

introduction to **SCRIPTING, DATABASES, SYSTEM ARCHITECTURE**

HTML Forms, Intro to PHP, HTML Validation



Claus Brabrand

(((brabrand@itu.dk)))

Associate Professor, Ph.D.

(((Programming, Logic, and Semantics)))

 IT University of Copenhagen

Agenda

- **1) HTML FORMS**

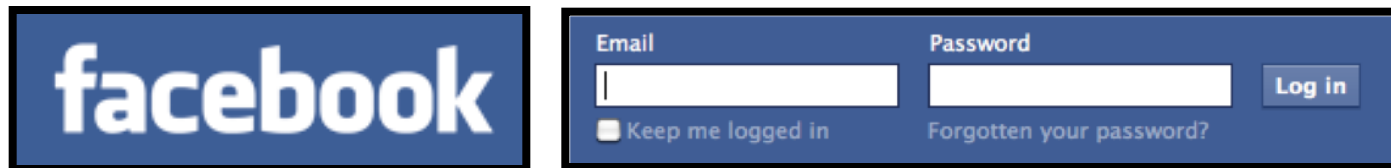
- **2) PHP (intro)**

- **3) HTML VALIDATION**

- **4) ASSIGNMENT**

HTML Forms

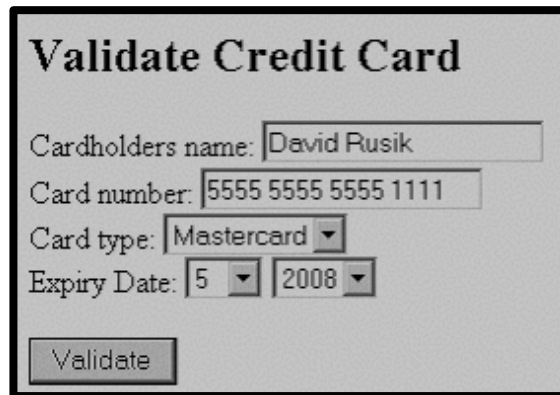
- Forms allow user to submit info to server:
 - (e.g. Login, Email, Search, Polls, Surveys, ...)



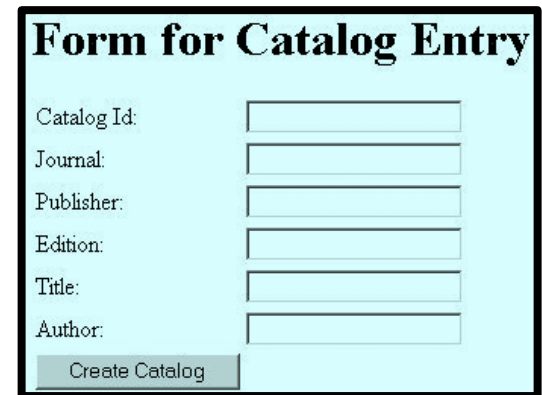
A screenshot of the Facebook login interface. It features the Facebook logo on the left. To the right, there are two input fields labeled 'Email' and 'Password'. Below the 'Email' field is a checkbox labeled 'Keep me logged in'. Below the 'Password' field is a link that says 'Forgotten your password?'. A 'Log in' button is positioned to the right of the password field.



A screenshot of a Google Account sign-in form. It has a title 'Sign in with your Google Account'. Below this, there are input fields for 'Email:' and 'Password:'. An example email 'ex: pat@example.com' is shown below the email field. There is a checkbox labeled 'Stay signed in' and a 'Sign in' button at the bottom.



A screenshot of a 'Validate Credit Card' form. It contains several input fields: 'Cardholders name:' with the value 'David Rusik', 'Card number:' with the value '5555 5555 5555 1111', 'Card type:' with a dropdown menu showing 'Mastercard', and 'Expiry Date:' with two dropdown menus showing '5' and '2008'. A 'Validate' button is at the bottom.

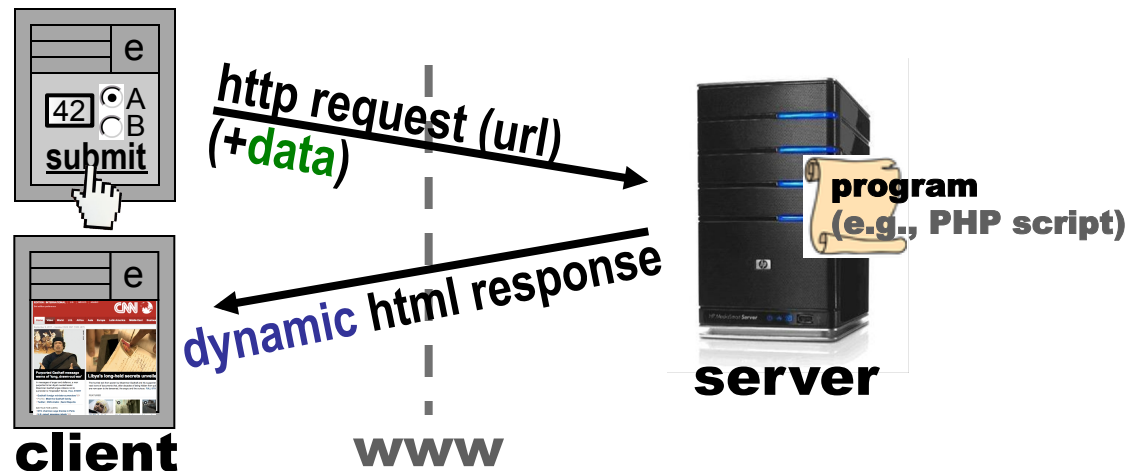


A screenshot of a 'Form for Catalog Entry'. It has a title 'Form for Catalog Entry' and several input fields: 'Catalog Id:', 'Journal:', 'Publisher:', 'Edition:', 'Title:', and 'Author:'. A 'Create Catalog' button is at the bottom.

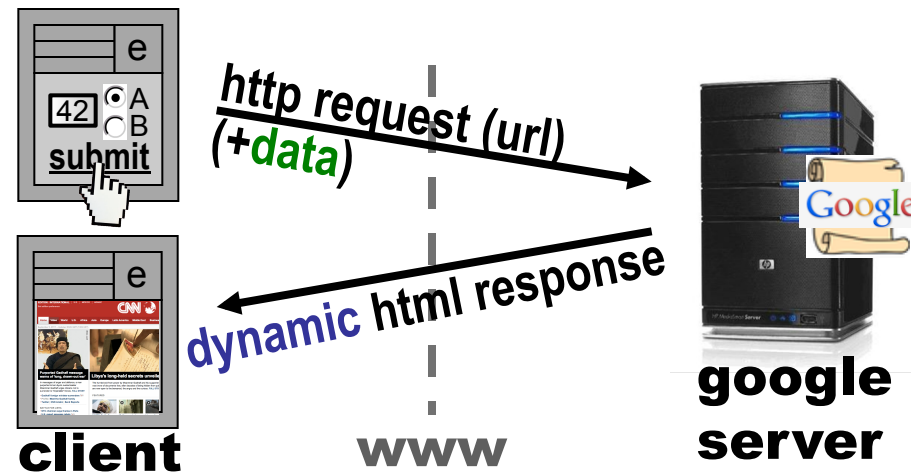
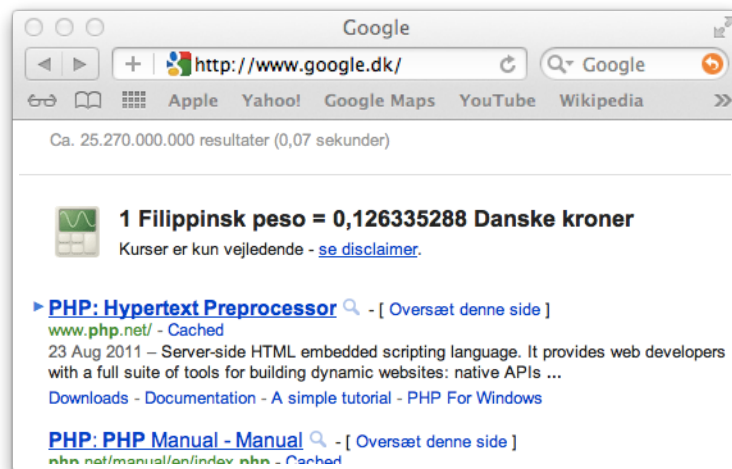
- This information is subsequently **processed** by a program on the server (e.g., PHP script)

Form Submission

- 1) The user **fills out the form** and clicks “submit” (which sends the **data** back to the server)
- 2) The server **runs a web service** (PHP program) that processes the **data** and constructs an HTML reply
- 3) The server sends back the **dynamically constructed HTML** document (that may depend on the **data!**):



Form Submission



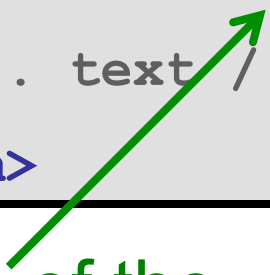
Dynamic HTML reply may depend on:

- **Data** (submitted by client)
- **Server state**
- **Time**
- **Other web sites...**

The **<form>** Element

- Form syntax:

```
<form action="address_of_script">  
    ... text / input fields / images / ...  
</form>
```



Value of the **action** attribute denotes the script that is to **process** the data of the form (when it is submitted)

- **Note:** forms cannot be *nested* (i.e., a “form” cannot contain a “form”)

Dictionary: 'nest'

Merriam Webster's('nest'):

Main Entry: **1nest**

Pronunciation: \ ' nest \

Function: **noun**

Etymology: Middle English, from Old English; akin to Old High German nest nest, latin nidus

First Known Use: before 12th century

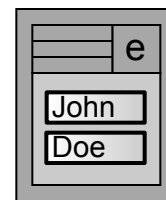
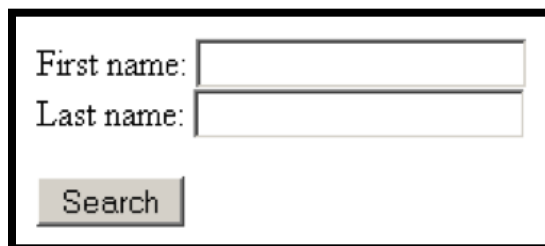
- 1a. A container or shelter made by a bird out of twigs, grass, or other material to hold its eggs and young.
- 1b. A similar structure in which fish, insects, or other animals deposit eggs or keep their young.
- 1c. A place in which young are reared; a lair.
- 1d. A number of insects, birds, or other animals occupying such a place: a nest of hornets.
2. A place affording snug refuge or lodging; a home.
- 3a. A place or environment that fosters rapid growth or development, especially of something undesirable; a hotbed: a nest of criminal activity.
- 3b. Those who occupy or frequent such a place or environment.
- 4a. A set of objects of graduated size that can be stacked together, each fitting within the one immediately larger; e.g., a nest of tables.**
- 4b. A cluster of similar things.
- 5. [Computer Science]: A set of data contained sequentially within another.**
6. A group of weapons in a prepared position: a machine-gun nest.



Example Form

■ Example Form:

```
<form action="http://www.brics.dk/ixwt/echo">
  <p>
    First name: <input type="text" name="firstname" /> <br />
    Last name:  <input type="text" name="lastname" /> <br />
  </p>
  <input type="submit" value="Search" />
</form>
```



client

firstname=John
lastname=Doe



server

Try it: (<http://itu.dk/people/brabrand/DSDS/form.html>)

More info: (http://www.w3schools.com/html/html_forms.asp)

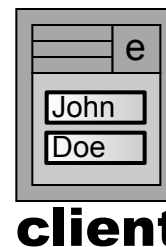
Example Form

■ Example Form:

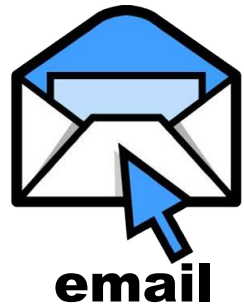
```
<form action="mailto:a@b.c" method="post" >
  <p>
    First name: <input type="text" name="firstname" /> <br />
    Last name:  <input type="text" name="lastname" /> <br />
  </p>
  <input type="submit" value="Search" />
</form>
```

First name:

Last name:



*firstname=John&
lastname=Doe*



<input ... />

■ Many “**type**”s of <input>:

```
<form action="address_of_script">
  <p>
    First name: <input type="text" name="firstname" /> <br />
    ...
</form>
```

■ <input type="text" .../>

■ <input type="password" .../>

■ <input type="radio" .../>

■ <input type="checkbox" .../>

■ <input type="submit" .../>

■ <input type="reset" .../>

■ ...

Form demo

file:///Users/brabrand/Desl Google

Apple Yahoo! Google Maps YouTube

First name:

Password:

Coke: ☒ Pepsi: ☐

Cheese: ☒

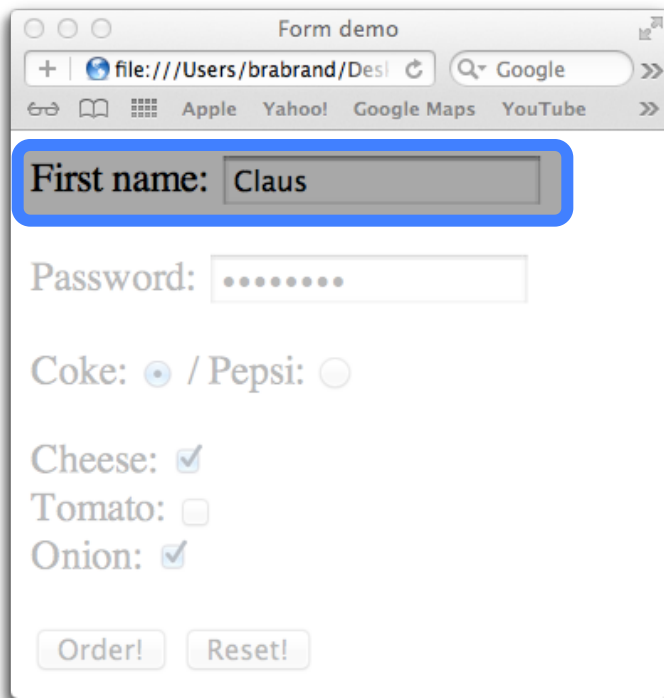
Tomato: ☐

Onion: ☒

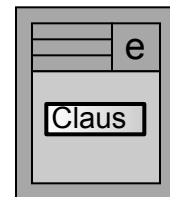
<input type="text" ... />

First name:

```
<input type="text" name="firstname" />
```



A screenshot of a web browser window titled "Form demo". The address bar shows a local file path. The form contains several fields: a text input for "First name:" with the value "Claus" (highlighted by a blue border), a password input field with masked characters, and two radio buttons for "Coke:" and "Pepsi:". Below these are three checkboxes for "Cheese:", "Tomato:", and "Onion:". At the bottom are "Order!" and "Reset!" buttons.



client

firstname=Claus



server

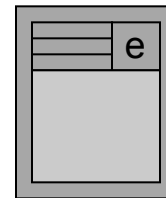
size="30" (width of text field) **maxlength="10"** (max text length allowed)

<input type="password" ... />

Password:

```
<input type="password" name="pass" />
```

A screenshot of a web browser window titled "Form demo". The address bar shows a local file path. The form contains several fields: "First name:" with the value "Claus", "Password:" with masked characters (highlighted by a blue box), "Coke:" and "Pepsi:" radio buttons, "Cheese:", "Tomato:", and "Onion:" checkboxes, and "Order!" and "Reset!" buttons at the bottom.



client

pass=12345678



server

<input type="radio" ... />

Coke:

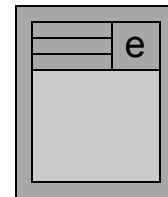
```
<input type="radio" name="drink" value="coke" />
```

/ Pepsi:

```
<input type="radio" name="drink" value="pepsi" />
```

Note: same name !
(radio button *group*)

A screenshot of a web browser window showing a form. The form has fields for 'First name' (Claus), 'Password' (masked with dots), and a radio button group for 'Coke' and 'Pepsi'. The 'Coke' radio button is selected. Below the radio buttons are checkboxes for 'Cheese' (checked), 'Tomato' (unchecked), and 'Onion' (checked). At the bottom are 'Order!' and 'Reset!' buttons. A blue box highlights the radio button group.



client

drink=coke



server

checked="checked" (to tick off by default, on page load)

<input type="checkbox" ... />

Cheese:

```
<input type="checkbox" name="ingr" value="cheese" /> <br />
```

Tomato:

```
<input type="checkbox" name="ingr" value="tomato" /> <br />
```

Onion:

```
<input type="checkbox" name="ingr" value="onion" /> <br />
```

Note: same name !
(checkbox *group*)

First name:

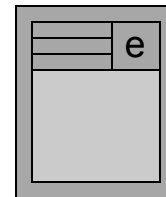
Password:

Coke: ☒ / Pepsi: ☐

Cheese: ☒

Tomato: ☐

Onion: ☒



client

ingr=cheese
ingr=onion



server

checked="checked" (to tick off by default, on page load)

<input type="submit" ... />

```
<input type="submit" value="Order!" />
```

Form demo

file:///Users/brabrand/Desi Google

First name:

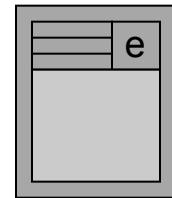
Password:

Coke: ☒ / Pepsi: ☐

Cheese: ☒

Tomato: ☐

Onion: ☒



client

*Clicking on a submit button causes all info to be **submitted** (sent) to the server*

the address of which was specified as the value of the action attribute of the form



server

`<input type="reset" ... />`

```
<input type="reset" value="Reset!" />
```

Form demo

file:///Users/brabrand/Desl Google

First name:

Password:

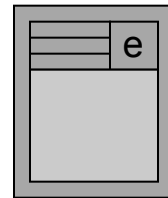
Coke: ☒ / Pepsi: ☐

Cheese: ☒

Tomato: ☐

Onion: ☒

Order!



client

Clicking reset
causes the info in
the form to be
....well, **reset**

<input/> attributes

type	name	value	checked	size	maxlength
text					
password					
hidden					
submit					
reset					
checkbox					
radio					

Obligatory	
Not obligatory	
Not allowed	

<input ... />

■ Many “**type**”s of <input>:

```
<form action="address_of_script">
  <p>
    First name: <input type="text" name="firstname" /> <br />
    ...
</form>
```

■ <input type="text" .../>

■ <input type="password" .../>

■ <input type="radio" .../>

■ <input type="checkbox" .../>

■ <input type="submit" .../>

■ <input type="reset" .../>

■ ...

Form demo

file:///Users/brabrand/Desl Google

Apple Yahoo! Google Maps YouTube

First name:

Password:

Coke: ☒ Pepsi: ☐

Cheese: ☒

Tomato: ☐

Onion: ☒

EXERCISE

For each of the “**type**”s of `<input>` fields (text, password, radio, checkbox) do:

- **1)** Make an HTML form with the input field:

```
<form>  
    ...input field goes here...  
    <input type="submit" value="Submit Form!" />  
    <input type="reset" value="Reset Form!" />  
</form>
```

- **2)** Submit to “echo service” (to see info submitted):

```
<form action="http://www.brics.dk/ixwt/echo">
```

- **3)** Submit form “via email” (to see info formatted):

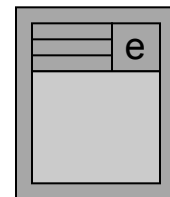
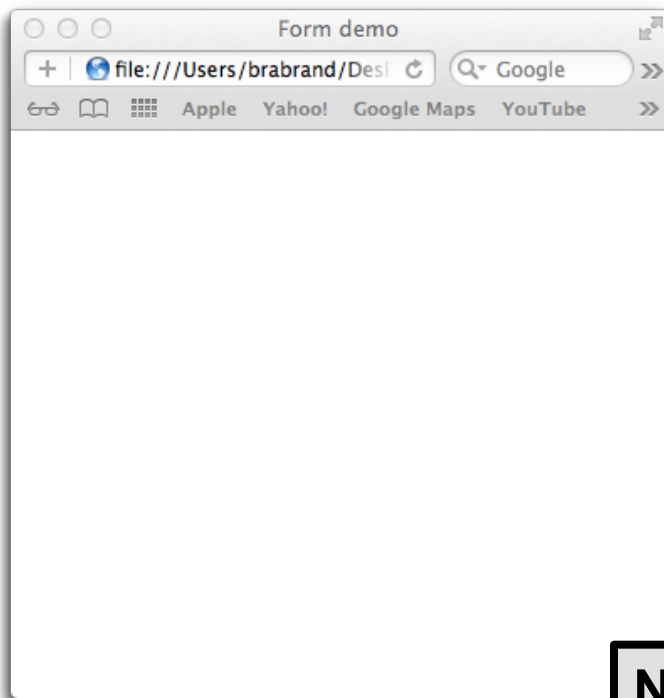
```
<form action="mailto:a@b.c" method="post" >
```

More `<input/>` fields...

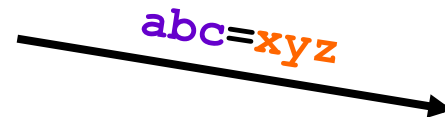


<input type="hidden" ... />

```
<input type="hidden" name="abc" value="xyz" />
```



client

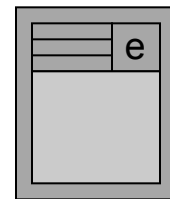
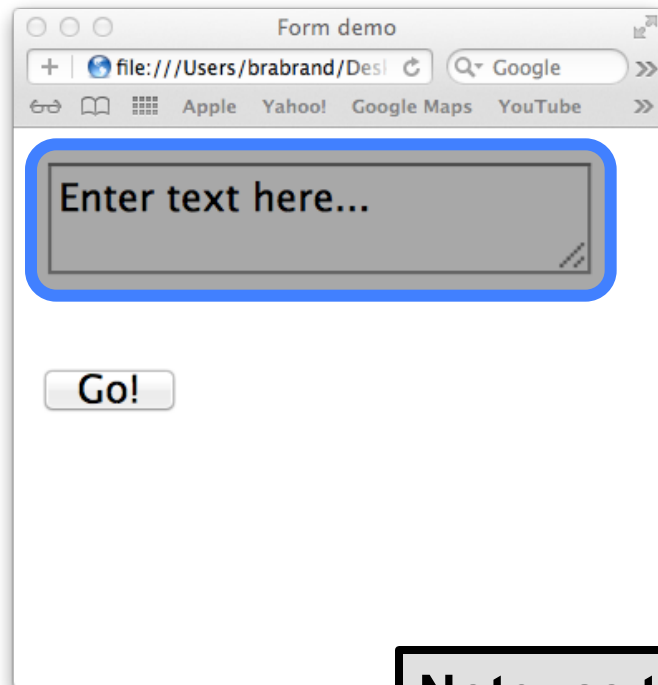


server

Note: serves to communicate information (state) from server to client and then back to server again

<textarea>

```
<textarea name="blah" rows="5" cols="30" >  
    Enter text here...  
</textarea>
```



client

blah=fgfweghefwh

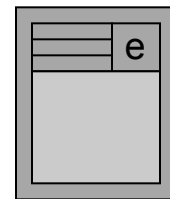
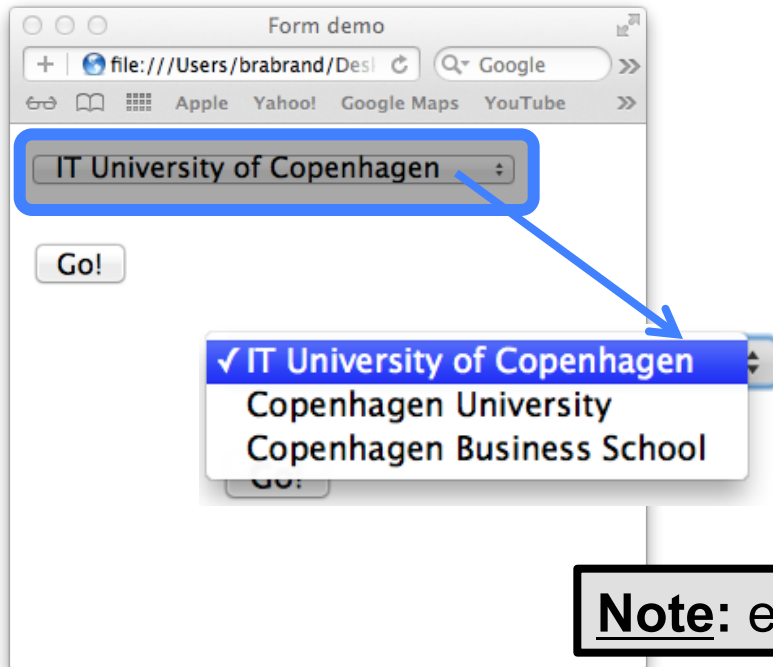


server

Note: as text input field, but permits multiple lines of input

<select>

```
<select name="uni" >
  <option value="ITU" >IT University of Copenhagen</option>
  <option value="KU" >Copenhagen University</option>
  <option value="CBS" >Copenhagen Business School</option>
</select>
```



client

uni=ITU



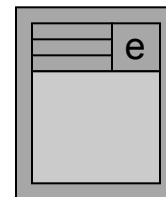
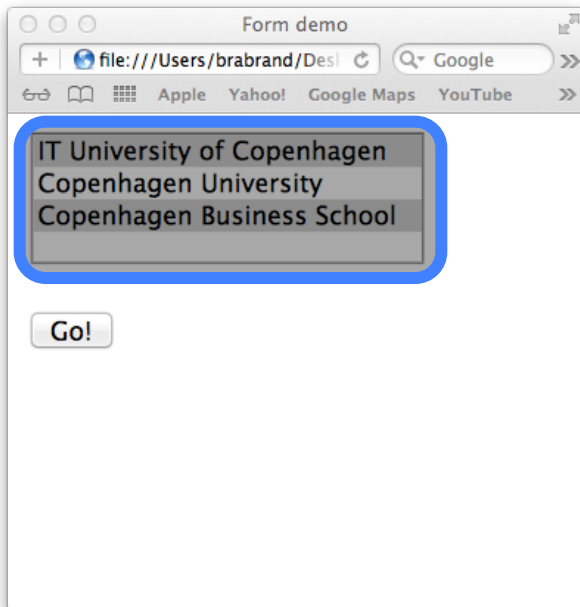
server

Note: essentially the same as a **radio button group**

selected="selected" (to tick off by default, on page load)

<select multiple="multiple">

```
<select name="uni" multiple="multiple" >
  <option value="ITU" >IT University of Copenhagen</option>
  <option value="KU" >Copenhagen University</option>
  <option value="CBS" >Copenhagen Business School</option>
</select>
```



client

uni=ITU
uni=CBS



server

Note: essentially the same as a **checkbox** group

selected="selected" (to tick off by default, on page load)

EXERCISE (II)

For each of the “**type**”s of `<input>` fields (hidden, textarea, select, select-multiple) do:

- 1) Make an HTML form with the input field:

```
<form>  
    ...input field goes here...  
    <input type="submit" value="Submit Form!" />  
    <input type="reset" value="Reset Form!" />  
</form>
```

- 2) Submit to “echo service” (to see info submitted):

```
<form action="http://www.brics.dk/ixwt/echo">
```

- 3) Submit form “via email” (to see info formatted):

```
<form action="mailto:a@b.c" method="post" >
```

Agenda

- **1) HTML FORMS**

- **2) PHP (intro)**

- **3) HTML VALIDATION**

- **4) ASSIGNMENT**

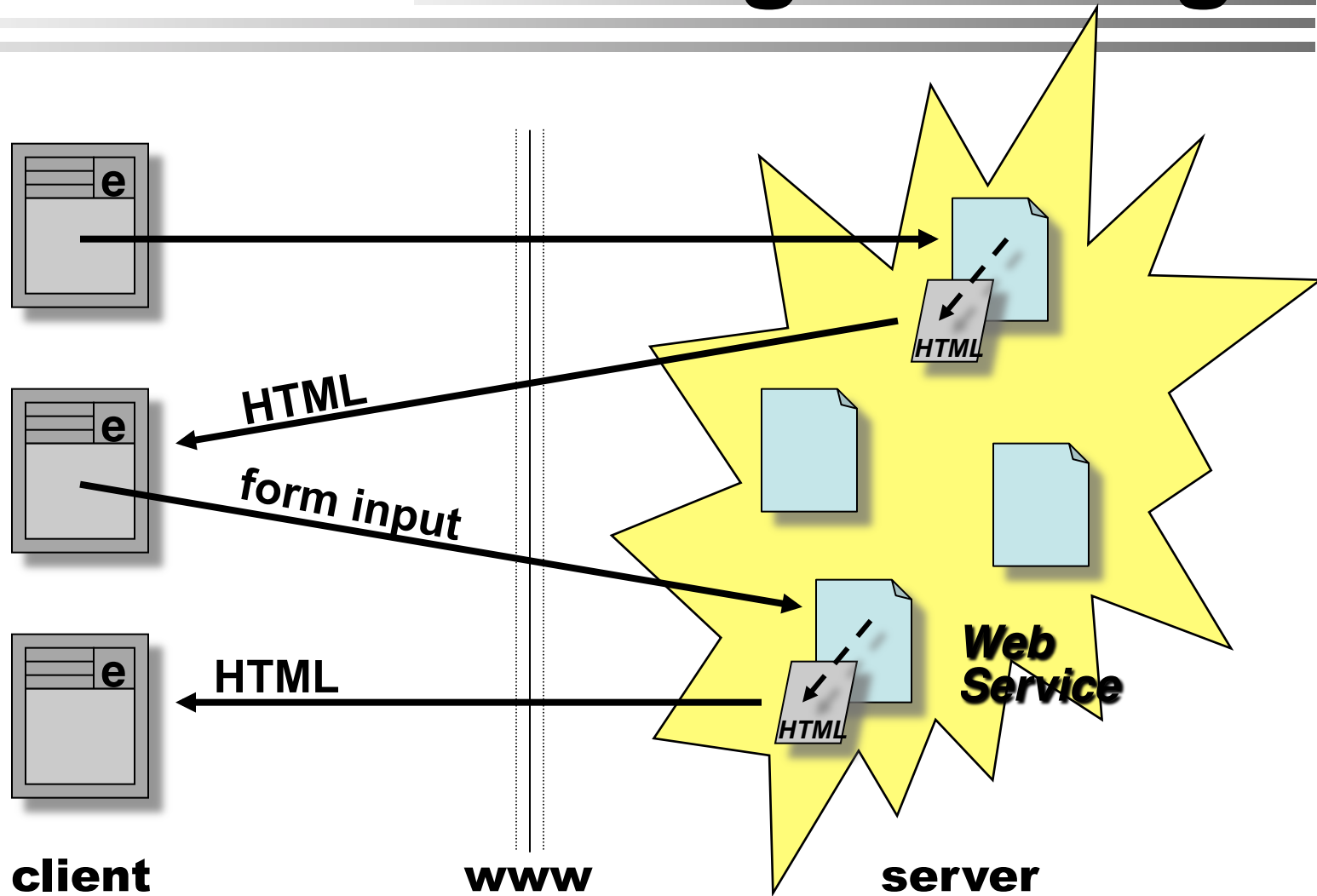
Programming Language

- A programming language allows us to tell (instruct) a computer what to do
- There are *many* different prog. languages:
 - PHP, Java, C, C++, Pascal, JavaScript, ... (10000s)
 - They all have different characteristics
 - Which one to use depends on what we want to do
 - **PHP**: made specifically for web service programming!
- Learning to program can be hard. Be patient!
 - Initially: 20% programming + 80% debugging

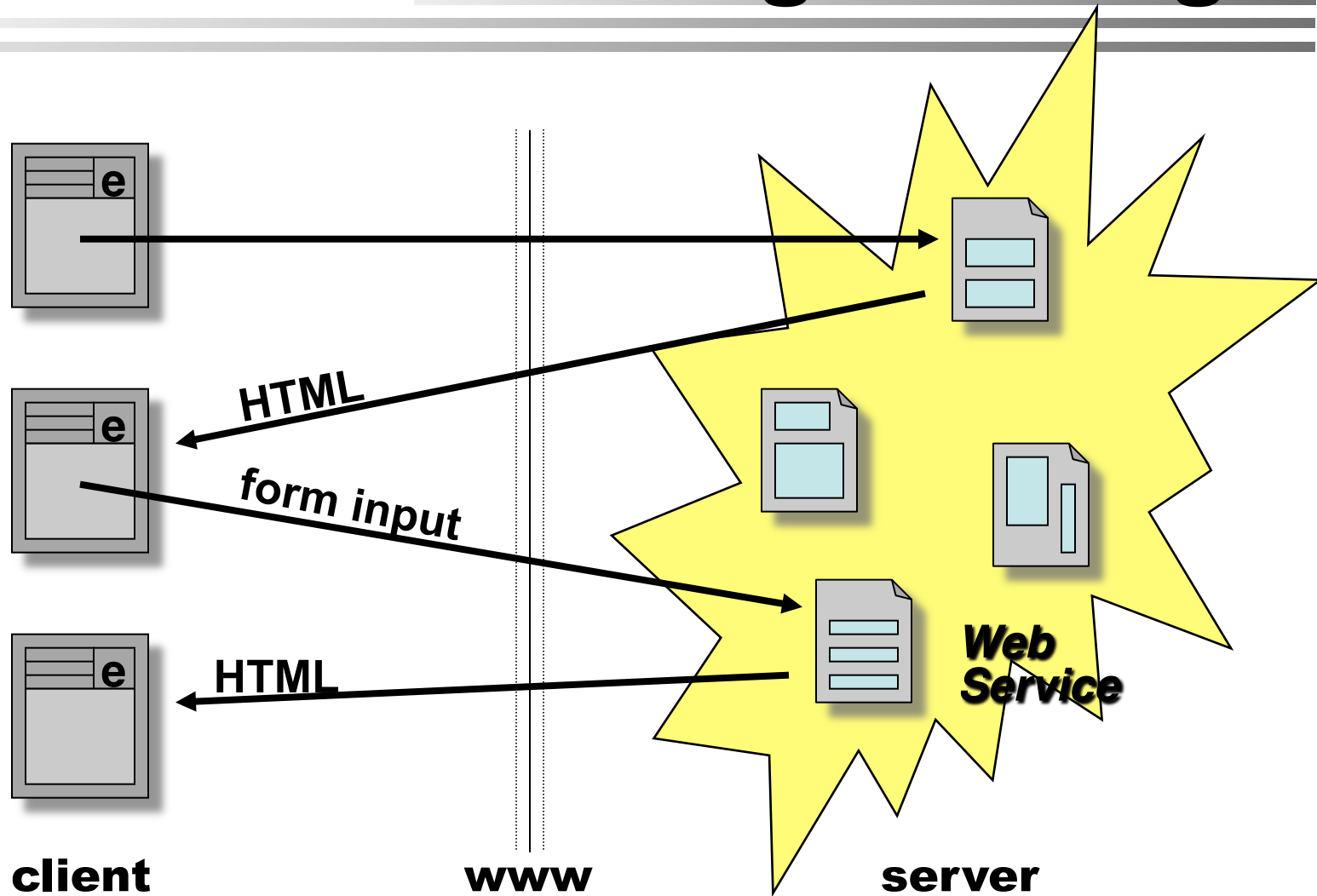


- **PHP: Hypertext Preprocessor**
- **Made for web service programming !!!**
- It is a “scripting language”
(used in Facebook, Wikipedia, Wordpress, ...)
- Other scripting languages:
 - ASP (Microsoft)
 - Javascript
 - Actionscript (Flash)

Traditional Web Programming



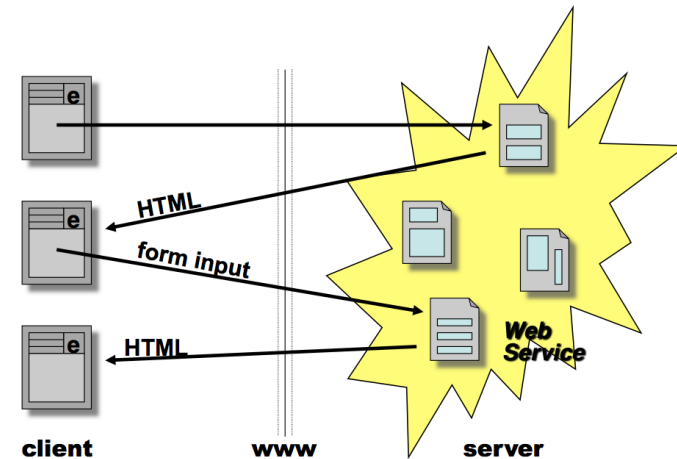
PHP Web Service Programming



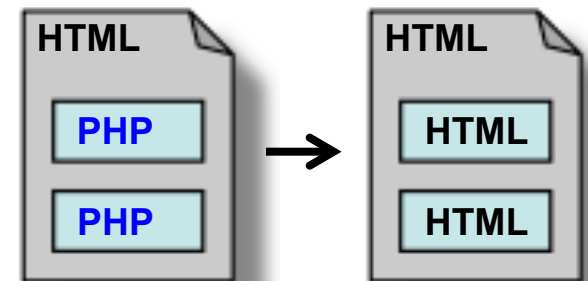
PHP

- PHP is a programming language made specifically for **web service programming**

- PHP code runs on the server (i.e., not on your computer)



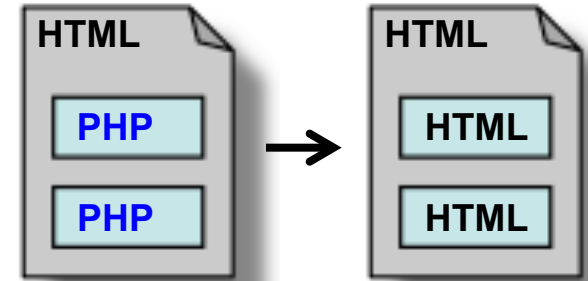
- Programming model of PHP:
 - HTML with special **PHP tags** (`<?php ... ?>`) that are evaluated and generate (dynamic) HTML



PHP Example: Hello World!

```
<html>
  <body>
    <?php
      echo "Hello World!" ;
    ?>
  </body>
</html>
```

Hello World!



- PHP code is written in `<?php ... ?>` tags inside regular HTML
- Each PHP command ends with “;” (semicolon)
- “echo” is a command that prints the argument (in this case it will print “Hello World!”)

PHP Example: Hello World!

```
<html>
  <body>
    <?php
      echo "<h1>Hello World!</h1>" ;
      echo "<p/>" ;
      echo "This is <b>bold</b> and <i>italic</i>" ;
    ?>
  </body>
</html>
```

Hello World!

This is **bold** and *italic*

- We can make HTML tags
(e.g., <h1>...</h1>, ..., ...)
- We can write multiple lines

PHP Example: Hello World!

```
<html>
  <body>
    <?php
      echo "<h1>Hello World!</h1>" ;
      echo "<p/>" ;
      echo "This is <b>bold</b> and <i>italic</i>" ;
    ?>
  </body>
</html>
```

||

```
<html>
  <body>
    <?php echo "<h1>Hello World!</h1>" ; ?>
    <p/>
    <?php echo "This is <b>bold</b> and <i>italic</i>" ; ?>
  </body>
</html>
```

Hello World!

This is **bold** and *italic*

PHP Example: Variables

```
<html>
  <body>
    <?php
      $year = 2012 ;
      echo "The year is:" ;
      echo $year ;
    ?>
  </body>
</html>
```

The year is: 2012

||

```
<html>
  <body>
    <?php $year = 2012 ; ?>
    The year is:
    <?php echo $year ; ?>
  </body>
</html>
```

The year is: 2012

PHP Example: Variables

```
<html>
  <body>
    <?php
      $year = 2012 ;
      echo "Next year is:" ;
      echo ($year + 1) ;
    ?>
  </body>
</html>
```

Next year is: 2013

- We can make **calculations**: (**\$year** + 1)

PHP Example: Today's date

```
<html>
  <body>
    Today is:
    <?php
      $today = date("d-m-Y") ;
      echo $today ;
    ?>
  </body>
</html>
```

Today is: 07-09-2012

- We can output *dynamic* information!
- **date**("...") is a **built-in function** that returns today's date according to some format "..."

(<http://php.net/manual/en/function.date.php>)

Agenda

- **1) HTML FORMS**

- **2) PHP (intro)**

- **3) HTML VALIDATION**

- **4) ASSIGNMENT**

HTML Document Structure

■ HTML Document Structure (refresh):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Hello!</title>
  </head>
  <body>
    <p>hello world!
  </body>
</html>
```



XHTML Document Structure

■ XHTML Document Structure (refresh):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Hello</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```



XHTML (Common Mistakes)

- Elements must be properly **nested**:

- `<i><b></b></i>`

(not `<i><b></i></b>`)

- Attributes values in single or double **quotes**:

- `<a href="xyz" />` or `<a href='xyz' />` (not `<a href=xyz />`)

- Always **lower case** element names:

- `<html>`

(not `<HTML>`)

- All elements must be **closed**:

- `<p>...</p>` or `<p />`

(not: `<p>...`)

- ``

Validation of (X)HTML

■ Static Validation:

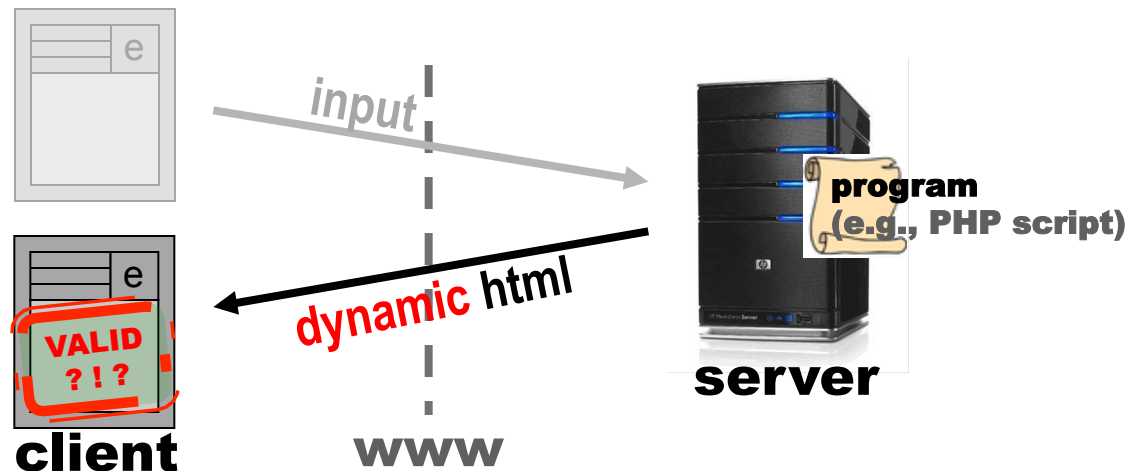
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">

<html>
  <head>
    <title>Hello!</title>
  </head>
  <body>
    <p>hello world!
  </body>
</html>
```

Validate:



■ Dynamic Validation:



Agenda



- **1) HTML FORMS**

- **2) PHP (intro)**

- **3) HTML VALIDATION**

- **4) ASSIGNMENT**

How to submit assignments

- We have created a special folder on ITU server:
 - <http://www.itu.dk/stud/e2012/DSDS/>
- You need to upload your files to your folder:
 - `W:/e2011/DSDS/username` (W-drive); or
 - `/import/stud/www/e2012/DSDS/username` (SFTP/SSH)
- You need to create a folder for each assignment **A1**, **A2**, etc. and put your final solution there:
 - `/import/stud/www/e2012/username/A1/`
 - (do not use your personal `public_html` folder)

FileZilla (FTP Client)

- FileZilla (FTP Client):
 - <http://filezilla-project.org/>
- Connect using your ITU username+password
- Hand-in folder:
</import/stud/www/e2012/DSDS/username/A1/>
<http://www.itu.dk/stud/e2012/DSDS/username/A1/>

Any questions?

A decorative graphic consisting of several horizontal lines of varying lengths and shades of gray, arranged in a stepped fashion on the right side of the slide.