DATABASE SYSTEMS

Introduction to web programming

Database Systems Course, 2016

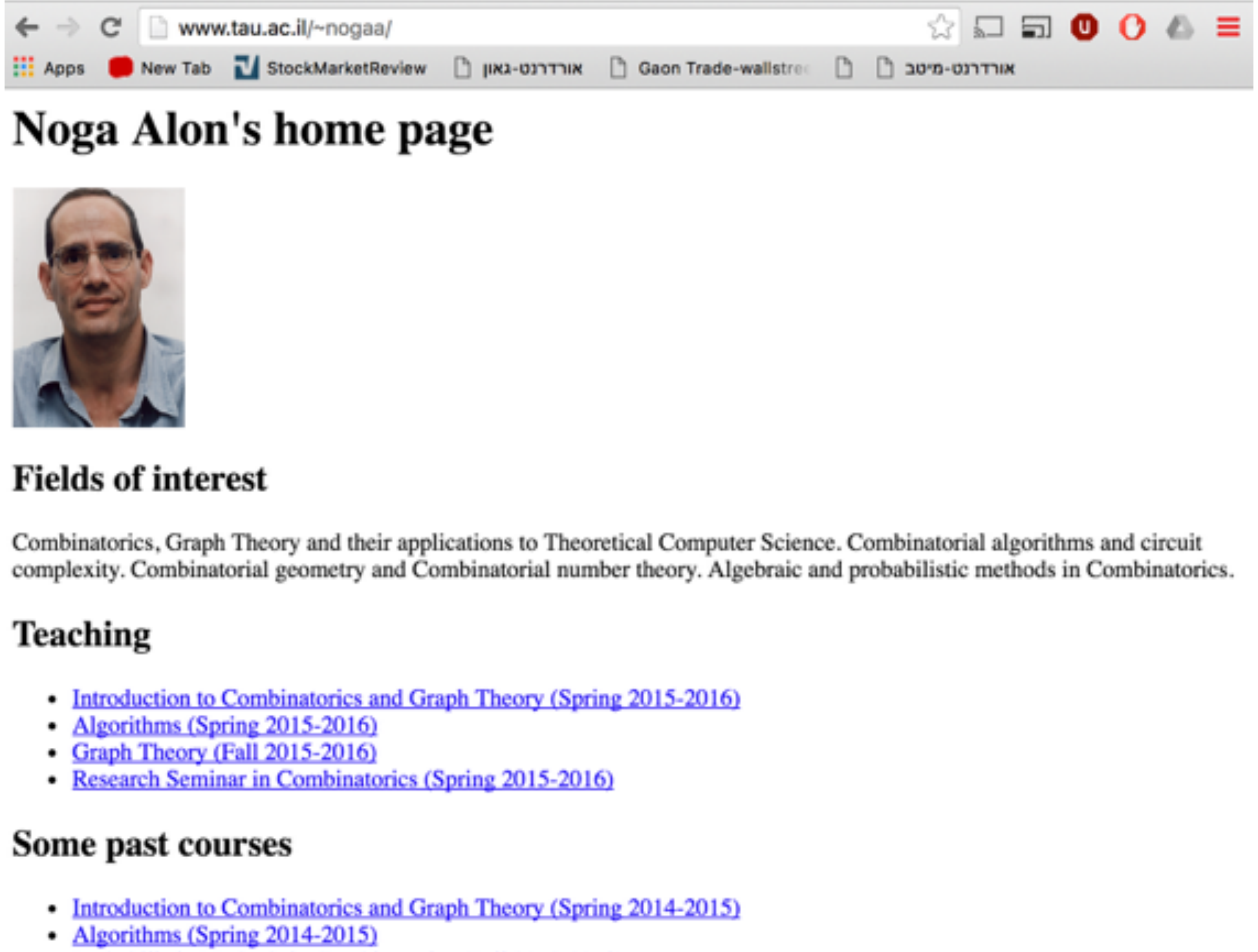
TEL AVIV UNIVERSITY

AGENDA FOR TODAY

- **Client side programming**
  - HTML
  - CSS
  - Javascript
  - Additional libraries: Bootstrap,Angular, Jquery
- **Server side programming: PHP**
  - Install XAMPP
  - Web server architecture
  - php+mysql
- **Web APIs: REST ,Json, and how to get them via Python**

A STATIC WEB PAGE

- Content is identical, regardless.
- To perform changes in content, the programmer has to change the HTML file.
- For example:



Noga Alon's home page

Fields of interest

Combinatorics, Graph Theory and their applications to Theoretical Computer Science. Combinatorial algorithms and circuit complexity. Combinatorial geometry and Combinatorial number theory. Algebraic and probabilistic methods in Combinatorics.

Teaching

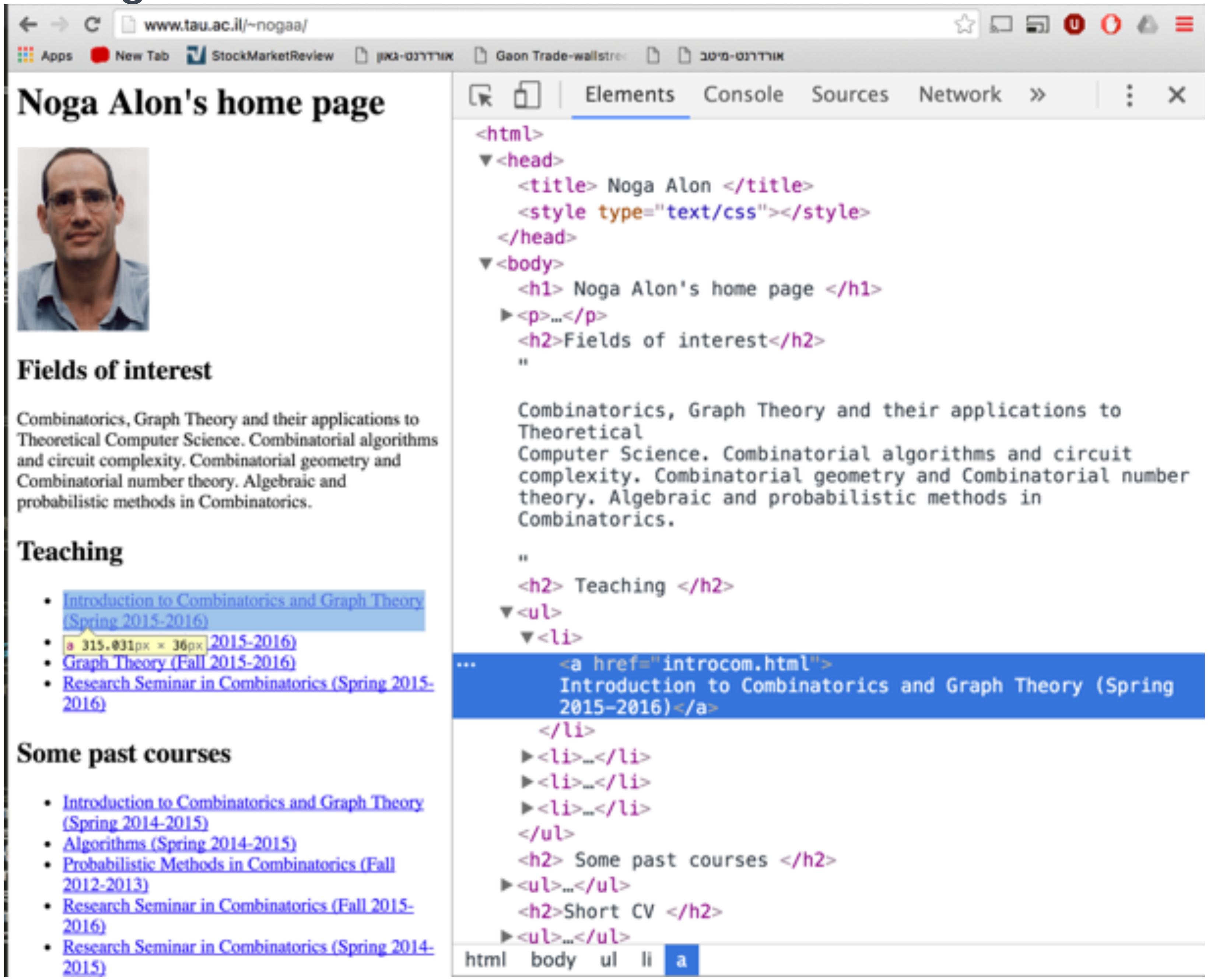
- Introduction to Combinatorics and Graph Theory (Spring 2015-2016)
- Algorithms (Spring 2015-2016)
- Graph Theory (Fall 2015-2016)
- Research Seminar in Combinatorics (Spring 2015-2016)

Some past courses

- Introduction to Combinatorics and Graph Theory (Spring 2014-2015)
- Algorithms (Spring 2014-2015)

A STATIC WEB PAGE

- To view the HTML source code, we can right click and select “view source”
- Or use the browser’s developer tools. e.g.



Noga Alon's home page

Fields of interest

Combinatorics, Graph Theory and their applications to Theoretical Computer Science. Combinatorial algorithms and circuit complexity. Combinatorial geometry and Combinatorial number theory. Algebraic and probabilistic methods in Combinatorics.

Teaching

- Introduction to Combinatorics and Graph Theory (Spring 2015-2016)
- 2015-2016)
- Graph Theory (Fall 2015-2016)
- Research Seminar in Combinatorics (Spring 2015-2016)

Some past courses

- Introduction to Combinatorics and Graph Theory (Spring 2014-2015)
- Algorithms (Spring 2014-2015)
- Probabilistic Methods in Combinatorics (Fall 2012-2013)
- Research Seminar in Combinatorics (Fall 2015-2016)
- Research Seminar in Combinatorics (Spring 2014-2015)

```
<html>
 ▼<head>
    <title> Noga Alon </title>
    <style type="text/css"></style>
  </head>
 ▼<body>
    <h1> Noga Alon's home page </h1>
  ▶<p>…</p>
    <h2>Fields of interest</h2>
    "

    Combinatorics, Graph Theory and their applications to
    Theoretical
    Computer Science. Combinatorial algorithms and circuit
    complexity. Combinatorial geometry and Combinatorial number
    theory. Algebraic and probabilistic methods in
    Combinatorics.

    "
    <h2> Teaching </h2>
  ▼<ul>
    ▼<li>
...       <a href="introcom.html">
        Introduction to Combinatorics and Graph Theory (Spring
        2015-2016)</a>
      </li>
    ▶<li>…</li>
    ▶<li>…</li>
    ▶<li>…</li>
    </ul>
    <h2> Some past courses </h2>
  ▶<ul>…</ul>
    <h2>Short CV </h2>
  ▶<ul>…</ul>
```

A STATIC WEB PAGE

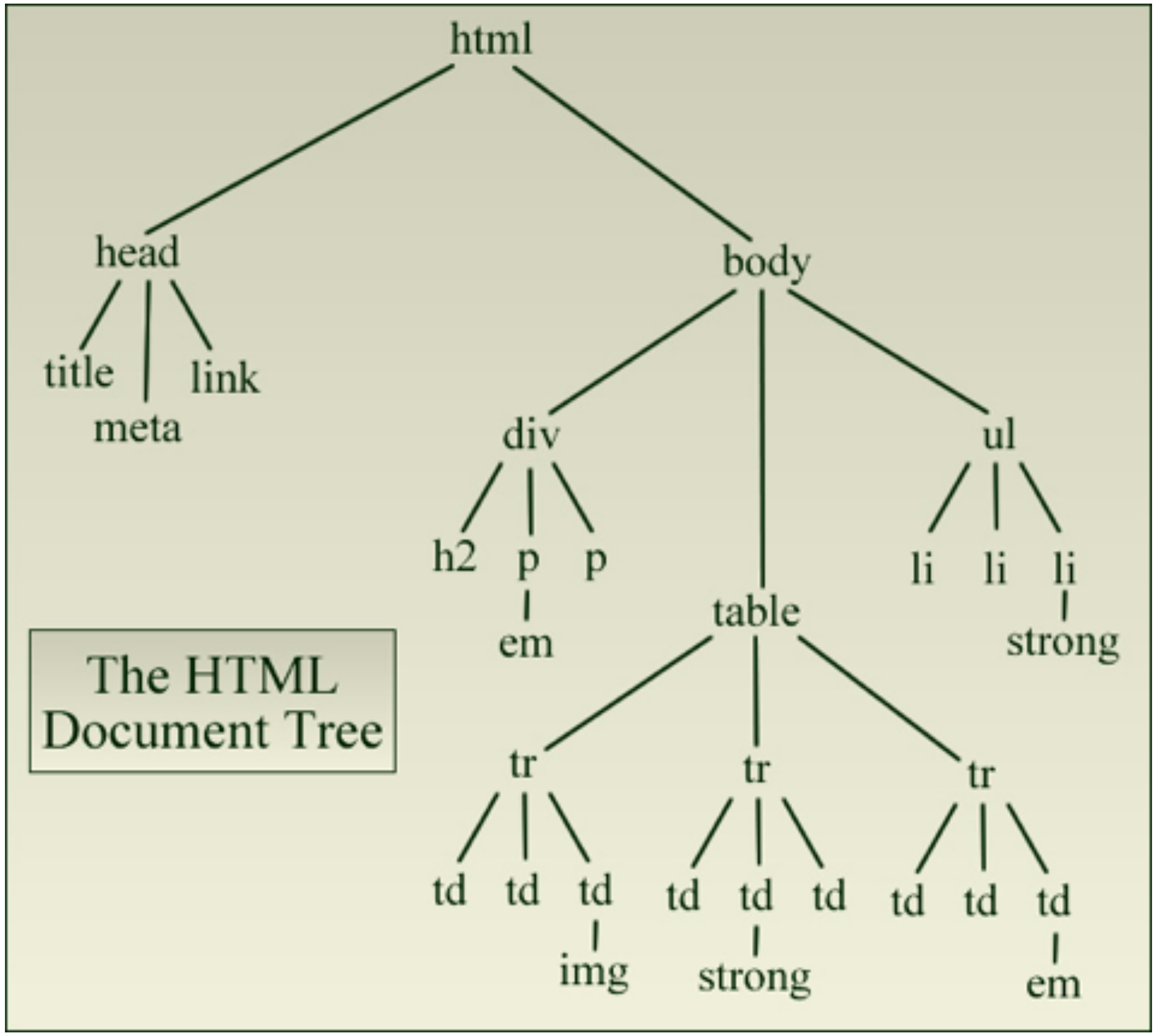
- **HTML web page is a document, orgnizied in a tree structure, according to the Document Object Model (DOM).**
- **The most important nodes:**

<html> the root of every web page

<head> containing meta-data and external sources

<body> holds the content of the webpage

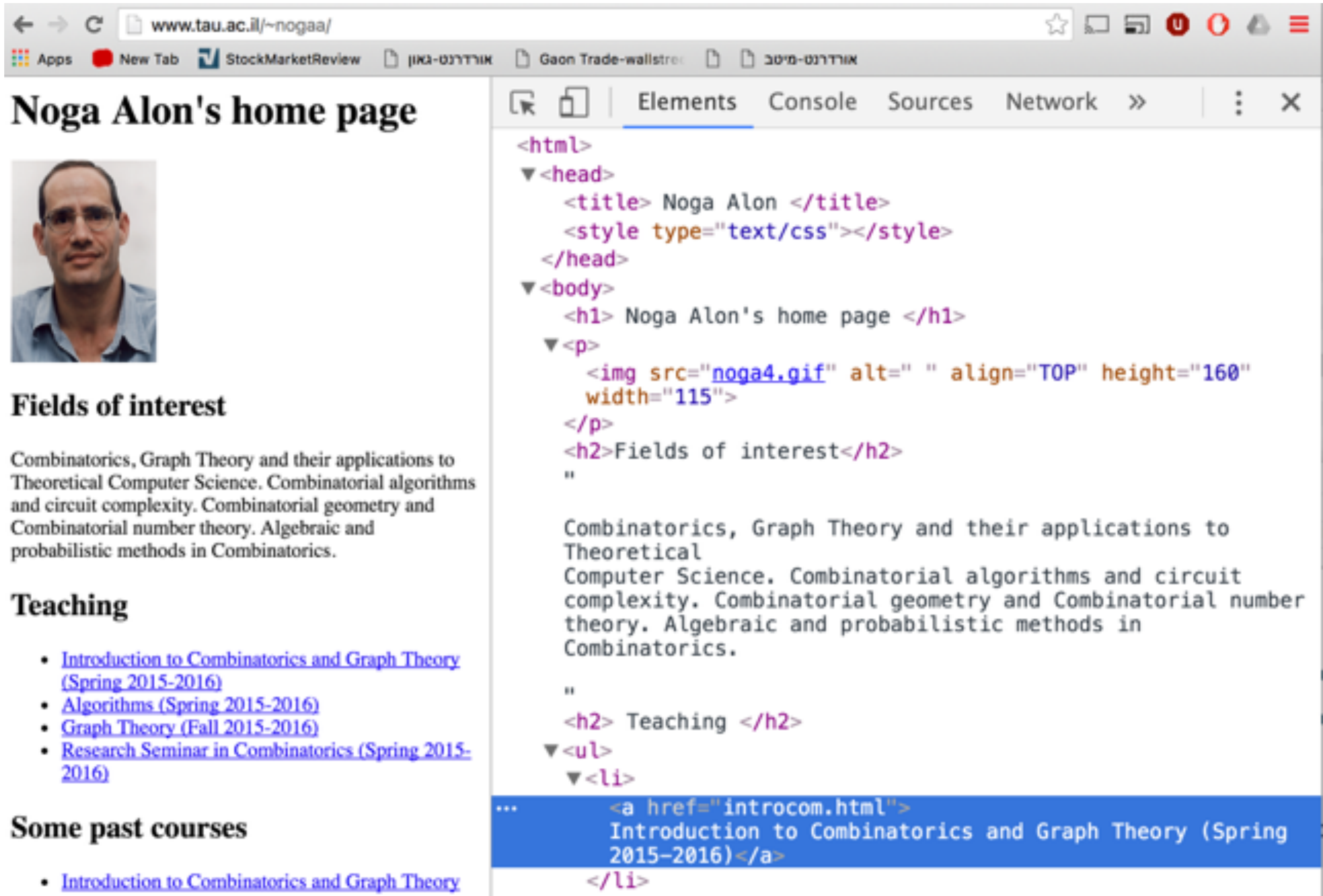
<div> is the basic content container.



The HTML Document Tree

A STATIC WEB PAGE

- Each node is an ***element***
- Each element beings and ends with a tag e.g. : <title> Noga Alon </title>
- Each ***element*** has ***a set of attributes***
  - **structure:** *attr = val*
  - *<img src="noga4.gif" alt=" ">*



Noga Alon's home page

Fields of interest

Combinatorics, Graph Theory and their applications to Theoretical Computer Science. Combinatorial algorithms and circuit complexity. Combinatorial geometry and Combinatorial number theory. Algebraic and probabilistic methods in Combinatorics.

Teaching

- Introduction to Combinatorics and Graph Theory (Spring 2015-2016)
- Algorithms (Spring 2015-2016)
- Graph Theory (Fall 2015-2016)
- Research Seminar in Combinatorics (Spring 2015-2016)

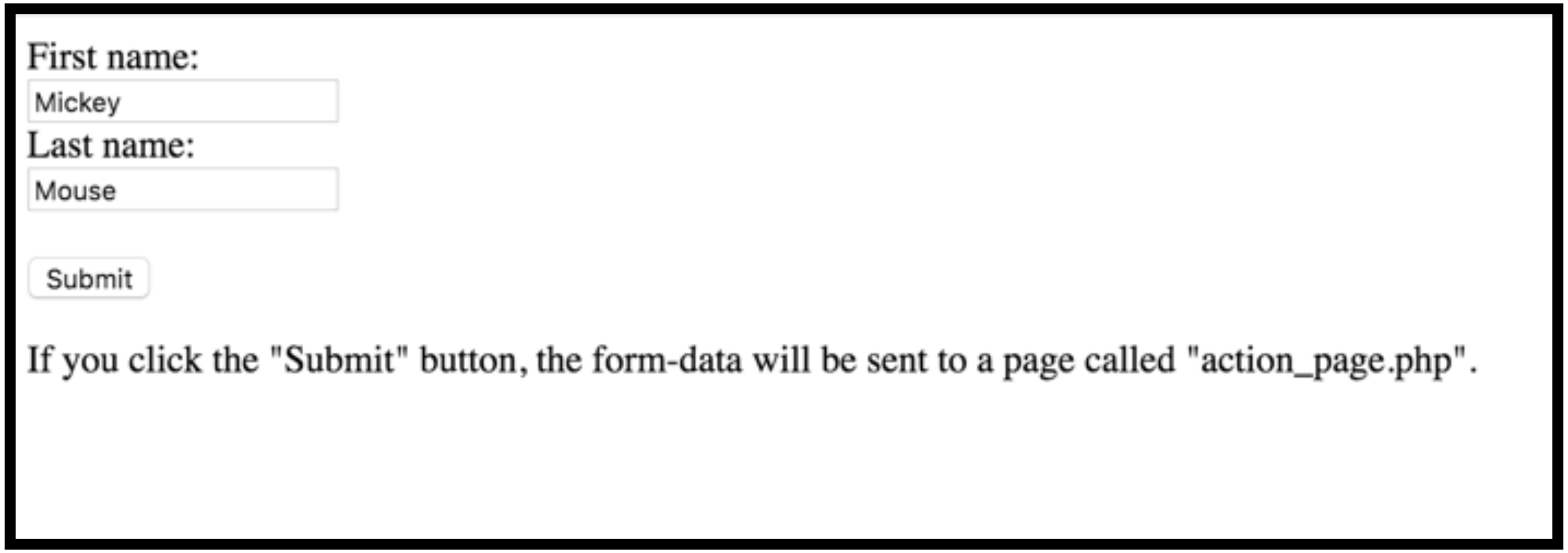
Some past courses

- Introduction to Combinatorics and Graph Theory

```
<html>
▼<head>
    <title> Noga Alon </title>
    <style type="text/css"></style>
  </head>
▼<body>
    <h1> Noga Alon's home page </h1>
  ▼<p>
      <img src="noga4.gif" alt=" " align="TOP" height="160"
      width="115">
    </p>
    <h2>Fields of interest</h2>
    "

    Combinatorics, Graph Theory and their applications to
    Theoretical
    Computer Science. Combinatorial algorithms and circuit
    complexity. Combinatorial geometry and Combinatorial number
    theory. Algebraic and probabilistic methods in
    Combinatorics.

    "
    <h2> Teaching </h2>
  ▼<ul>
    ▼<li>
...       <a href="introcom.html">
          Introduction to Combinatorics and Graph Theory (Spring
          2015-2016)</a>
      </li>
```

AN (ADVANCED) STATIC PAGE

- **Web forms:**
  - Used to collect **input** from the user and **submit** it to the server
  - The values are sent to the **web server** via and **HTTP GET** request (by default)



First name:

Mickey

Last name:

Mouse

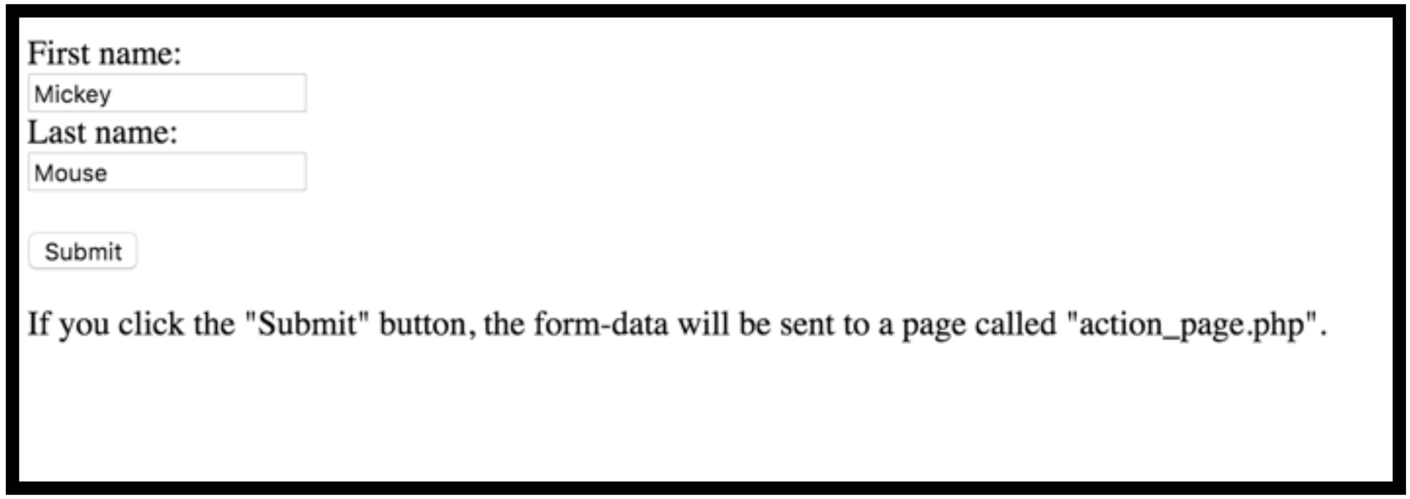
If you click the "Submit" button, the form-data will be sent to a page called "action_page.php".

AN (ADVANCED) STATIC PAGE

Web forms:

- The attribute ***action*** sets the web URI that will handle the request
- The attribute ***method*** will set the HTTP request method ("get" or "post")

```
<!DOCTYPE html>
<html>
<body>

<form action="action_page.php" method="post">
  First name:<br>
  <input type="text" name="firstname" value="Mickey">
  <br>
  Last name:<br>
  <input type="text" name="lastname" value="Mouse">
  <br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called
"action_page.php".</p>

</body>
</html>
```



First name:

Mickey

Last name:

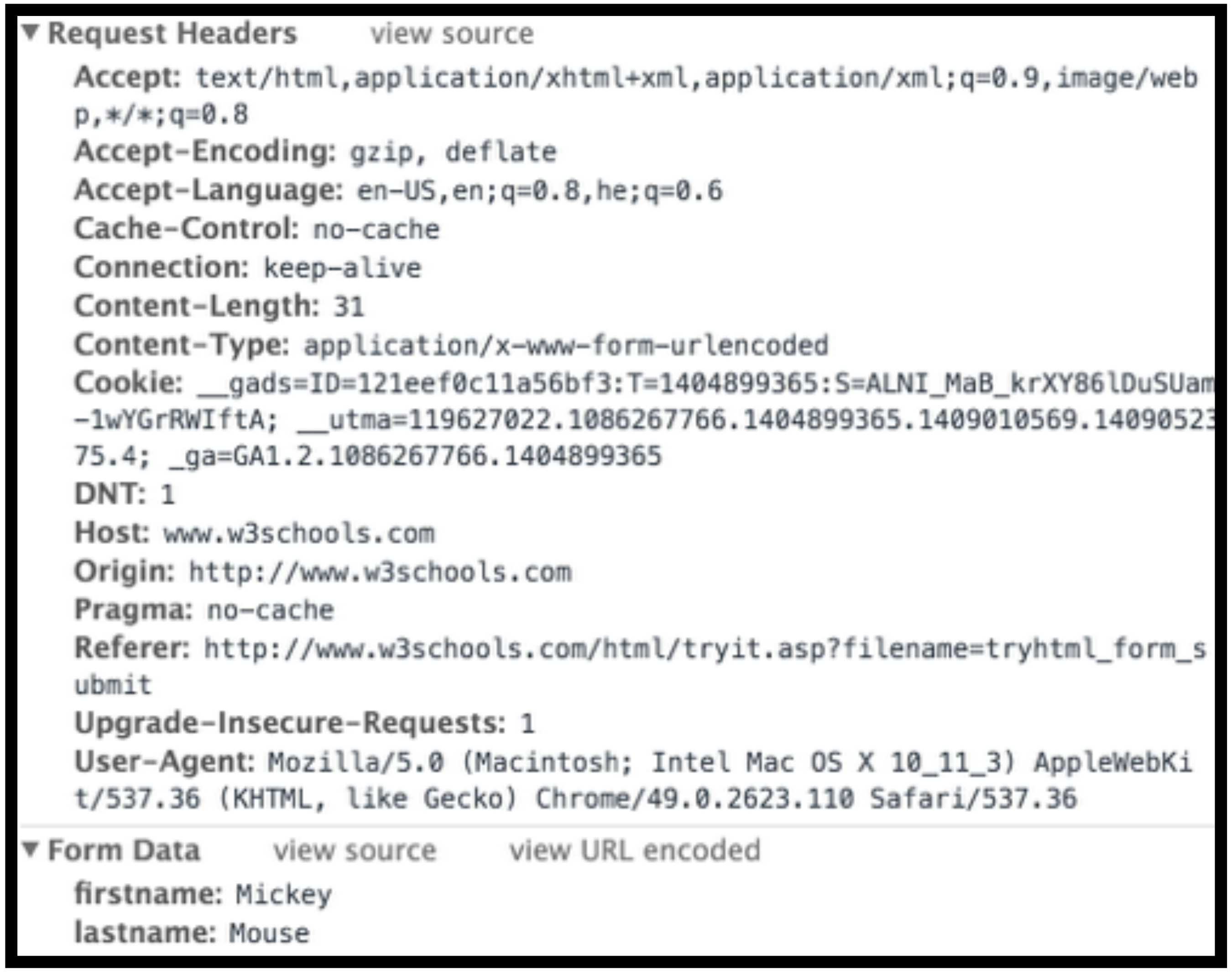
Mouse

If you click the "Submit" button, the form-data will be sent to a page called "action_page.php".

AN (ADVANCED) STATIC PAGE

Web forms:

- The attribute ***action*** sets the web URI that will handle the request
- The attribute ***method*** will set the HTTP request method ("get" or "post")
- When clicking "submit" the following HTTP request os generated:



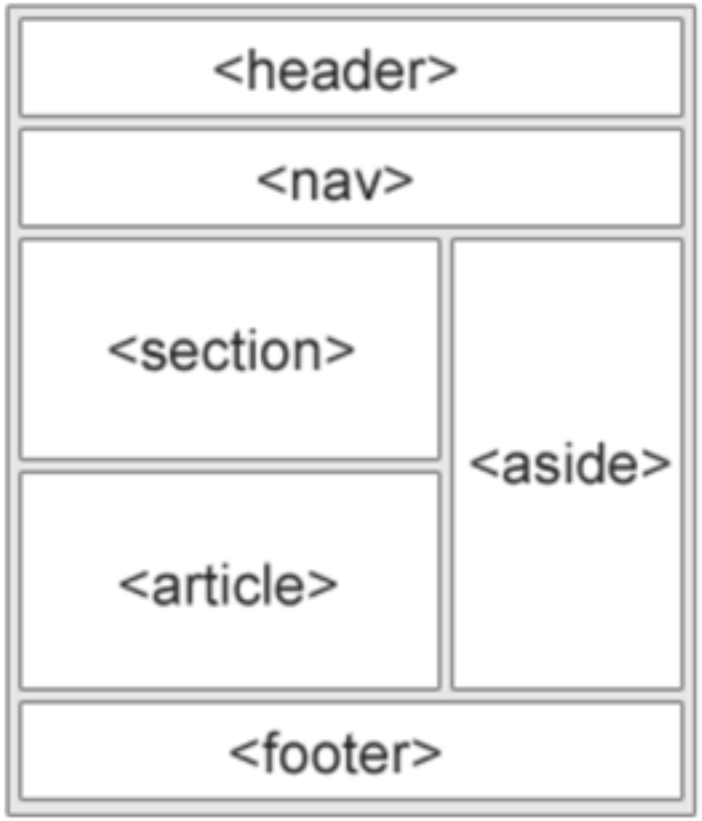
```
▼ Request Headers      view source
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/web
    p,*/*;q=0.8
    Accept-Encoding: gzip, deflate
    Accept-Language: en-US,en;q=0.8,he;q=0.6
    Cache-Control: no-cache
    Connection: keep-alive
    Content-Length: 31
    Content-Type: application/x-www-form-urlencoded
    Cookie: __gads=ID=121eef0c11a56bf3:T=1404899365:S=ALNI_MaB_krXY86lDuSUam
    -1wYGrRWIftA; __utma=119627022.1086267766.1404899365.1409010569.14090523
    75.4; _ga=GA1.2.1086267766.1404899365
    DNT: 1
    Host: www.w3schools.com
    Origin: http://www.w3schools.com
    Pragma: no-cache
    Referer: http://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_s
    ubmit
    Upgrade-Insecure-Requests: 1
    User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_3) AppleWebKi
    t/537.36 (KHTML, like Gecko) Chrome/49.0.2623.110 Safari/537.36
▼ Form Data     view source     view URL encoded
    firstname: Mickey
    lastname: Mouse
```

AN (ADVANCED) STATIC PAGE

HTML5:

- HTML was pretty "basic" and needed many 3rd party plugins (e.g. Adobe Flash)
- HTML was not standardised and the programmer had to check the rendering of her code in all browser and handle irrational browser e.g., Internet Explorer.
- HTML5 was introduced in 2014 and includes new tags, attributes and cool features such as:
  - **Graphic elements**: <canvas>, <svg>, <video>, <audio>
  - **Semantic elements:** <footer>, <article>, <section>
  - **APIs:** Geolocation, Drag and Drop, Local Storage

Server: fraglimit changed to 70
Server: g_respawnDelay changed to 5
0:00
ZM LR300 30



<header>

<nav>

<section>

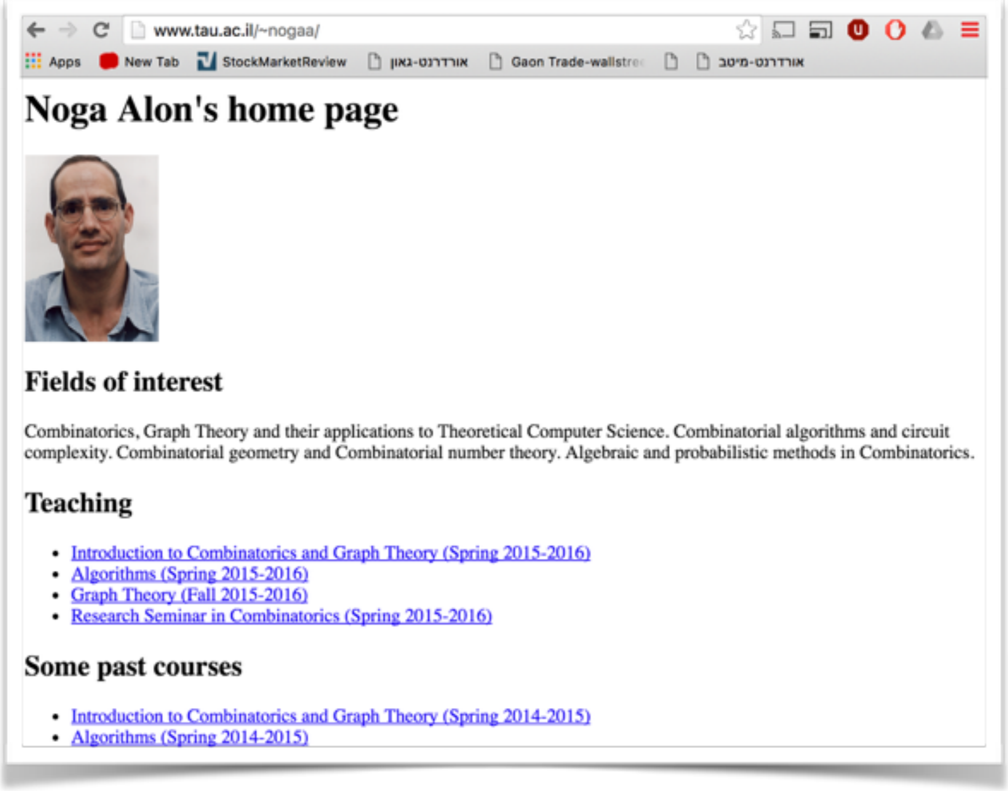
<article>

<aside>

<footer>

A STATIC WEB PAGE (WITH STYLE)

- So far we learned how to structure the content of a webpage (Like Noga)
- This is Tova’s website without style. Looks familiar?



Noga Alon's home page

Fields of interest

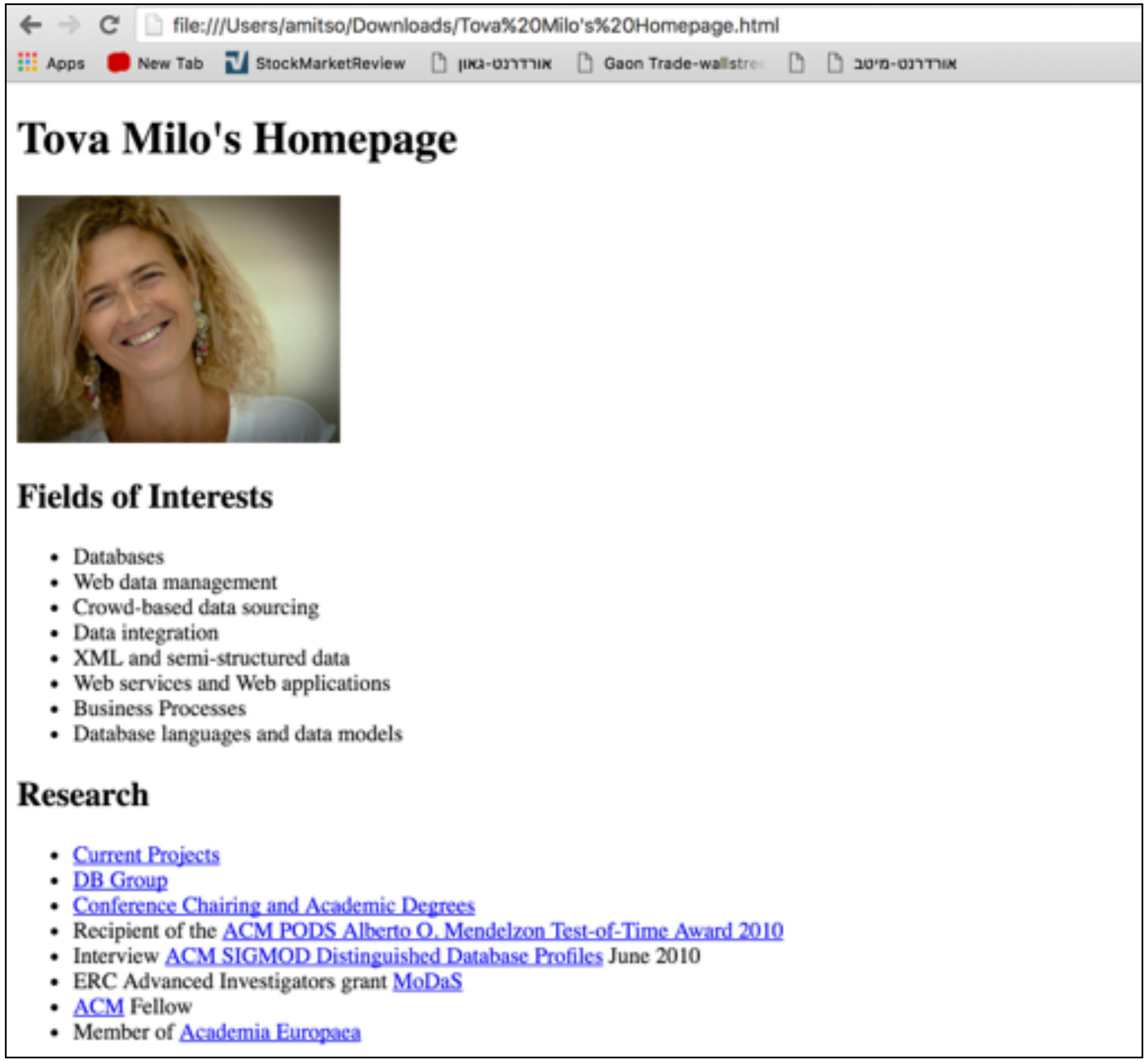
Combinatorics, Graph Theory and their applications to Theoretical Computer Science. Combinatorial algorithms and circuit complexity. Combinatorial geometry and Combinatorial number theory. Algebraic and probabilistic methods in Combinatorics.

Teaching

- Introduction to Combinatorics and Graph Theory (Spring 2015-2016)
- Algorithms (Spring 2015-2016)
- Graph Theory (Fall 2015-2016)
- Research Seminar in Combinatorics (Spring 2015-2016)

Some past courses

- Introduction to Combinatorics and Graph Theory (Spring 2014-2015)
- Algorithms (Spring 2014-2015)



Tova Milo's Homepage

Fields of Interests

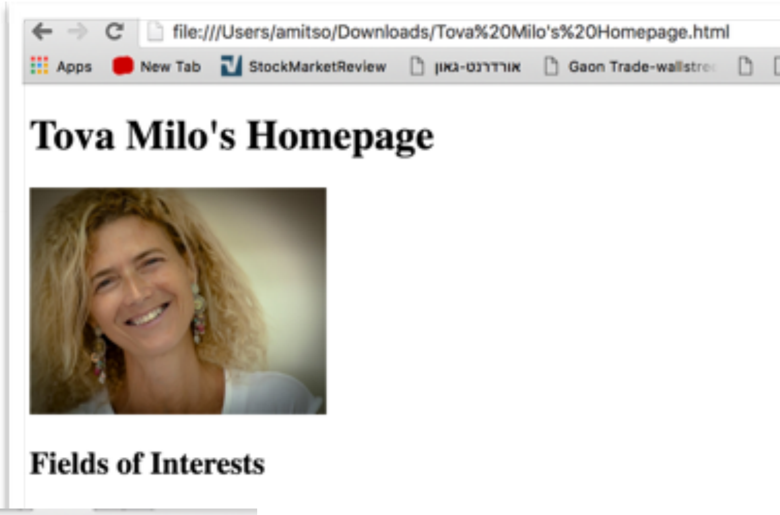
- Databases
- Web data management
- Crowd-based data sourcing
- Data integration
- XML and semi-structured data
- Web services and Web applications
- Business Processes
- Database languages and data models

Research

- Current Projects
- DB Group
- Conference Chairing and Academic Degrees
- Recipient of the ACM PODS Alberto O. Mendelzon Test-of-Time Award 2010
- Interview ACM SIGMOD Distinguished Database Profiles June 2010
- ERC Advanced Investigators grant MoDaS
- ACM Fellow
- Member of Academia Europaea

A STATIC WEB PAGE (WITH STYLE)

- For adding some “style”, we use a CSS (Cascading Style Sheet) file.

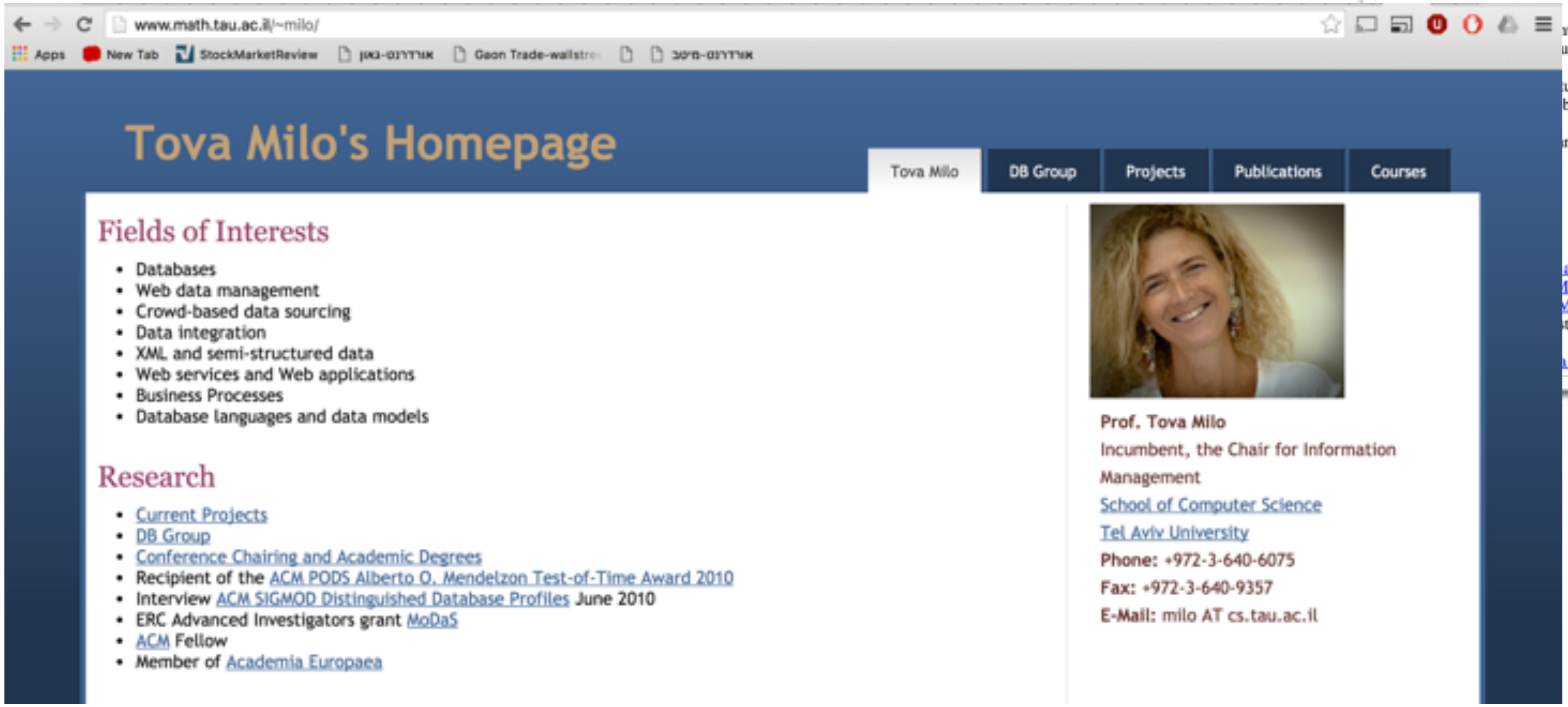

file:///Users/amitso/Downloads/Tova%20Milo's%20Homepage.html
Apps
New Tab
StockMarketReview
Gaon Trade-wallstre

Tova Milo's Homepage

Fields of Interests

…nt
…urcing
…ured data
…b applications
…nd data models

…and Academic Degrees
…I PODS Alberto O. Mendelzon Test-of-Time Award 2010
…MOD Distinguished Database Profiles June 2010
…tigators grant MoDaS
…Europaea



Tova Milo's Homepage

Fields of Interests

- Databases
- Web data management
- Crowd-based data sourcing
- Data integration
- XML and semi-structured data
- Web services and Web applications
- Business Processes
- Database languages and data models

Research

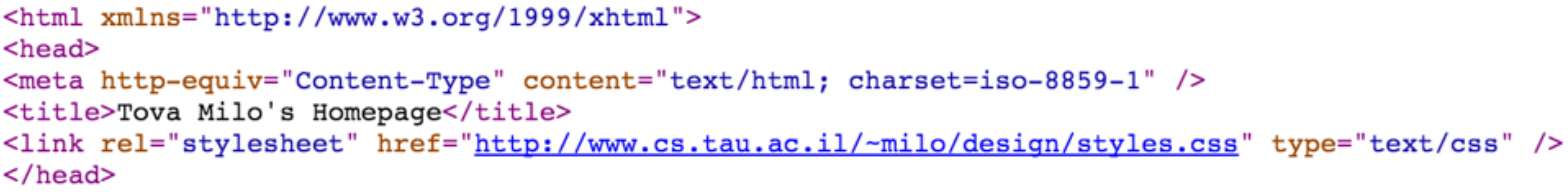
- Current Projects
- DB Group
- Conference Chairing and Academic Degrees
- Recipient of the ACM PODS Alberto O. Mendelzon Test-of-Time Award 2010
- Interview ACM SIGMOD Distinguished Database Profiles June 2010
- ERC Advanced Investigators grant MoDaS
- ACM Fellow
- Member of Academia Europaea

Prof. Tova Milo
Incumbent, the Chair for Information Management
School of Computer Science
Tel Aviv University
Phone: +972-3-640-6075
Fax: +972-3-640-9357
E-Mail: milo AT cs.tau.ac.il

A STATIC WEB PAGE (WITH STYLE)

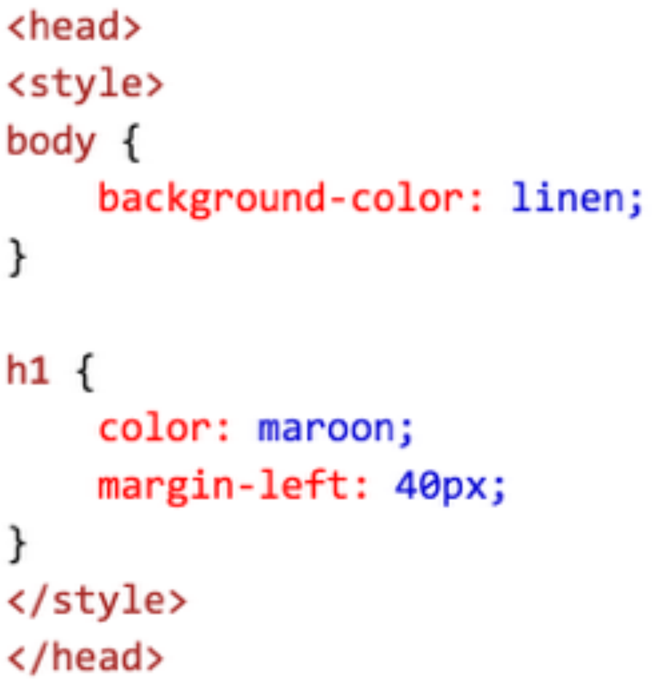
- There are 3 ways to include a CSS file. (you will use the first only).

1.External CSS file: Include a link to the stylesheet file under the <head> tag of your HTML file:



```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Tova Milo's Homepage</title>
<link rel="stylesheet" href="http://www.cs.tau.ac.il/~milo/design/styles.css" type="text/css" />
</head>
```

2.Internal Stylesheet: Include a tag <style> under the <head> tag:

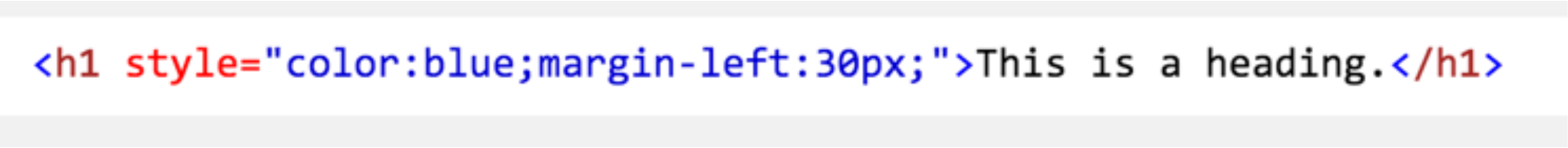


```
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
```

A STATIC WEB PAGE (WITH STYLE)

- There are 3 ways to include a CSS file. (you will use the first only).

3.Inline styling: by adding the attribute *style:*



```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

- You can use multiple style sheets.
- FYI: Your browser has its own CSS file that is used by default.
- Cascading order (first one has the highest priority):

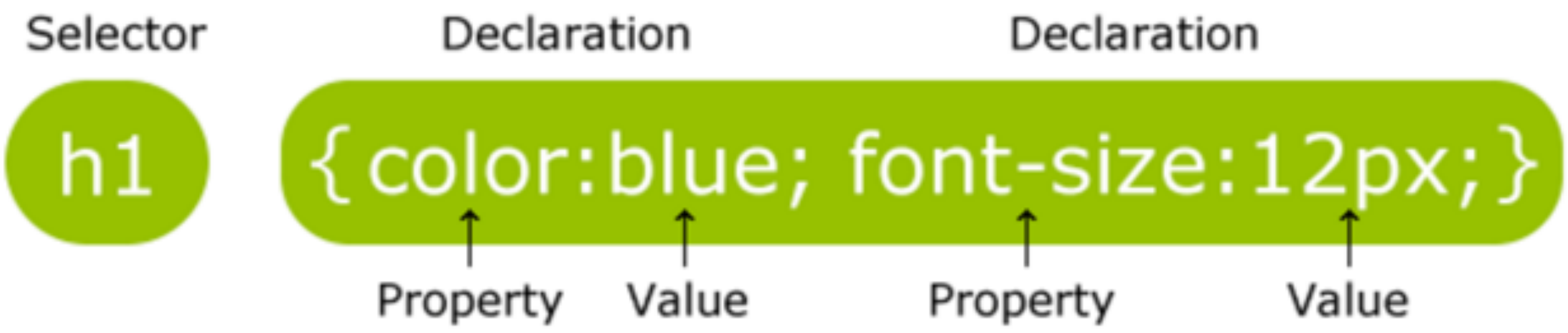
1.Inline style (inside an HTML element)

2.External and internal style sheets (in the head section)

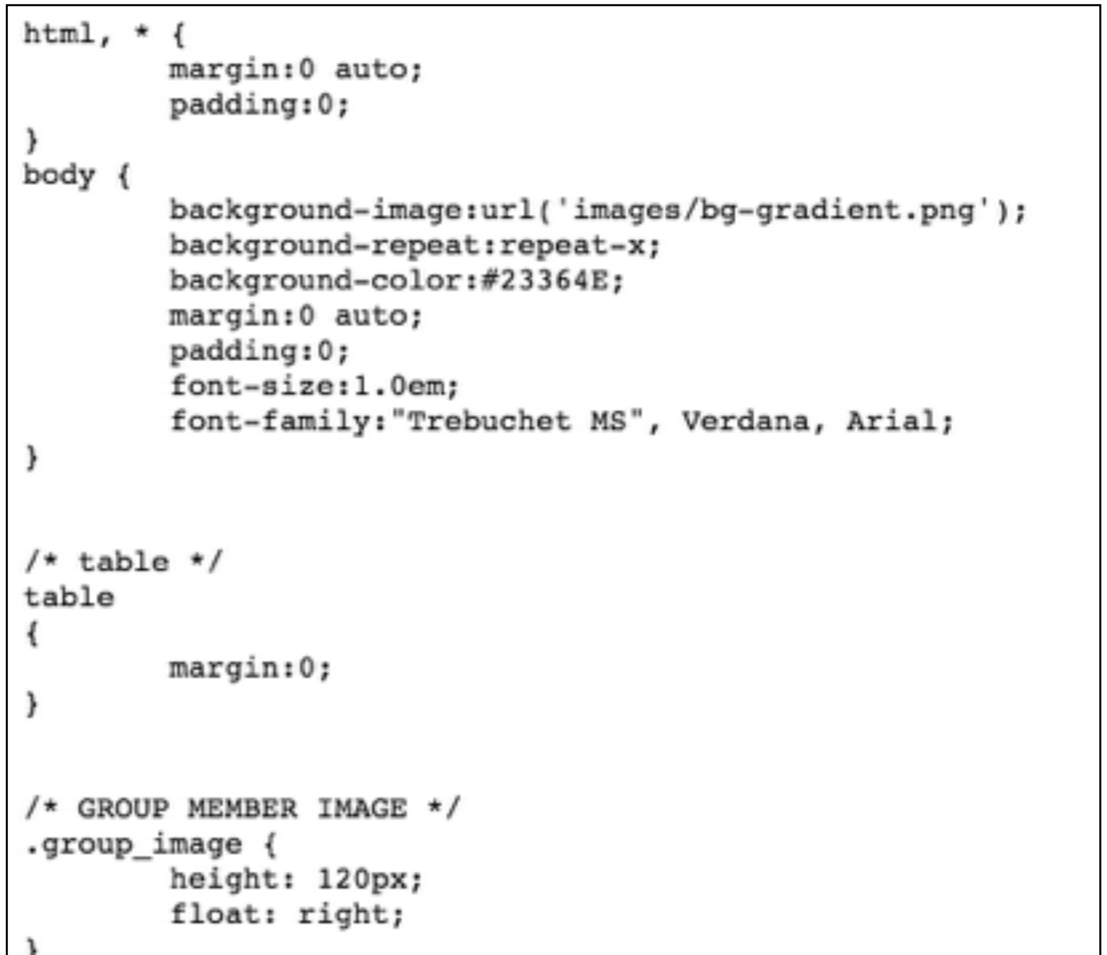
3.Browser default

A STATIC WEB PAGE (WITH STYLE)

- How to set the style of an element:



- Example of a CSS file:



```
html, * {
        margin:0 auto;
        padding:0;
}
body {
        background-image:url('images/bg-gradient.png');
        background-repeat:repeat-x;
        background-color:#23364E;
        margin:0 auto;
        padding:0;
        font-size:1.0em;
        font-family:"Trebuchet MS", Verdana, Arial;
}


/* table */
table
{
        margin:0;
}


/* GROUP MEMBER IMAGE */
.group_image {
        height: 120px;
        float: right;
}
```

CSS SELECTORS

- **Selection by element ID:** (Use when addressing unique elements)

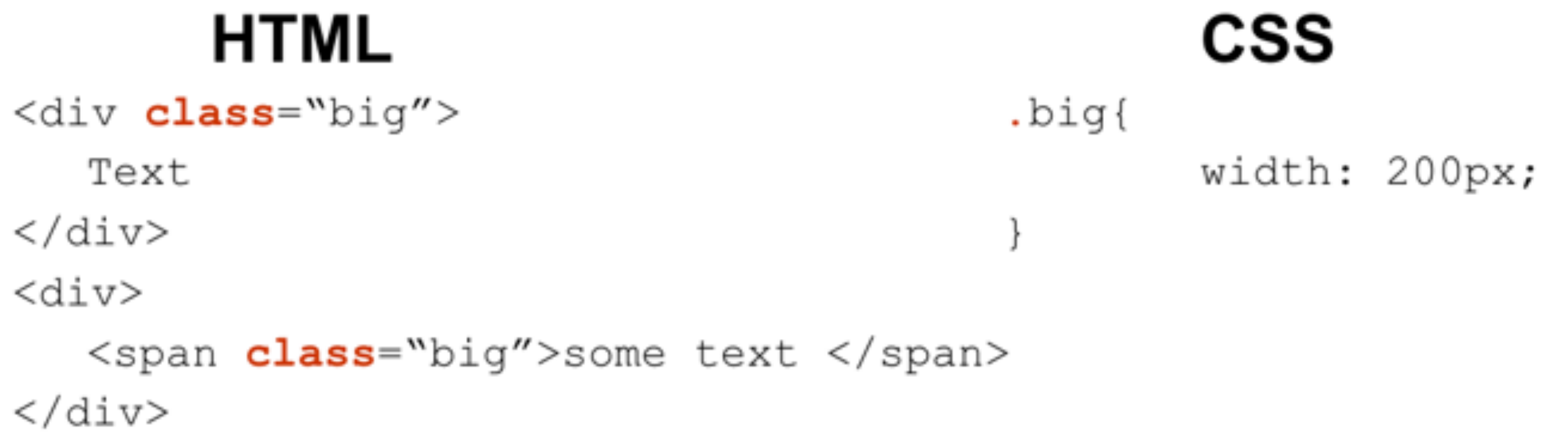
HTML

```
<div id="content">
  Text
</div>
```

CSS

```
#content {
        width: 200px;
}
```

- **Selection by element class:** (can be used for multiple elements)



HTML

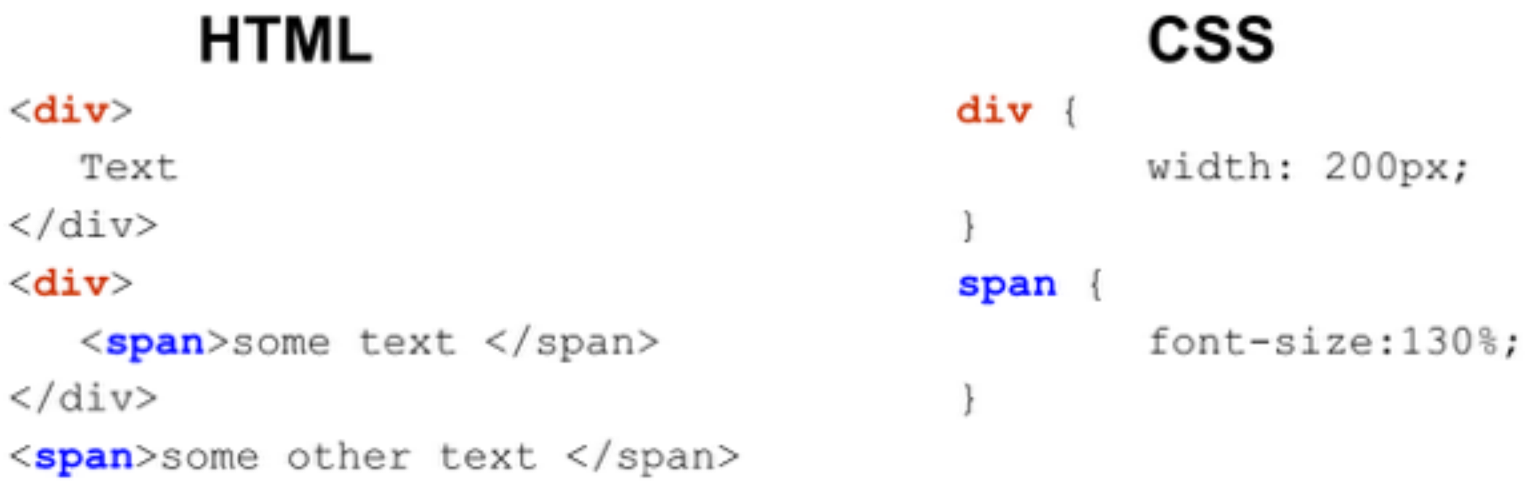
```
<div class="big">
  Text
</div>
<div>
  <span class="big">some text </span>
</div>
```

CSS

```
.big{
        width: 200px;
}
```

CSS SELECTORS

- Selection by tag:



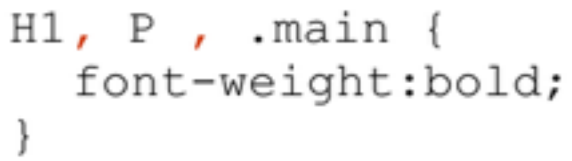
HTML

```
<div>
   Text
</div>
<div>
   <span>some text </span>
</div>
<span>some other text </span>
```

CSS

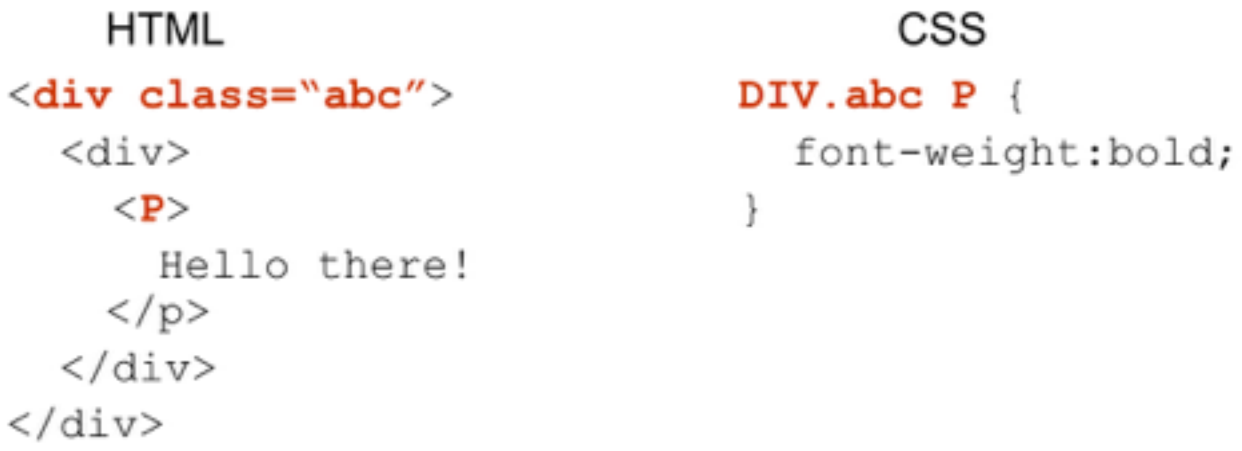
```
div {
        width: 200px;
}
span {
        font-size:130%;
}
```

- Grouping selection:



```
H1, P , .main {
   font-weight:bold;
}
```

- Descendant selection:



HTML

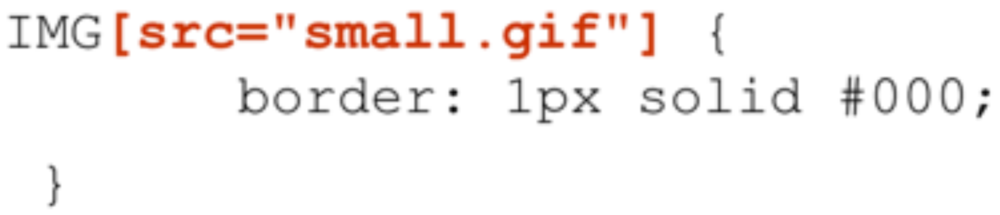
```
<div class="abc">
  <div>
    <P>
      Hello there!
    </p>
  </div>
</div>
```

CSS

```
DIV.abc P {
  font-weight:bold;
}
```

CSS SELECTORS

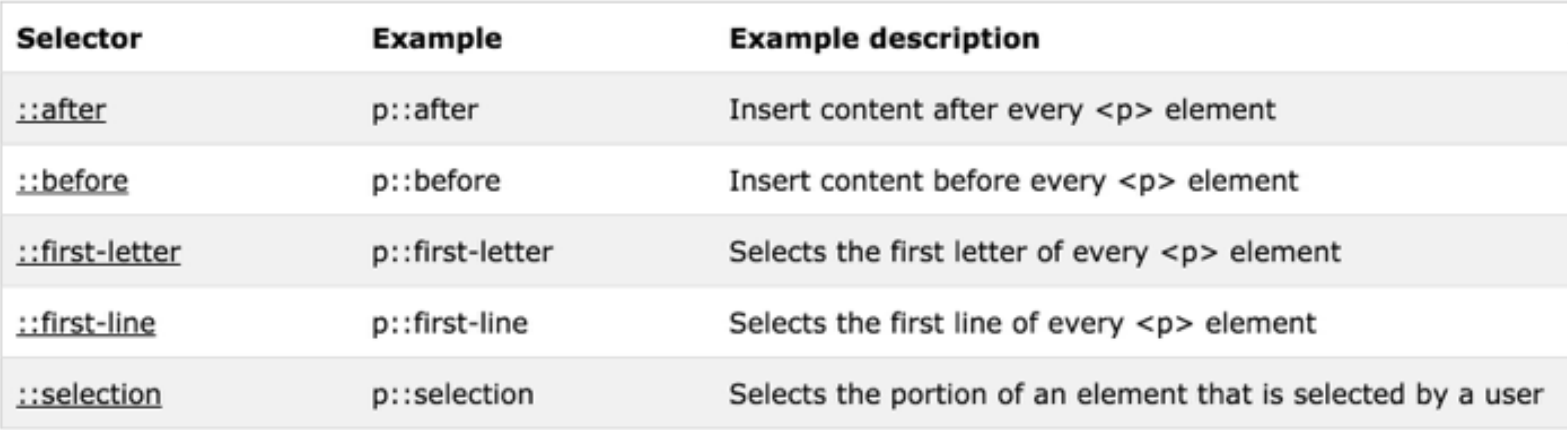
- **Attributes selection** (Attribute selectors selects elements based upon the attributes present in the HTML Tags and their value)**:**



```
IMG[src="small.gif"] {
        border: 1px solid #000;
 }
```

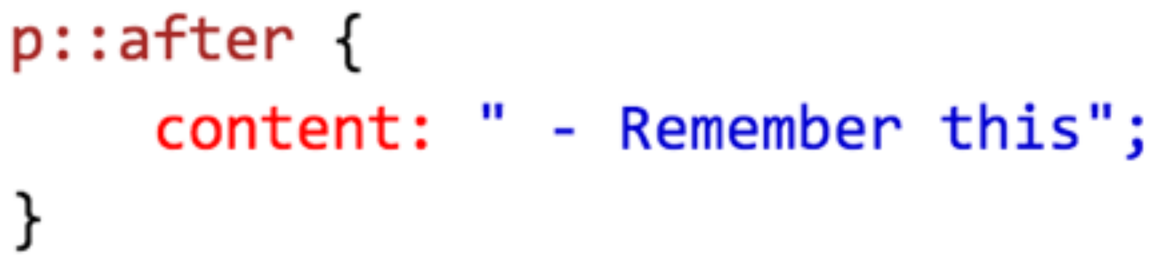
CSS PSEUDO-ELEMENTS

- Used to generate HTML content automatically.



Selector	Example	Example description
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

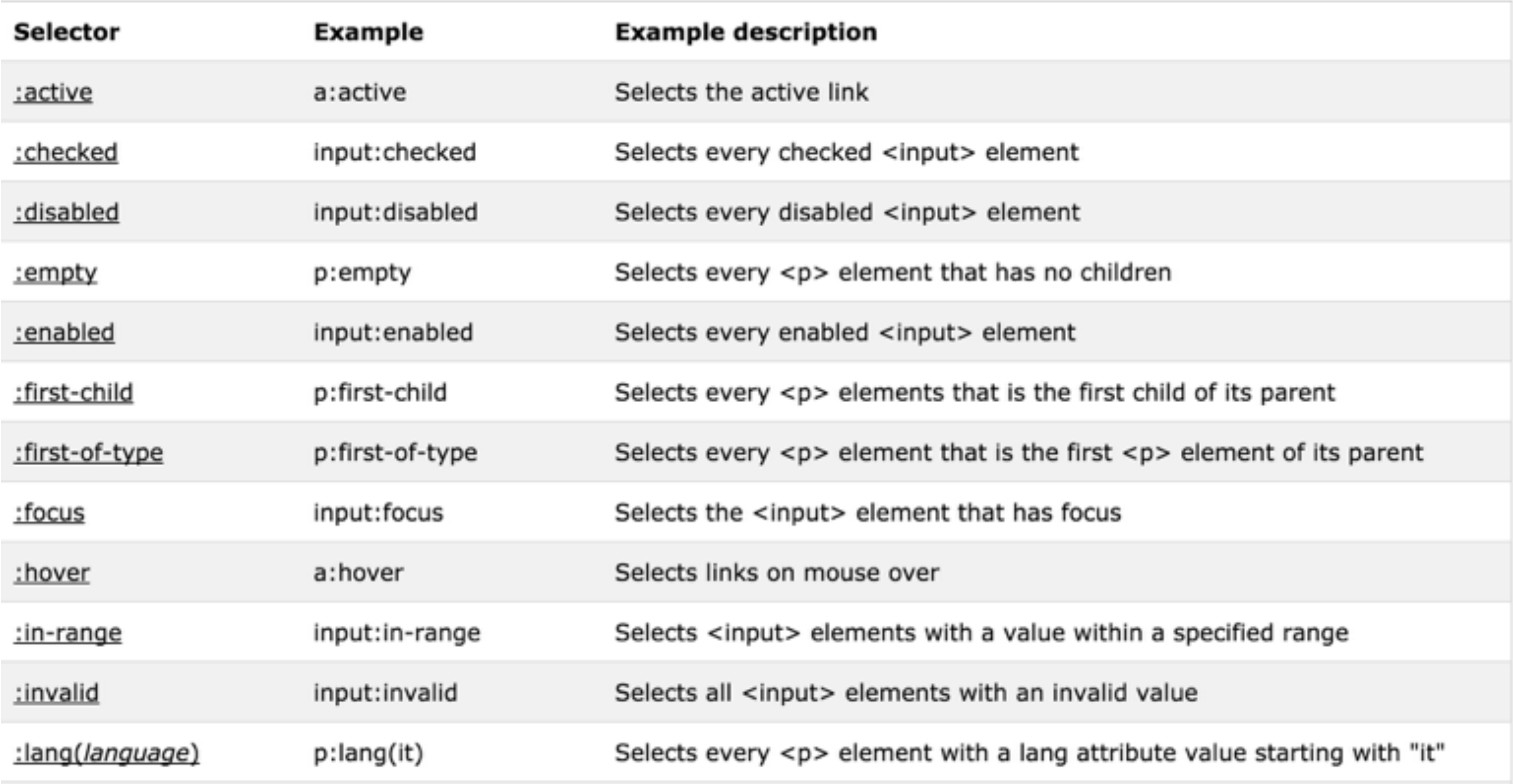
- Using the special attribute *Content:*



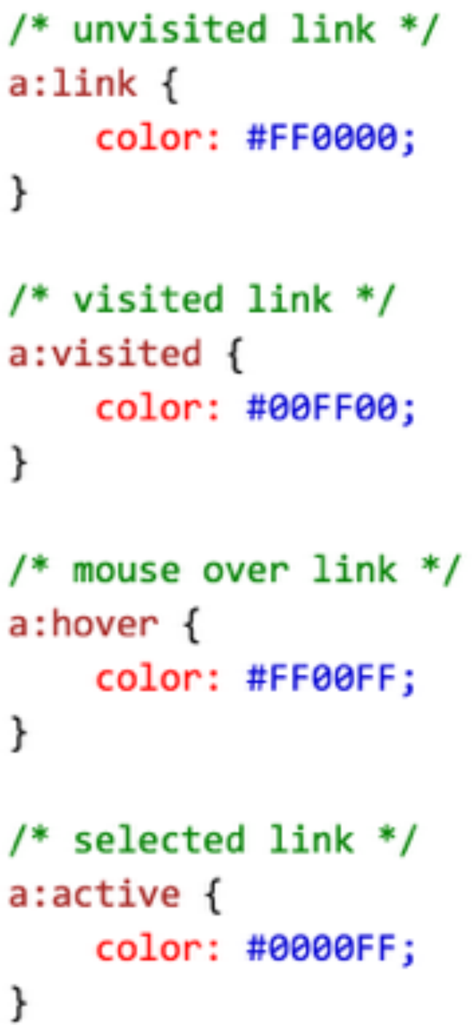
```
p::after {
    content: " - Remember this";
}
```

CSS PSEUDO-CLASSES

- Use to refer elements in different stages of execution.



Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"



```
/* unvisited link */
a:link {
    color: #FF0000;
}

/* visited link */
a:visited {
    color: #00FF00;
}

/* mouse over link */
a:hover {
    color: #FF00FF;
}

/* selected link */
a:active {
    color: #0000FF;
}
```

THE BOX MODEL

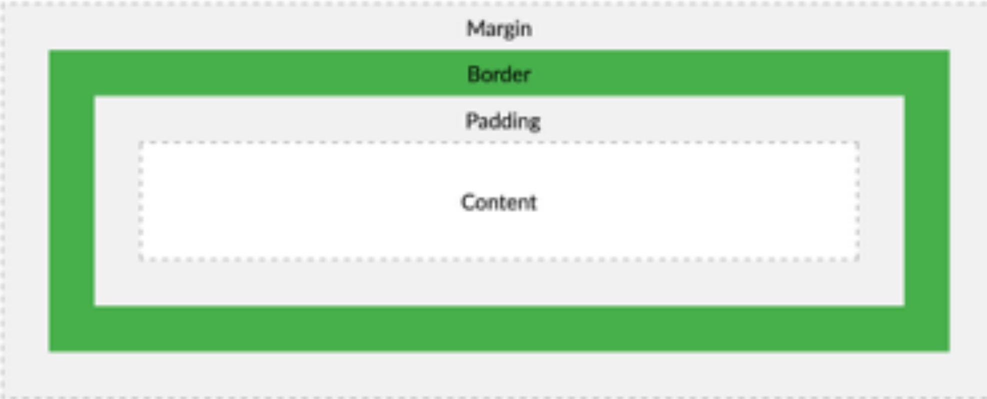
- **All HTML elements are considered as “boxes” . The box model allows us to add a border around elements, and to define space between elements:**

  Content: The content of the box, where text and images appear

  Padding: Clears an area around the content. Padding is transparent

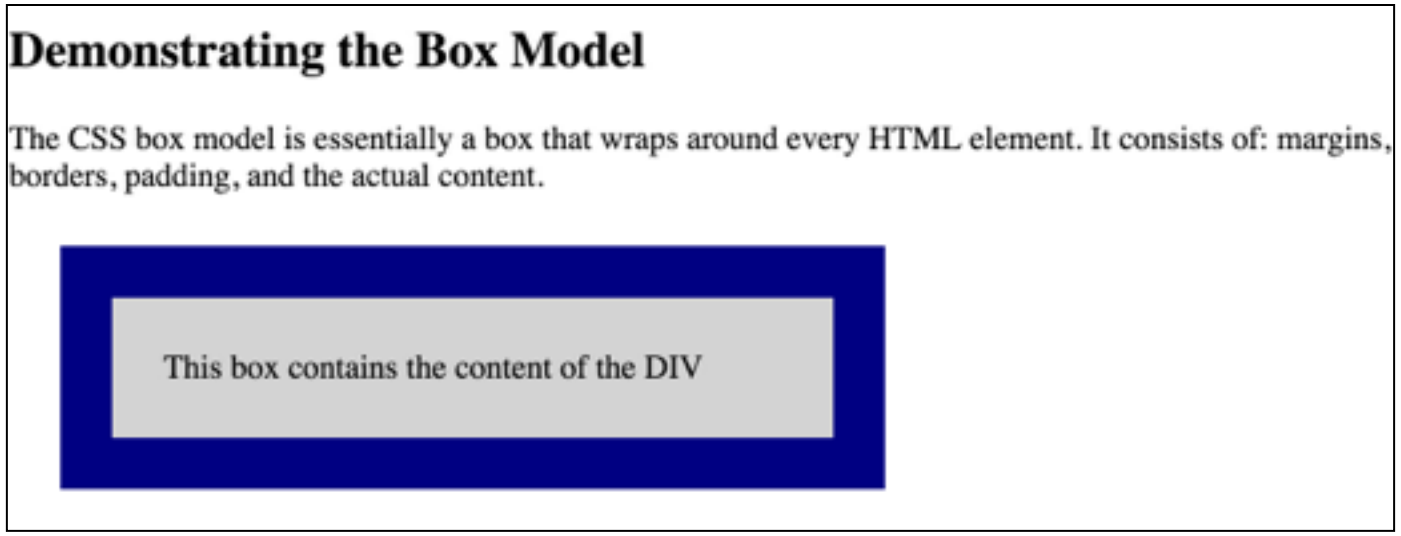
  Border: A border that goes around the padding and content

  Margin: Clears an area outside the border.The margin is transparent



- **For Example:**

```
div {
    width: 300px;
    padding: 25px;
    border: 25px solid navy;
    margin: 25px;
}
```



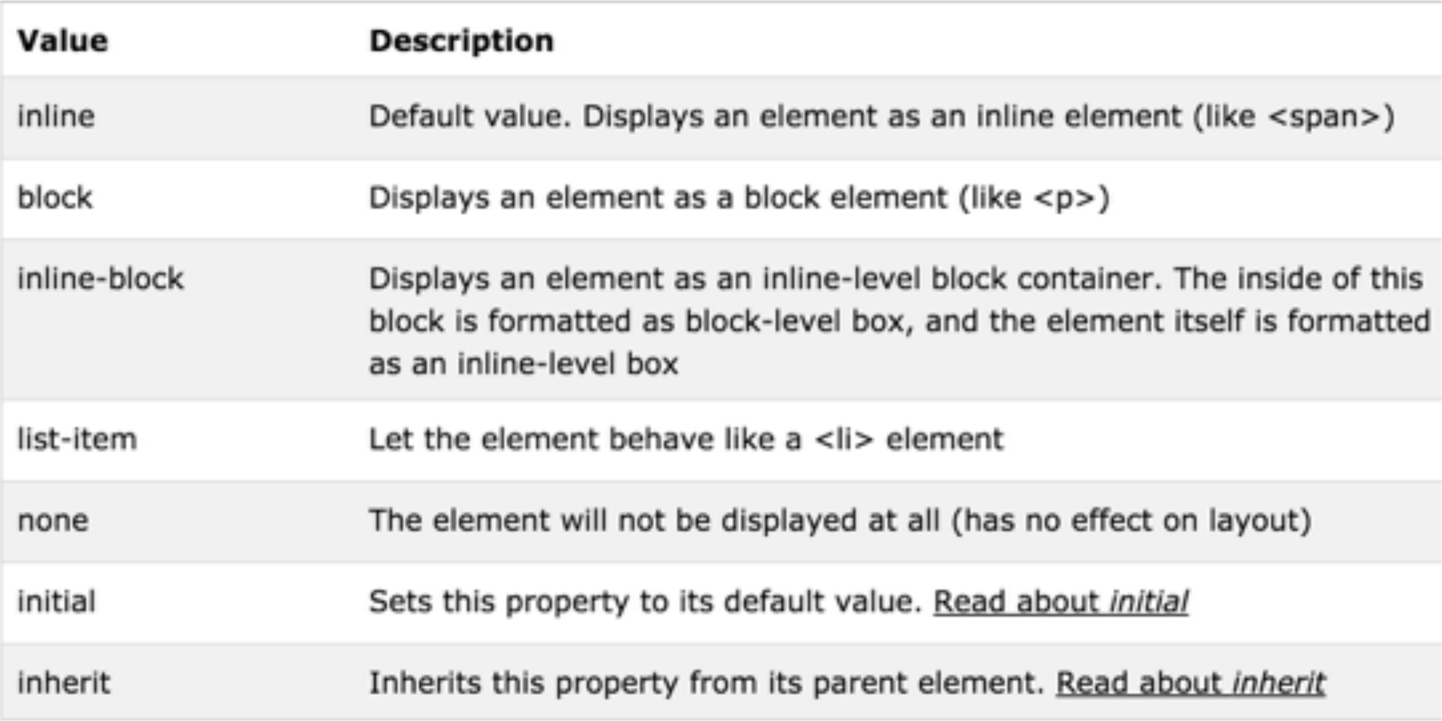
Demonstrating the Box Model

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.

This box contains the content of the DIV

CSS: DISPLAY AND POSITION

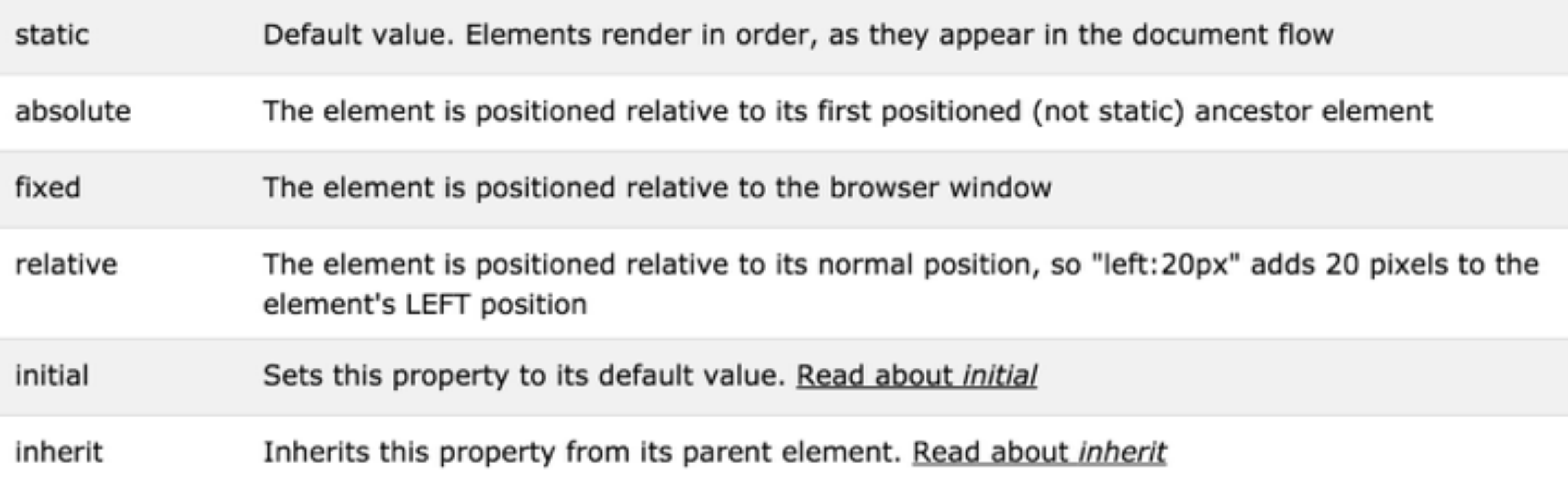
- **These are the most important attributes in CSS.**
  - FYI: It is a nightmare to deal with. Go through this tutorial : http://www.w3schools.com/css/css_positioning.asp
- **There are 2 types of elements: *Block* and *Inline***
  - **Block (e.g. DIV, FORM,H1..H6):** starts in new line , always extend to the full width available.
  - **Inline (e.g. SPAN, IMG, A)** does not start on a new line and only takes up as much width as necessary
- **The *Display* attribute:** can alter the element's type or hide it completely.



Value	Description
inline	Default value. Displays an element as an inline element (like <span>)
block	Displays an element as a block element (like <p>)
inline-block	Displays an element as an inline-level block container. The inside of this block is formatted as block-level box, and the element itself is formatted as an inline-level box
list-item	Let the element behave like a <li> element
none	The element will not be displayed at all (has no effect on layout)
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element. Read about *inherit*

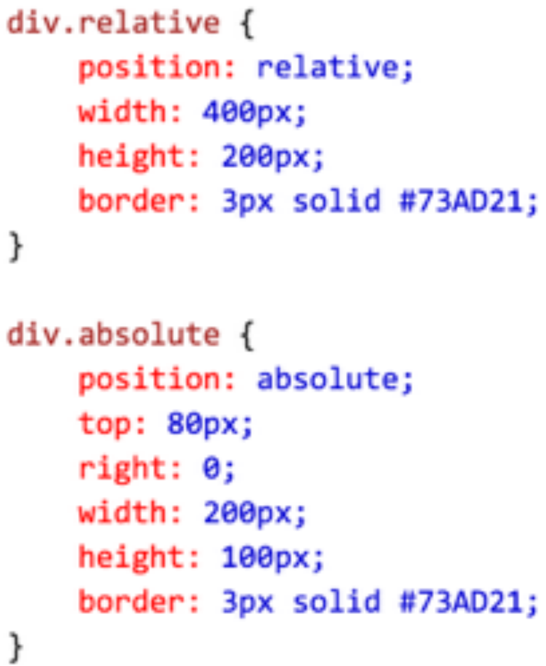
CSS: DISPLAY AND POSITION

- Positioning of elements:

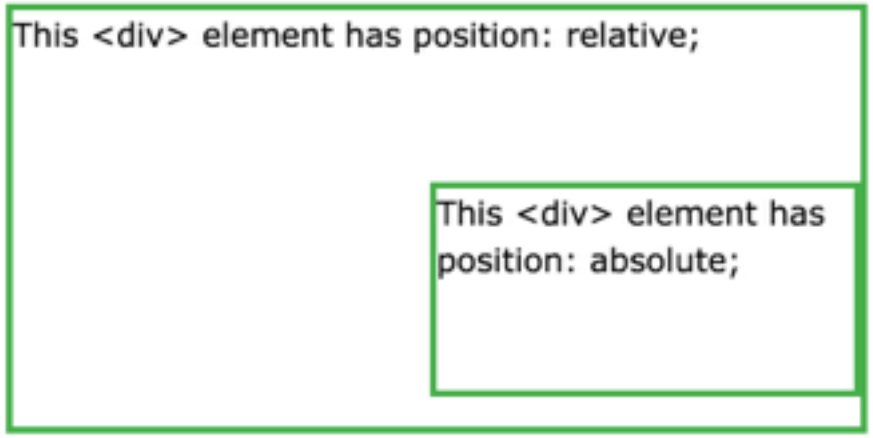


static	Default value. Elements render in order, as they appear in the document flow
absolute	The element is positioned relative to its first positioned (not static) ancestor element
fixed	The element is positioned relative to the browser window
relative	The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element. Read about *inherit*

- Example:



```
div.relative {
    position: relative;
    width: 400px;
    height: 200px;
    border: 3px solid #73AD21;
}

div.absolute {
    position: absolute;
    top: 80px;
    right: 0;
    width: 200px;
    height: 100px;
    border: 3px solid #73AD21;
}
```



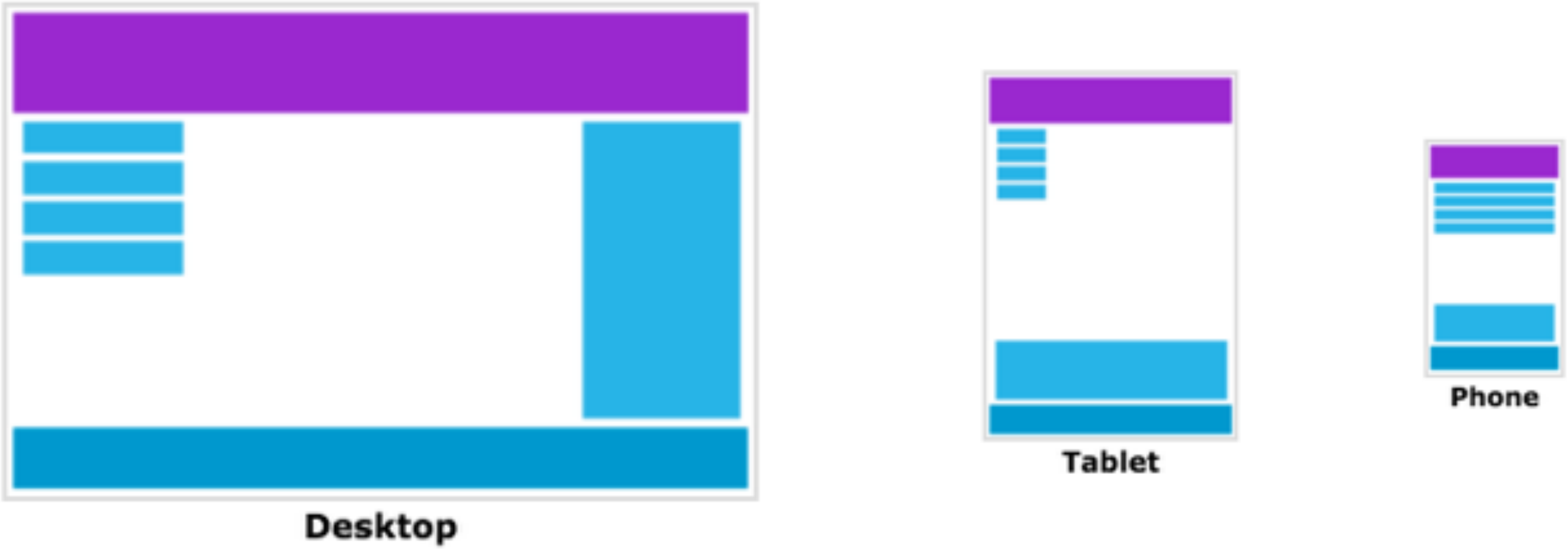
This <div> element has position: relative;

This <div> element has position: absolute;

CSS AND RESPONSIVE WEB DESIGN

Responsive web design makes your web page look good on all devices.

- Responsive web design uses only HTML and CSS.
- Responsive web design is not a program or a JavaScript.



Defining the viewport: the main visible area for the user on any device

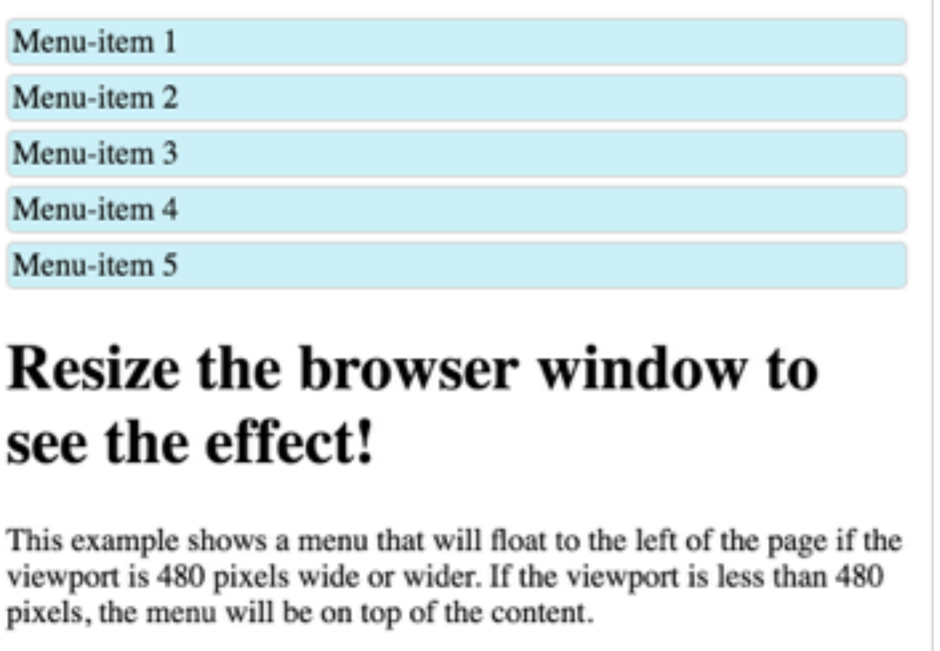
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

CSS AND RESPONSIVE WEB DESIGN

- **Media queries: Use CSS media queries to apply different styling for small and large screens.**
- **Example:**

•Original styling is for mobile

```
#main {margin-left: 4px;}
#leftsidebar {float: none;width: auto;}
```



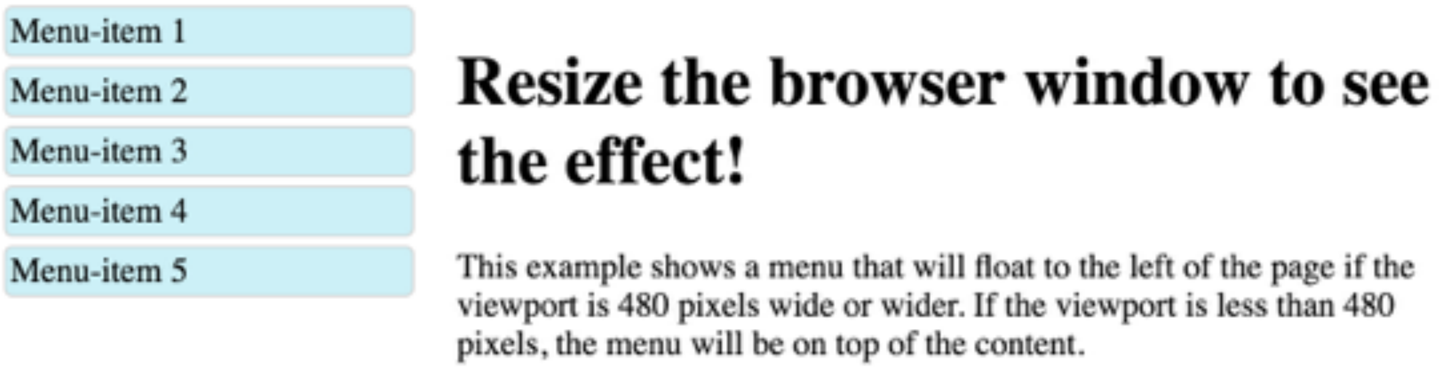
Menu-item 1
Menu-item 2
Menu-item 3
Menu-item 4
Menu-item 5

Resize the browser window to see the effect!

This example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider. If the viewport is less than 480 pixels, the menu will be on top of the content.

•Applying media query for bigger screens:

```
@media screen and (min-width: 480px) {
    #leftsidebar {width: 200px; float: left;}
    #main {margin-left:216px;}
}
```



Menu-item 1
Menu-item 2
Menu-item 3
Menu-item 4
Menu-item 5

Resize the browser window to see the effect!

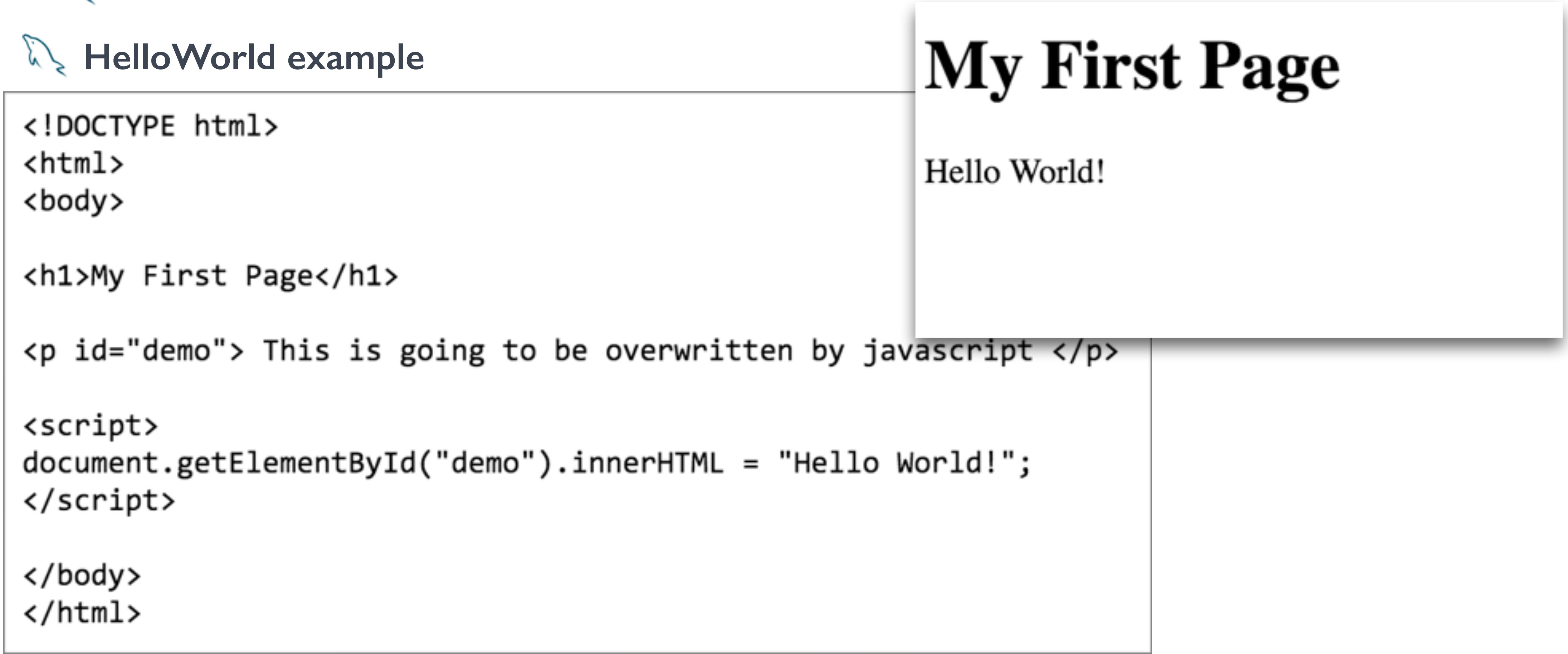
This example shows a menu that will float to the left of the page if the viewport is 480 pixels wide or wider. If the viewport is less than 480 pixels, the menu will be on top of the content.

JAVASCRIPT: WHAT AND WHY

- **Javascript is the client-side programming language.**
  - It is not Java and not related to Java by nothing. (Sun was involved somehow and therefore the name)
  - It is high-level, dynamic, untyped, and interpreted programming language
  - Syntax is C based (but semi-colon is not obligatory)
  - Code is evaluated by the web browser
  - It can traverse the DOM tree and handle browser events (e.g. click on a link, pressing a key)
- **To include a Javascript file use the tag <script>**
  - Internal: Just type your js code
  - External (recommended) **<script src=external.js > </script>**

JAVASCRIPT: HELLO WORLD

- **Javascript basic features:**
  - **Traverse the tree using the *document* *reserved word.***
  - **Function *getElementByID("id"):*** finds the HTML element
  - **Variable *innerHTML*** : holds the element HTML content
- **HelloWorld example**



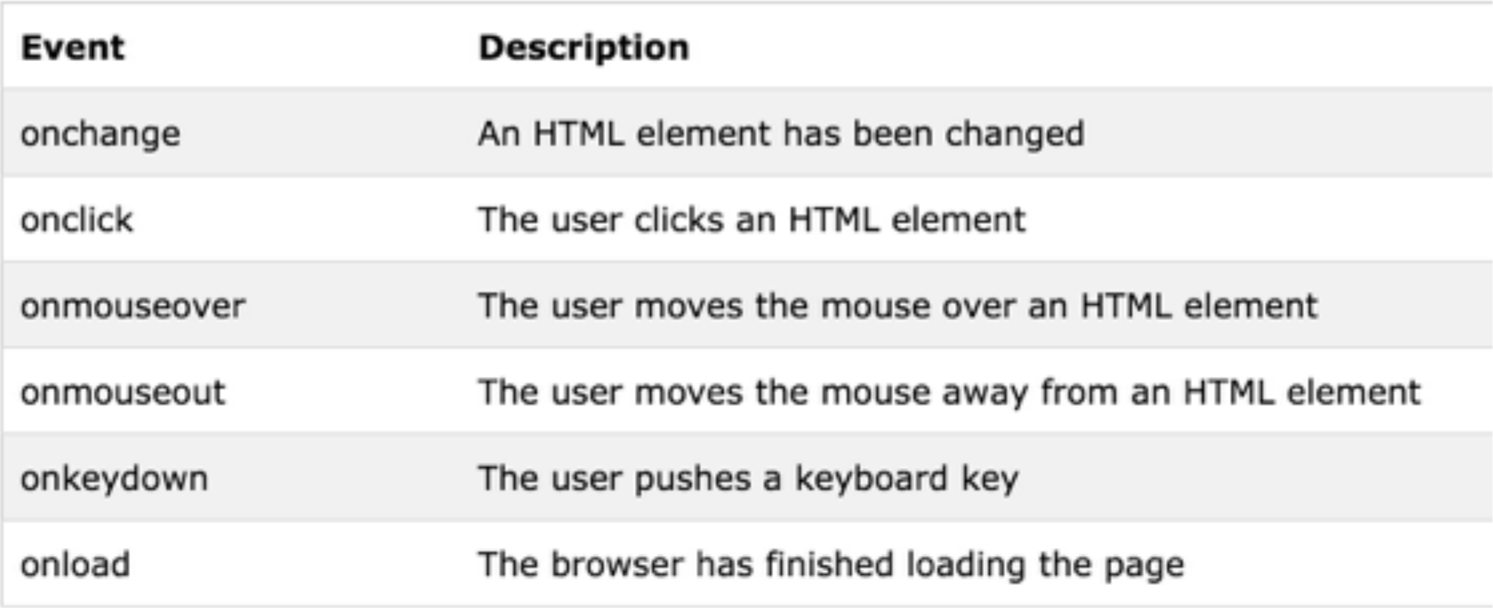
```
<!DOCTYPE html>
<html>
<body>

<h1>My First Page</h1>

<p id="demo"> This is going to be overwritten by javascript </p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

My First Page

Hello World!

JAVASCRIPT: DOM EVENTS

This are the main events that happen in the web browser (there are more):



Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

With these events JS can do:

- Things that should be done every time a page loads/closed.
- Action that should be performed when a user clicks a button.

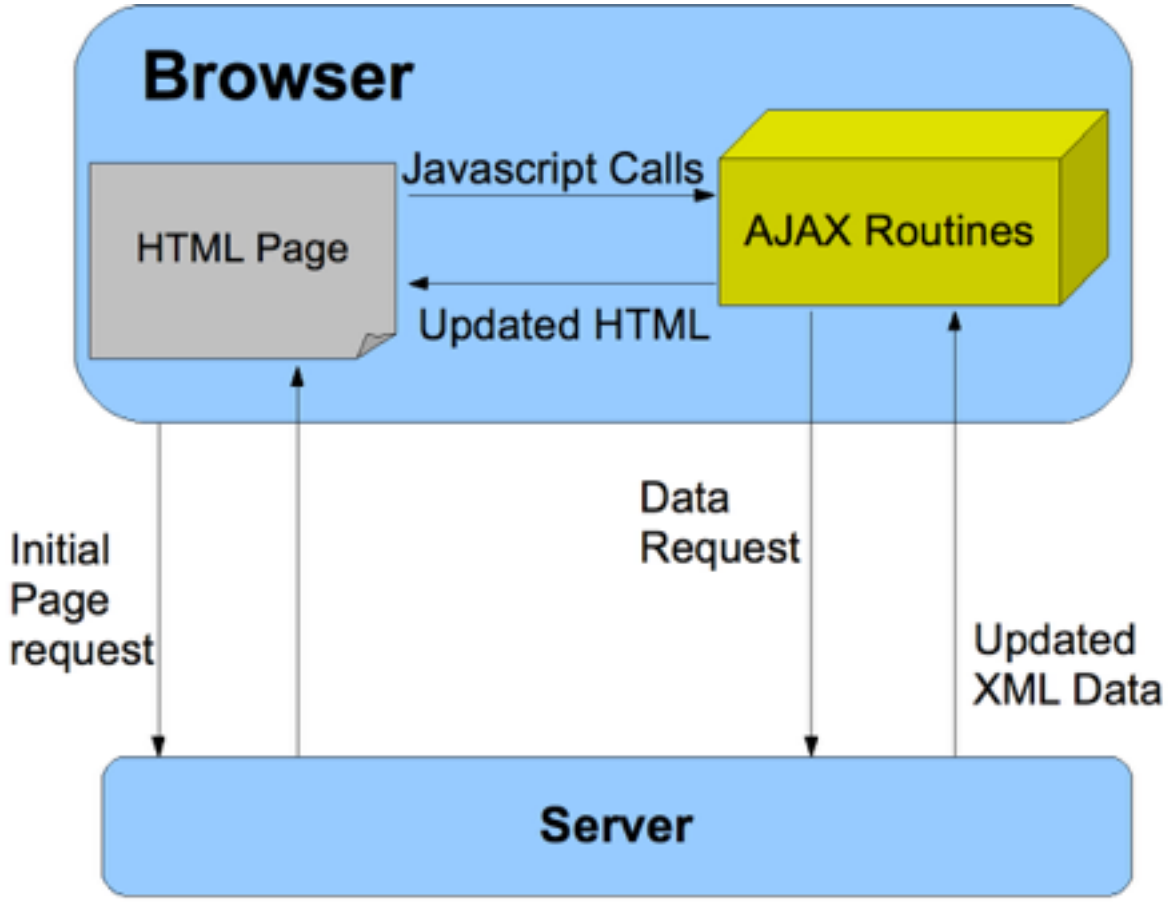
Important:

- HTML event attributes can execute JavaScript code directly / call JavaScript Functions.
- You can assign your own event handler functions to HTML elements/ prevent handling events

JAVASCRIPT: AJAX

AJAX: asynchronous JavaScript and XML. Lets you:

- Update a web page without reloading the page
- Request and receive data from a server - after the page has loaded
- Send data to a server - in the background

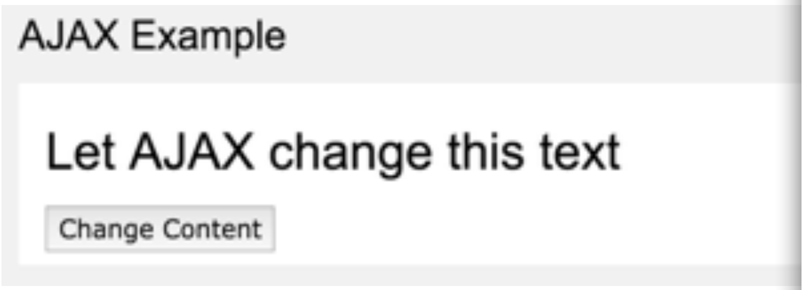


JAVASCRIPT: AJAX

An example showing everything :

- The HTML page contains a <div> section and a <button>.
- The <div> section is used to display information from a server.
- The <button> calls a function (if it is clicked).
- The function requests data from a web server and displays it:

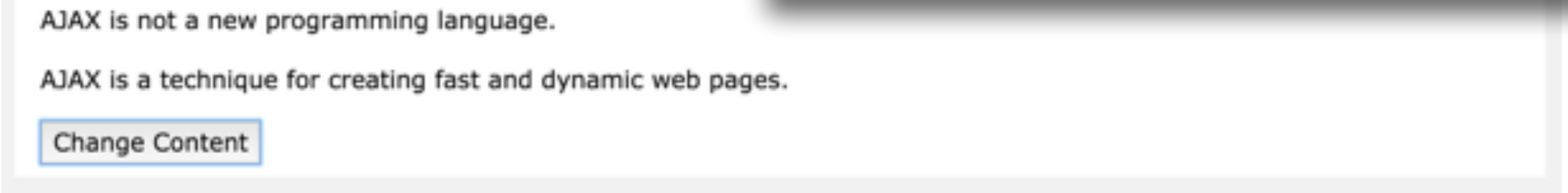
•BEFORE CLICKING



AJAX Example

Let AJAX change this text

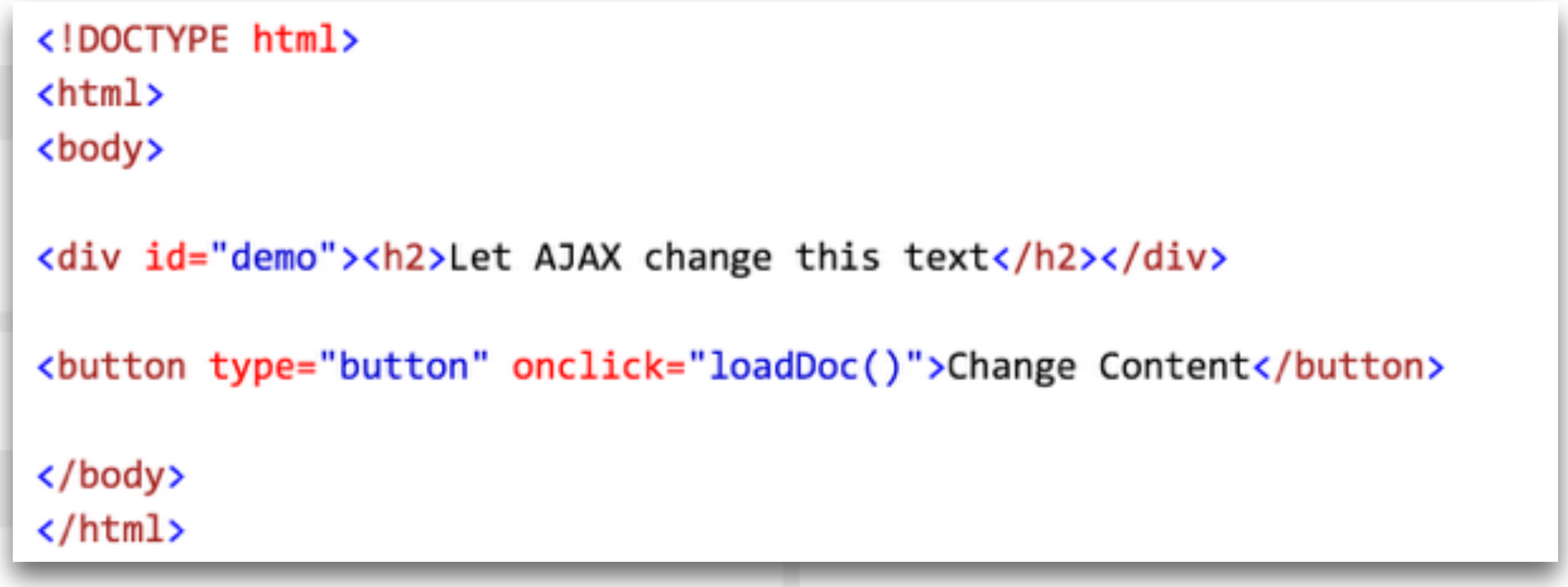
•AFTER CLICKING



AJAX Example

AJAX is not a new programming language.

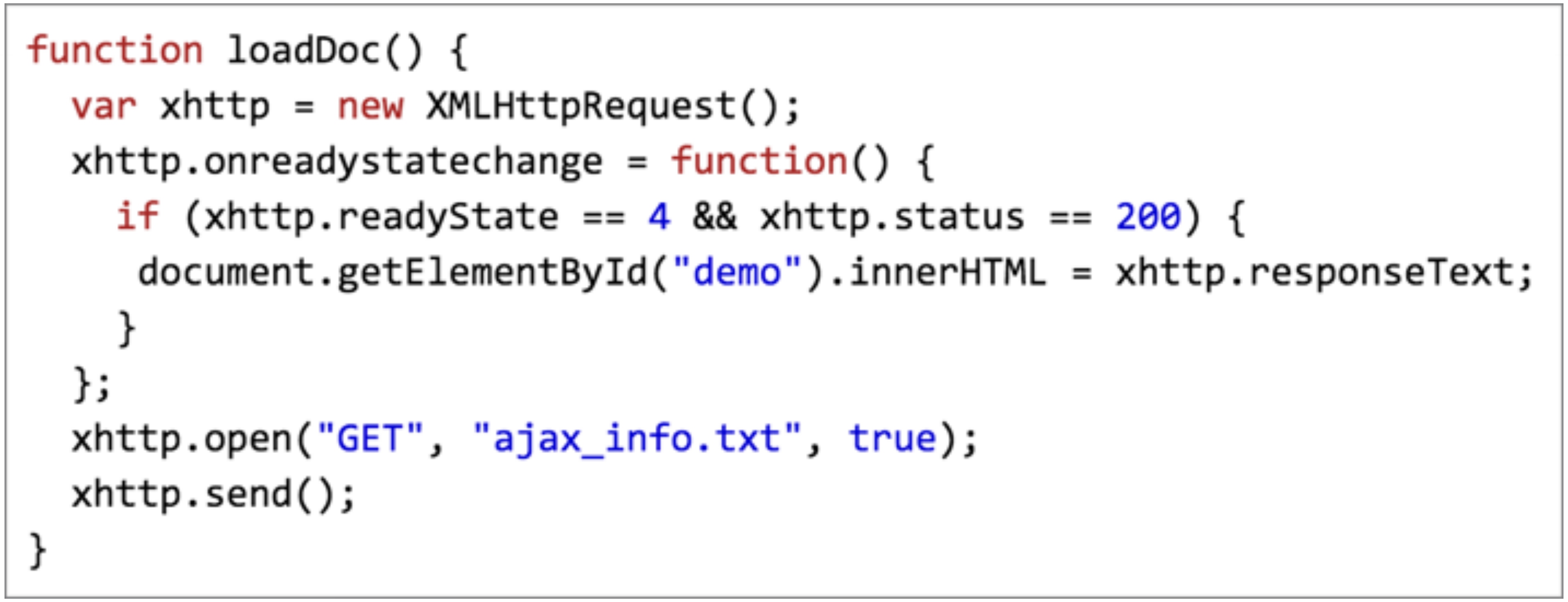
AJAX is a technique for creating fast and dynamic web pages.



```
<!DOCTYPE html>
<html>
<body>

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

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

</body>
</html>
```

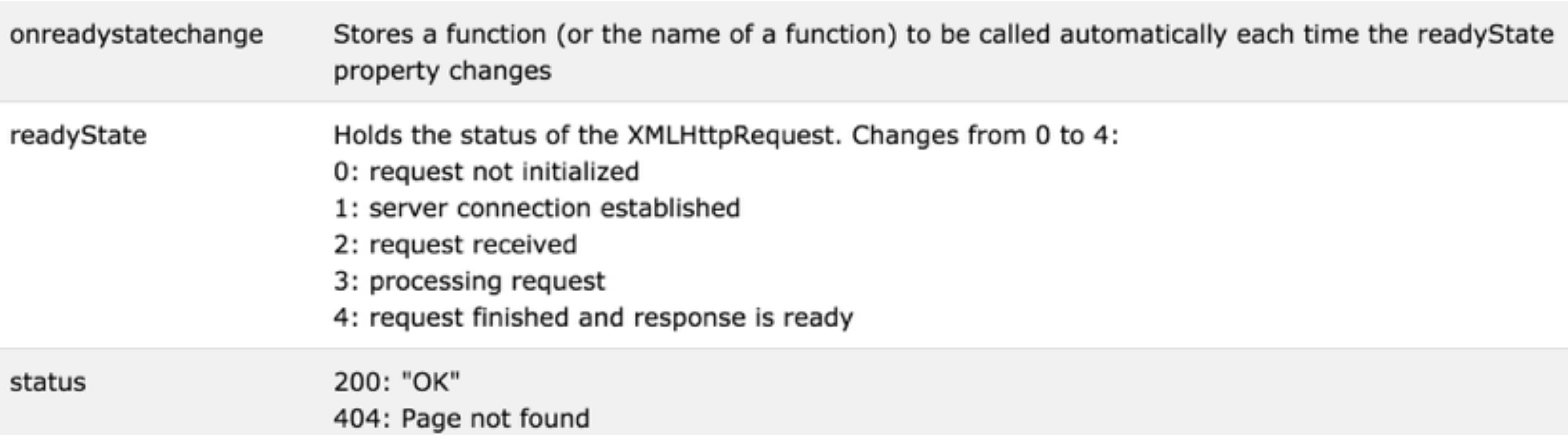
JAVASCRIPT: AJAX

- The JavaScript Code:



```
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();
}
```

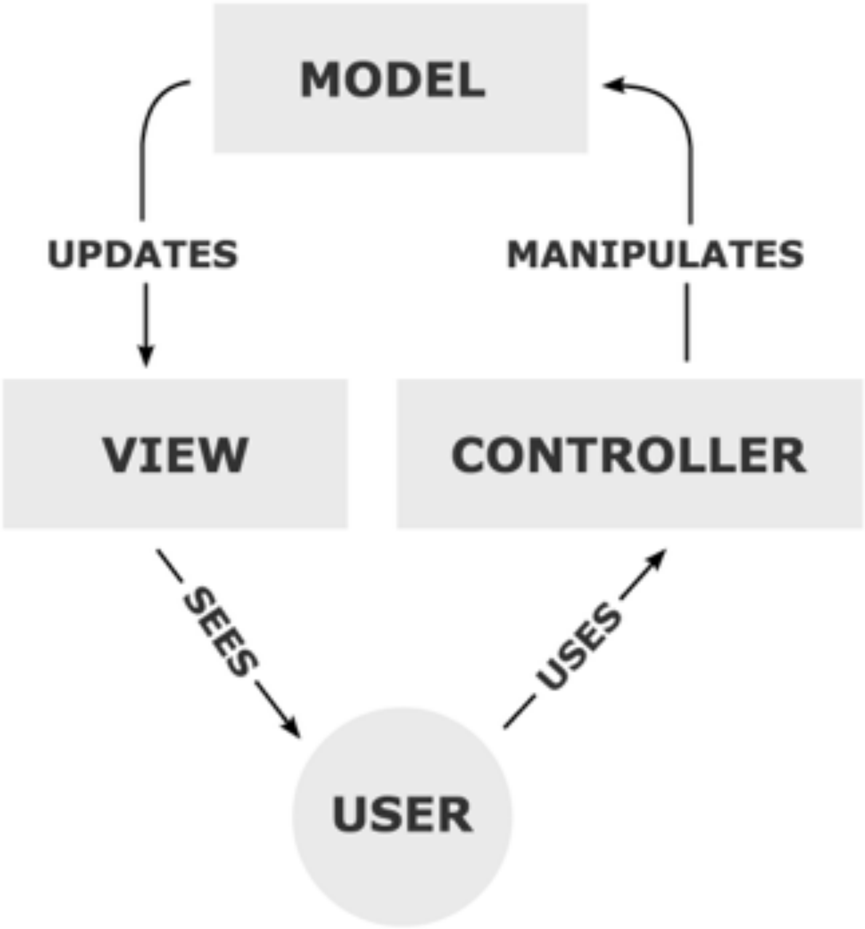
- Ready states:



onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

MVC: MODEL-VIEW-CONTROLLER

- **Why designing a page from scratch when you can rely on existing libraries that extend HTML, CSS and JavaScript?**
  - ★ **jQuery:** a JavaScript Library that simplifies JavaScript Programming
  - ★ **Angular JS:** extends HTML by adding new tags and features
  - ★ **Bootstrap:** An HTML+CSS+JavaScript framework for developing **Responsive** websites
- **The MVC approach separates the UI (views) from the data and logics (models) and let them communicate via designated I/O methods (controllers)**

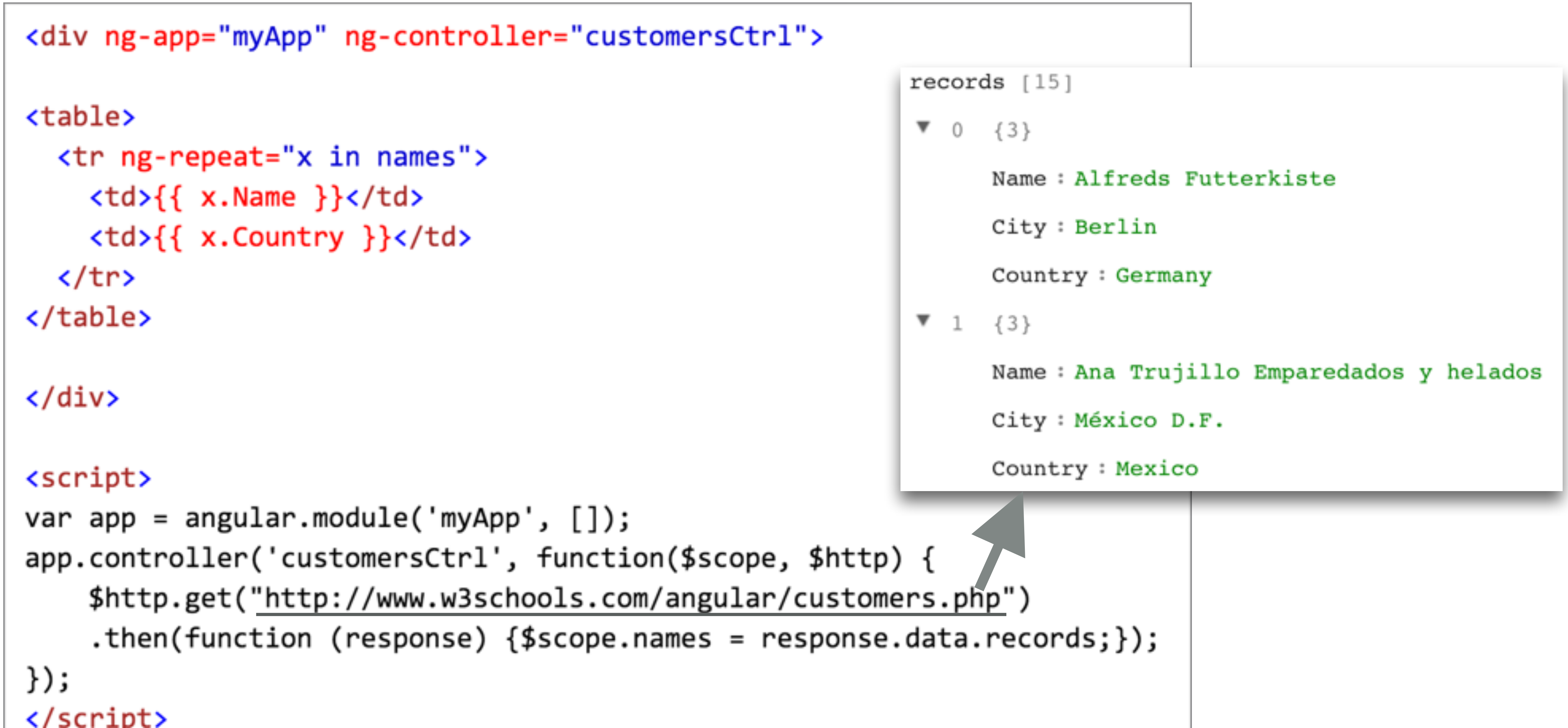


ANGULAR JS: EXAMPLE

- How to install Angular? Simply include the following script in your <head> scope:

```
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
```

- A bit complicated example for creating a table with data from the server:

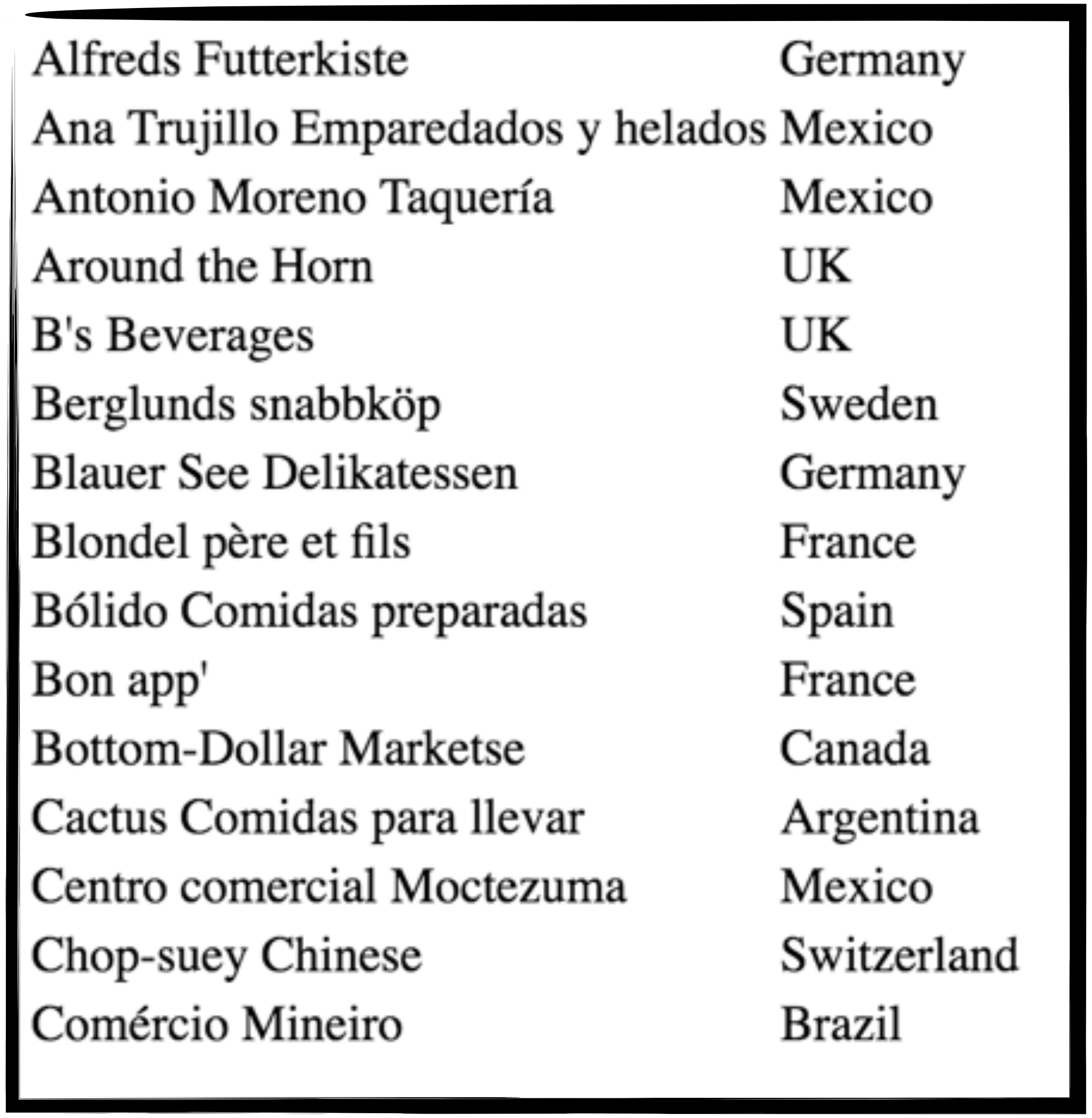


```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("http://www.w3schools.com/angular/customers.php")
    .then(function (response) {$scope.names = response.data.records;});
});
</script>
```

```
records [15]
▼ 0   {3}
        Name : Alfreds Futterkiste
        City : Berlin
        Country : Germany
▼ 1   {3}
        Name : Ana Trujillo Emparedados y helados
        City : México D.F.
        Country : Mexico
```

ANGULAR JS: EXAMPLE

- **The results:**



Alfreds Futterkiste	Germany
Ana Trujillo Emparedados y helados	Mexico
Antonio Moreno Taquería	Mexico
Around the Horn	UK
B's Beverages	UK
Berglunds snabbköp	Sweden
Blauer See Delikatessen	Germany
Blondel père et fils	France
Bólido Comidas preparadas	Spain
Bon app'	France
Bottom-Dollar Marketse	Canada
Cactus Comidas para llevar	Argentina
Centro comercial Moctezuma	Mexico
Chop-suey Chinese	Switzerland
Comércio Mineiro	Brazil

AGENDA FOR TODAY

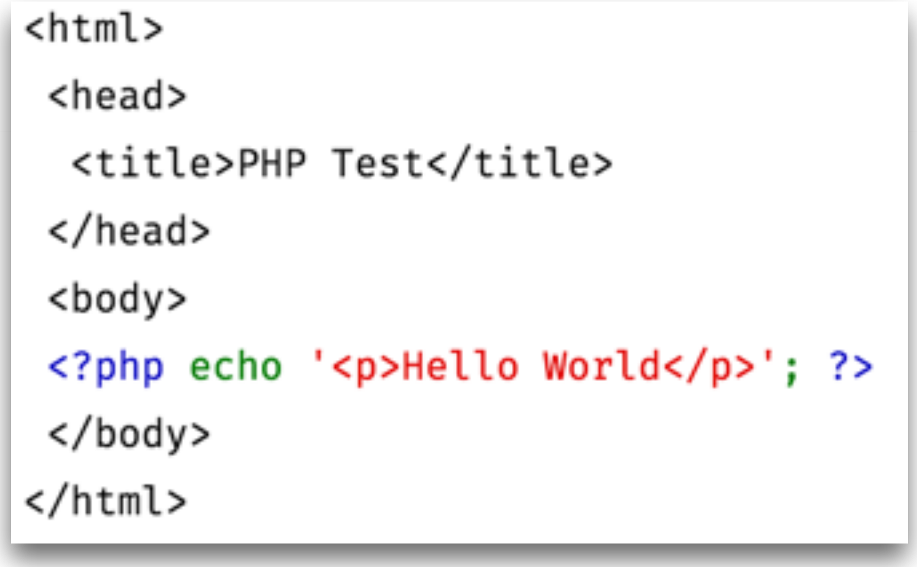
- **Client side programming**
  - HTML
  - CSS
  - Javascript
  - Additional libraries: Bootstrap,Angular, Jquery
- **Server side programming: PHP**
  - Install XAMPP
  - Web server architecture
  - php+mysql
- **Web APIs: REST ,Json, and how to get them via Python**

php: INTRODUCTION

- **PHP stands for *“PHP: Hypertext Preprocessor”***
- **What is PHP:** a **server-side scripting** language designed for **web development**
- **Why PHP:**
  - ★PHP is one of the leading web development languages.
  - ★PHP is compatible with almost all servers used today (Apache, IIS, etc.)
  - ★PHP is free. Download it from the official PHP resource: www.php.net
  - ★PHP is easy to learn and runs efficiently on the server side
- **What can PHP do:**
  - ★PHP can generate dynamic HTML content
  - ★PHP can collect and process user input from GET and POST requests
  - ★PHP can send and receive cookies
  - ★PHP can add, delete, modify data in your database

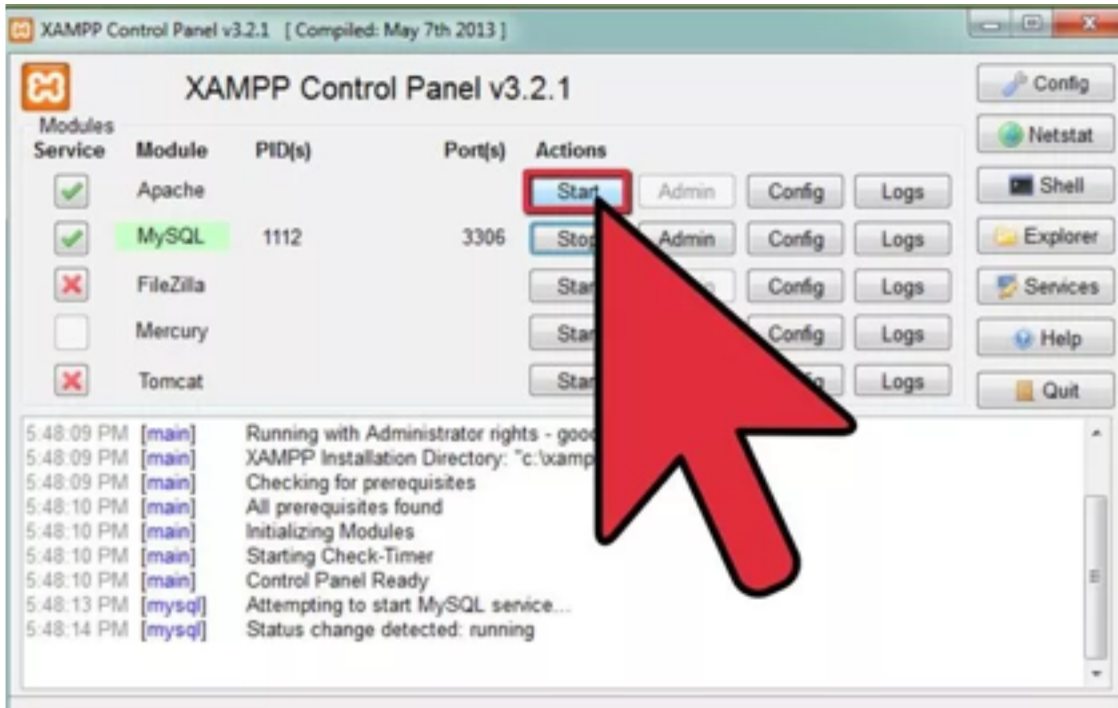
PHP: INSTALLING + HELLOWORLD

- **PHP is already installed in the university servers. Do the following:**
  1. Connect to NOVA
  2. Create a new directory called html
  3. create a hello.php file —>
  4. access the url: cs.tau.ac.il/~<your_username>/hello.php



```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <?php echo '<p>Hello World</p>'; ?>
 </body>
</html>
```

- **On your local machine (Windows):**
  1. Install XAMPP (or WAMP) from https://www.apachefriends.org/download.html
  2. It will install Apache (web server) + MySQL database + PHP + phpMyAdmin
  3. Open the XAMPP control panel and click **Start** on everything.
  4. Open the folder C:/xampp/htdocs and create hello.php
  5. Access from your web browser:
     1. http://localhost/hello.php



XAMPP Control Panel v3.2.1 [Compiled: May 7th 2013]

XAMPP Control Panel v3.2.1

Modules Service	Module	PID(s)	Port(s)	Actions
✓	Apache			Start Admin Config Logs
✓	MySQL	1112	3306	Stop Admin Config Logs
☒	FileZilla			Star Config Logs
□	Mercury			Star Config Logs
☒	Tomcat			Star Logs

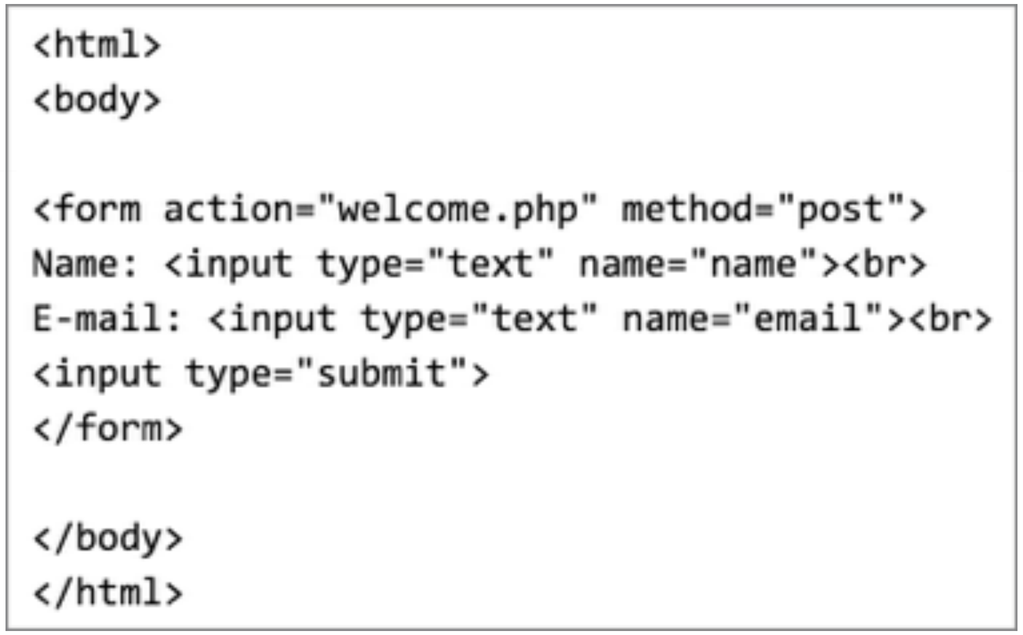
```
5:48:09 PM  [main]   Running with Administrator rights - good
5:48:09 PM  [main]   XAMPP Installation Directory: "c:\xamp
5:48:09 PM  [main]   Checking for prerequisites
5:48:10 PM  [main]   All prerequisites found
5:48:10 PM  [main]   Initializing Modules
5:48:10 PM  [main]   Starting Check-Timer
5:48:10 PM  [main]   Control Panel Ready
5:48:13 PM  [mysql]  Attempting to start MySQL service...
5:48:14 PM  [mysql]  Status change detected: running
```

PHP: SUPER GLOBALS

- **SuperGlobals:** PHP has several predefined arrays that are “super globals”: means they are available in all scopes throughout a script without using any special prefix.
- **And they are:**
    - ***$GLOBALS*** stores all global variables
    - ***$_SERVER*** stores information about the current server e.g. path of the script, server name.
    - ***$_GET*** stores the parameters that are passed via HTTP GET
    - ***$_POST*** stores the parameters that are passed via HTTP POST
    - ***$_FILES*** stores **files** that are uploaded to the server via HTTP POST
    - ***$_COOKIE*** stores the parameters of the HTTP cookie
    - ***$_SESSION*** stores information for a user in a **session**.
    - ***$_REQUEST*** *stores all data passed via GET and POST*

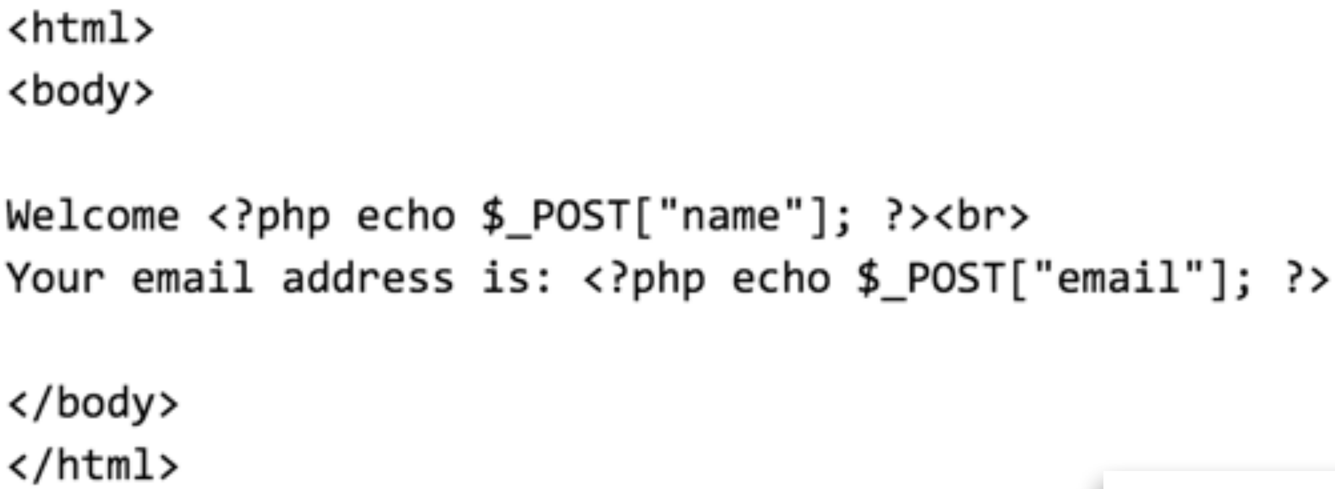
PHP: HANDLING REQUESTS

- **This is a simple HTML form:**
  - Note that it contains the parameters **Name** and **Email**.
  - These are sent to **welcome.php** via a POST request



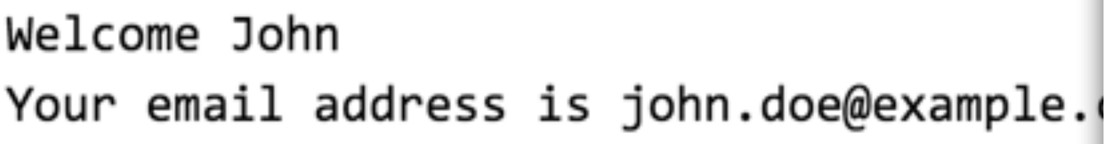
```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

- **What does welcome.php looks like?**

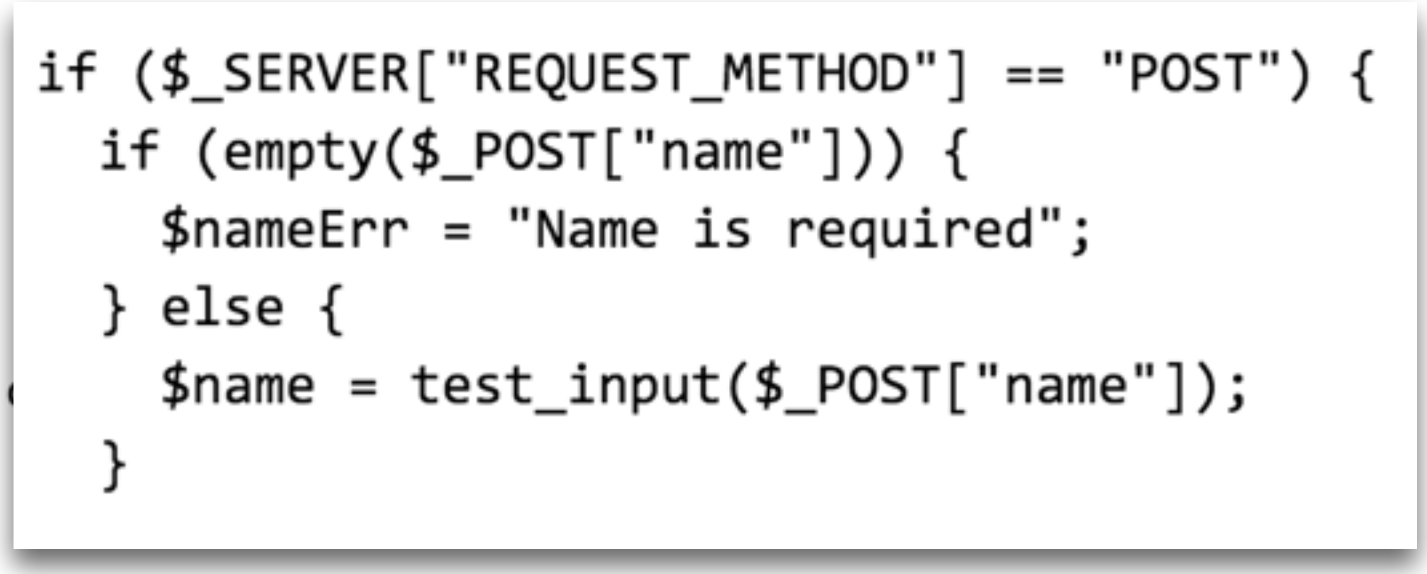


```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

- **What does the user see?**



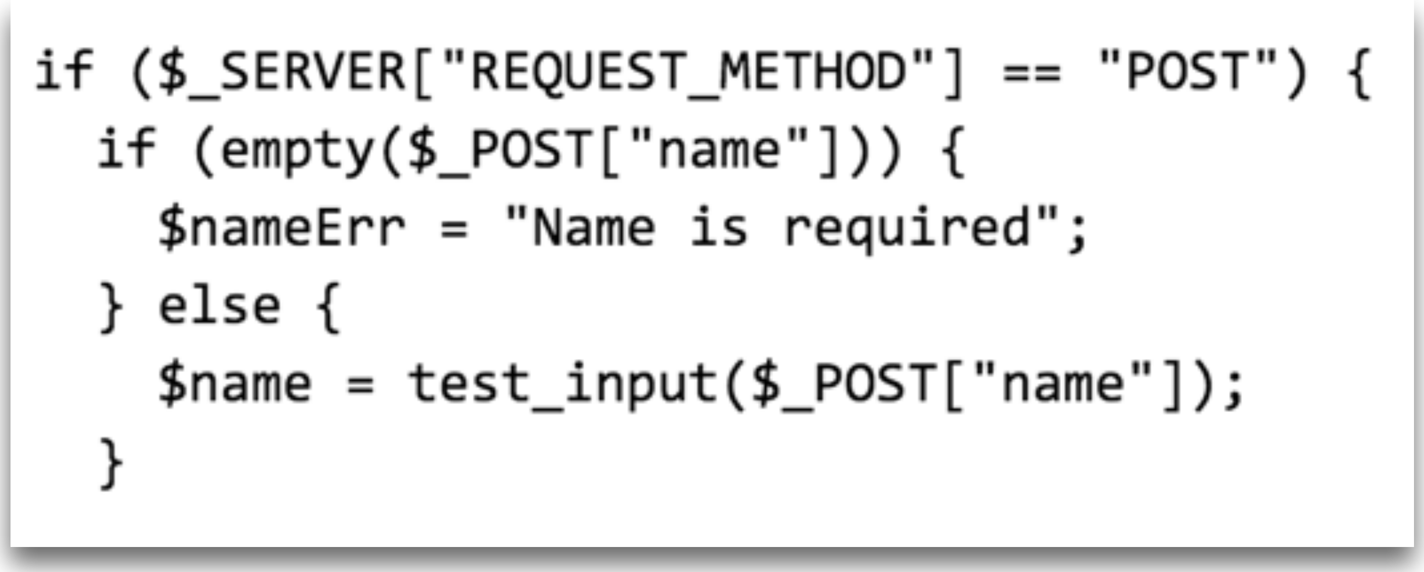
```
Welcome John
Your email address is john.doe@example.
```



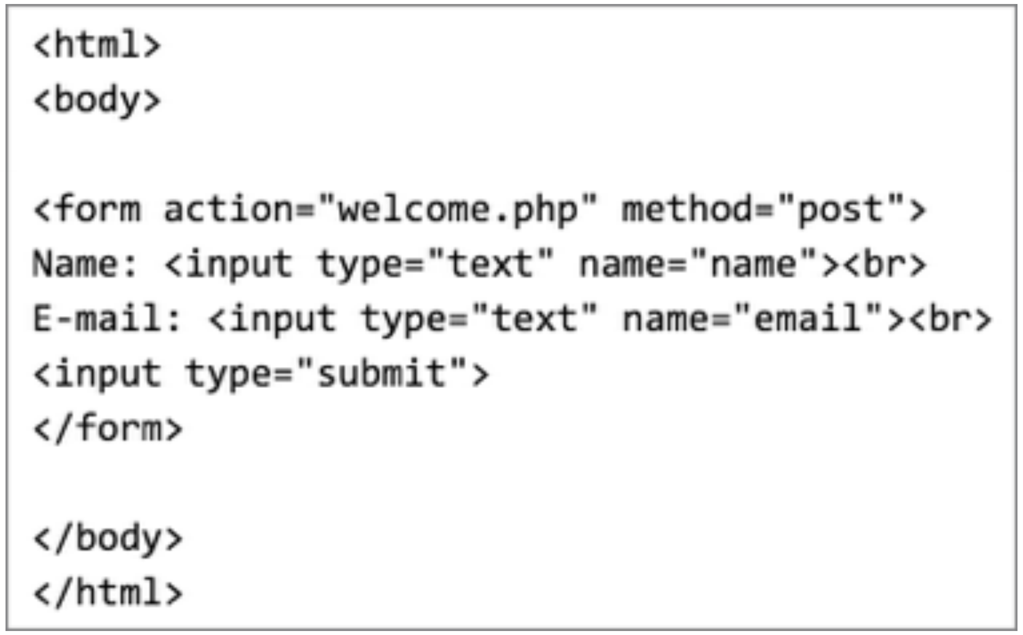
```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }
```

PHP: HANDLING REQUESTS

- **What if some of the user didn't send a required parameter?**
  - Never trust the user, always validate input **on the server side!**
- **Like that:**



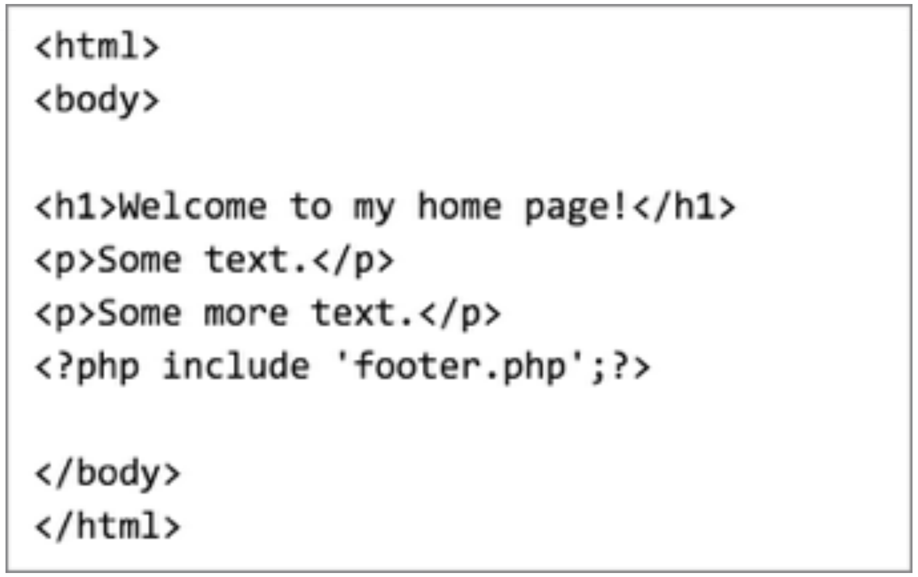
```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }
```



```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

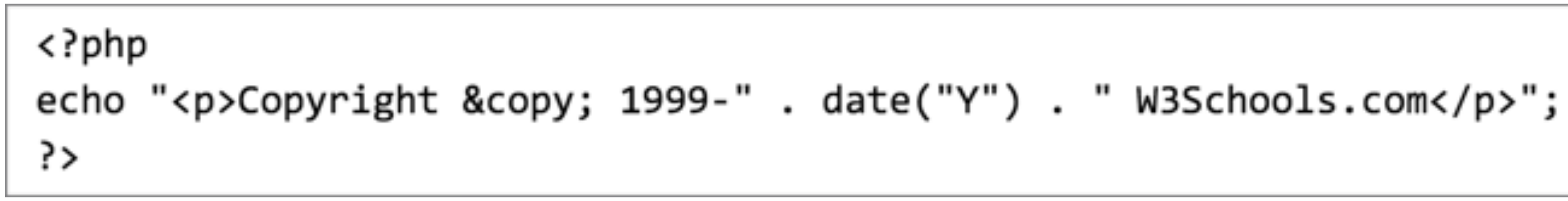
PHP: INCLUDE FILES

- **PHP allows the programmer to include (or require) other scripts.**



```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

- **footer.php can be**



```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . " W3Schools.com</p>";
?>
```

- **When to use:**

  ★Separate DB handling from the logics.

  ★Separate HTML generation from the logic.

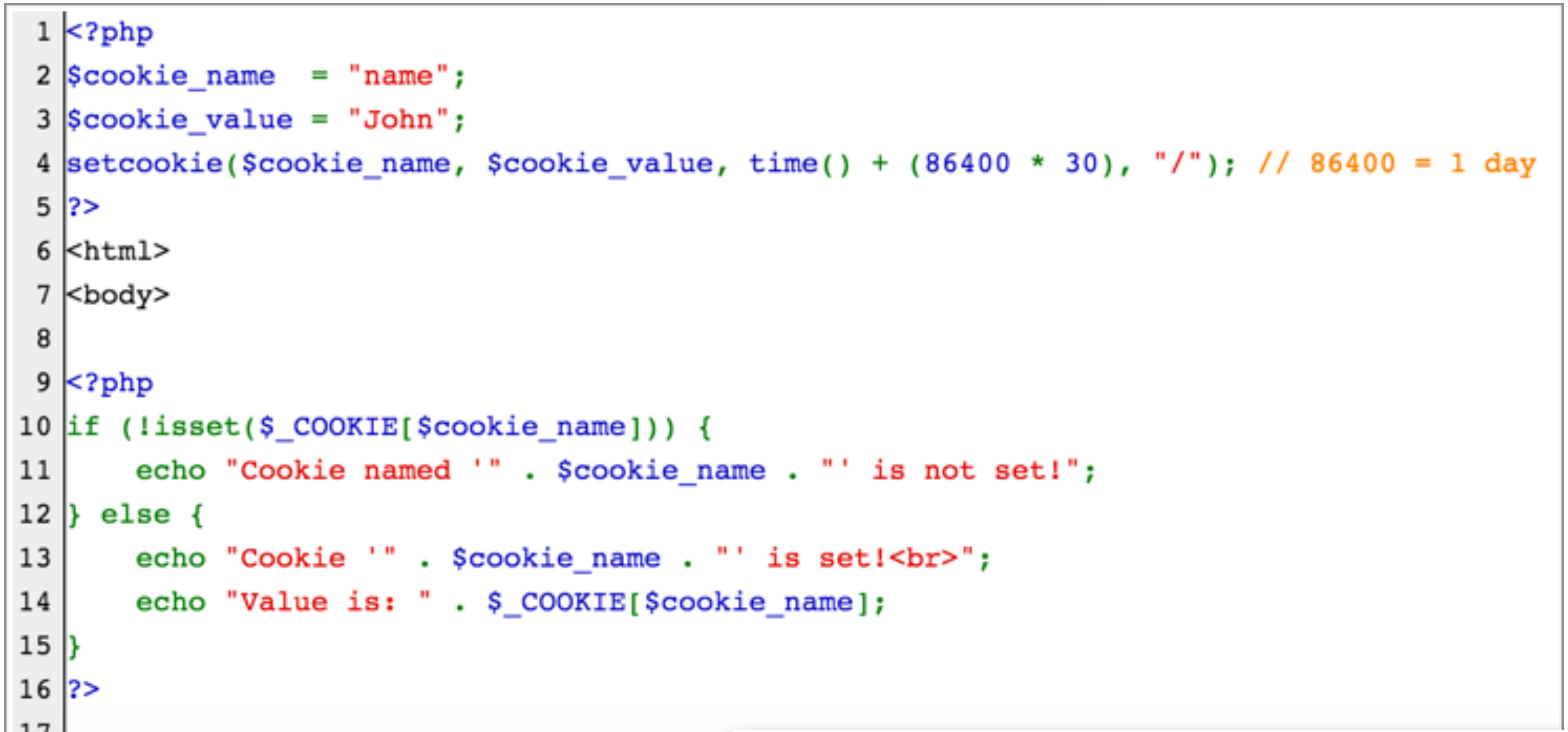
  ★Dedicated files for methods and functions

PHP: SESSIONS & COOKIES

- **HTTP Cookie (Browser cookie)** is a small piece of data stored in the web browser
  - ★Useful since the internet (and HTTP) (and PHP) are **STATELESS** .
  - ★Example use: Your shopping cart in **amazon.com**.
  - ★Cookies are saved per web page.
  - ★The browser will send the cookie content to the web-page if it contains one.
- **In PHP we can read and set an HTTP cookie (we can not “modify”)**
  - •**Set cookie via :**
    - •Only the ***name*** `setcookie(name, value, expire, path, domain, secure, httponly);`
    - •***path*** and ***domain*** variables set the sub-domains and pages that will include the cookie
    - •***secure*** *indicates the the cookie will be transferred in a secured (https) session only*
    - •***httponly*** *indicates that the that the cookie is not accessible from the browser*
    - •Should be done before the **<html>** tag!
  - •**Read cookie content via $_COOKIE super global**

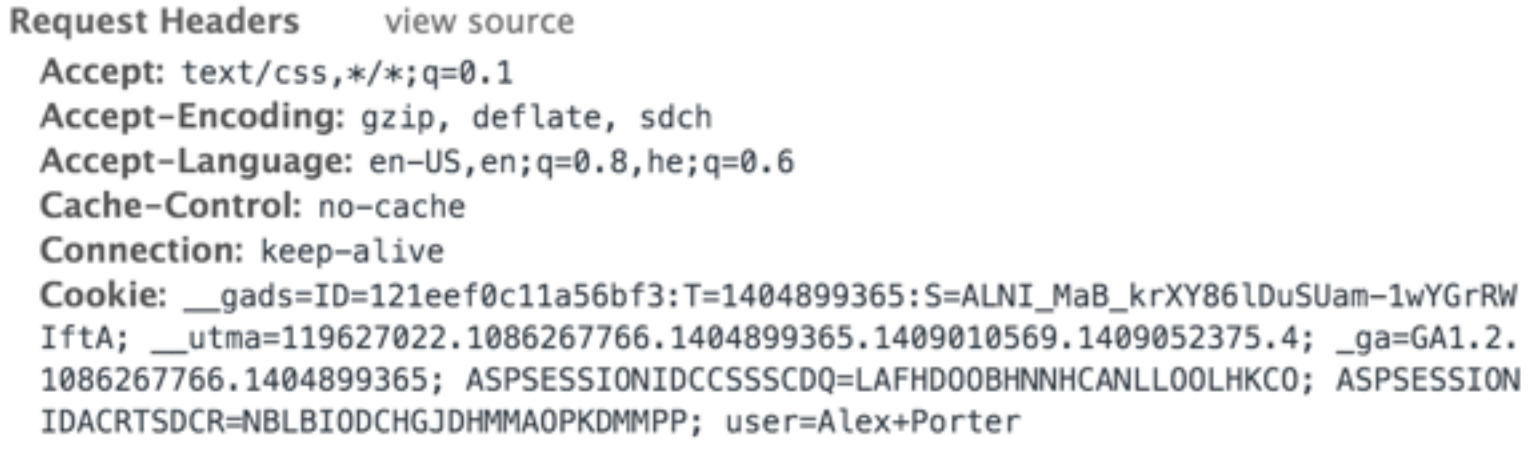
PHP: SESSIONS & COOKIES

Example:



```
<?php
$cookie_name  = "name";
$cookie_value = "John";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>
<html>
<body>

<?php
if (!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
```

The browser will sent the request :



```
Request Headers      view source
  Accept: text/css,*/*;q=0.1
  Accept-Encoding: gzip, deflate, sdch
  Accept-Language: en-US,en;q=0.8,he;q=0.6
  Cache-Control: no-cache
  Connection: keep-alive
  Cookie: __gads=ID=121eef0c11a56bf3:T=1404899365:S=ALNI_MaB_krXY86lDuSUam-1wYGrRW
  IftA; __utma=119627022.1086267766.1404899365.1409010569.1409052375.4; _ga=GA1.2.
  1086267766.1404899365; ASPSESSIONIDCCSSSCDQ=LAFHDOOBHNNHCANLLOOLHKCO; ASPSESSION
  IDACRTSDCR=NBLBIODCHGJDHMMAOPKDMMPP; user=Alex+Porter
```

PHP: SESSIONS & COOKIES

- **PHP Sessions: are a way to store information (in variables) to be used across multiple pages.**
- **Unlike a cookie, the information is not stored in the browser but in the server!**
- **How to:**

1. **Session_start()** , the first thing on the page **even if the session is not new!**
2. Read and write to **$_SESSION** super global
3. **session_unset()** will delete all session variables
4. **session_destroy()** will destroy the session

PHP: SESSIONS & COOKIES

How is a PHP session created?

- PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.

- A cookie called PHPSESSID is automatically sent to the user's computer to store unique session identification string.

- A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_ ie sess_3c7foj34c3jj973hjkop2fc937e3443.

How does PHP retrieves session information?

- PHP automatically gets the unique session identifier string from the PHPSESSID cookie.

- then looks in its temporary directory for the file bearing that name and a validation can be done by comparing both values.

How do sessions end?

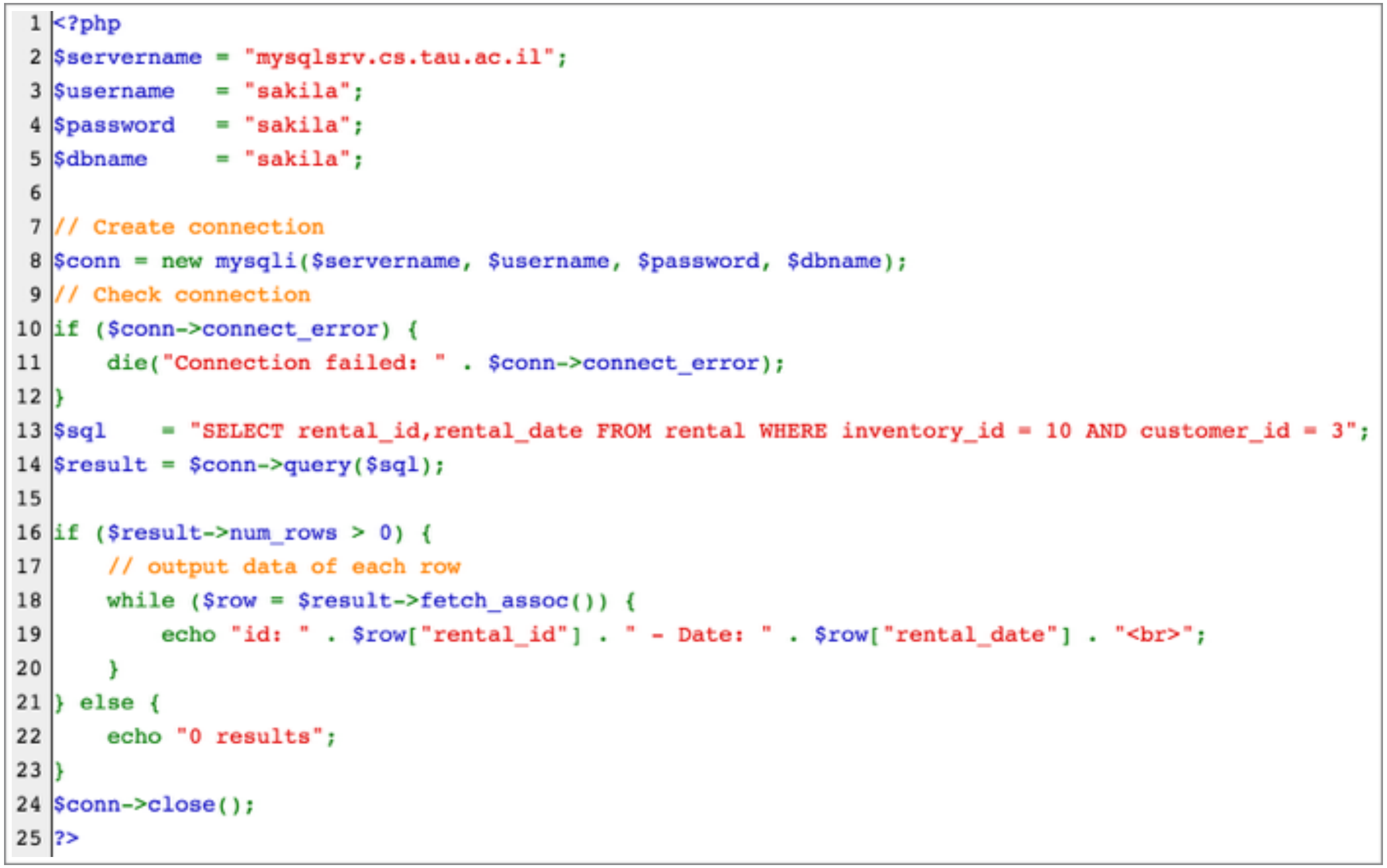
- When the cookie is lost.

- the server will terminate the session after a predetermined period of time, commonly 30 minutes duration.

PHP + MYSQL

PHP can use one of two methods for Database handling:

- MySQLi extension (the "i" stands for improved)
- PDO (PHP Data Objects)

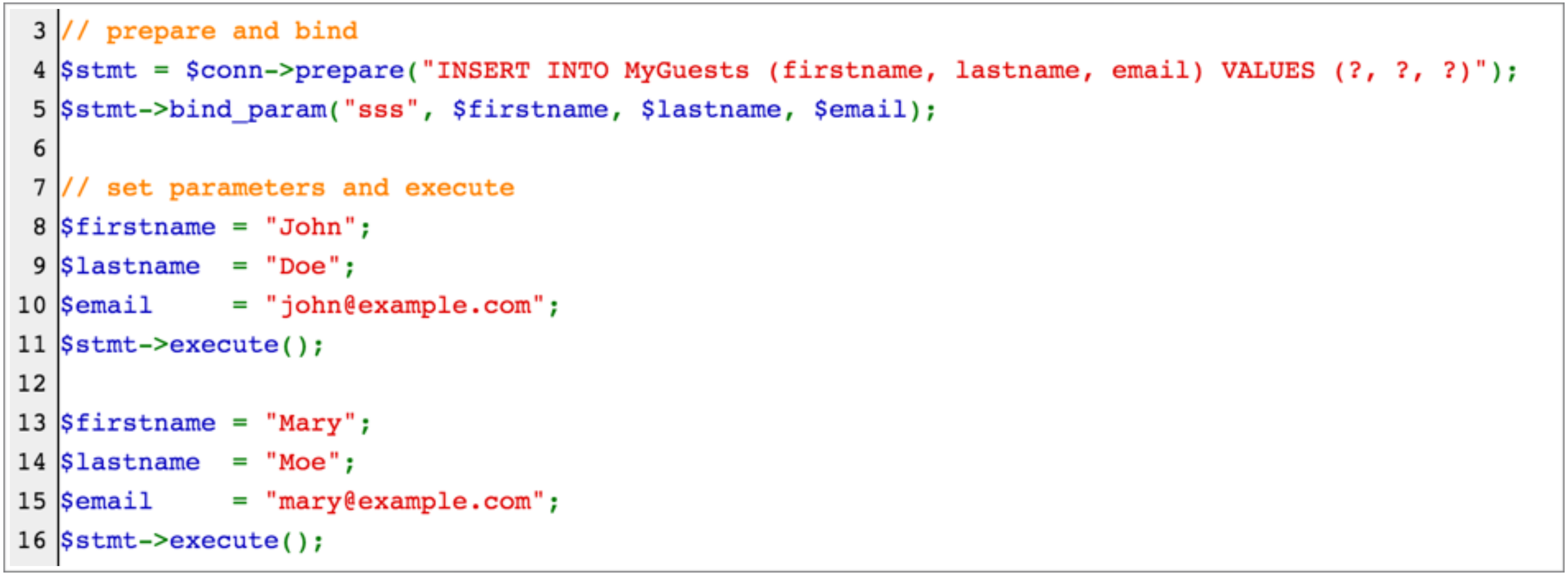


```php
<?php
$servername = "mysqlsrv.cs.tau.ac.il";
$username   = "sakila";
$password   = "sakila";
$dbname     = "sakila";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql     = "SELECT rental_id,rental_date FROM rental WHERE inventory_id = 10 AND customer_id = 3";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while ($row = $result->fetch_assoc()) {
        echo "id: " . $row["rental_id"] . " - Date: " . $row["rental_date"] . "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

PHP + MYSQL

Prepared statements using:

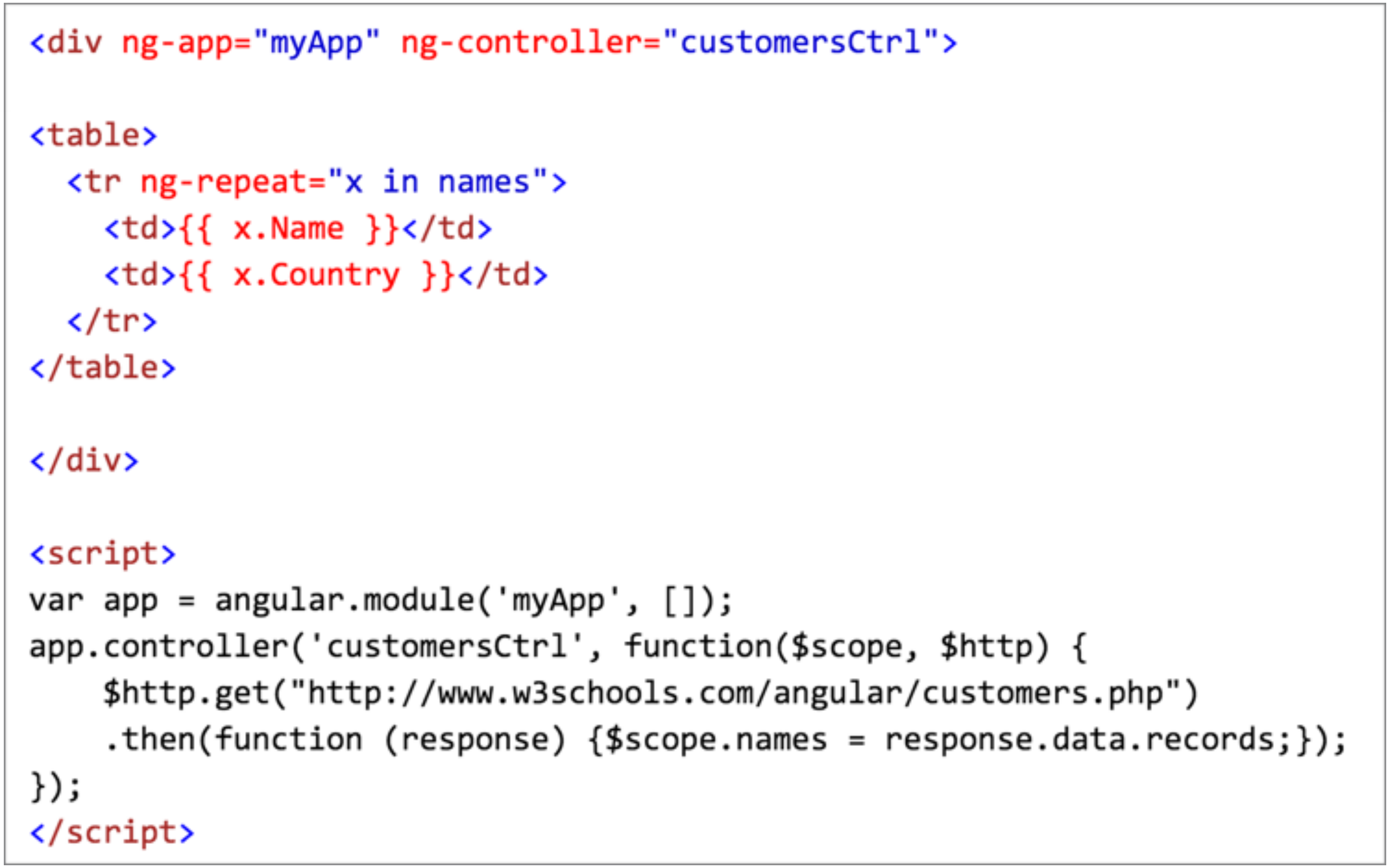
- **prepare**
- **bind**
- **execute**



```php
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname  = "Doe";
$email     = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname  = "Moe";
$email     = "mary@example.com";
$stmt->execute();
```

PHP SERVES JSON

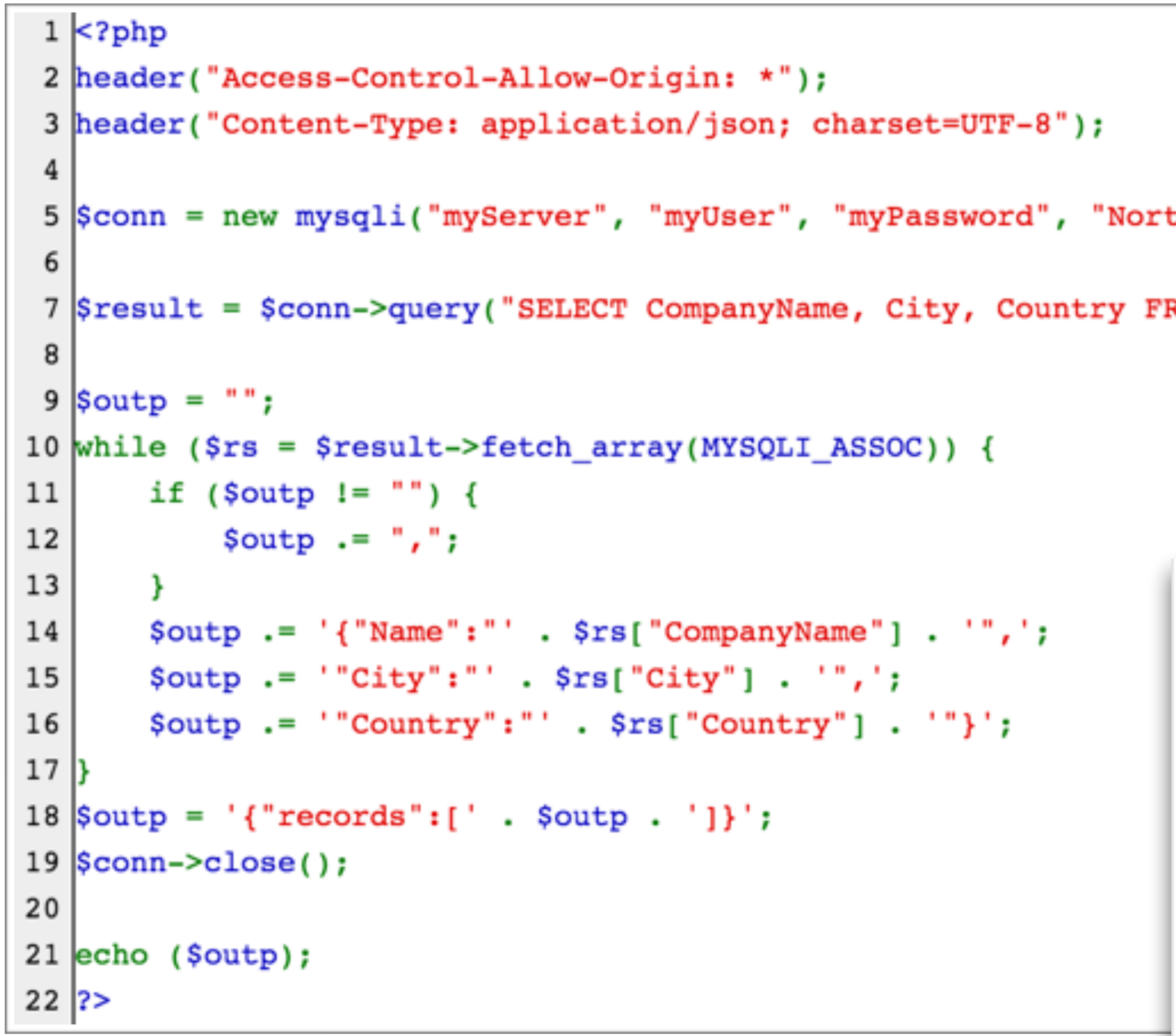
Remember this?



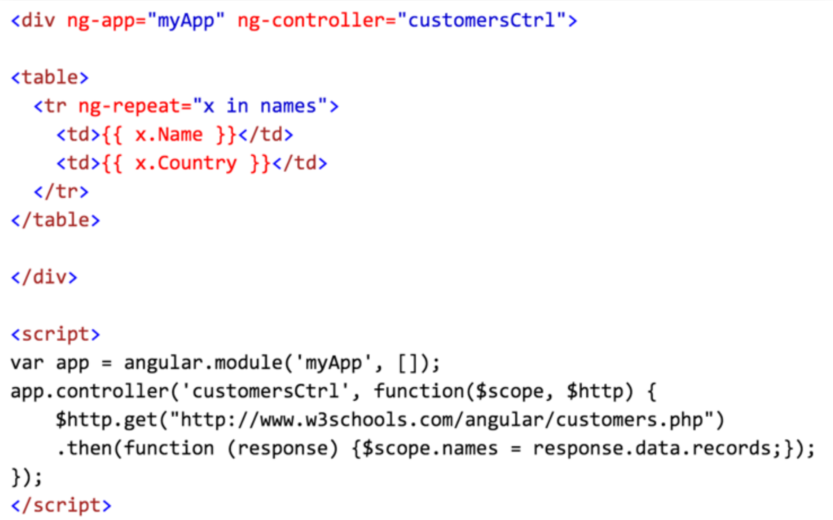
```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("http://www.w3schools.com/angular/customers.php")
    .then(function (response) {$scope.names = response.data.records;});
});
</script>
```

PHP SERVES JSON

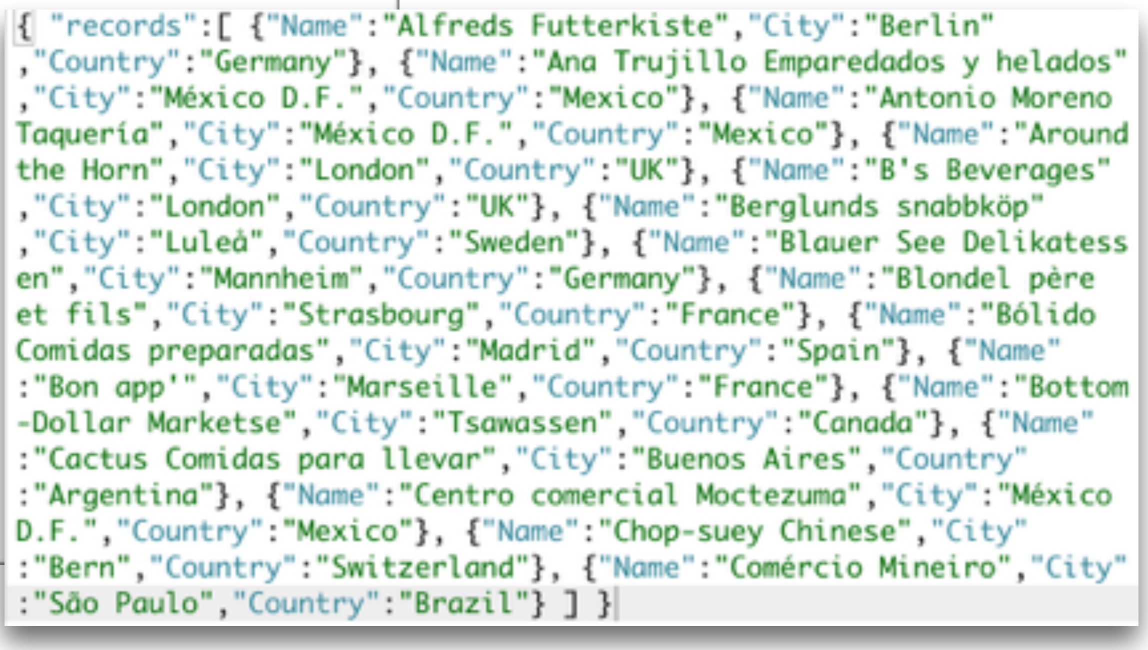
- Now we can see what happens in *customers.php*



```php
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

$conn = new mysqli("myServer", "myUser", "myPassword", "Northwind");

$result = $conn->query("SELECT CompanyName, City, Country FROM Customers");

$outp = "";
while ($rs = $result->fetch_array(MYSQLI_ASSOC)) {
    if ($outp != "") {
        $outp .= ",";
    }
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'   . $rs["City"]        . '",';
    $outp .= '"Country":"'. $rs["Country"]     . '"}';
}
$outp ='{"records":['.$outp.']}';
$conn->close();

echo ($outp);
?>
```



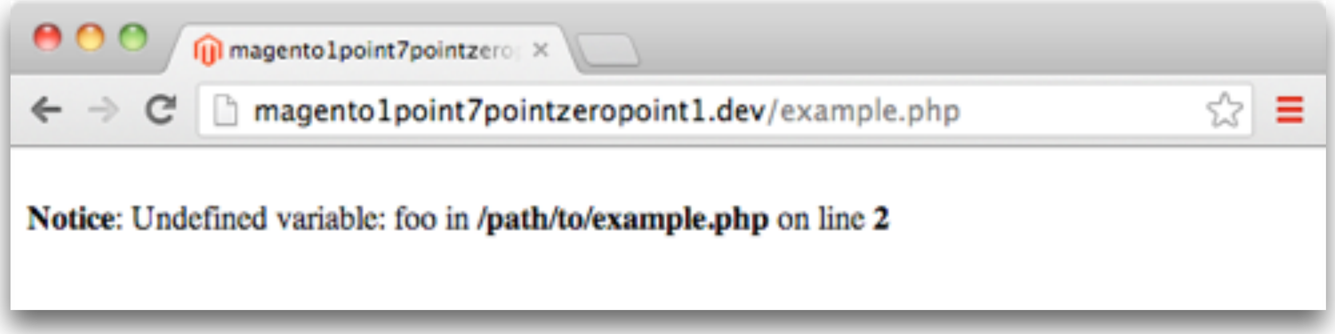
```html
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("http://www.w3schools.com/angular/customers.php")
    .then(function (response) {$scope.names = response.data.records;});
});
</script>
```



```
{ "records":[ {"Name":"Alfreds Futterkiste","City":"Berlin"
,"Country":"Germany"}, {"Name":"Ana Trujillo Emparedados y helados"
,"City":"México D.F.","Country":"Mexico"}, {"Name":"Antonio Moreno
Taquería","City":"México D.F.","Country":"Mexico"}, {"Name":"Around
the Horn","City":"London","Country":"UK"}, {"Name":"B's Beverages"
,"City":"London","Country":"UK"}, {"Name":"Berglunds snabbköp"
,"City":"Luleå","Country":"Sweden"}, {"Name":"Blauer See Delikatess
en","City":"Mannheim","Country":"Germany"}, {"Name":"Blondel père
et fils","City":"Strasbourg","Country":"France"}, {"Name":"Bólido
Comidas preparadas","City":"Madrid","Country":"Spain"}, {"Name"
:"Bon app'","City":"Marseille","Country":"France"}, {"Name":"Bottom
-Dollar Marketse","City":"Tsawassen","Country":"Canada"}, {"Name"
:"Cactus Comidas para llevar","City":"Buenos Aires","Country"
:"Argentina"}, {"Name":"Centro comercial Moctezuma","City":"México
D.F.","Country":"Mexico"}, {"Name":"Chop-suey Chinese","City"
:"Bern","Country":"Switzerland"}, {"Name":"Comércio Mineiro","City"
:"São Paulo","Country":"Brazil"} ] }
```

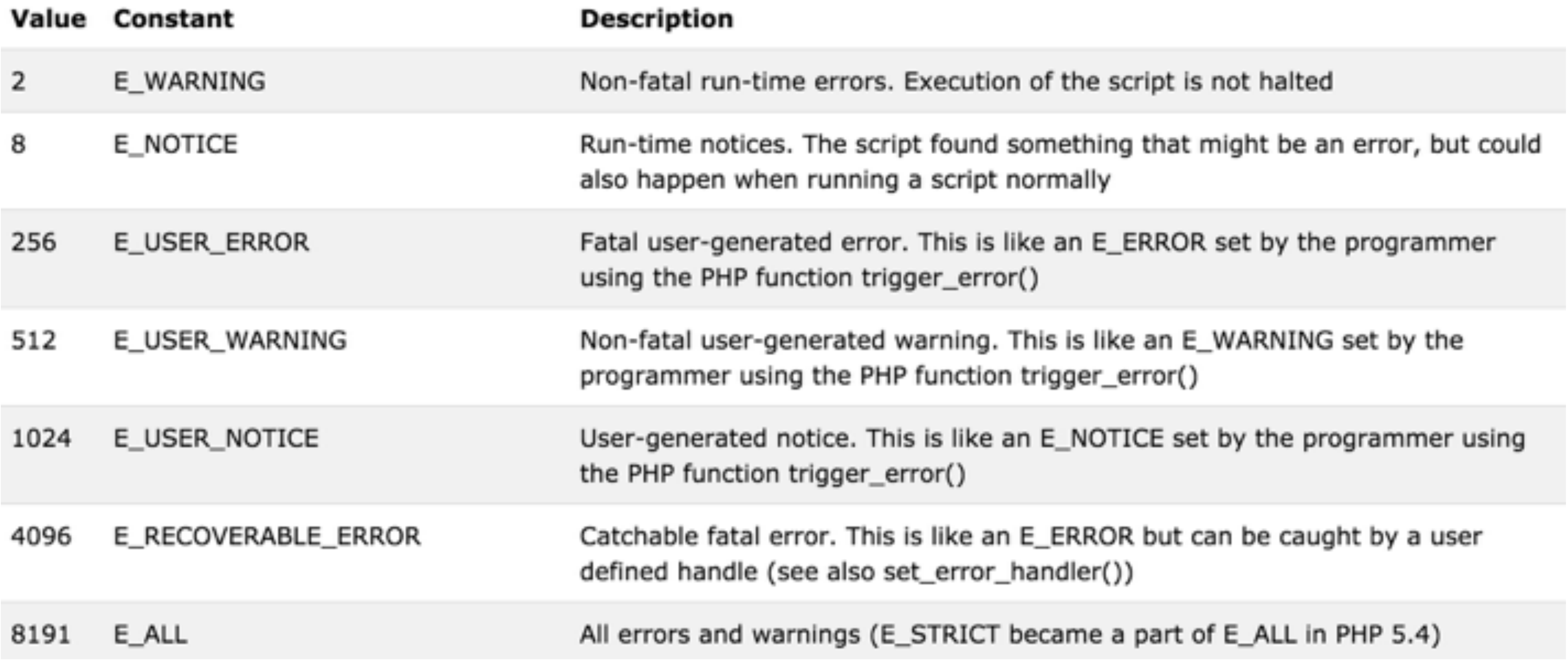
PHP ERROR HANDLING

- PHP stores all error and warning to a log.
- Depends on the configuration, it also prints annoying messages to the screen such as :



```
Notice: Undefined variable: foo in /path/to/example.php on line 2
```

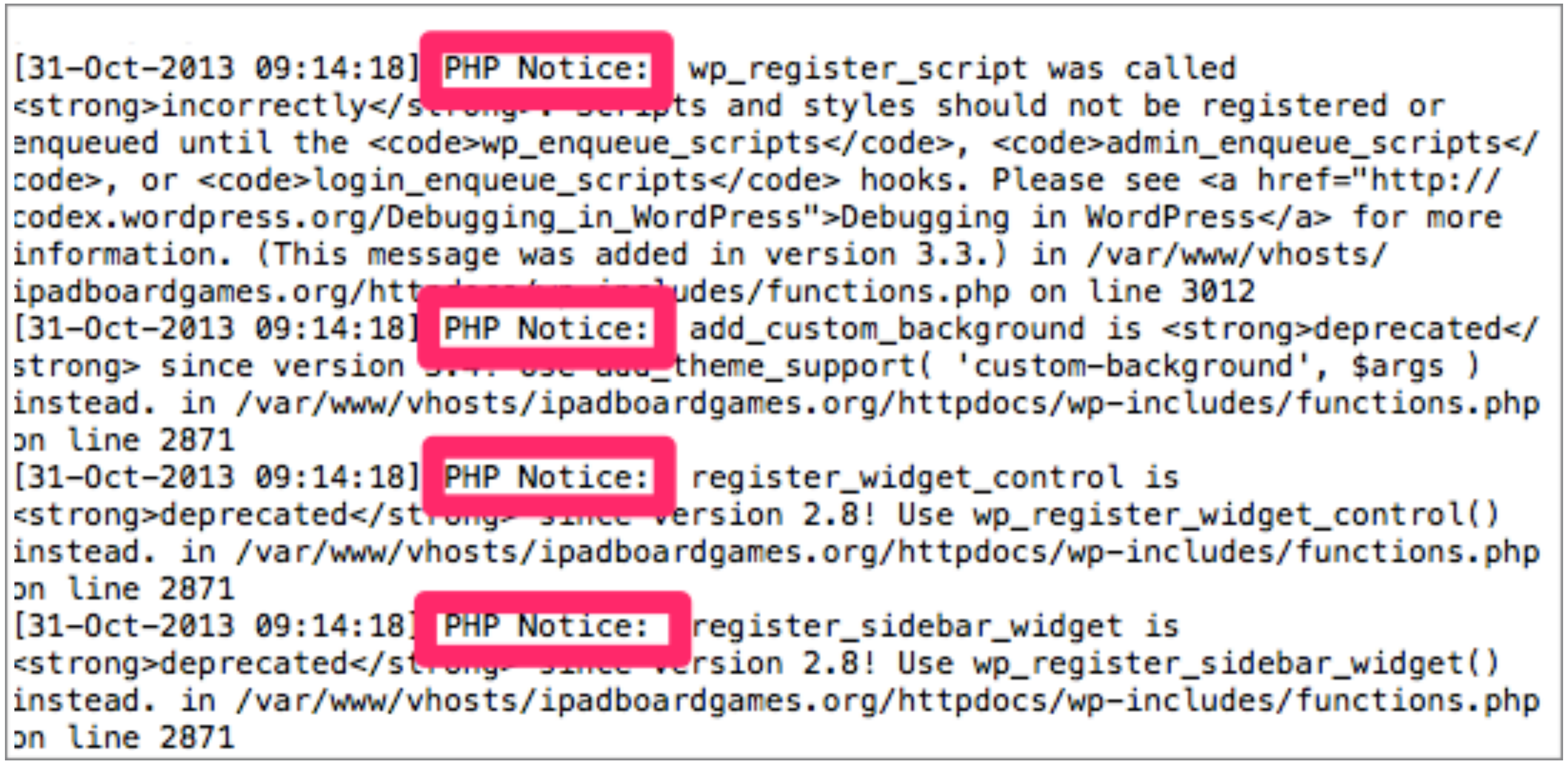
- Theses are the available error levels: use error_reporting() to control it:



Value	Constant	Description
2	E_WARNING	Non-fatal run-time errors. Execution of the script is not halted
8	E_NOTICE	Run-time notices. The script found something that might be an error, but could also happen when running a script normally
256	E_USER_ERROR	Fatal user-generated error. This is like an E_ERROR set by the programmer using the PHP function trigger_error()
512	E_USER_WARNING	Non-fatal user-generated warning. This is like an E_WARNING set by the programmer using the PHP function trigger_error()
1024	E_USER_NOTICE	User-generated notice. This is like an E_NOTICE set by the programmer using the PHP function trigger_error()
4096	E_RECOVERABLE_ERROR	Catchable fatal error. This is like an E_ERROR but can be caught by a user defined handle (see also set_error_handler())
8191	E_ALL	All errors and warnings (E_STRICT became a part of E_ALL in PHP 5.4)

APACHE+PHP ERROR LOGS

- **Are store in a file called *error.log*** that can be found in the apache directory



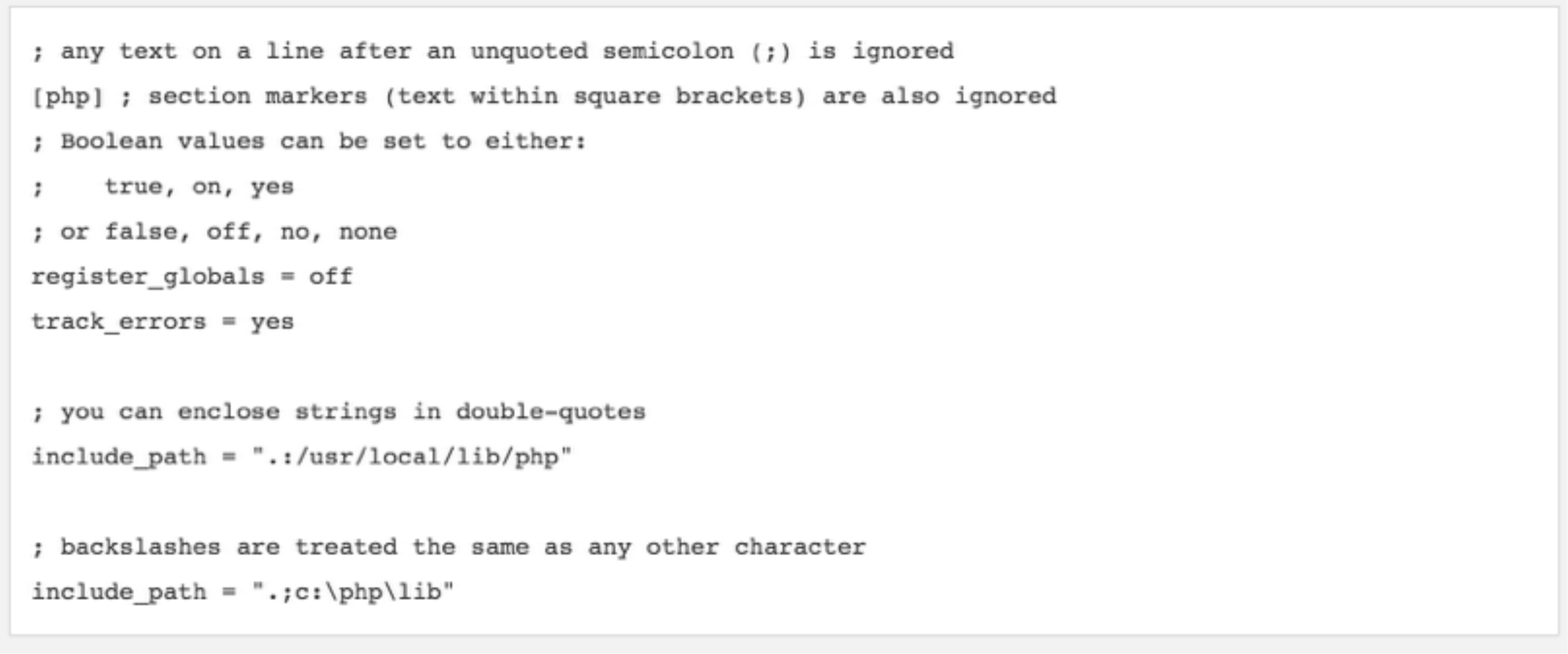
```
[31-Oct-2013 09:14:18] PHP Notice:  wp_register_script was called
<strong>incorrectly</strong>. Scripts and styles should not be registered or
enqueued until the <code>wp_enqueue_scripts</code>, <code>admin_enqueue_scripts</
code>, or <code>login_enqueue_scripts</code> hooks. Please see <a href="http://
codex.wordpress.org/Debugging_in_WordPress">Debugging in WordPress</a> for more
information. (This message was added in version 3.3.) in /var/www/vhosts/
ipadboardgames.org/httpdocs/wp-includes/functions.php on line 3012
[31-Oct-2013 09:14:18] PHP Notice:  add_custom_background is <strong>deprecated</
strong> since version 3.4! Use add_theme_support( 'custom-background', $args )
instead. in /var/www/vhosts/ipadboardgames.org/httpdocs/wp-includes/functions.php
on line 2871
[31-Oct-2013 09:14:18] PHP Notice:  register_widget_control is
<strong>deprecated</strong> since version 2.8! Use wp_register_widget_control()
instead. in /var/www/vhosts/ipadboardgames.org/httpdocs/wp-includes/functions.php
on line 2871
[31-Oct-2013 09:14:18] PHP Notice:  register_sidebar_widget is
<strong>deprecated</strong> since version 2.8! Use wp_register_sidebar_widget()
instead. in /var/www/vhosts/ipadboardgames.org/httpdocs/wp-includes/functions.php
on line 2871
```

APACHE+PHP CONF. FILES

Apache configurations are stored in a file called httpd.conf

- To make changes: You can make changes in the httpd.conf file then restart the server
- Holds informations such as the server port, supported modules etc.

PHP configurations are stored in a file called PHP.ini



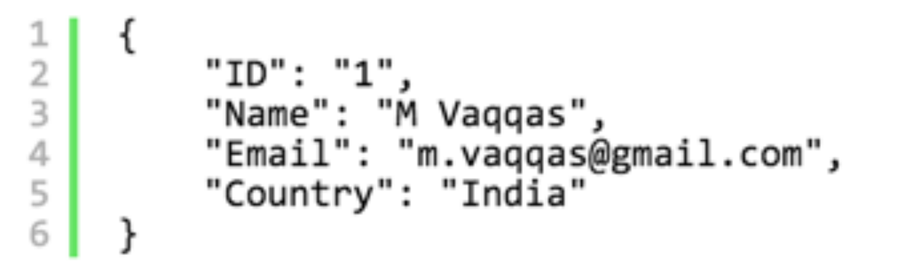
```
; any text on a line after an unquoted semicolon (;) is ignored
[php] ; section markers (text within square brackets) are also ignored
; Boolean values can be set to either:
;    true, on, yes
; or false, off, no, none
register_globals = off
track_errors = yes

; you can enclose strings in double-quotes
include_path = ".:/usr/local/lib/php"

; backslashes are treated the same as any other character
include_path = ".;c:\php\lib"
```

AGENDA FOR TODAY

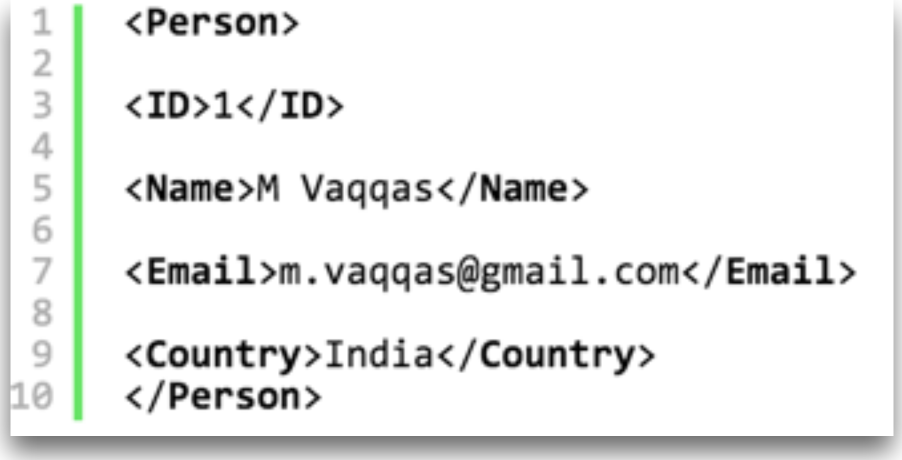
- **Client side programming**
  - HTML
  - CSS
  - Javascript
  - Additional libraries: Bootstrap,Angular, Jquery
- **Server side programming: PHP**
  - Install XAMPP
  - Web server architecture
  - php+mysql
- **Web APIs: REST ,Json, and how to get them via Python**

WEB SERVICES

- A web service is like a website but is ***structured***.
- It is for programs, not for humans.
- RESTful: REpresentational State Transter (ful)
- REST APIs have the following characteristics:
    - •**Representations:** which are the objects like in OOP
    - •**Messages:** the client and the servers are sending messages to each other
    - •**Stateless:** Like the internet. REST is stateless.
    - •**Links between resources:** Same as in URI and URLs.
- The response message will be in JSON or XML



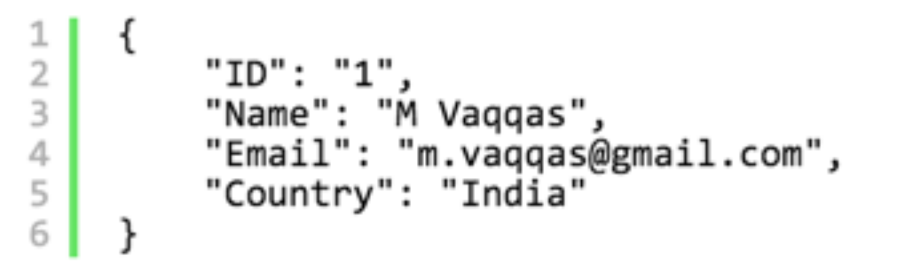
```
{
        "ID": "1",
        "Name": "M Vaqqas",
        "Email": "m.vaqqas@gmail.com",
        "Country": "India"
}
```



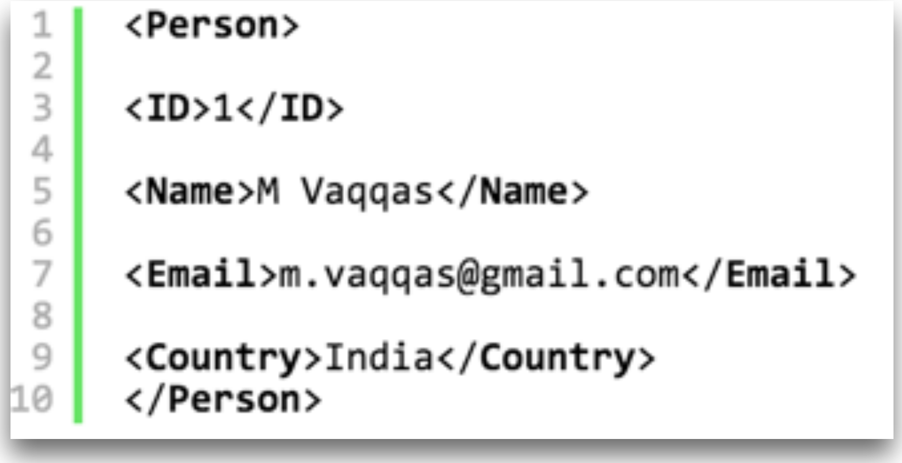
```
<Person>

<ID>1</ID>

<Name>M Vaqqas</Name>

<Email>m.vaqqas@gmail.com</Email>

<Country>India</Country>
</Person>
```

WEB SERVICES

- A web service is like a website but is ***structured***.
- It is for programs, not for humans.
- RESTful: REpresentational State Transter (ful)
- REST APIs have the following characteristics:
    - **Representations:** which are the objects like in OOP
    - **Messages:** the client and the servers are sending messages to each other
    - **Stateless:** Like the internet. REST is stateless.
    - **Links between resources:** Same as in URI and URLs.
- The response message will be in JSON or XML



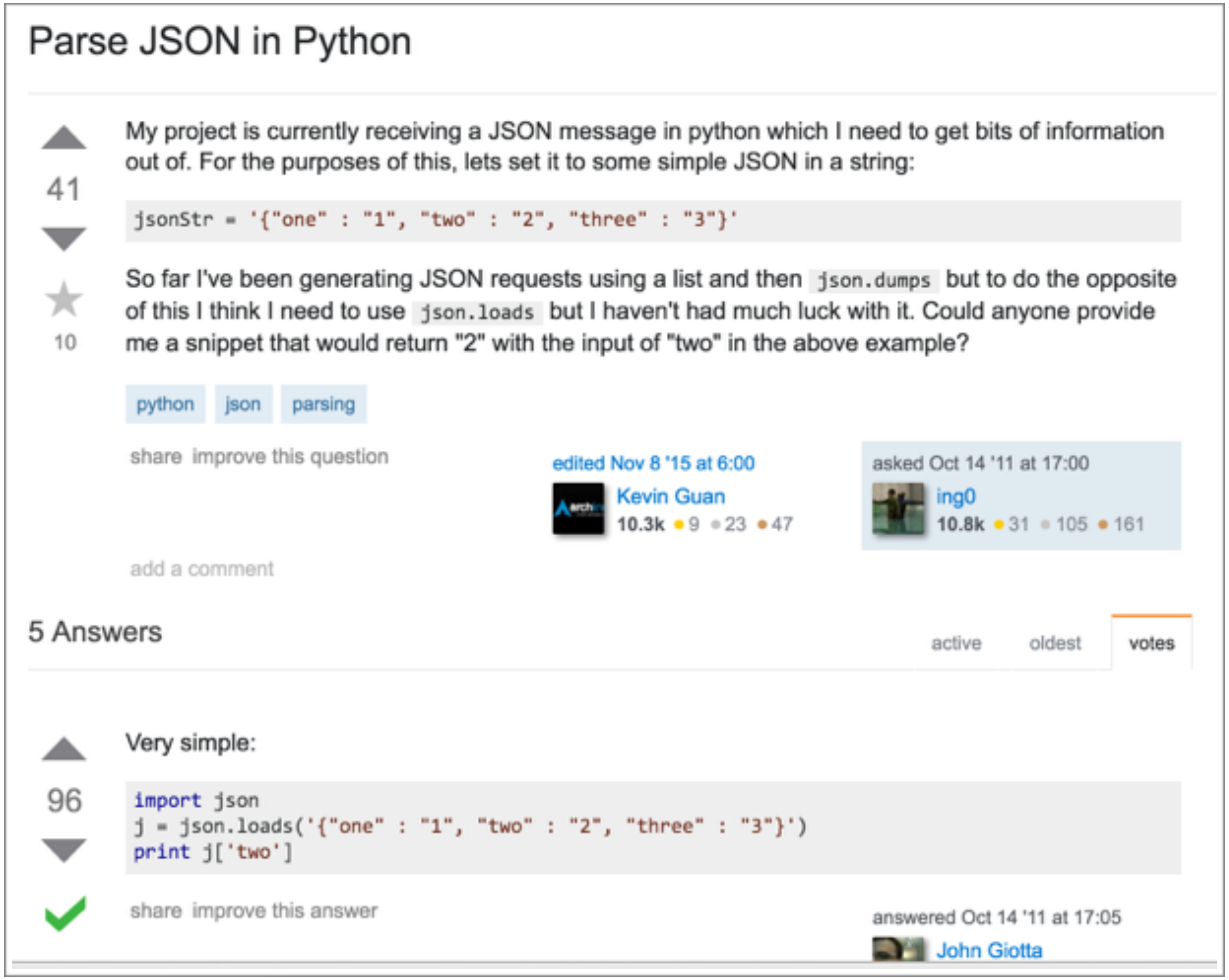
```
{
        "ID": "1",
        "Name": "M Vaqqas",
        "Email": "m.vaqqas@gmail.com",
        "Country": "India"
}
```



```
<Person>

<ID>1</ID>

<Name>M Vaqqas</Name>

<Email>m.vaqqas@gmail.com</Email>

<Country>India</Country>
</Person>
```

STACKEXCHANGE API

- Q&A platform , one of its known instances is stack overflow



Parse JSON in Python

41

10

My project is currently receiving a JSON message in python which I need to get bits of information out of. For the purposes of this, lets set it to some simple JSON in a string:

```
jsonStr = '{"one" : "1", "two" : "2", "three" : "3"}'
```

So far I've been generating JSON requests using a list and then `json.dumps` but to do the opposite of this I think I need to use `json.loads` but I haven't had much luck with it. Could anyone provide me a snippet that would return "2" with the input of "two" in the above example?

python json parsing

share improve this question

edited Nov 8 '15 at 6:00
Kevin Guan
10.3k ●9 ●23 ●47

asked Oct 14 '11 at 17:00
ing0
10.8k ●31 ●105 ●161

add a comment

5 Answers

active oldest votes

96

Very simple:

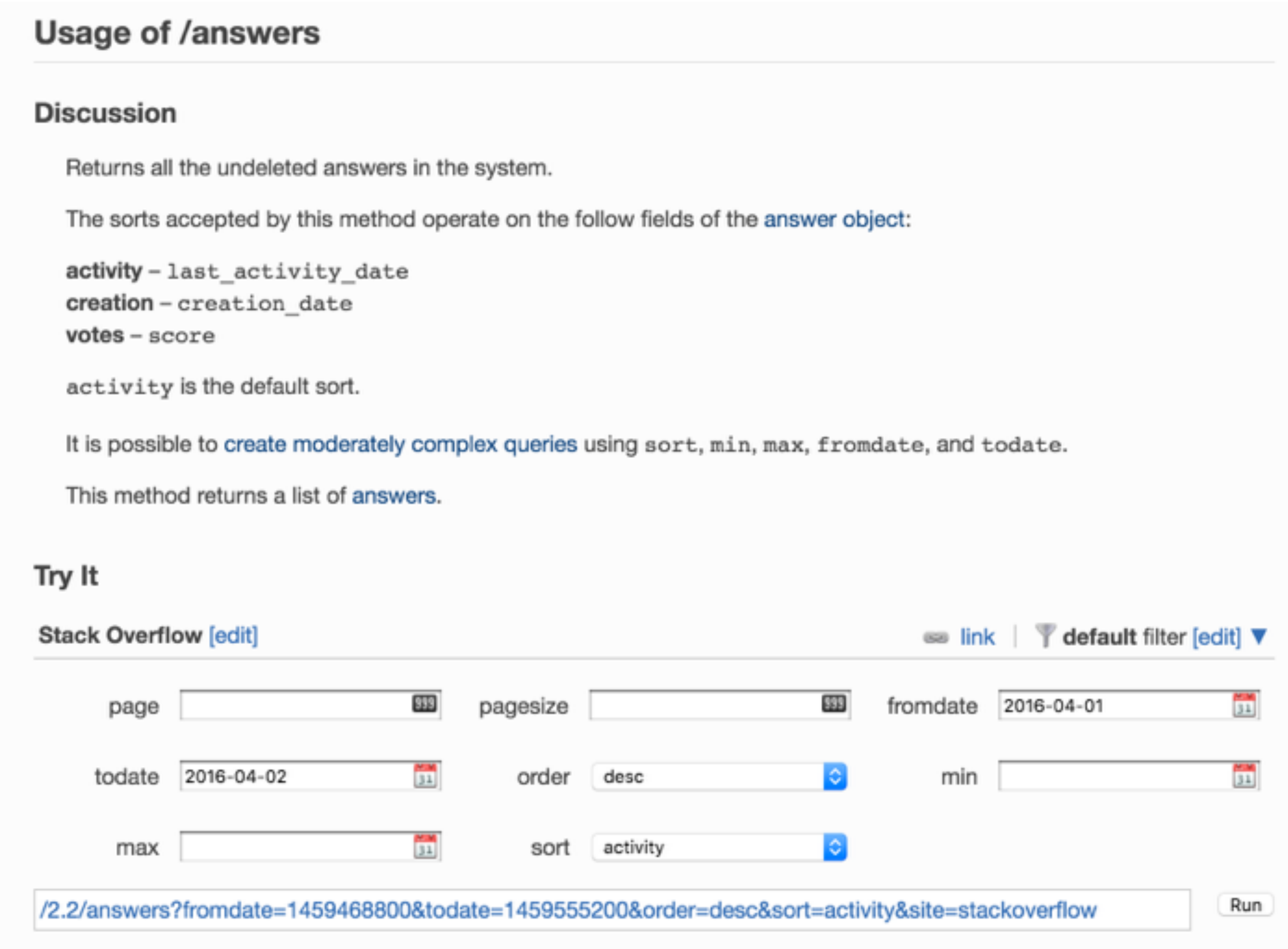
```
import json
j = json.loads('{"one" : "1", "two" : "2", "three" : "3"}')
print j['two']
```

share improve this answer

answered Oct 14 '11 at 17:05
John Giotta

STACKEXCHANGE API

- **Stack exchange API example :**



Usage of /answers

Discussion

Returns all the undeleted answers in the system.

The sorts accepted by this method operate on the follow fields of the answer object:

activity – `last_activity_date`
creation – `creation_date`
votes – `score`

`activity` is the default sort.

It is possible to create moderately complex queries using `sort`, `min`, `max`, `fromdate`, and `todate`.

This method returns a list of answers.

Try It

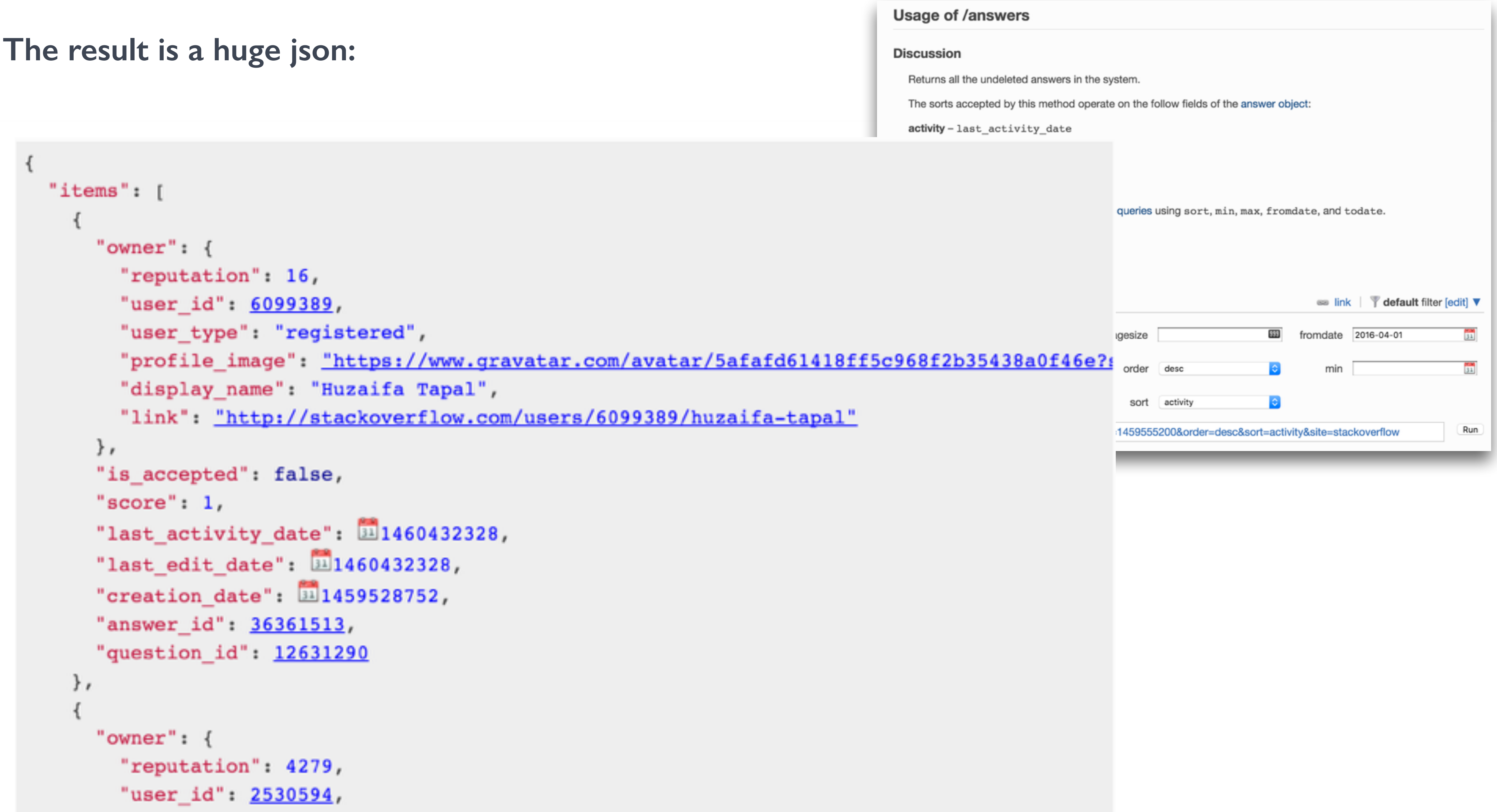
Stack Overflow [edit]

page:
pagesize:
fromdate: 2016-04-01
todate: 2016-04-02
order: desc
min:
max:
sort: activity

/2.2/answers?fromdate=1459468800&todate=1459555200&order=desc&sort=activity&site=stackoverflow

STACKEXCHANGE API

- The result is a huge json:



```
{
  "items": [
    {
      "owner": {
        "reputation": 16,
        "user_id": 6099389,
        "user_type": "registered",
        "profile_image": "https://www.gravatar.com/avatar/5afafd61418ff5c968f2b35438a0f46e?s
        "display_name": "Huzaifa Tapal",
        "link": "http://stackoverflow.com/users/6099389/huzaifa-tapal"
      },
      "is_accepted": false,
      "score": 1,
      "last_activity_date": 1460432328,
      "last_edit_date": 1460432328,
      "creation_date": 1459528752,
      "answer_id": 36361513,
      "question_id": 12631290
    },
    {
      "owner": {
        "reputation": 4279,
        "user_id": 2530594,
```

Usage of /answers

Discussion

Returns all the undeleted answers in the system.

The sorts accepted by this method operate on the follow fields of the answer object:

activity – `last_activity_date`

queries using `sort`, `min`, `max`, `fromdate`, and `todate`.

gesize:

fromdate: 2016-04-01

order: desc

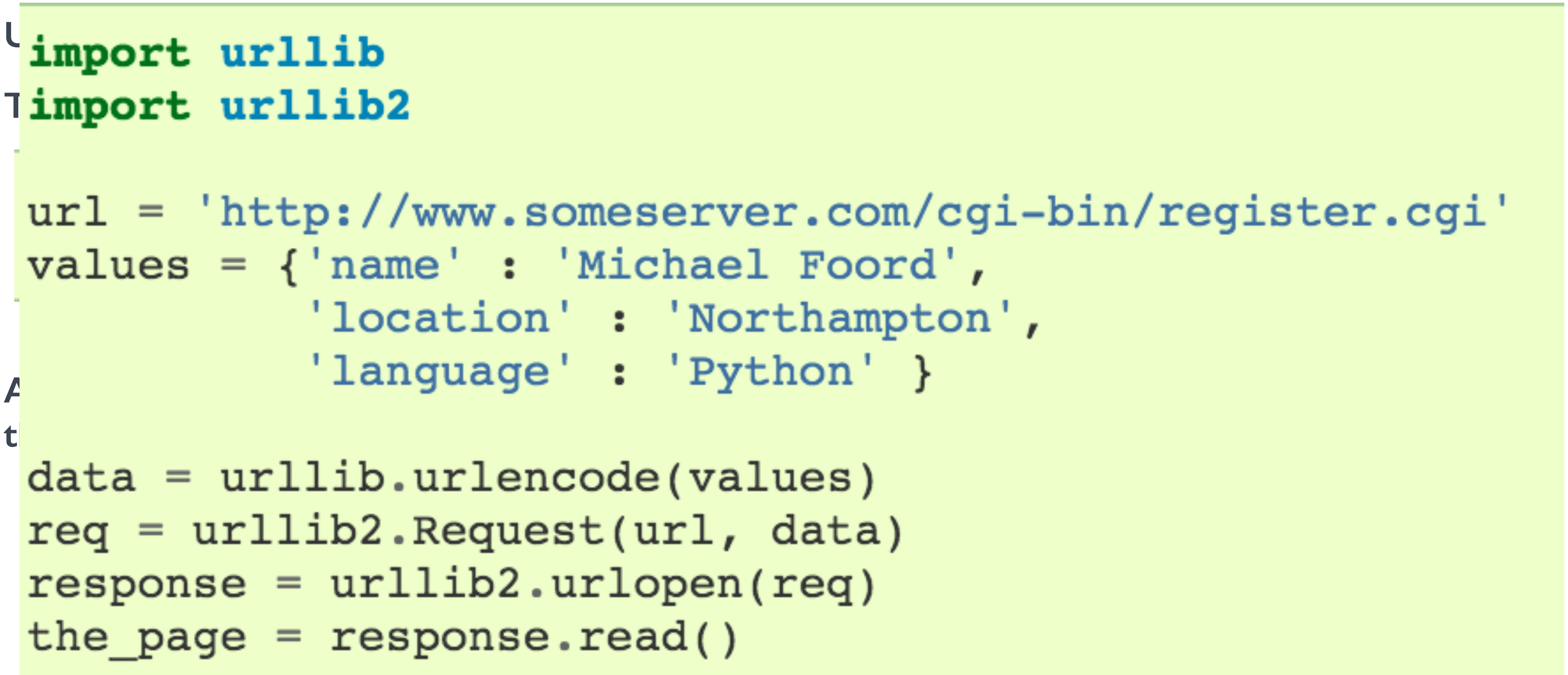
min:

sort: activity

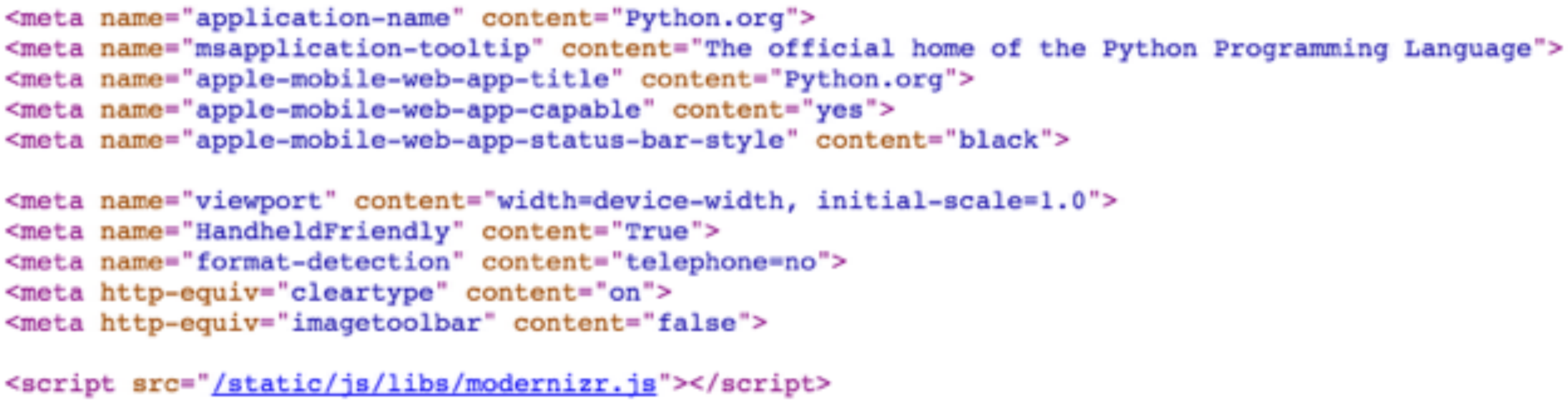
1459555200&order=desc&sort=activity&site=stackoverflow

USING PYTHON FOR WEB API

- U
- T
- A
  t



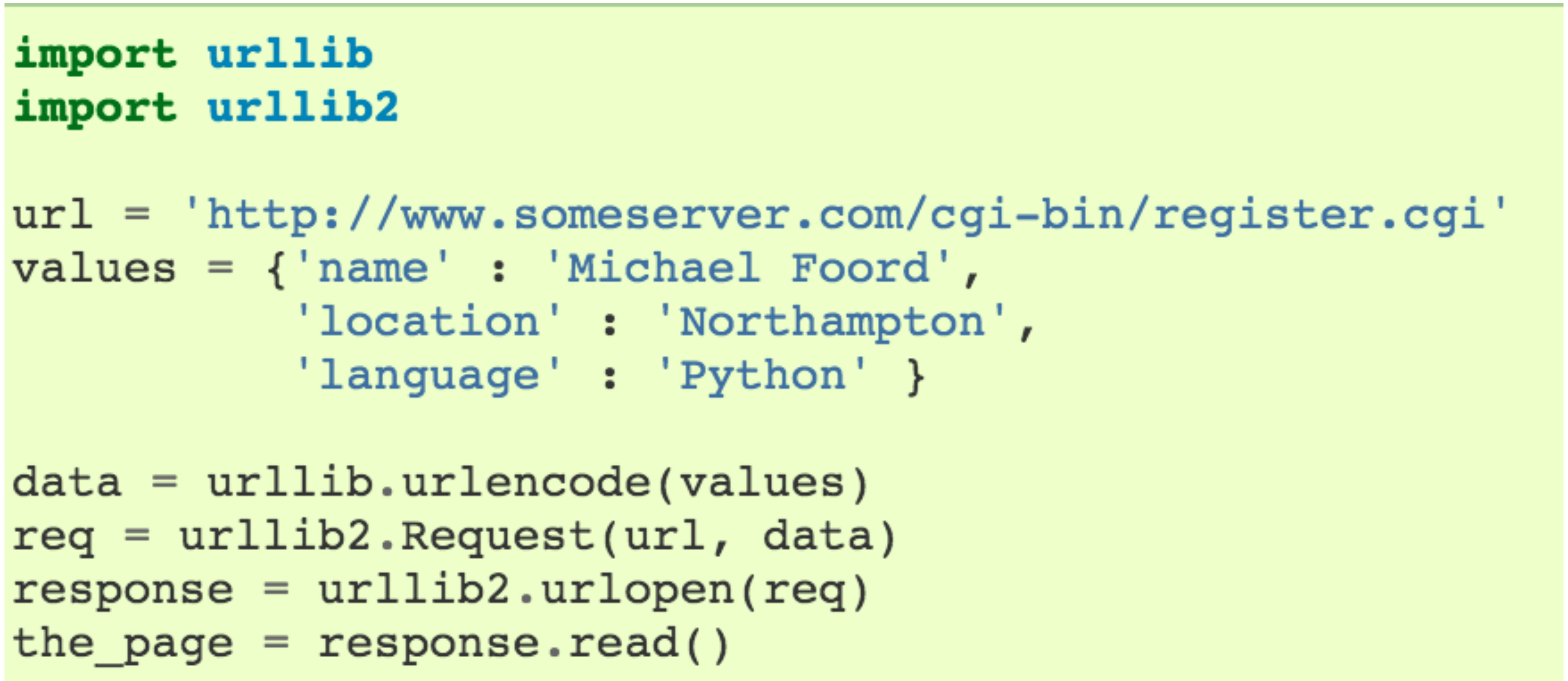
```
import urllib
import urllib2

url = 'http://www.someserver.com/cgi-bin/register.cgi'
values = {'name' : 'Michael Foord',
          'location' : 'Northampton',
          'language' : 'Python' }

data = urllib.urlencode(values)
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
the_page = response.read()
```



```
<meta name="application-name" content="Python.org">
<meta name="msapplication-tooltip" content="The official home of the Python Programming Language">
<meta name="apple-mobile-web-app-title" content="Python.org">
<meta name="apple-mobile-web-app-capable" content="yes">
<meta name="apple-mobile-web-app-status-bar-style" content="black">

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="HandheldFriendly" content="True">
<meta name="format-detection" content="telephone=no">
<meta http-equiv="cleartype" content="on">
<meta http-equiv="imagetoolbar" content="false">

<script src="/static/js/libs/modernizr.js"></script>
```

USING PYTHON FOR WEB API

- Using a “request” object, you can generate a post request:
  - Create a dictionaries with variables and values
  - Create a new Request object and load it with the URL and the dict.
  - Execute the request via urlopen



```
import urllib
import urllib2

url = 'http://www.someserver.com/cgi-bin/register.cgi'
values = {'name' : 'Michael Foord',
          'location' : 'Northampton',
          'language' : 'Python' }

data = urllib.urlencode(values)
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
the_page = response.read()
```

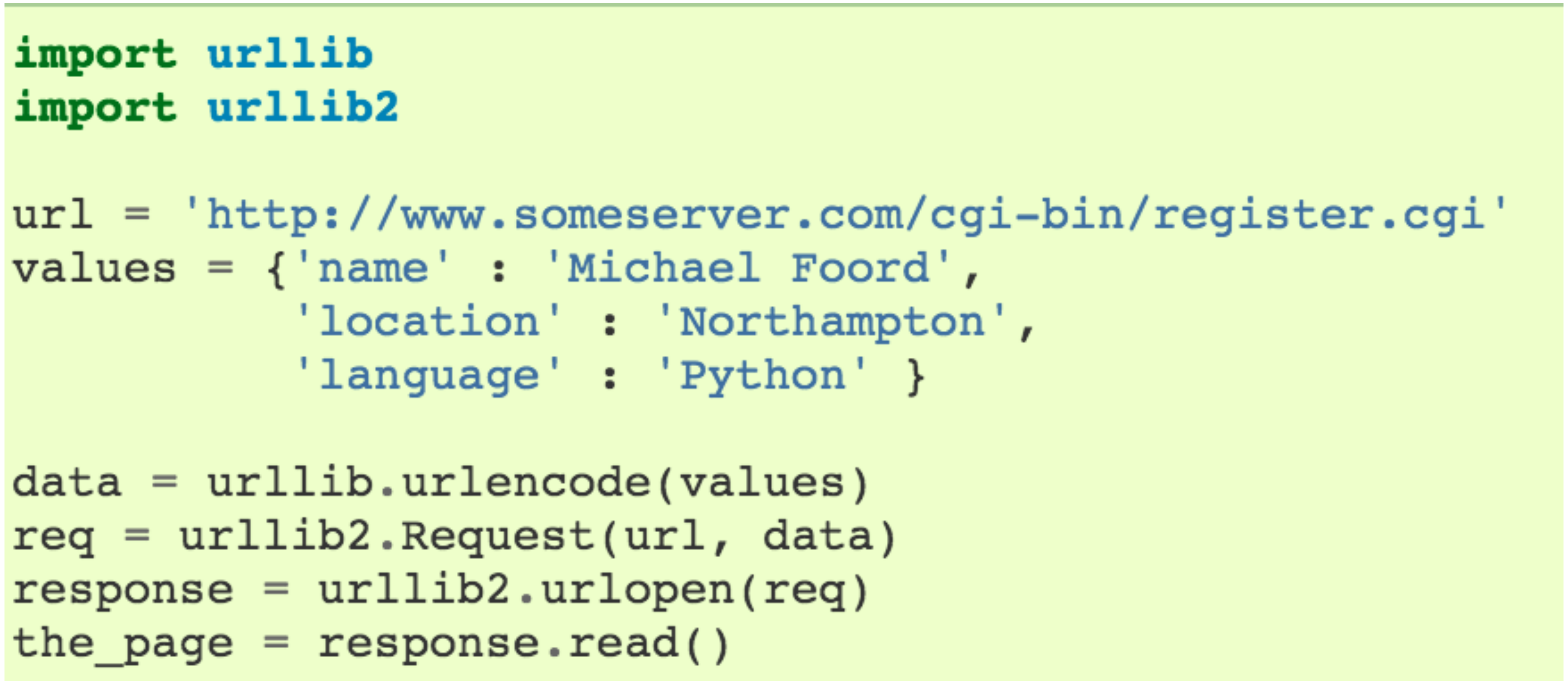
USING PYTHON FOR WEB API

- Using a “request” object, you can generate a post request:

•Create a dictionaries with variables and values

•Create a new Request object and load it with the URL and the dict.

•Execute the request via urlopen



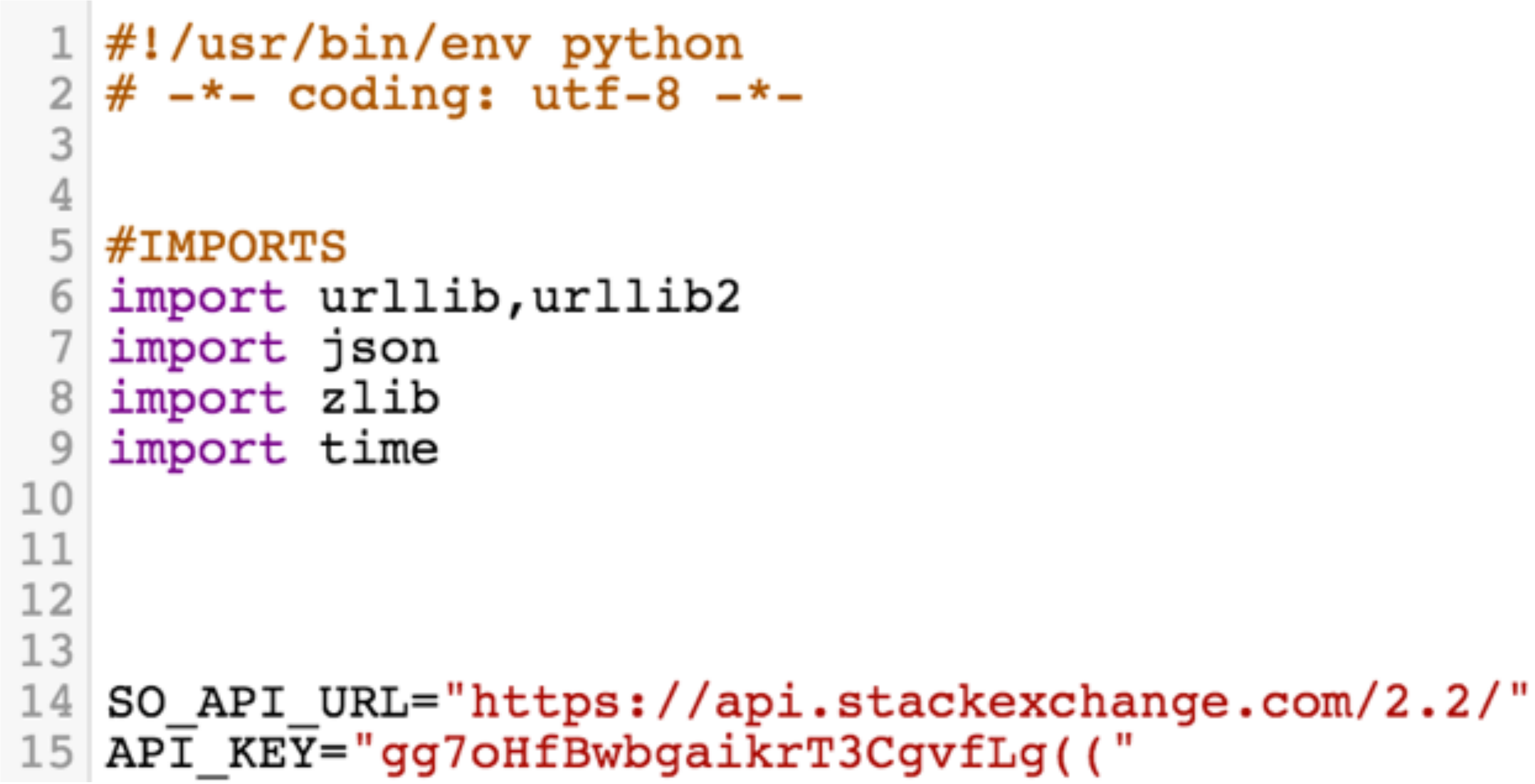
```
import urllib
import urllib2

url = 'http://www.someserver.com/cgi-bin/register.cgi'
values = {'name' : 'Michael Foord',
          'location' : 'Northampton',
          'language' : 'Python' }

data = urllib.urlencode(values)
req = urllib2.Request(url, data)
response = urllib2.urlopen(req)
the_page = response.read()
```

USING PYTHON FOR STACK EXCHANGE API

SETUP

- We will need to import libraries for HTTP handling, JSON handling and Zlib compression handling.

- Using the stack exchange API key we get more quota.

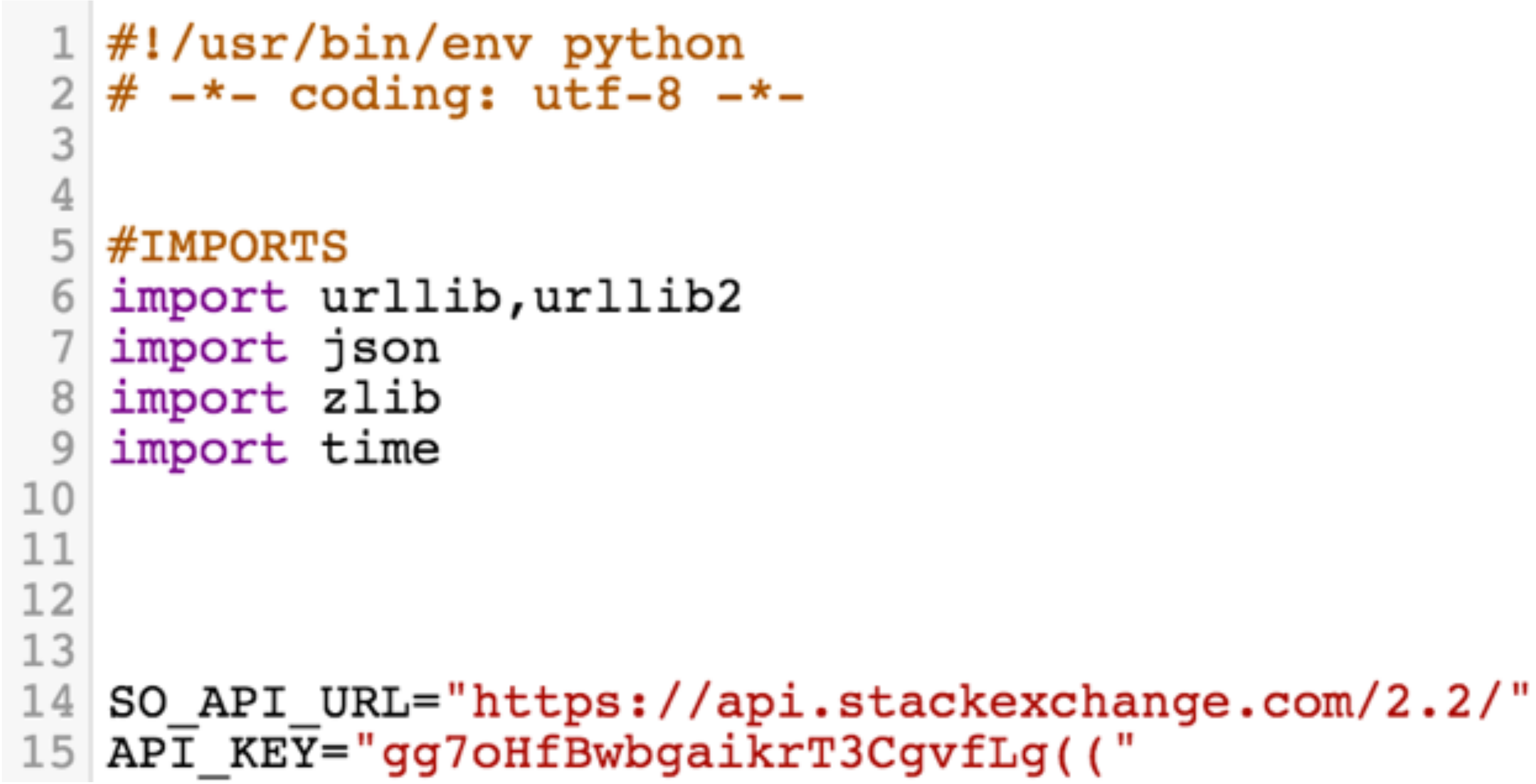


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


#IMPORTS
import urllib,urllib2
import json
import zlib
import time




SO_API_URL="https://api.stackexchange.com/2.2/"
API_KEY="gg7oHfBwbgaikrT3CgvfLg(("
```

USING PYTHON FOR STACK EXCHANGE API

SETUP

- We will need to import libraries for HTTP handling, JSON handling and Zlib compression handling.
- Using the stack exchange API key we get more quota.

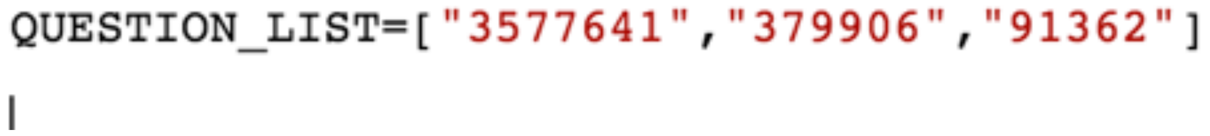


```
#!/usr/bin/env python
# -*- coding: utf-8 -*-


#IMPORTS
import urllib,urllib2
import json
import zlib
import time




SO_API_URL="https://api.stackexchange.com/2.2/"
API_KEY="gg7oHfBwbgaikrT3CgvfLg(("
```

USING PYTHON FOR STACK EXCHANGE API

We want to get answers to questions by their question ID.

- Assume this is the question ID list :



```
QUESTION_LIST=["3577641","379906","91362"]
```

- The basic method for retrieving:

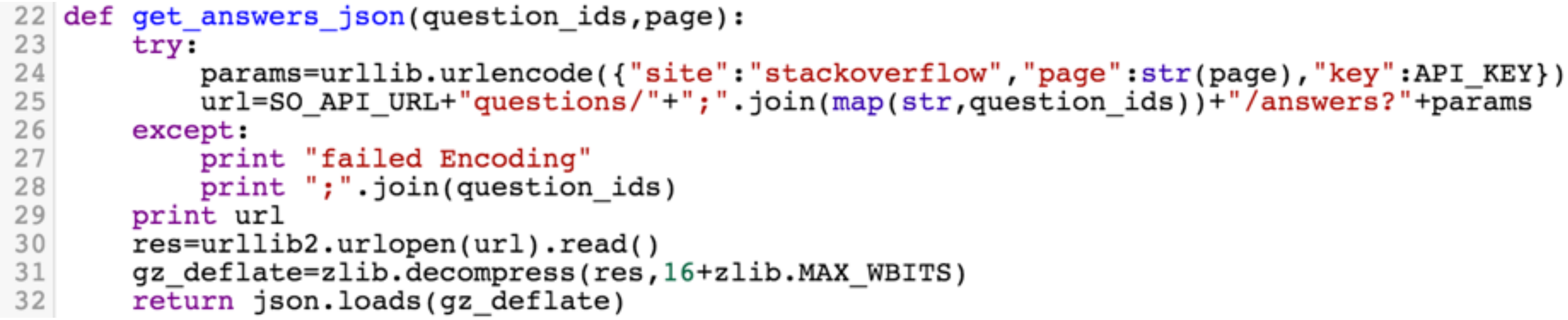
1. Preparing list of ***url encoded*** parameters (line 24)

2.compiling the URL (line 25)

3. Executing the request (line 30)

4.decompressing the results (31)

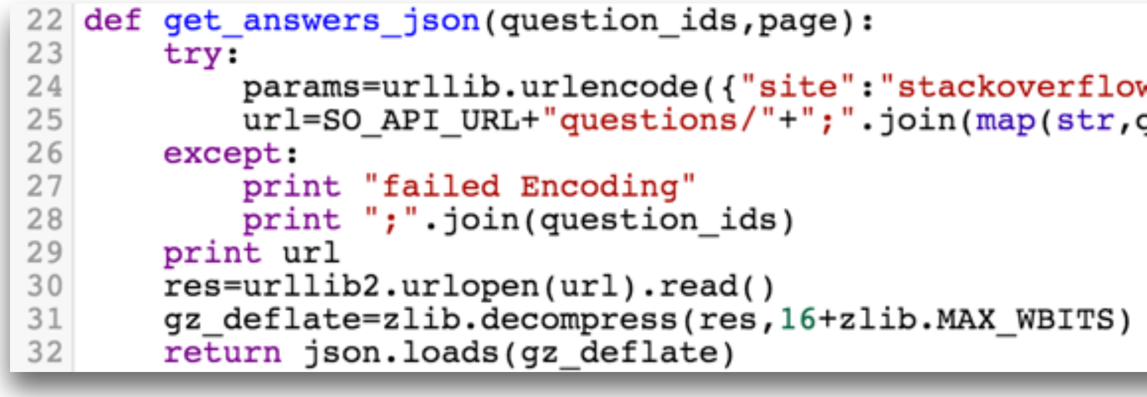
5.Parsing the Json into a dictionary and return it (line 32)



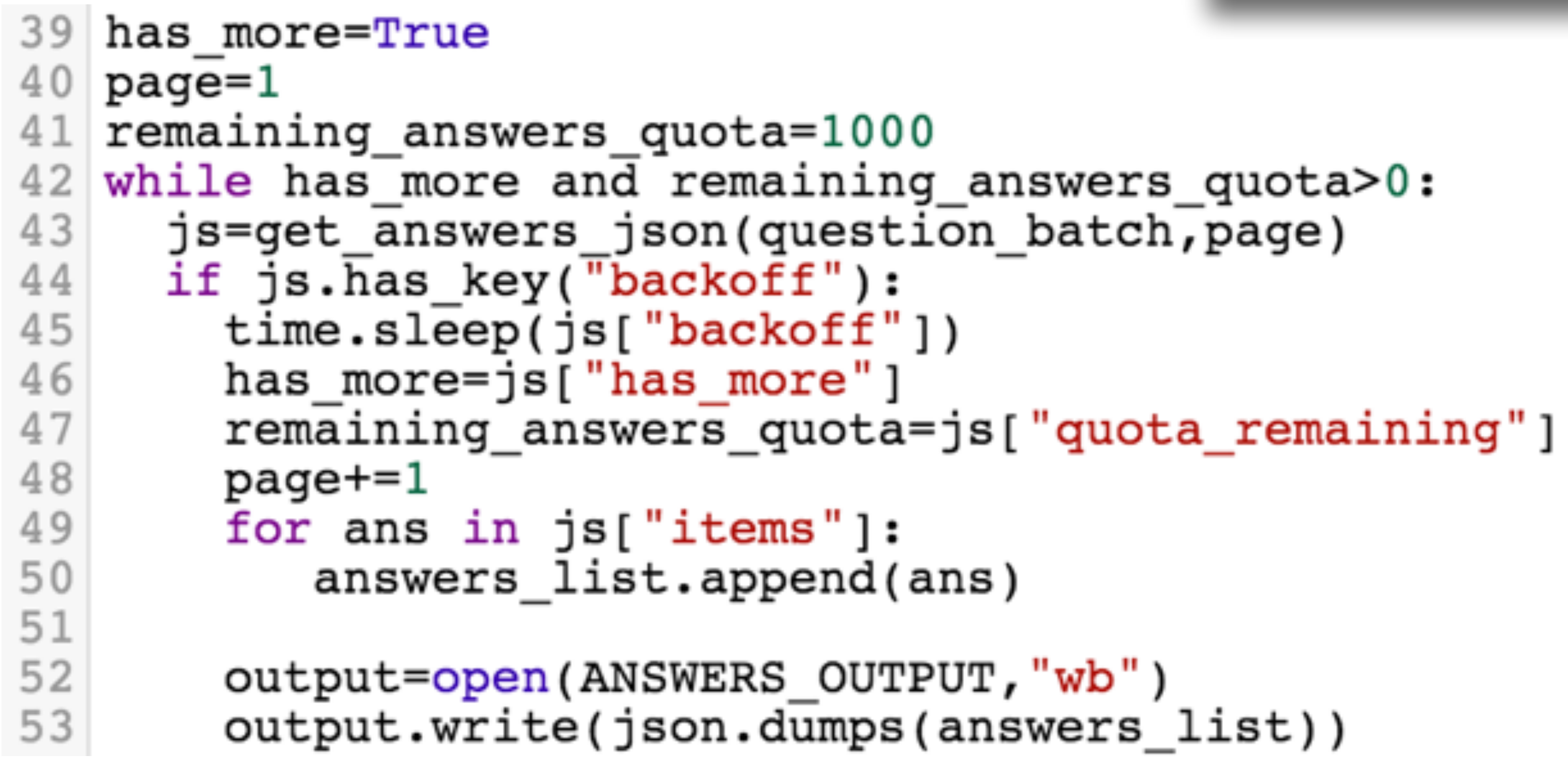
```
22 def get_answers_json(question_ids,page):
23     try:
24         params=urllib.urlencode({"site":"stackoverflow","page":str(page),"key":API_KEY})
25         url=SO_API_URL+"questions/"+";".join(map(str,question_ids))+"/answers?"+params
26     except:
27         print "failed Encoding"
28         print ";".join(question_ids)
29     print url
30     res=urllib2.urlopen(url).read()
31     gz_deflate=zlib.decompress(res,16+zlib.MAX_WBITS)
32     return json.loads(gz_deflate)
```

USING PYTHON FOR STACK EXCHANGE API

- Still it is not so simple as stack exchange are not פרייארים:
  - ★Requests quota is limited
  - ★“Backoff”: If you don’t wait the backoff, you are banned.
  - ★They don't send all results at once (“hasMore”)
  - ★No more than 100 questions IDs can be sent at once.



```
def get_answers_json(question_ids,page):
    try:
        params=urllib.urlencode({"site":"stackoverflow
        url=SO_API_URL+"questions/"+";".join(map(str,
    except:
        print "failed Encoding"
        print ";".join(question_ids)
    print url
    res=urllib2.urlopen(url).read()
    gz_deflate=zlib.decompress(res,16+zlib.MAX_WBITS)
    return json.loads(gz_deflate)
```



```
has_more=True
page=1
remaining_answers_quota=1000
while has_more and remaining_answers_quota>0:
  js=get_answers_json(question_batch,page)
  if js.has_key("backoff"):
    time.sleep(js["backoff"])
    has_more=js["has_more"]
    remaining_answers_quota=js["quota_remaining"]
    page+=1
    for ans in js["items"]:
       answers_list.append(ans)

    output=open(ANSWERS_OUTPUT,"wb")
    output.write(json.dumps(answers_list))
```

SUMMARY

- **We learned the basics for client side developments.**

  In reality you will use frameworks (bootstrap,angular)

  Use browser "developer tools" for adjusting CSS properties

  For all you need to know: w3schools.org

- **We learned PHP.**

  Install XAMPP to have it locally

  adjust settings in the php.ini file and httpd.conf

  Make sure you have file permissions (both unix /windows)

  For all you need to know: w3schools.org and stackoverflow.com

- **We learned RESTful services.**
  - Read the docs carefully
  - Register for an API key
  - Use the online "demo" tool first to understand the JSON structure