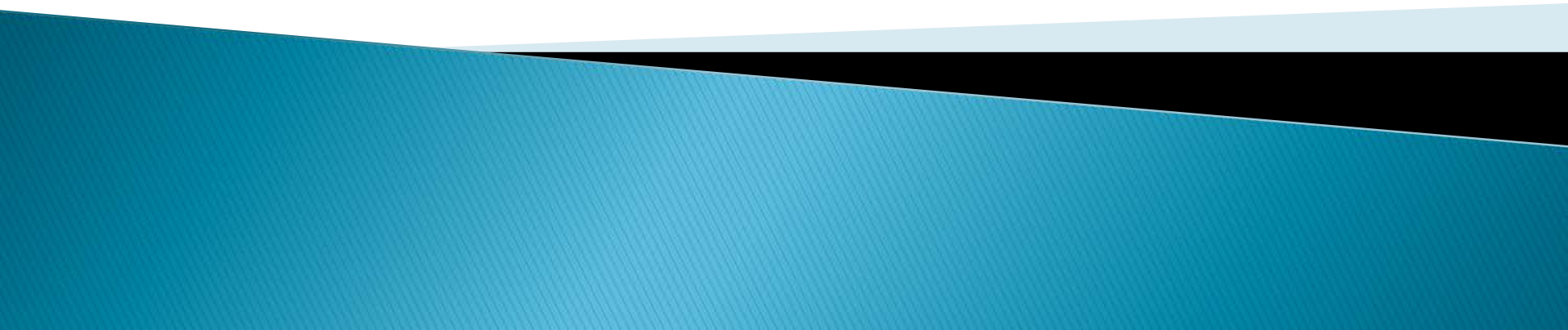
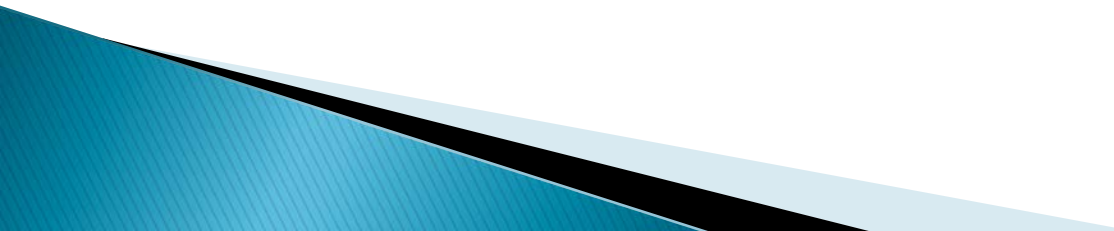


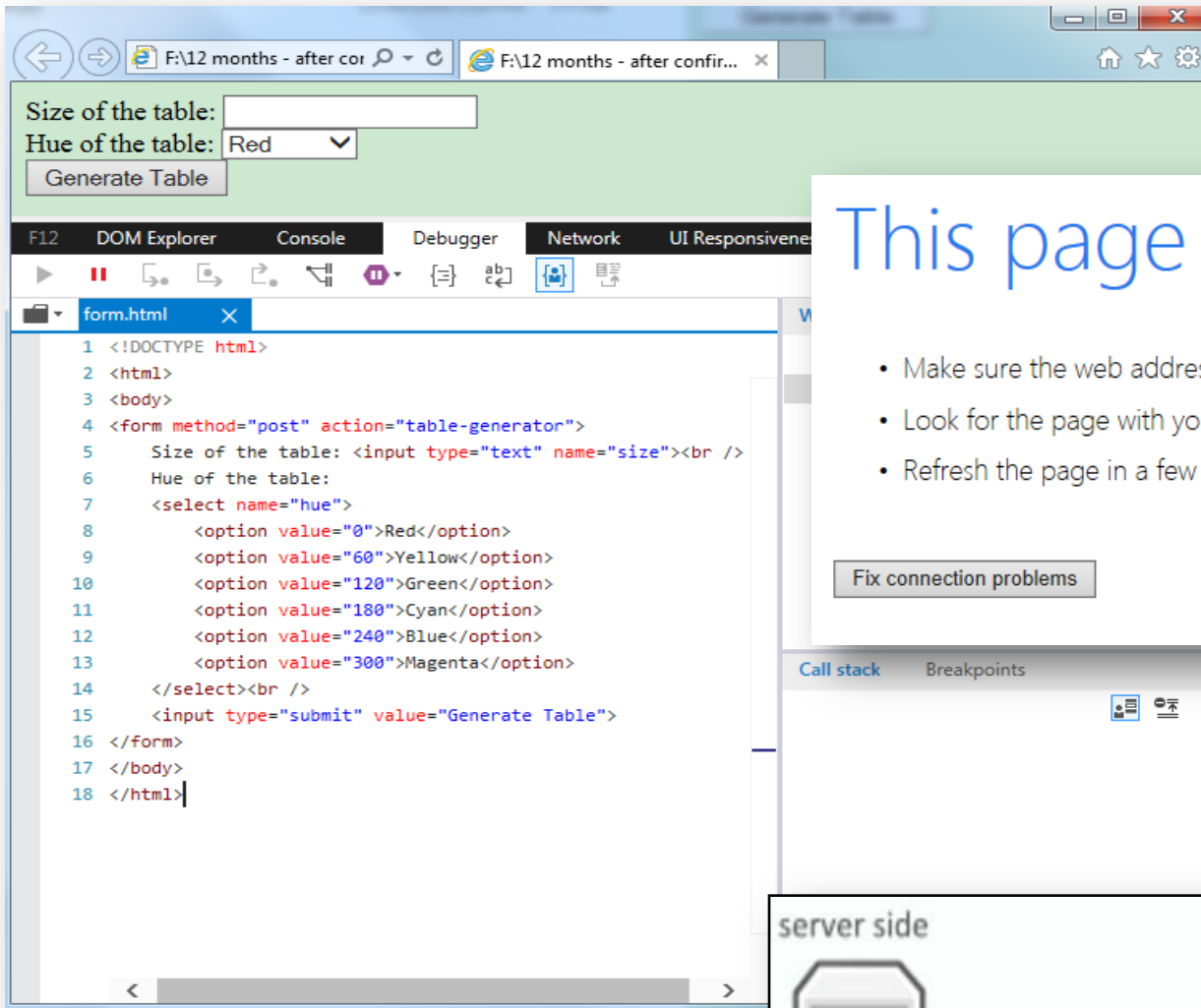
# Interactivity

Foundation of Informatics – Week 9



# Outcomes:

- ▶ Recap the mechanism in terms of HTML/CSS web application
  - ▶ Learn JavaScript and jQuery development
    - DOM manipulation
    - Event-driven programming
    - Ajax techniques
- 



# This page can't be displayed

- Make sure the web address is correct.
- Look for the page with your search engine.
- Refresh the page in a few minutes.

Fix connection problems

server side

client side (browser)



HTML page  
with JavaScript



final HTML page  
with content generated  
from the client side script

## Client Side Scripting

Size of the table:

Hue of the table: Yellow ▼

```
<!DOCTYPE html>
<html>
<body>
<form method="post" action="table-generator">
  Size of the table: <input type="text" name="size"><br />
  Hue of the table:
  <select name="hue">
    <option value="0">Red</option>
    <option value="60">Yellow</option>
    <option value="120">Green</option>
    <option value="180">Cyan</option>
    <option value="240">Blue</option>
    <option value="300">Magenta</option>
  </select><br />
  <input type="submit" value="Generate Table">
</form>
</body>
</html>
```

Client: browsing the information and sending requests (interactive functions)

```
@app.route("/table-generator" methods=['POST'])
def root():
    styles, rows = process()
    html = '''
    <!DOCTYPE html>
    <html>
    <head>
      <style>%s</style>
    </head>
    <body>
    <table>
      <tbody>%s</tbody>
    </table>
    </body>
    </html>''' % (styles, rows)
    return html, 200, {'Content-Type': 'text/html; charset=utf-8'}

if __name__ == "__main__":
    app.run(debug=True, host='127.0.0.1', port=80)
```

Server function: generating the HTML/CSS for clients

# Distinguish the server-scripting and client-scripting web app

- ▶ Download and unzip the two versions of Hello World applications on a local folder.
- ▶ Run the application (keep two pages opened)
  - >>> `http://localhost:8765/helloworld`
  - >>> `http://localhost:8765/helloworld.html`

Inspect the source code respectively...

Hello World!

Hello World!

Hello World!

http://localhost:8765/helloworld

- It is exactly same as *.py* output
- “html generator”
- Server-scripting

```
<html lang="en">
  ▼<head>
    <meta charset="utf-8">
    <title>Server Side Demo</title>
  </head>
  ▼<body>
    <p>Hello World!</p>
    <p>Hello World!</p>
    <p>Hello World!</p>
  </body>
</html>
```

http://localhost:8765/helloworld.html

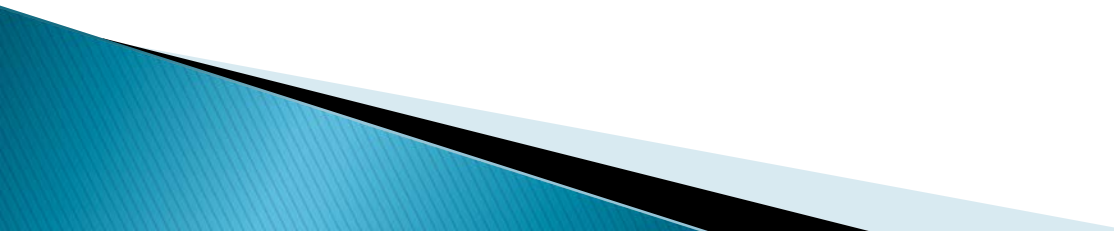
- It takes the *.html* output
- Java script
- Client-scripting

```
<html lang="en">
  ▼<head>
    <meta charset="utf-8">
    <title>Javascript Demo</title>
  </head>
  ▼<body>
    <script type="text/javascript">

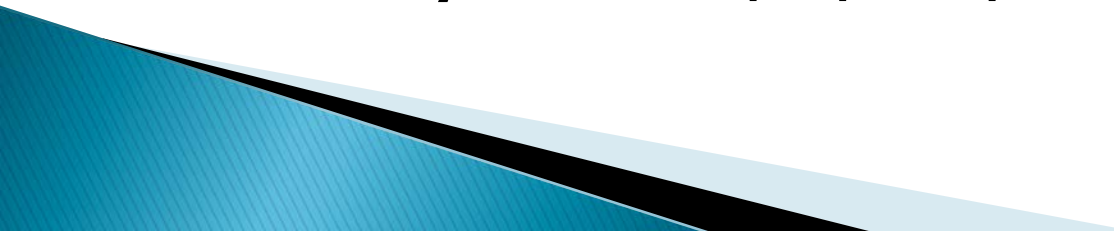
      for (i = 0; i < 3; i++) {
        document.write('<p>Hello World!</p>');
      }

    </script>
    <p>Hello World!</p>
    <p>Hello World!</p>
    <p>Hello World!</p>
  </body>
</html>
```

# Pros/cons – server

- ▶ Server-side programming, is the general name for the kinds of **programs which are run on the Server.**
  - ▶ Example Languages
    - PHP
    - **Python**
    - ASP.Net in C#, C++, or Visual Basic.
  - ▶ Server-side processing is used to interact with permanent storage like SQL-based databases or files. – **“resource sharing”**
  - ▶ **Postback** – client must wait for the server to process the request and send the page back to the client.
- 

# Pros/cons-client

- ▶ Send requests to and retrieve responses from the server.
  - ▶ Usually a browser process the scripts on the end user computers.
  - ▶ Example languages
    - HTML – for structure
    - CSS – for style
    - JavaScript – for interactive reasons
  - ▶ **Hand over the “controlling” to user**
  - ▶ **Security risks – pops up**
- 



# DOM (Document Object Model)

- ▶ The way how HTML elements are organised
  - DOM tree – similar to `tree = etree.parse('X.xml')`
  - In the JavaScript, the whole DOM is reflected by a global variable `document`
  - `dom-0.html`

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

```
function initContent() {  
    var content = document.getElementById("content");  
    content.innerHTML = "<h1>Hello World from JavaScript</h1>";  
}
```

```
window.onload = initContent; // call initContent once this HTML page is fully loaded
```

```
</script>
```

# jQuery – a lightweight JS library

- Compared with pure js, jQuery provides a simple way to realise the HTML operating.
- Reduce codes at the client side.
- “dom-1.html”

```
<script src="jquery.js" type="text/javascript"></script>  

```

# Inspect...

- How many CSS segments are working on this output
- About `<h1>hello work from jQuery</h1>`, which CSS segment is actually rendered
- Can you identify the “conflict”
- `$(‘p’)` refers to what?
  - Element selector
- `$(‘.First-three-para’)`?
  - Class selector

## Hello World from jQuery

Paragraph 0

Paragraph 1

Paragraph 2

Last paragraph

```
▼<script type="text/javascript">

function populateParagraphs(){
    var html = '';
    for (i = 0; i < 3; i++) {
        html += '<p class="the-first-three-para" id="para-' + i + '>Paragraph ' + i + '</p>';
    }
    html += '<p id="" + i + '>Last paragraph</p>';

    // use CSS selector to manipulate the DOM objects
    $('#content').append(html);
    <!-- which means the 'html' will be formatted as '#content' format -->
    $('p').css({ "font-weight": "bold" });
    $('.the-first-three-para').css({ "font-style": "italic" });
}

$(document).ready( // call the following function
function (){
    $('#content').html('<h1>Hello World from jQuery</h1>');
    attributes = {"font-family": "Verdana", "color": "#ff9900"};
    $('#content h1').css(attributes);
    <!-- based on the styling sentence in CSS, here the #content h1 potentially override the #content in CSS-->
    populateParagraphs();
}
); // you can make initContent as inline anonymous function
```

# Event driven paradigm

- ▶ The common paradigm of js
- ▶ Rely on the *'event-handler'*
  - HTML element can be associated with an “event”
  - A function is the ‘handler’ triggered by events
  - `$(selector).click(function_that_handle_the_click_event);`
  - `event-0.html`

```
<input id="show_button" type="button" value="Show Content" />
<input id="hide_button" type="button" value="Hide Content" />
```

```
function showContent() { $('#content').show('slow'); }
function hideContent() { $('#content').hide('slow'); }
...
```

```
$('#show_button').click(showContent);
$('#hide_button').click(hideContent);
```

```
$('#show_button').click(
    function(){
        $('#content').show('slow');
    }
);
```

# Exercise

- ▶ Try to “hover....fadeIn” effect
- ▶ Solution:

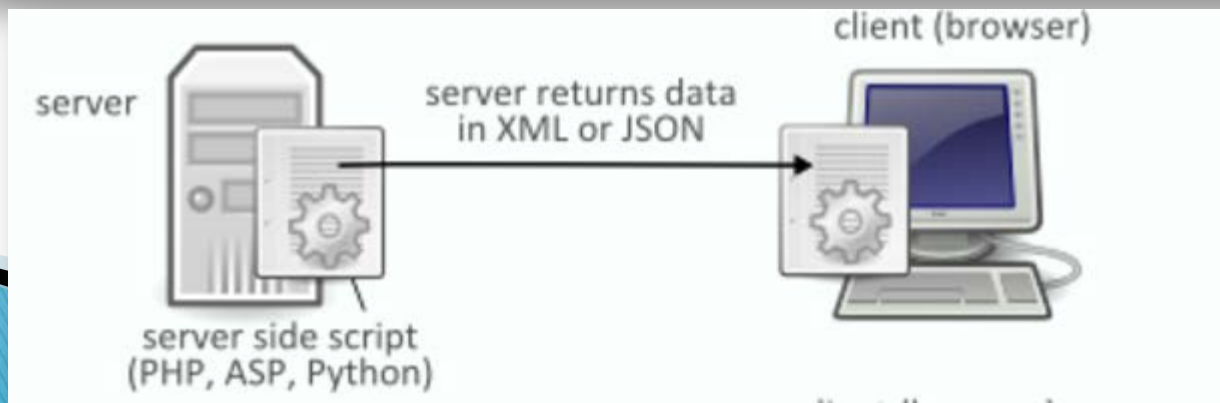
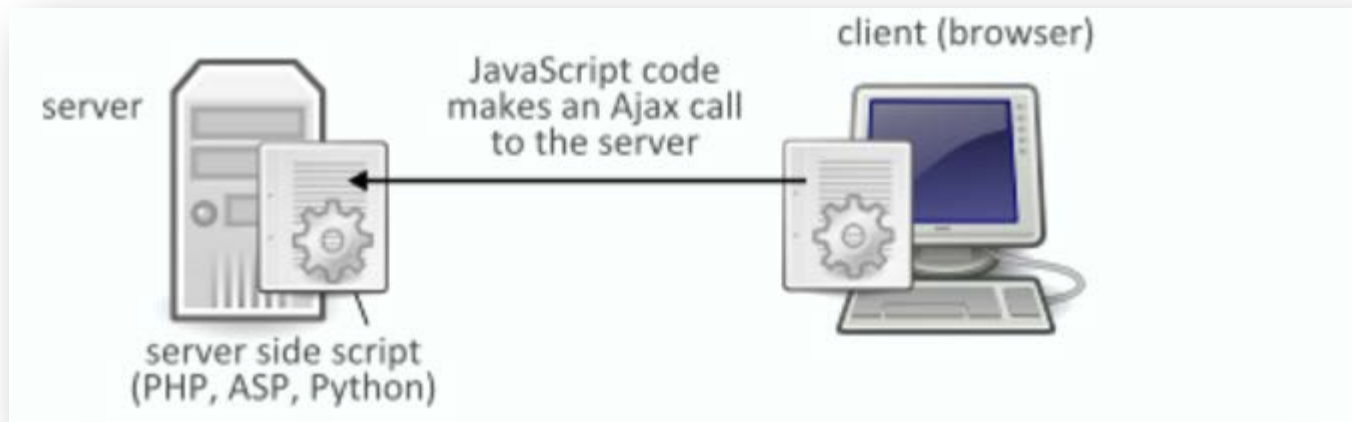
```
$("#para-0").hover(function() {$("#para-1").fadeOut();}, function() {$("#para-1").fadeIn();});
```

# Exercise – after the class

- ▶ Download `jquery-demo.zip`, load the `gen-table.html` in your browser, and observe the behaviour of this simple application.
- ▶ Without looking at the underlying code, try to recreate the page.
- ▶ Solution is the “source code”

# Ajax – Asynchronous JS and XML

- ▶ Group of technologies used to communicate between client and server without interfering with display or behaviour of a page



# Ajax

- HTML/CSS for presentation
- DOM for dynamic display
- XML for data interchange
  - **XMLHttpRequest** object (xhr)
    - It is used to exchange data with a server behind the scenes.
    - With the xhr object you can update parts of a web page, without reloading the whole page
    - [http://www.w3schools.com/xml/dom\\_http.asp](http://www.w3schools.com/xml/dom_http.asp)
- JavaScript to tie all the technologies together



# Ajax-xml.zip – observe...

```
function ajaxRequest(e) {  
  var responseType = 'xml'  
  var expression = $('#input[name=expression]').val();  
  var requestType = 'application/json';  
  //var formData = $('#ajax-form').serializeArray();  
  var requestData = {  
    format: responseType,  
    expression: expression  
  };  
};
```

```
<input type="text"  
      id="expression"  
      name="expression"  
      placeholder="Enter a mathematical expres  
      required="required"  
      value="2+2">
```

- "application/json" – type/format of the request
- MIME Type – Multipurpose Internet Mail Extensions – extends the format of email
- MIME was designed mainly for SMTP (Simple Mail Transfer Protocol), the content types defined by MIME standards are also of importance in communication protocols such as HTTP for the World Wide Web.
  - <http://www.sitepoint.com/web-foundations/mime-types-complete-list/>

# jQuery.ajax([setting])

- ▶ Perform an asynchronous HTTP (Ajax) request.
  - A set of key/value pairs that configure the Ajax request.
  - When sending data to the server, use this **content type**.
  - **dataType**: The type of data that you're expecting back from the server.
  - **Data** to be sent to the server.
  - **success()** only gets called if your webserver responds with a 200 OK HTTP header – basically when everything is fine.
  - **complete()** will always get called when the request finishes (no matter if the ajax call was successful or not)
    - If it is successful, the **.complete()** will get called *after* **.success()** gets called.

```
$.ajax({
  type: "POST",
  url: "ajax-handler",
  contentType: requestType,
  dataType: responseType, // server will return data in either json or xml
  data: JSON.stringify(requestData),
  success: function(responseData) {
    // process result in xml
    expression = $(responseData).find('expression').text();
    result = $(responseData).find('result').text();
    $('#id-log').prepend(expression + " = " + result + "\n");
  },
  complete: function(XMLHttpRequest, textStatus) { // display the raw response
    $('#id-row').val(XMLHttpRequest.responseText)
  }
});
```

# Exercise

- ▶ Modify the application to provide an option to the user for **selecting the format of the Ajax response** (either as XML or JSON).
  - You need to add a radio button `ajax.html` for format selection.
  - Modify `ajax.py` to include response generation in JSON format.

# Remind...

## ▶ Submission

- The **final team members**
- Final choice of domain and data-sets

## ▶ Presentation

- Final week of the workshop (24<sup>th</sup> May)
  - It is worth 3%
  - 10–15 min
  - **Attendance\***
  - Slides are allowed to use (not required)
  - Four issues should be covered in the presentation
- 