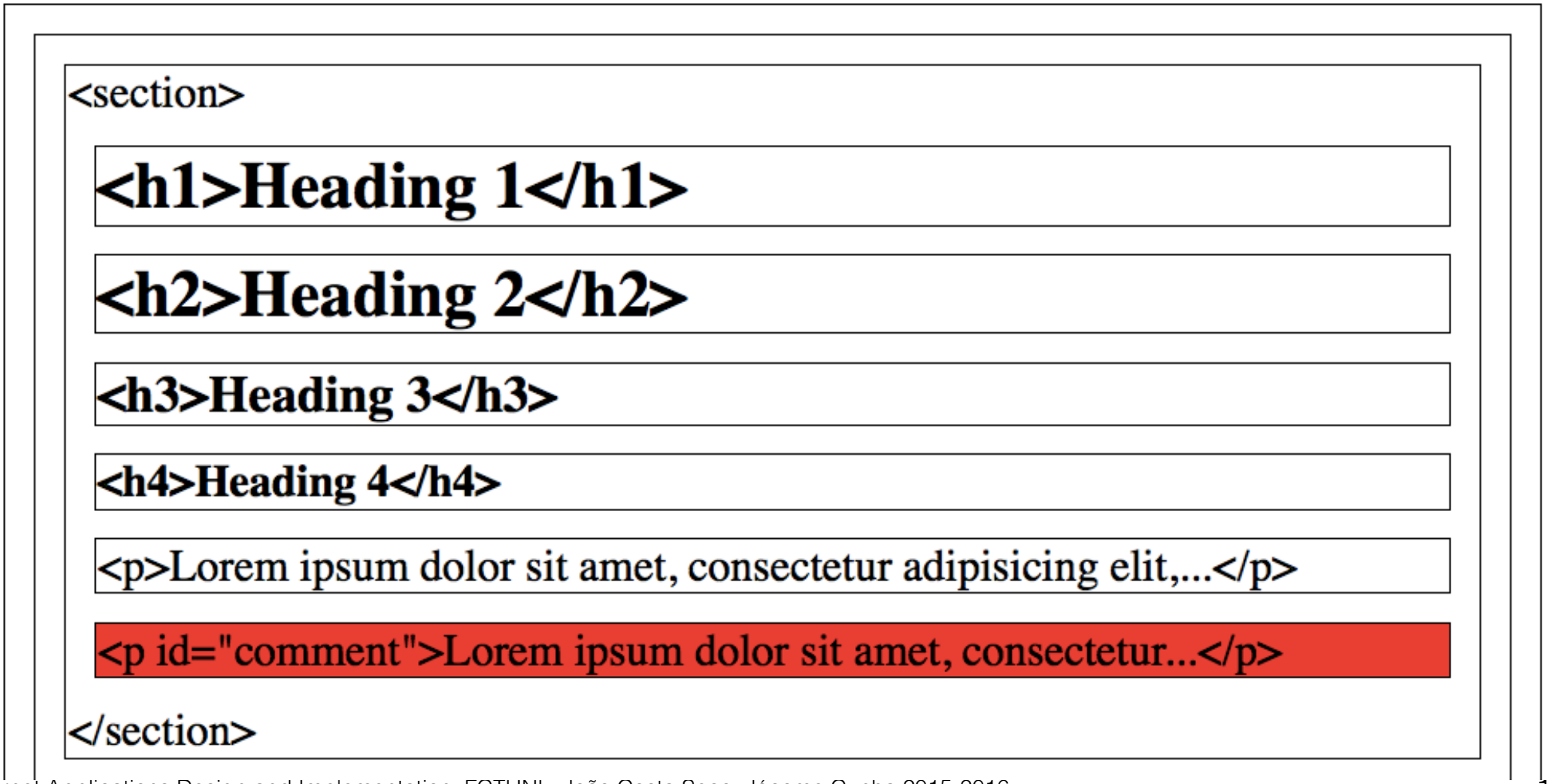
```
h1, h2, h4 {background-color: red;}
```



CSS Selectors

- h1 {background-color: red;}

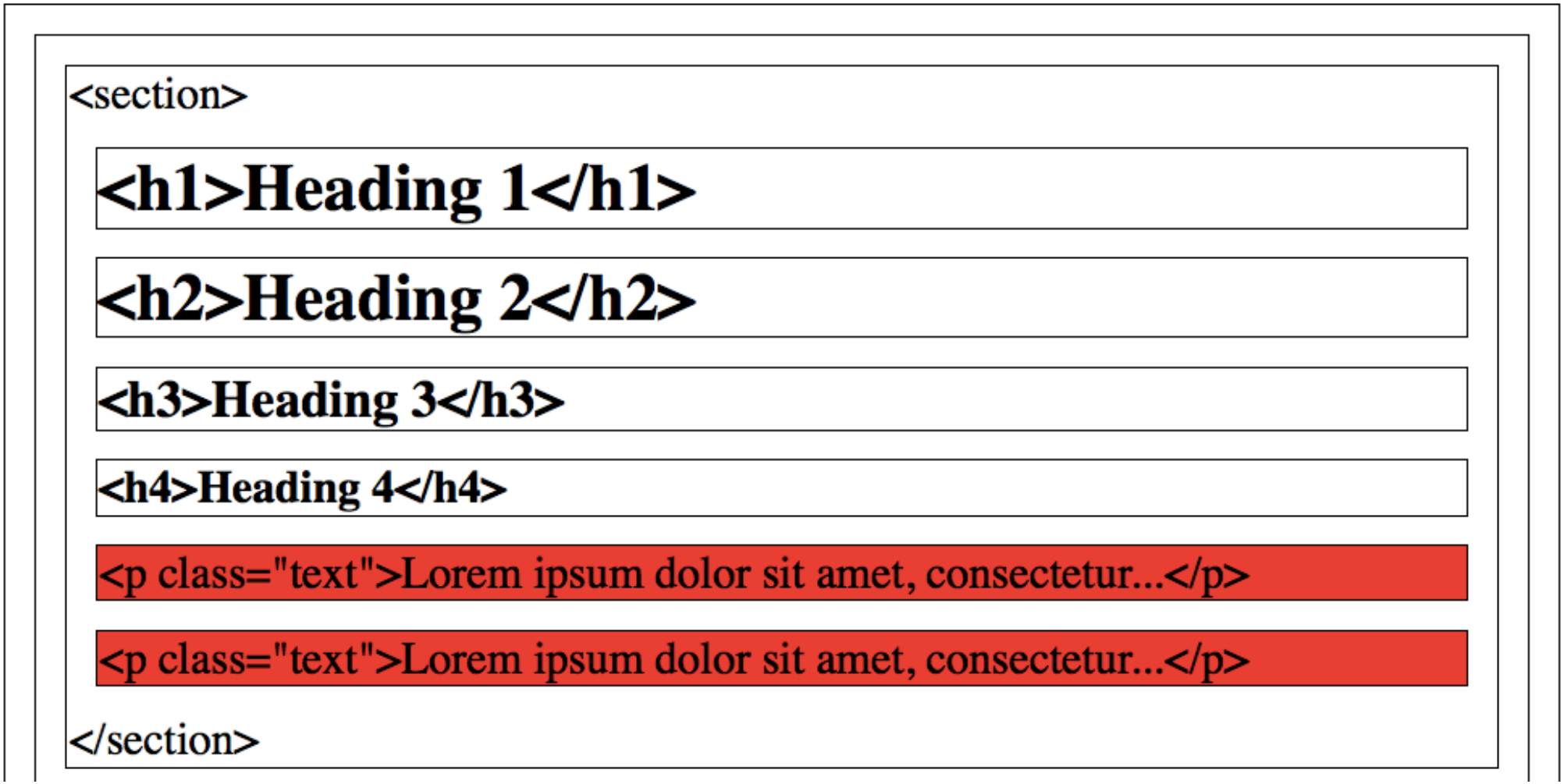
```
#comment {background-color: red;}
```



CSS Selectors

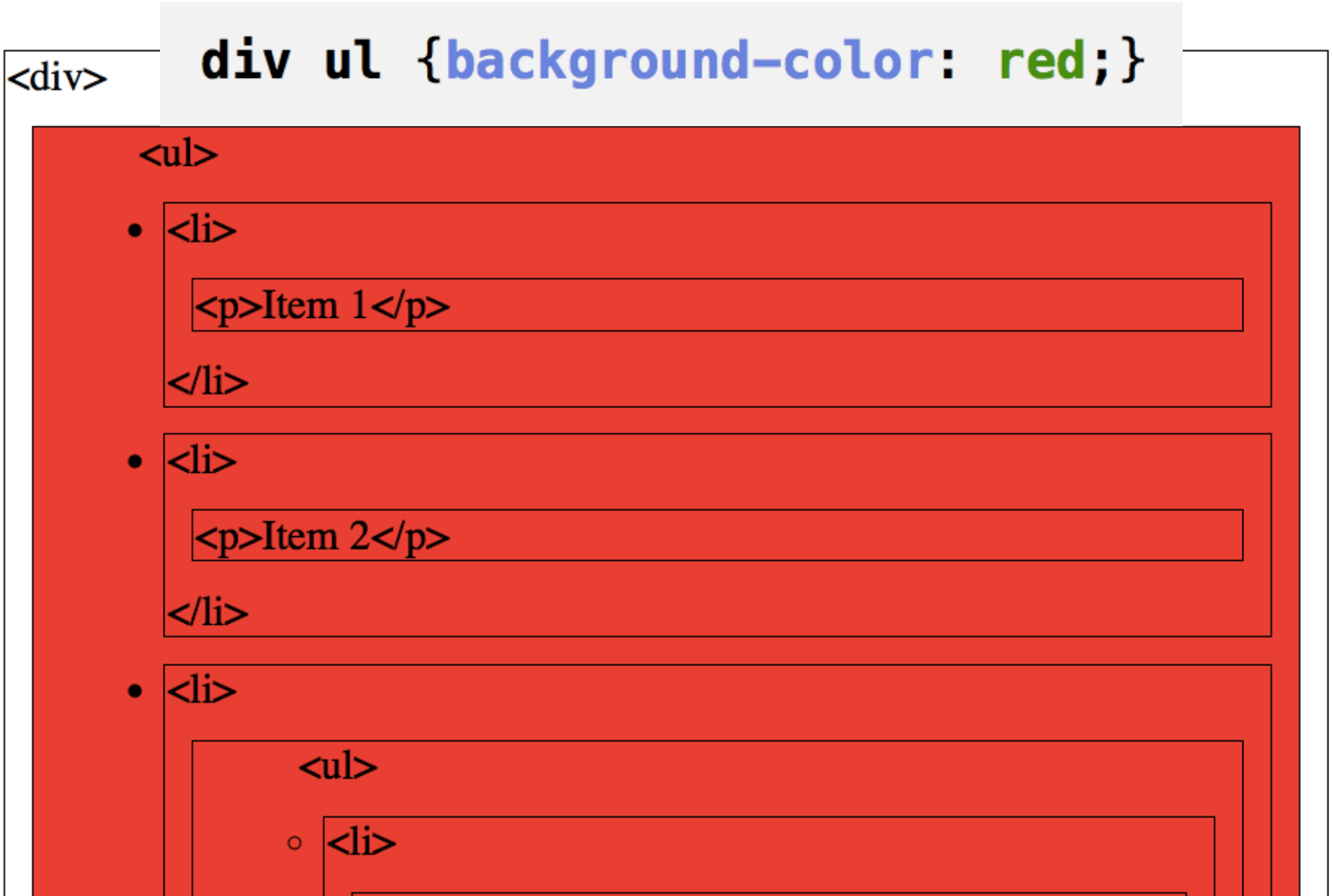
- h1 {background-color: red;}

```
.text {background-color: red;}
```



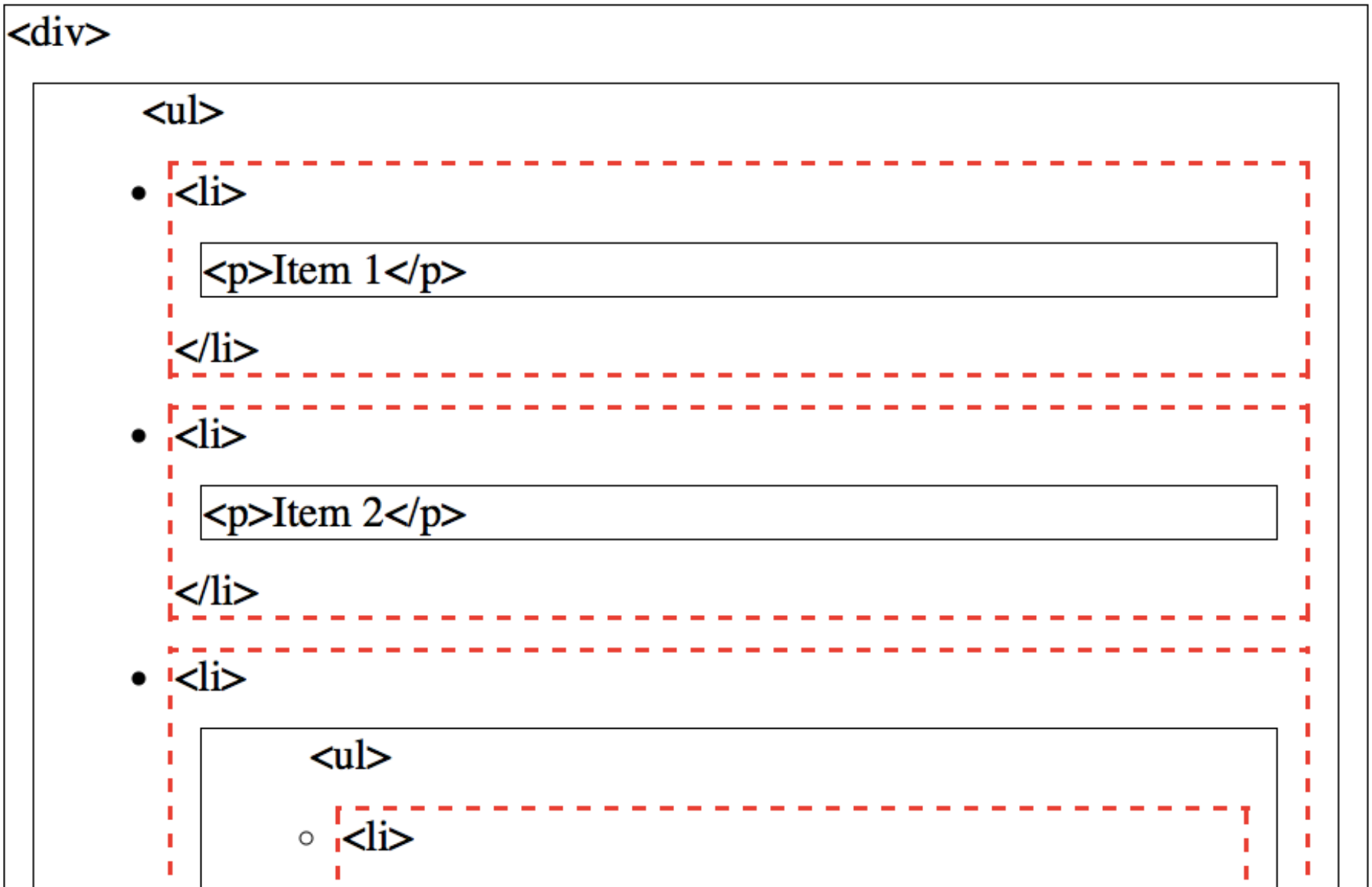
CSS Selectors

```
div ul {background-color: red;}
```



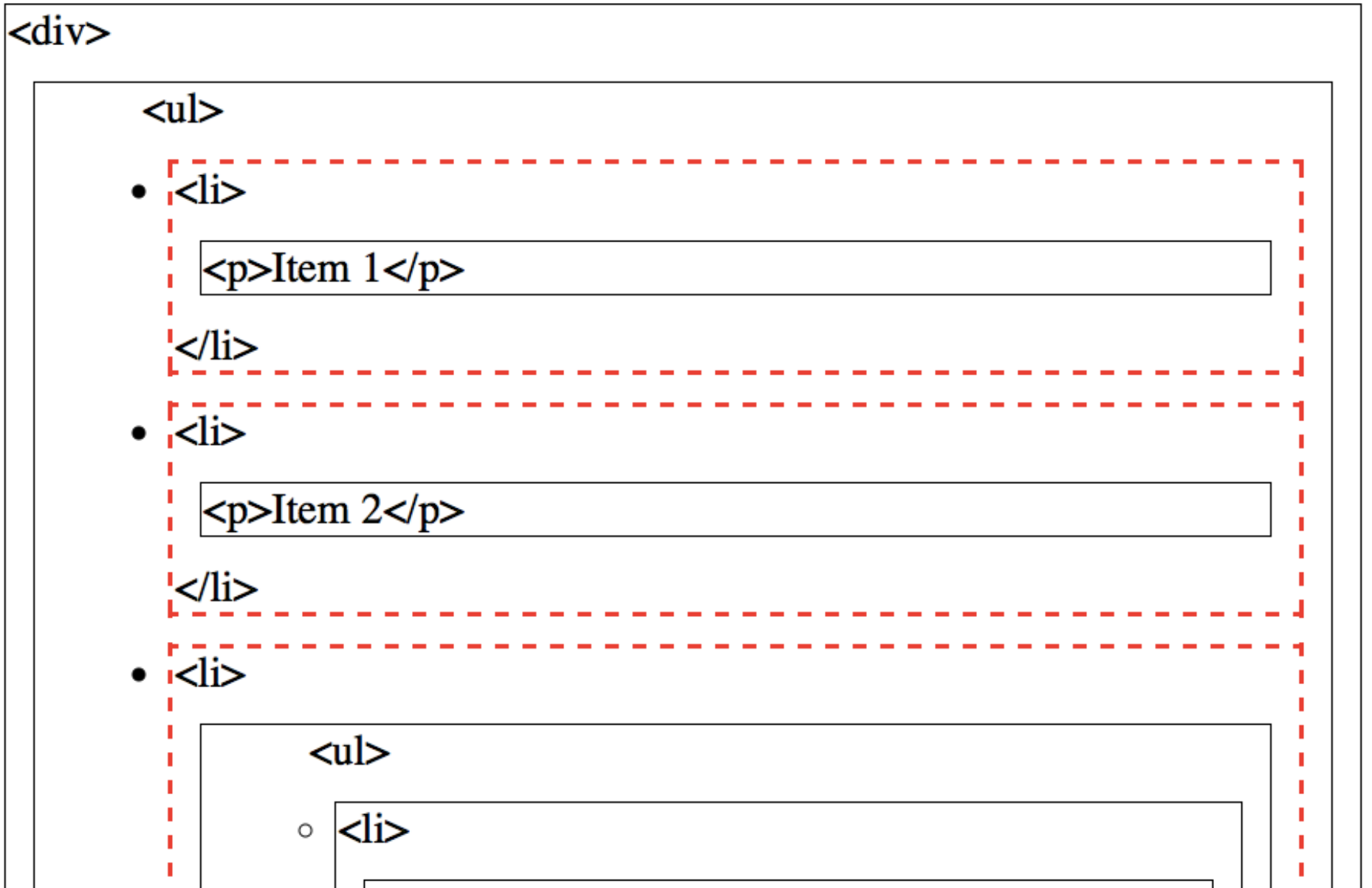
CSS Selectors

```
div li {border: dashed red 2px;}
```



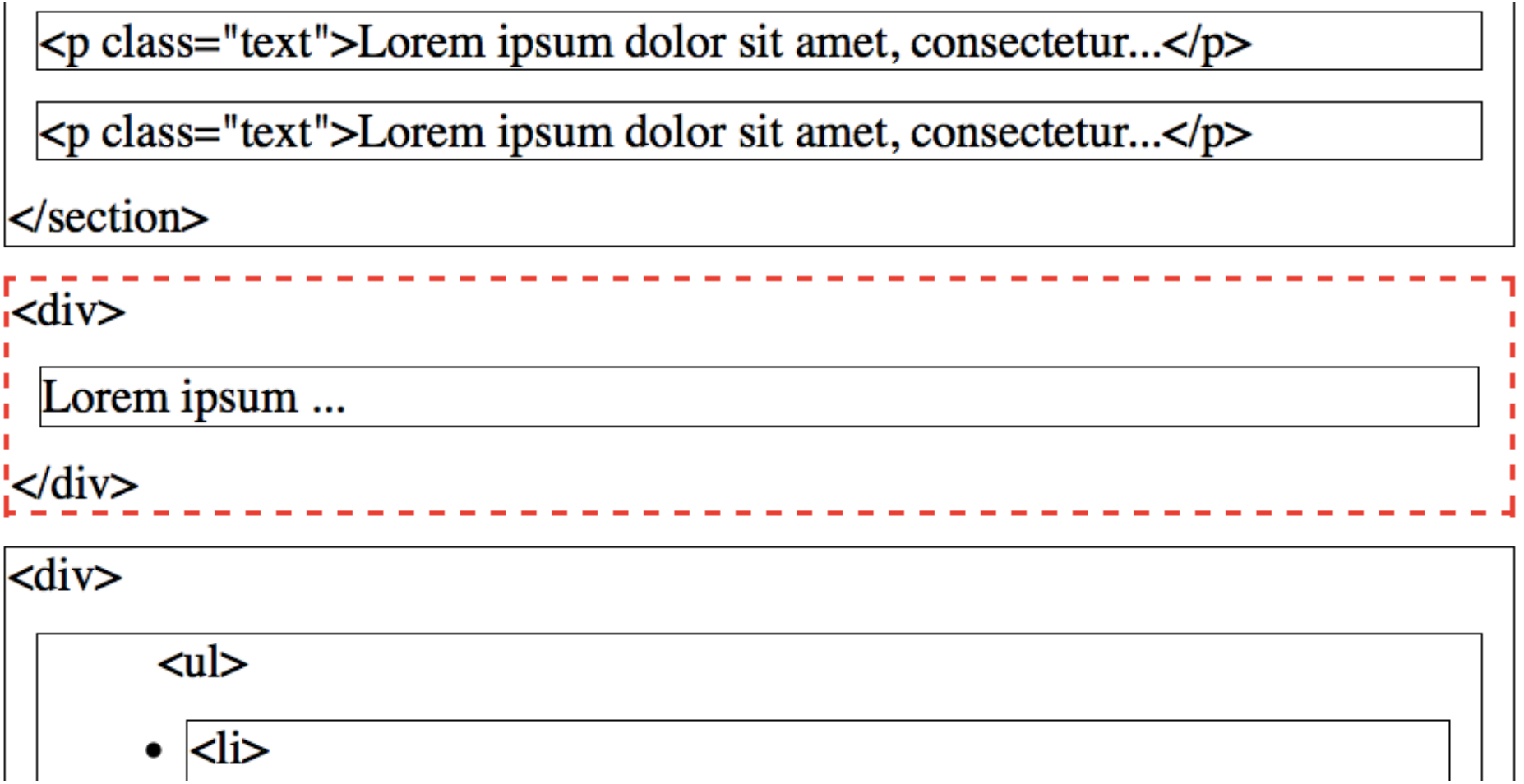
CSS Selectors

```
div > ul > li {border: dashed red 2px;}
```



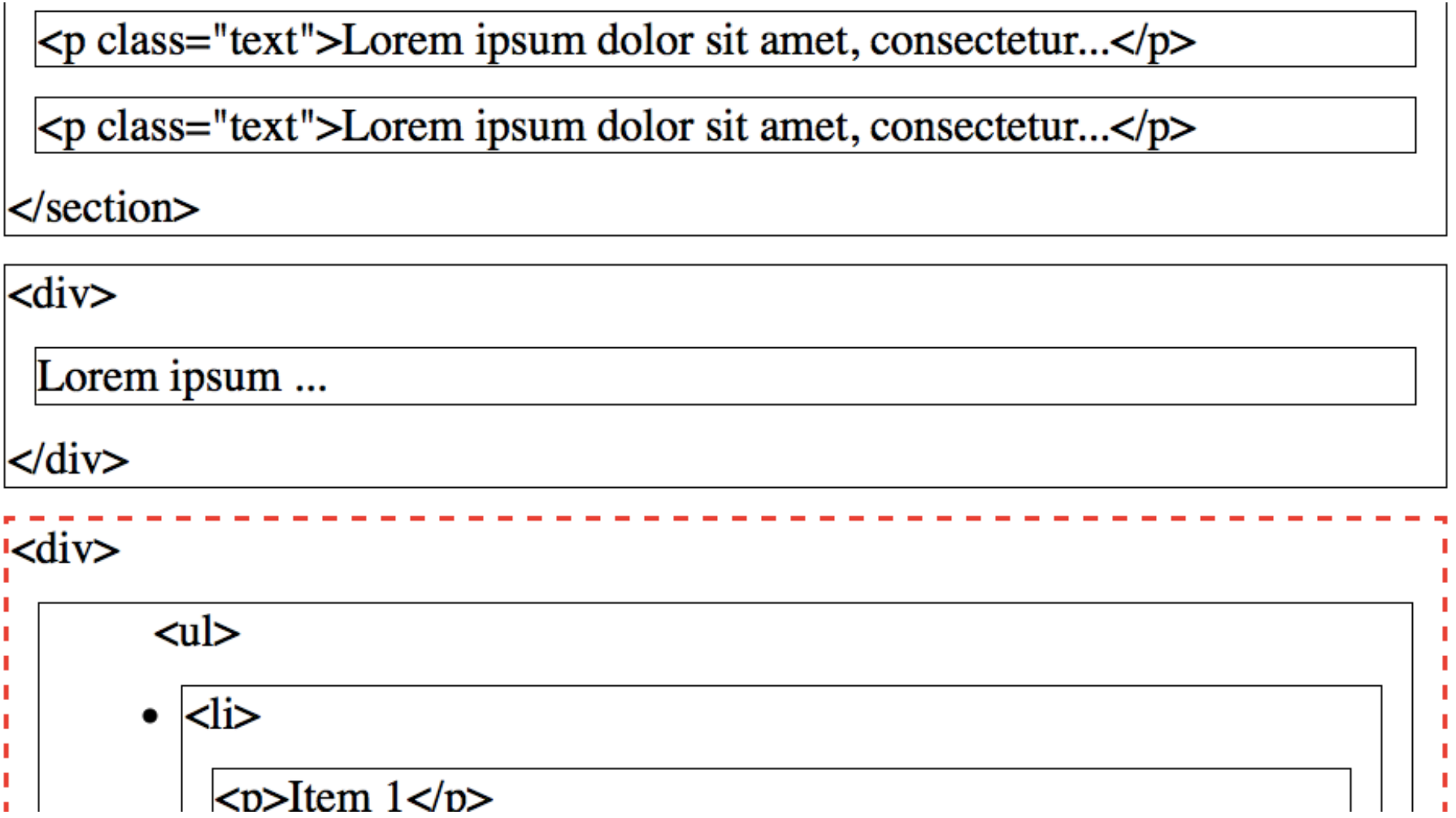
CSS Selectors

```
section+div {border: dashed red 2px;}
```



CSS Selectors

```
div~div {border: dashed red 2px;}
```



CSS Properties

CSS Properties

CSS Property Groups

- Color
- Background and Borders
- Basic Box
- Flexible Box
- Text
- Text Decoration
- Fonts
- Writing Modes
- Table
- Lists and Counters
- Animation
- Transform
- Transition
- Basic User Interface
- Multi-column
- Paged Media
- Generated Content
- Filter Effects
- Image/Replaced Content
- Masking
- Speech
- Marquee

CSS Properties

Color Properties

Property	Description	CSS
color	Sets the color of text	1
opacity	Sets the opacity level for an element	3

Background and Border Properties

Property	Description	CSS
background	Sets all the background properties in one declaration	1
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page	1
background-color	Sets the background color of an element	1
background-image	Sets the background image for an element	1
background-position	Sets the starting position of a background image	1
background-repeat	Sets how a background image will be repeated	1
background-clip	Specifies the painting area of the background	3
background-origin	Specifies the positioning area of the background images	3
background-size	Specifies the size of the background images	3
border	Sets all the border properties in one declaration	1

The Box Model

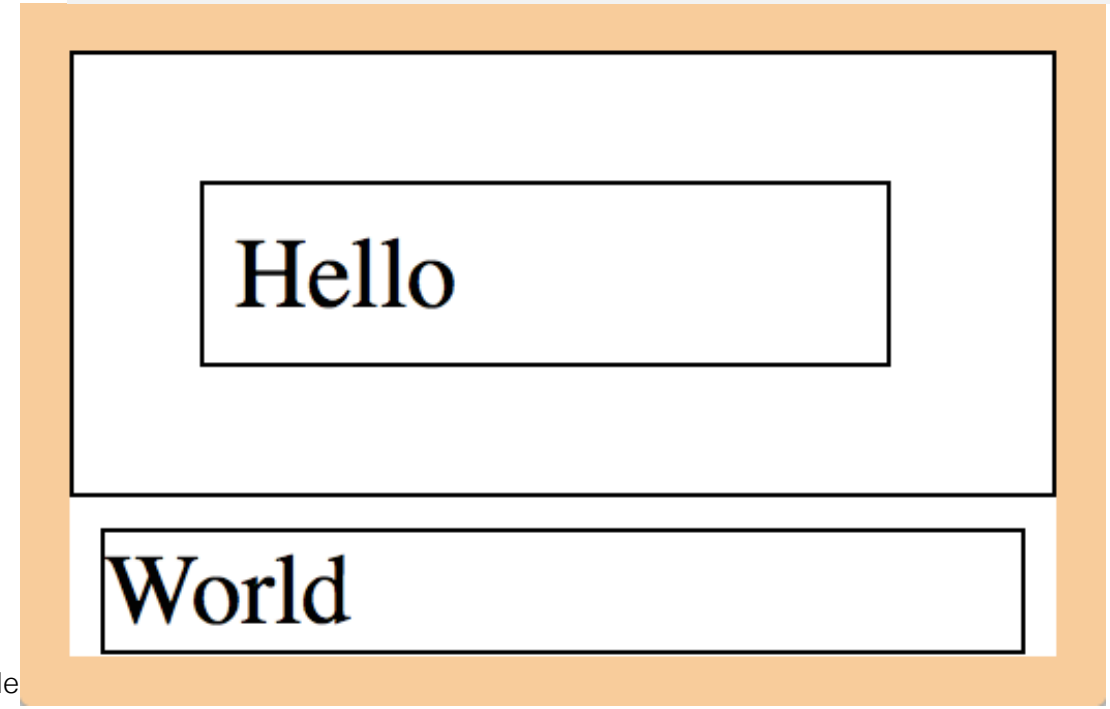
Basic Box Properties

Property	Description	CSS
bottom	Specifies the bottom position of a positioned element	2
clear	Specifies which sides of an element where other floating elements are not allowed	1
clip	Clips an absolutely positioned element	2
display	Specifies how a certain HTML element should be displayed	1
float	Specifies whether or not a box should float	1
height	Sets the height of an element	1
left	Specifies the left position of a positioned element	2
overflow	Specifies what happens if content overflows an element's box	2
overflow-x	Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area	3
overflow-y	Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area	3
padding	Sets all the padding properties in one declaration	1
padding-bottom	Sets the bottom padding of an element	1
padding-left	Sets the left padding of an element	1
padding-right	Sets the right padding of an element	1

The Box Model

- The layout of web pages is based on the setting of box dimensions, using 4 different measures
  - Width and height; padding; border; margin

```
#a1 { width:100px; margin:20px; padding: 5px;}

#a2 { margin: 5px; }
```



Display, Visible, Float, and Position

- There are three ways (properties) that control the positioning of an HTML element in a rendered page.
- **`display`**: inline, block, inline-block, none
- **`visibility`**: visible, hidden, collapse (tables)
- **`float`**: left, right…
- **`position`**: static, absolute, fixed, relative
  - associated properties: left, right, top, bottom

Media queries

- Media queries are extra conditions on the application of style rules that can target media parameters like the width and height of a screen, orientation of a device and kind of device.

```
<!-- CSS media query on a link element -->
<link rel="stylesheet" media="(max-width: 800px)" href="example.css" />

<!-- CSS media query within a stylesheet -->
<style>
@media (max-width: 600px) {
  .facet_sidebar {
    display: none;
  }
}
</style>
```

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

Media queries

- Media queries are extra conditions on the application of style rules that can target media parameters like the width and height of a screen, orientation of a device and kind of device.
- @media <media type> and <media features>

```
@media (min-width: 700px) { ... }
```

```
@media (min-width: 700px) and (orientation: landscape) { ... }
```

```
@media tv and (min-width: 700px) and (orientation: landscape) { ... }
```

```
@media (min-width: 700px), handheld and (orientation: landscape) { ... }
```

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

Media queries

Pseudo-BNF (for those of you that like that kind of thing)

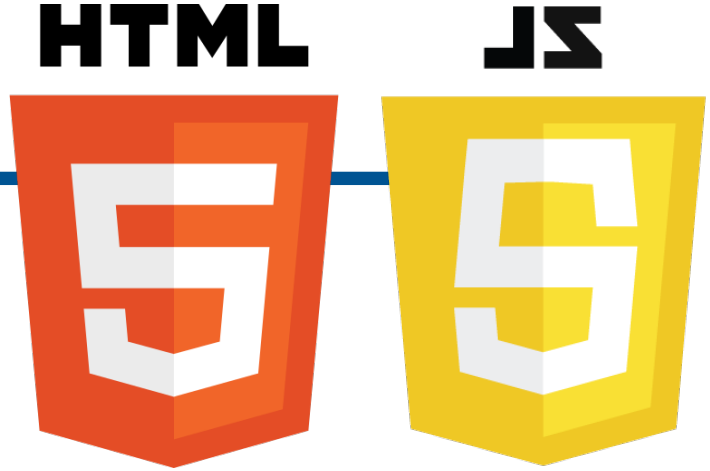
```
media_query_list: <media_query> [, <media_query> ]*
media_query: [[only | not]? <media_type> [ and <expression> ]*]
  | <expression> [ and <expression> ]*
expression: ( <media_feature> [: <value>]? )
media_type: all | aural | braille | handheld | print |
  projection | screen | tty | tv | embossed
media_feature: width | min-width | max-width
  | height | min-height | max-height
  | device-width | min-device-width | max-device-width
  | device-height | min-device-height | max-device-height
  | aspect-ratio | min-aspect-ratio | max-aspect-ratio
  | device-aspect-ratio | min-device-aspect-ratio | max-device-aspect-ratio
  | color | min-color | max-color
  | color-index | min-color-index | max-color-index
  | monochrome | min-monochrome | max-monochrome
  | resolution | min-resolution | max-resolution
  | scan | grid
```

https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Media_queries

Responsive design

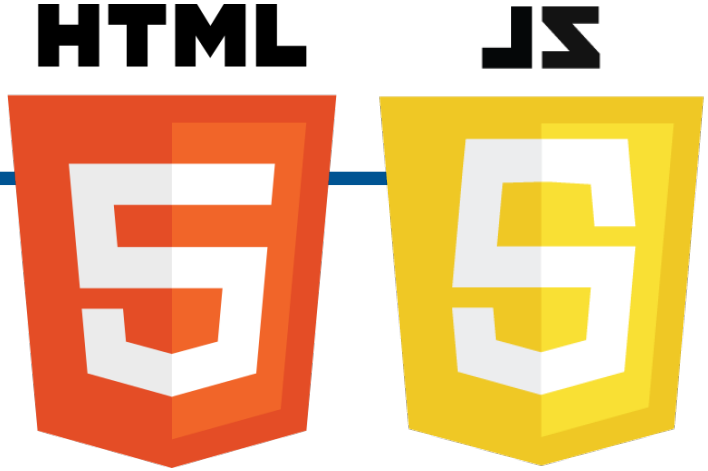
- Sets of methods and techniques to optimize reading and navigation of a certain web design to a myriad of devices and displays.
- Usually achieved by gracious resizing, hiding, and rearrangement of page sections.
  - Flexible (flow and grid) layouts
  - Media queries
  - Javascript reactive interfaces

Javascript in the browser



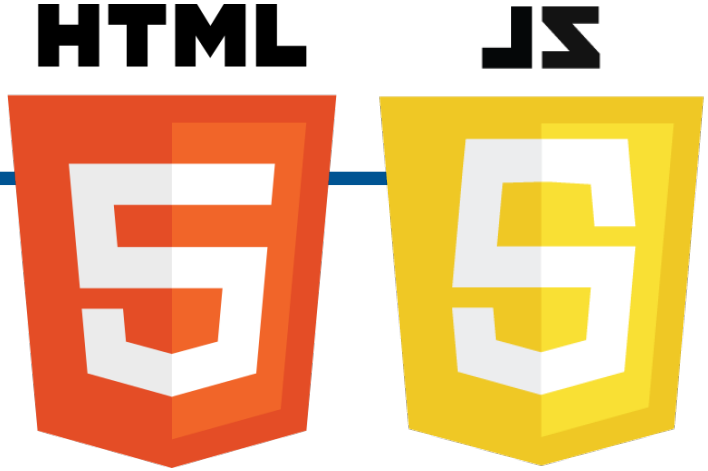

```
<h1>JavaScript Can Validate Input</h1>

<p>Please input a number between 1 and 10:</p>

<input id="numb" type="number">

<button type="button" onclick="myFunction()">Submit</button>

<p id="demo"></p>

<script>
function myFunction() {
    var x, text;

    // Get the value of the input field with id="numb"
    x = document.getElementById("numb").value;

    // If x is Not a Number or less than one or greater than 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
```

Javascript in the browser




```
<h1>JavaScript Can Validate Input</h1>

<p>Please input a number between 1 and 10:</p>

<input id="numb" type="number">

<button type="button" onclick="myFunction()">Submit</button>

<p id="demo"></p>

<script>
function myFunction() {
    var x, text;

    // Get the value of the input field with id="numb"
    x = document.getElementById("numb").value;

    // If x is Not a Number or less than one or greater than 10
    if (isNaN(x) || x < 1 || x > 10) {
        text = "Input not valid";
    } else {
        text = "Input OK";
    }
    document.getElementById("demo").innerHTML = text;
}
</script>
```

Javascript + AJAX



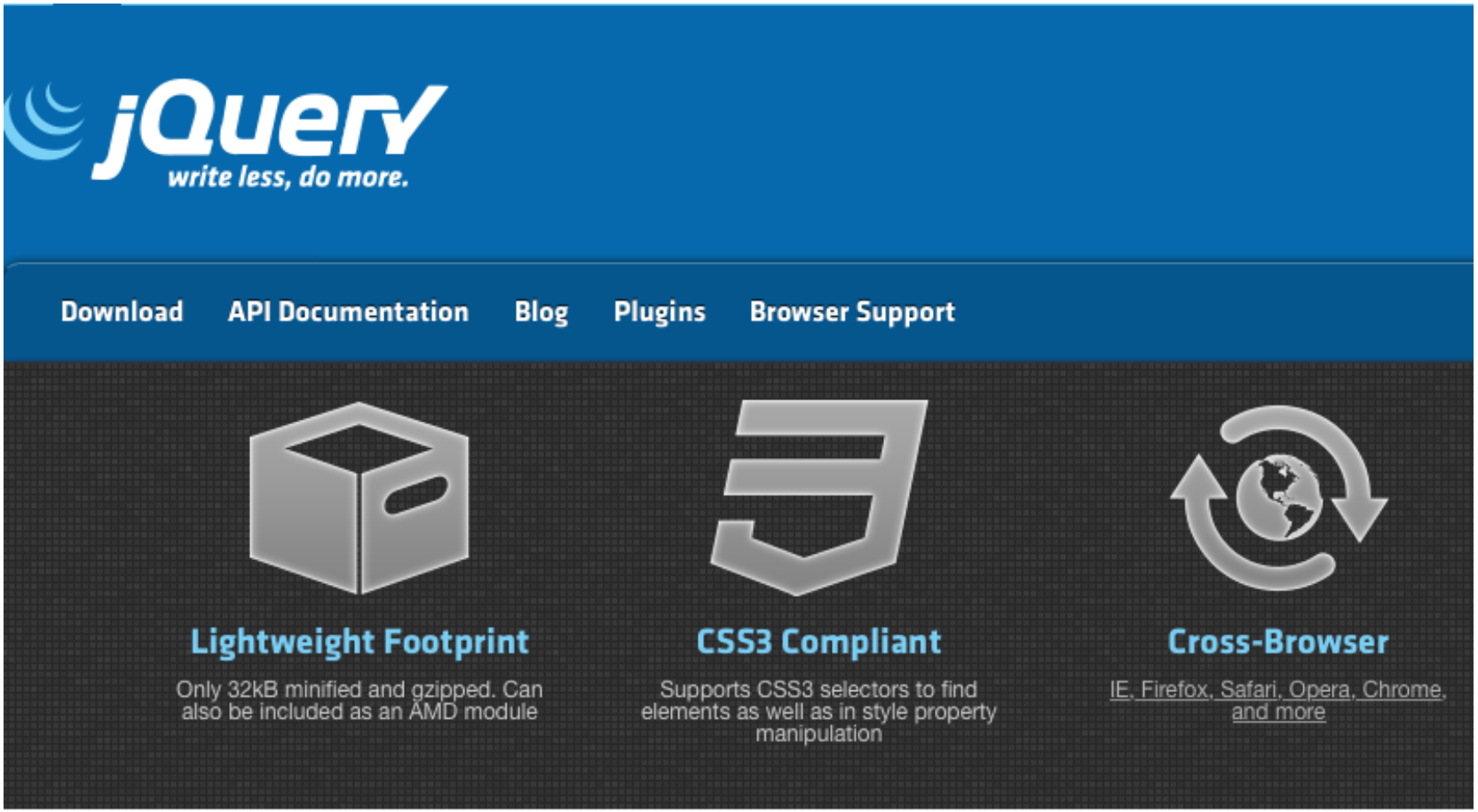

```
<div id="demo"><h2>Let AJAX change this text</h2></div>

<button type="button" onclick="loadDoc()">Change Content</button>

<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      document.getElementById("demo").innerHTML = xhttp.responseText;
    }
  }
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>
```

Javascript Frameworks

Javascript Frameworks

- Abstraction of browser related details
  (No, they are not the same as the standard !).
- Higher abstraction level in browser related operations.
  - Searching the DOM and iterating elements (jQuery)
  - Updating values of elements in the DOM (AngularJS)
- Inversion of control
  - Basic event model already provide it
  - frameworks extend it further…
- **jQuery** is really an extensible library…
- **jQuery UI/Mobile** are frameworks…
- **Bootstrap**
- **AngularJS**

jQuery



What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

jQuery

```
<script type="text/javascript">

$(document).ready(function(){
    $("#msgid").html("This is Hello World by JQuery");
});

</script>

This is Hello World by HTML

<div id="msgid">
</div>
```

Bootstrap

Preprocessors

Bootstrap ships with vanilla CSS, but its source code utilizes the two most popular CSS preprocessors, Less and Sass. Quickly get started with precompiled CSS or build on the source.

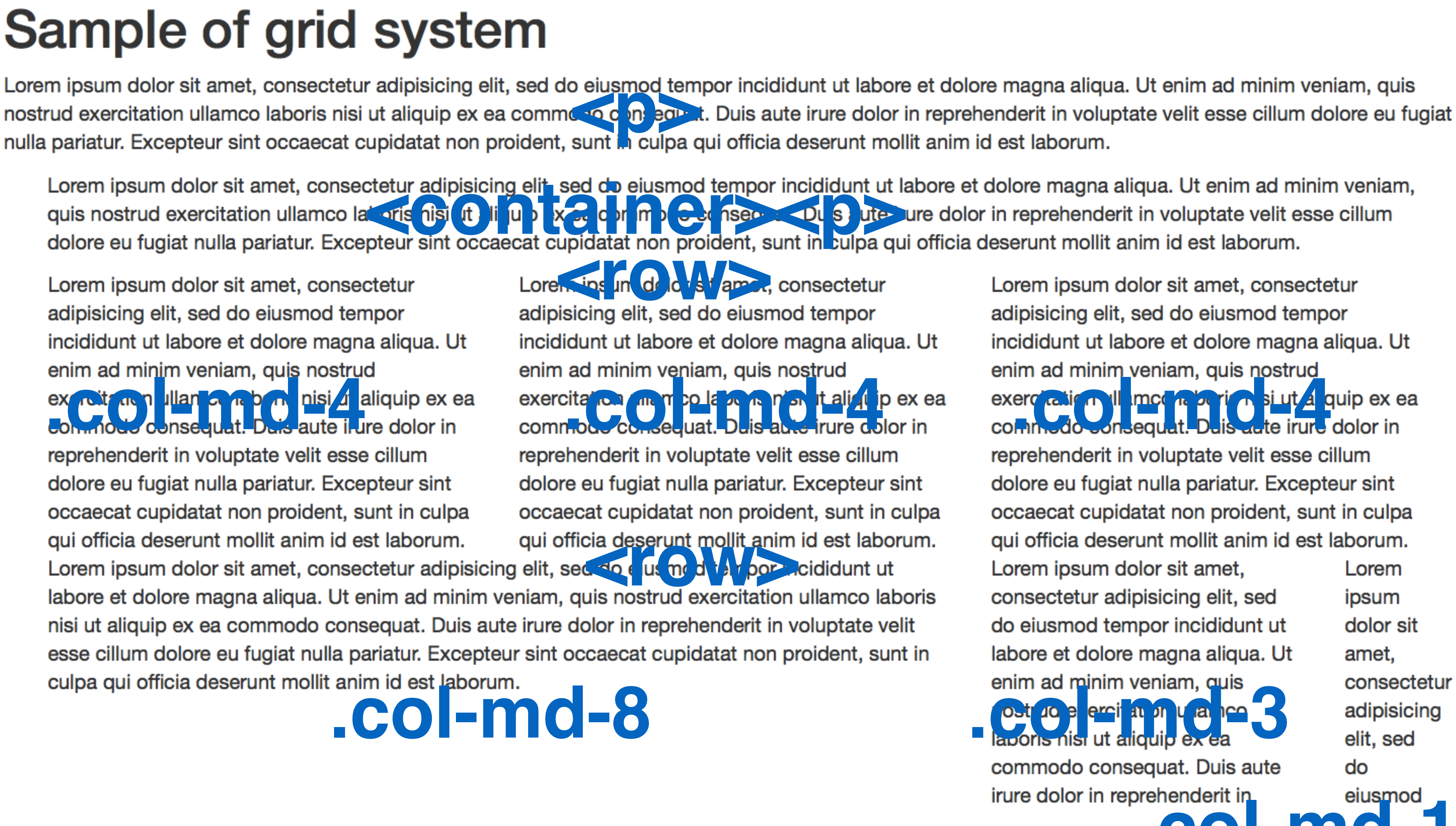
One framework, every device.

Bootstrap easily and efficiently scales your websites and applications with a single code base, from phones to tablets to desktops with CSS media queries.

Full of features

With Bootstrap, you get extensive and beautiful documentation for common HTML elements, dozens of custom HTML and CSS components, and awesome jQuery plugins.

Bootstrap Grid System



Bootstrap Grid System

- Bootstrap is a CSS/Javascript client framework that provides simpler and effective layout tools.
- Bootstrap divides a page into 12 columns and provides classes for rows and columns:
  - `container`, `row`, `col-X-Y`, etc.
  - X: **xs** (phones), **sm** (tablets), **md** (desktops), and **lg** (larger desktops)
  - Y: 1 .. 12
  - Explore bootstrap grids at: http://getbootstrap.com/examples/grid/

AngularJS

- AngularJS is a MVC framework that embeds reactivity in the client side. It simplifies greatly the process of watching variable modifications and changing the appropriate places.

```
1.  <!doctype html>
2.  <html ng-app>
3.    <head>
4.      <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.4/
    angular.min.js"></script>
5.    </head>
6.    <body>
7.      <div>
8.        <label>Name:</label>
9.        <input type="text" ng-model="yourName" placeholder="Enter a name here">
10.       <hr>
11.       <h1>Hello {{yourName}}!</h1>
12.     </div>
13.   </body>
14. </html>
```

Web Applications

Web application variants

- “Traditional” request - response (HTML view)
  - CRUD + Custom controller/view definition
  - Full DOM reload on each request
- Dynamic HTML Views
  - Ajax HTML requests load partial views into the DOM
- Single page application (SPA)
  - One page, decoupled from server
  - All data is dynamically loaded by requests to web services
- Hybrid
  - Main template loaded statically
  - Secondary data loaded dynamically and asynchronously
- Mashups

Templating

HTML - Templating

- In a web application HTML are “almost never” static. Templating is a mechanism that produces dynamic content out of “templates”
- Templates
  - Parametric
  - Dynamic (loops, decisions, …)
  - Abstract (meta-inf, scripts, links, …)
  - Reuse (by including common blocks)
- Templating depend on the host language:
  - Java: JSP, thymeleaf
  - Rails: ERB, haml, …
  - NodeJS: handlebars, jade, …

HTML - Templating

- Thymeleaf (thymeleaf.org)
  - XML based
  - special attributes in tags (th:*)
  - internationalisation expressions #{}
  - variable / model expressions ${}
  - navigation expressions @{}

```
1   <table>
2     <thead>
3       <tr>
4         <th th:text="#{msgs.headers.name}">Name</th>
5         <th th:text="#{msgs.headers.price}">Price</th>
6       </tr>
7     </thead>
8     <tbody>
9       <tr th:each="prod : ${allProducts}">
10        <td th:text="${prod.name}">Oranges</td>
11        <td th:text="${#numbers.formatDecimal(prod.price,1,2)}">0.99</td>
12      </tr>
13    </tbody>
14  </table>
```

Shells for web applications

- Mobile Applications
  - PhoneGap / Apache Cordova

Client Web App Design
Process

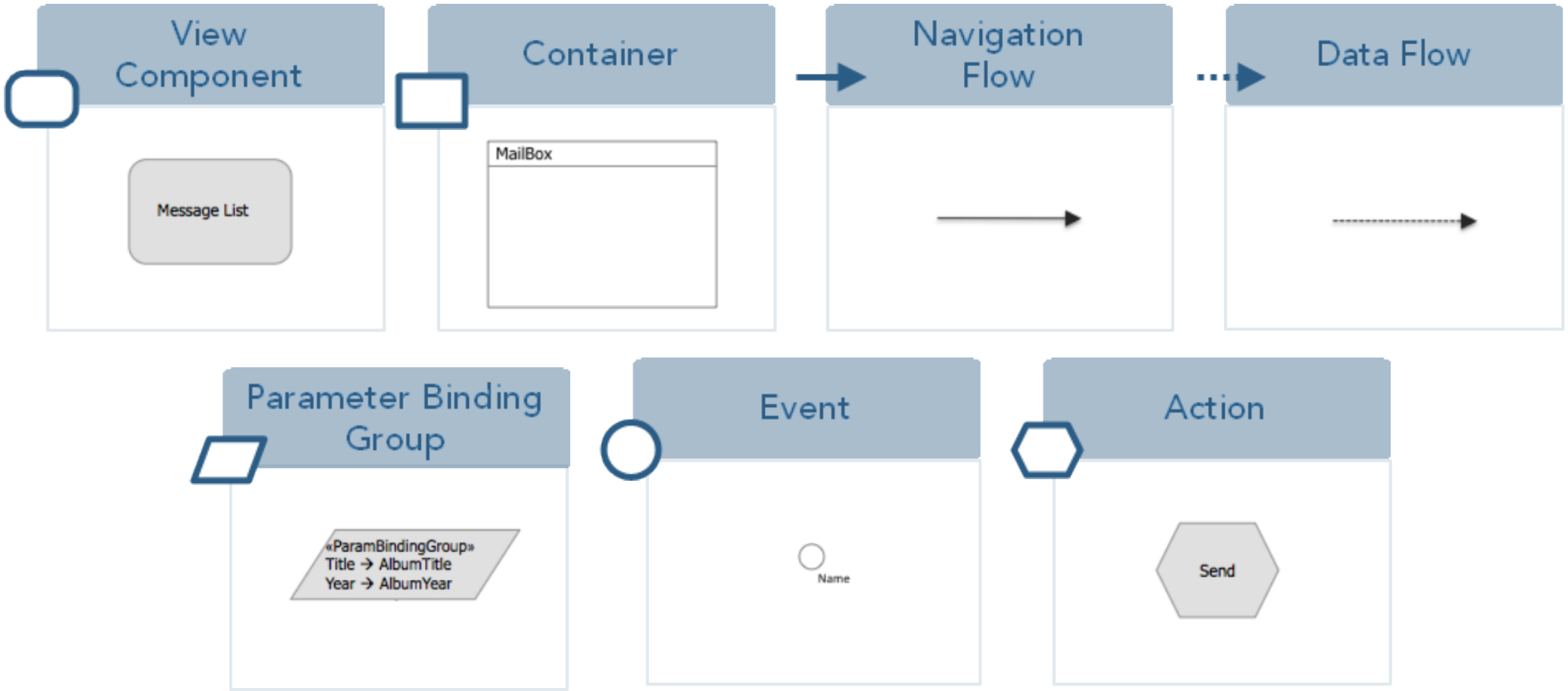
Client Application Design

- Specify the data structure using data-centric resource based method (UML, ER, etc…) — REST interface
- Specify the interaction using IFML
  - Interaction and (basic and compound) view components
- Connect basic (CRUD) and custom controllers to IFML elements
  - Provides a decoupled design
  - Each IFML element is mapped to a resource access point:
    - View Components —— GET requests
    - Forms and Actions —— POST requests

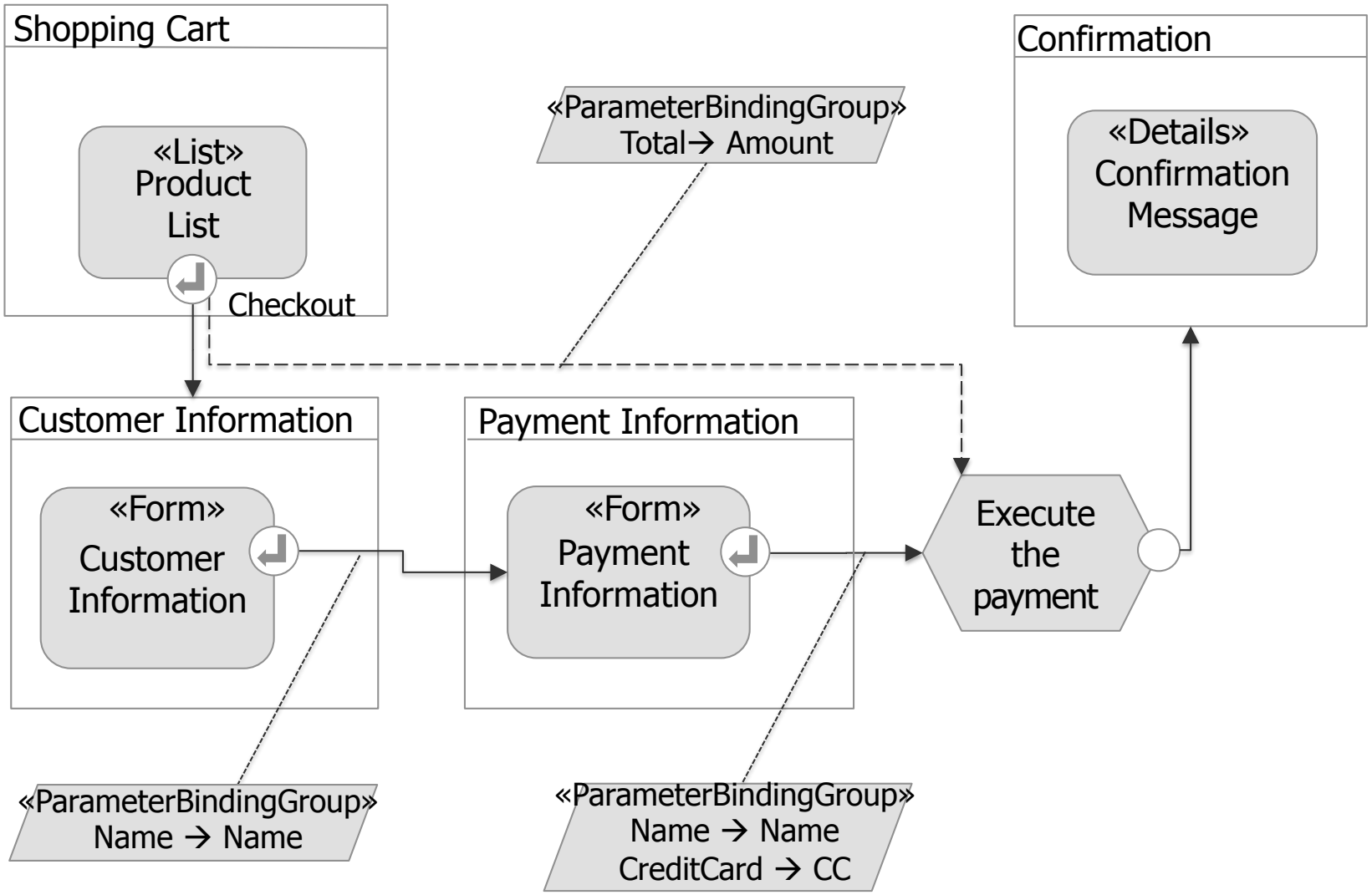
Specifying Web Client Applications (Recap)

IFML

IFML elements



IFML diagrams



Connecting Web Client Applications
REST

Restful interface design (Recap)

- Follows an architectural style (convention)
  - Architectural style that promotes a simpler and more efficient way of providing and connecting web services. Built on top of basic HTTP
- Promotes the decoupling from Data-centric server side applications and client user-centric applications
- Implementations provides (convenient) flavours
  - Web-service style pure JSON/XML Data
  - Complete/partial HTML view responses
  - Javascript code responses (e.g. Rails AJAX responses)

REST - Representational State Transfer

- Resource Based
- Representation
- Uniform Interface
- Stateless
- Cacheable
- Client-Server
- Layered System
- Code on Demand (optional)

Representational State Transfer

- Resource Based
  - vs Action Based
  - Nouns and not verbs to identify data in the system
  - Identified (represented) by URI
  - Aliasing is admissible
- Representation
- Uniform Interface
- Stateless
- Cacheable
- Client-Server
- Layered System
- Code on Demand (not talking about it)

Representational State Transfer

- Resource Based
- Representation
  - JSON or XML representation of the state of a given resource transferred between client and server at a given verb in a given URL.
  - Well identified interface (the information retrieved at an URL — the type)
- Uniform Interface
- Stateless
- Cacheable
- Client-Server
- Layered System
- Code on Demand (not talking about it)

Representational State Transfer

- Resource Based
- Representation
- Uniform Interface
  - standard HTTP verbs (GET, PUT, POST, DELETE)
  - standard HTTP response (status code, info in the response body)
  - Uniform structure of URIs with a name, identifying the resource
  - References inside responses must be complete.
- Stateless
- Cacheable
- Client-Server
- Layered System
- Code on Demand (not talking about it)

Representational State Transfer

- Resource Based
- Representation
- Uniform Interface
- Stateless
  - Server does not hold session state
  - Messages are self contained
- Cacheable
- Client-Server
- Layered System
- Code on Demand (not talking about it)

Representational State Transfer

- Resource Based
- Representation
- Uniform Interface
- Stateless
- Cacheable
  - Responses can be tagged as cacheable (in the server)
  - (also) Bookmarkable
- Layered System
- Code on Demand (not talking about it)

Representational State Transfer

- Resource Based
- Representation
- Uniform Interface
- Stateless
- Cacheable
- Layered System
  - Establishes an API between a client and a “database”
- Code on Demand (not talking about it)

EXAMPLES

6. Real REST Examples

Here's a very partial list of service providers that use a REST API. Note that some of them also support a WSDL (Web Services) API, in addition, so you can pick which to use; but in most cases, when both alternatives are available, REST calls are easier to create, the results are easier to parse and use, and it's also less resource-heavy on your system.

So without further ado, some REST services:

- The Google Glass API, known as "Mirror API", is a pure REST API. Here is an excellent video talk about this API. (The actual API discussion starts after 16 minutes or so.)
- Twitter has a **REST API** (in fact, this was their original API and, so far as I can tell, it's still the main API used by Twitter application developers),
- **Flickr**,
- **Amazon.com** offer several REST services, e.g., for their S3 storage solution,
- **Atom** is a RESTful alternative to RSS,
- **Tesla Model S** uses an (undocumented) REST API between the car systems and its Android/iOS apps.

in … http://rest.elkstein.org/2008/02/real-rest-examples.html

Mirror API - Google Glasses

Contacts

For Contacts Resource details, see the resource representation page.

Method	HTTP request	Description
URIs relative to https://www.googleapis.com/mirror/v1, unless otherwise noted		
delete	`DELETE /contacts/id`	Deletes a contact.
get	`GET /contacts/id`	Gets a single contact by ID.
insert	`POST /contacts`	Inserts a new contact.
list	`GET /contacts`	Retrieves a list of contacts for the authenticated user.
patch	`PATCH /contacts/id`	Updates a contact in place. This method supports patch semantics.
update	`PUT /contacts/id`	Updates a contact in place.

in … https://developers.google.com/glass/v1/reference/

Mirror API - Google Glasses

Timeline

For Timeline Resource details, see the resource representation page.

Method	HTTP request	Description
URIs relative to https://www.googleapis.com/mirror/v1, unless otherwise noted		
delete	`DELETE /timeline/id`	Deletes a timeline item.
get	`GET /timeline/id`	Gets a single timeline item by ID.
insert	`POST https://www.googleapis.com/upload/mirror/v1/timeline` and `POST /timeline`	Inserts a new item into the timeline.
list	`GET /timeline`	Retrieves a list of timeline items for the authenticated user.
patch	`PATCH /timeline/id`	Updates a timeline item in place. This method supports patch semantics.
update	`PUT https://www.googleapis.com/upload/mirror/v1/timeline/id` and `PUT /timeline/id`	Updates a timeline item in place.

Mirror API - Google Glasses

Timeline.attachments

For Timeline.attachments Resource details, see the resource representation page.

Method	HTTP request	Description
URIs relative to https://www.googleapis.com/mirror/v1, unless otherwise noted		
delete	`DELETE /timeline/itemId/attachments/attachmentId`	Deletes an attachment from a timeline item.
get	`GET /timeline/itemId/attachments/attachmentId`	Retrieves an attachment on a timeline item by item ID and attachment ID.
insert	`POST https://www.googleapis.com/upload/mirror/v1/timeline/itemId/attachments`	Adds a new attachment to a timeline item.
list	`GET /timeline/itemId/attachments`	Returns a list of attachments for a timeline item.

in … https://developers.google.com/glass/v1/reference/

Exercise

1. Consider the scenario of a web application for booking Hotel services.

2. Define the standard REST interface for the application (resources).

3. Define the basic hotel reservation use case using IFML.

4. Define the resources/action used by each IFML view component and action.