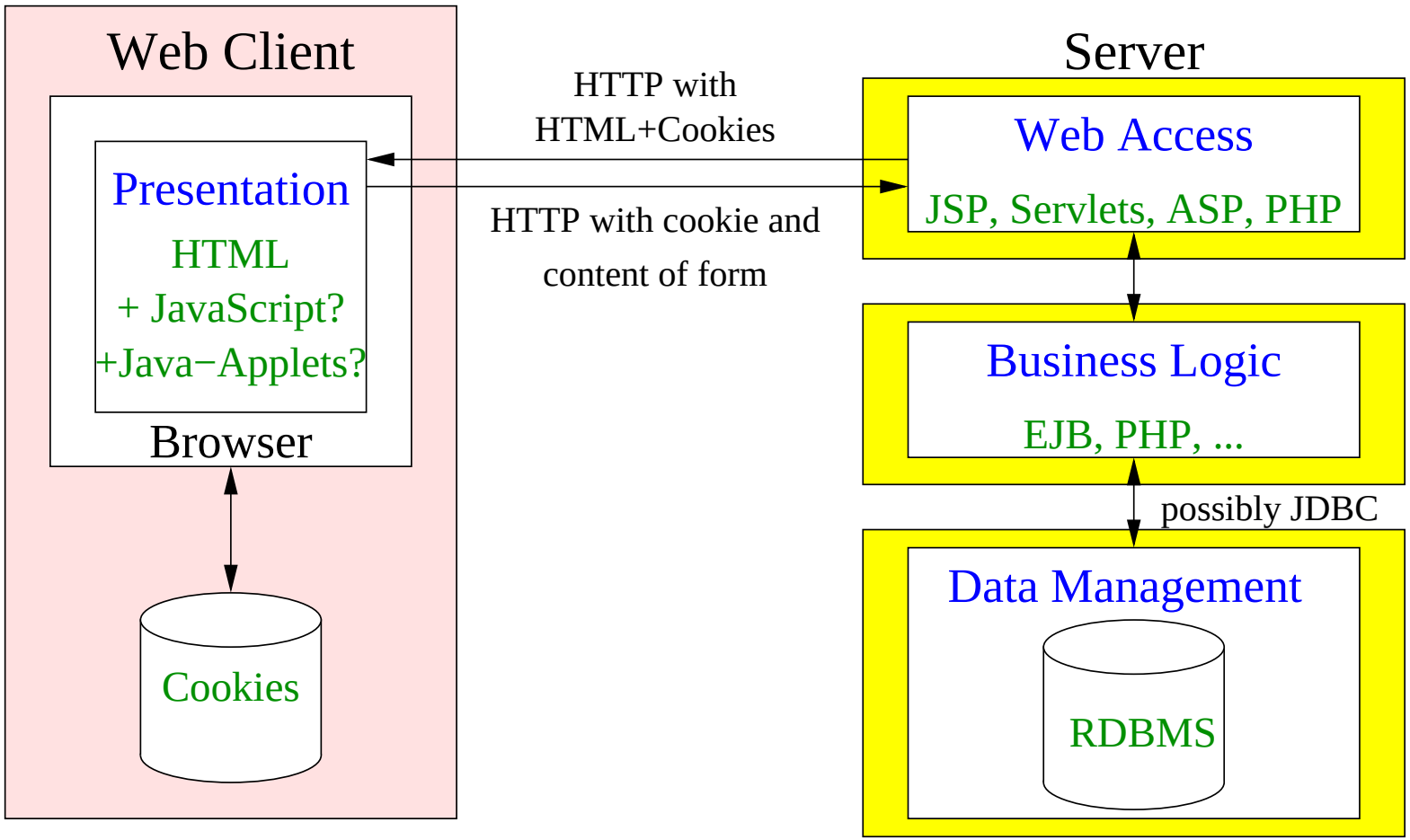
Part II

Web Applications

Web Applications



Outline

HTML

- “markup” language for describing structure ($\rightarrow$ tags), content, and (possibly) layout of a web page
- offers: text, lists, tables, links, pictures, frames, forms, . . .
- for each HTML version:
  own predefined document type definition (DTD)
  ```
  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
  ```
- structure of a HTML document:
  ASCII file with content, tags, and meta-information (among others) for search engines
- meta information: author, key words, editor, classification

The Most Important Tags

tag name	function
<HTML> ... </HTML>	declares a HTML page
<HEAD> ... </HEAD>	contains the header of the HTML page
<TITLE> ... </TITLE>	defines the title of the page
<BODY> ... </BODY>	contains the body of the HTML page
<Hn> ... </Hn>	defines a heading of level n
<B> ... </B>	sets ... in boldface
<I> ... </I>	sets ... in italics
<UL> ... </UL> and <OL> ... </OL>	delimit an unordered and ordered list, respectively
<LI> ... </LI>	list item
 	forces a line break
<P>... </P>	paragraph
<HR/>	inserts a horizontal line
<PRE> ... </PRE>	preformatted text
<IMG SRC="address"/>	loads a picture
<A HREF="URL"> ... </A>	inserts a link with description ...
<!-- ...-->	comment

see web page for examples

HTML 5

- new semantic markup. e.g. <section>, <article>, <footer>, <header>
- micro data: key-value-pairs for search engines
- new testable input-types, e.g. tel, url, number, email for forms
- client-site storage (web storage, indexed database, files, Web SQL)
- offline applications possible ($\rightarrow$ cache)
- access to (mobile) device, e.g. geolocation API

HTML 5 (continued)

- communication over websockets and cross document messaging
- multimedia (videos with WebM, OGG, H.264; Audio) without plug-ins
- 2D and 3D graphics (e.g. SVG), MathML
- web workers enable JavaScript computations in background ($\rightarrow$ multithreading)
- XMLHttpRequest Level 2 ($\rightarrow$ Ajax)
- device-dependent layout by CSS 3

Cascading Style Sheets (CSS)

- layout properties are modified by tags, e.g.

```
<H2 ALIGN="center"><FONT COLOR="red">Introduction</FONT></H2>
```

- style sheets fix layout properties for a whole document, e.g.

```
H2 text-align: center; color: red
```

- properties: color, font family, font style (e.g. italic), font size, character distance, word distance, text decoration (e.g. underline, blink), horizontal and vertical alignment and indentation, background color, ...

Ways of Using Style Sheets

1) `<LINK>` tag in header of a document

```
<LINK REL=StyleSheet HREF="style.css" TYPE="text/css">
```

2) using an `@import` element, e.g.

```
<STYLE TYPE="text/css">
<!-
@import url(http://mymachine/style.css);->
</STYLE>
```

Ways of Using Style Sheets (2)

3) embedding the styles in a document, e.g.

```
<STYLE TYPE="text/css">
<!-
  H2 text-align: center; color: red
  H3 text-align: left; color: blue
->
</STYLE>
```

4) inlining: bad, since content and layout are not separated, e.g.

```
<H2 STYLE="color: red; text-align: center">Introduction</H2>
```

- in case of contradicting layout information, the more specific one counts: tag $>$ 4) $>$ 3) $>$ 2) $>$ 1) $>$ browser setting

Forms

- enable the interactive composition of information and its transmission to a web server
- part of a web page

Tags in Forms

1) Start and End of a Form

```
<FORM ACTION="/cgi-bin/script1.pl" METHOD="GET"> ... </FORM>
```

- `ACTION`-attribute defines URL of the processing program
- `METHOD`-attribute determines method of the transmission to the server:

  `GET:` inputs are transmitted as part of the URL and accessed via environment variables `QUERY_STRING` and/or `PATH_INFO`

  `POST:` inputs transmitted in “payload” and read from standard input

Tags in Forms (2)

2) Text- and Password-Input

```
<INPUT TYPE="text" NAME="Kunde">
<INPUT TYPE="password" NAME="passwd">
```

3) Buttons

```
<INPUT TYPE="submit" VALUE="Submit">
<INPUT TYPE="image" SRC="picture.gif">
<INPUT TYPE="reset" VALUE="Reset Form">
```

4) Radio Buttons and Check Boxes

```
<INPUT TYPE="radio" NAME="Payment Method" VALUE="cash">
<INPUT TYPE="checkbox" NAME="with dinner">
```

Tags in Forms (3)

5) Menus and Scroll Lists

```
<SELECT NAME="place of residence" SIZE=1>
<OPTION>London
<OPTION>Paris
</SELECT>
```

- for SIZE $>$ 1: scroll list (possibly MULTIPLE)

6) Text Areas

```
<TEXTAREA ROWS=5 COLS=20 NAME="remark">
</TEXTAREA>
```

7) select and upload file to server

```
<INPUT TYPE="file" SIZE=20>
```

see web page for examples

Cookie

- short information, which the client assigns to a page and which it stores locally (with expiration date)
- it provides memory to the HTTP connection between browser and server
- it is included in each HTTP message to its corresponding server
- e.g.: when visiting a web site again,
  the server can recognize the user
- (normally) it is not protected during transmission

Components of a Cookie

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Yes
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	No
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No
sneaky.com	/	UserID=3627239101	31-12-12 23:59	No

- domain: generating web server ($\leq$ 20 Cookies per client and domain)
- path in file system of the server to programs, which may use the cookie
- content: (name,value) pair
- after expiration: cookie is deleted
- overwriting a cookie by an expired one, deletes it
- secure?: if yes: cookie is only propagated to an https server
- application: e.g. shopping cart, user options,
  CRM (user profile), log-in information

Outline

Client-Site Program Logic Using JavaScript

- scripting language developed by Netscape
- typically used in web pages
- syntax similar to Java; semantics is different ($\rightarrow$ types)
- JavaScript code is executed by the browser at the client site
- enables interaction between web page and user without involving the server
- major ingredient of AJAX (Asynchronous JavaScript and XML), which allows to communicate with the server without loading a new web page
- alternatives: Java applets, Flash, ...

JavaScript (continued)

- can provide a usibility similar to desktop applications
- can validate forms
- can communicate with web server
- can hide different behaviors caused by different web browsers ($\rightarrow$ jQuery)
- cannot read and write abritrary local files
- cannot communicate with other servers
- can only simulate multithreading ($\rightarrow$ web workers)
- ...