Authentication

TI1506: Web and Database Technology

Claudia Hauff
ti1506-ewi@tudelft.nl

Lecture 2 [Web], 2014/15

Authentication

So far: HTTP as **anonymous**, **stateless** request/response protocol. The same request, sent by different clients, is treated in exactly the same manner.

Now: identification via

A. HTTP headers
B. Client IP address tracking
C. User login (HTTP Basic Authentication)
D. Fat URLs

In later lectures: Cookies & Sessions.

User-related HTTP header fields

From	Request	User's email addres	mainly used by Web crawlers
User-Agent	Request	User's browser	allows customisation based on the device
Referer	Request	Page the user came	crude way to learn about user interests
Authorization	Request	Username & password	
Client-IP	Request (Extension)	Client's IP address	
X-Forwarded-For	Request (Extension)	Client's IP address	
Cookie	Request (Extension)	Server-generated ID label	

Client IP Address

- **Client IP address** as user identifier (if not in HTTP header, then via the TCP connection)
- Possible, **if** every user has a distinct IP address, a user's IP rarely changes and the Web server can determine the IP address for every request
- **Problems**:
  A. IP addresses describe the **machine**, not the user
  B. Internet service providers **dynamically assigned IP** addresses to users
  C. Users may access the Web through **firewalls** (obscures the real IP address)
  D. HTTP **proxies** and **gateways** open new TCP connections (IP of the proxy/gateway is shown), `X-Forwarded-For` might help

Not reliable, rarely used nowadays

Client IP Address

- **Client IP address** as user identifier (if not in HTTP header, then via the TCP connection)
- Possible, **if** every user has a distinct IP address, a user's IP rarely changes and the Web server can determine the IP address for every request
- **Problems**:
  A. IP addresses describe the **machine**, not the user
  B. Internet service providers **dynamically assigned IP** addresses to users
  C. Users may access the Web through **firewalls** (obscures the real IP address)

Fat URLs

```
<a href="/browse/-/229220/ref=gr_gifts/002-1145265-8016838">Gifts</a>
<a href="/wishlist/ref=gr_pl1_/002-1145265-8016838">Wish List</a>
```

- Tracking through the **generation of unique URLs** for each user
  - First time a user visits a page within a Web site, a **unique ID** is generated by the server
  - Server **redirects** client to the fat URL (recall status code 3**)
  - Server on the fly **rewrites the HTML** when an HTTP request with a fat URL is received (adds ID to all hyperlinks to maintain the knowledge)
- Independent HTTP transactions can be tied into a single "session"

Question: what is a problem of fat URLs?

Fat URLs

- Fat URLs are ugly
- Fat URLs cannot be shared (URL may not exist later on or user inadvertently shares private information)
- Fat URLs break caching mechanisms (one URL per user/page instead of one URL per page)
- Extra server load (HTML page rewrites necessary)
- Users can “escape” (ID is lost when user navigates outside the Web site, unless he bookmarks it)

HTTP basic authentication

- Server explicitly asks the user for authentication (username and password)

- HTTP has a built-in mechanism to pass user information to web sites: `WWW-Authentication` and `Authorization` headers

- HTTP is stateless: once logged in, the browser sends the login information with each request

HTTP basic authentication

Put into practice in Assignment 1

1

```
GET /index.html HTTP/1.1
host: www.microsoft.com
```

2

```
HTTP/1.1 401 Login Required
WWW-Authenticate: Basic realm="B&R"
```

security realm and authentication algorithm

Browser presents login screen

3

```
GET /index.html HTTP/1.1
host: www.microsoft.com
Authorization: Basic am910jRmdW4=
```

```
HTTP/1.1 200 OK
Content-length: 1234
Content-type: text/html
…
```

base-64 encoding

In future HTTP requests to the site, the browser automatically issues the stored username/password when asked (or even without triggering)

HTTP basic authentication

- Username and password are joined together by a colon and converted to base-64 encoding (e.g. `john:mypwd`)

- Base-64 encoding ensures that only HTTP compatible characters are entered into the message (takes as input binary, text and international character data strings)

images can be encoded this way

Normandië	Tm9ybWFuZGnDqw==
Delft	RGVsZnQ=
España	RXNwYcOxYQ==

HTTP basic authentication: secure?

- Username and password can be **decoded trivially** (sent over the network “in the clear”)

- Users tend to **reuse** login/password combinations; a non-critical web site may use basic authentication without SSL that an opponent can capture and try on critical sites

- No protection against **counterfeit servers** (that act instead of the original server)

HTTP basic authentication: overall

Basic authentication prevents **accidental** or **casual access** by curious users (privacy is desired but not essential).

Basic authentication is useful for **personalisation** and access control within a “friendly” environment (intranet).

“In the wild”, basic authentication should always be used in combination with **secure HTTP (e.g. https)** — avoids sending username/password **in the clear** across the network.

Secure HTTP

Secure HTTP

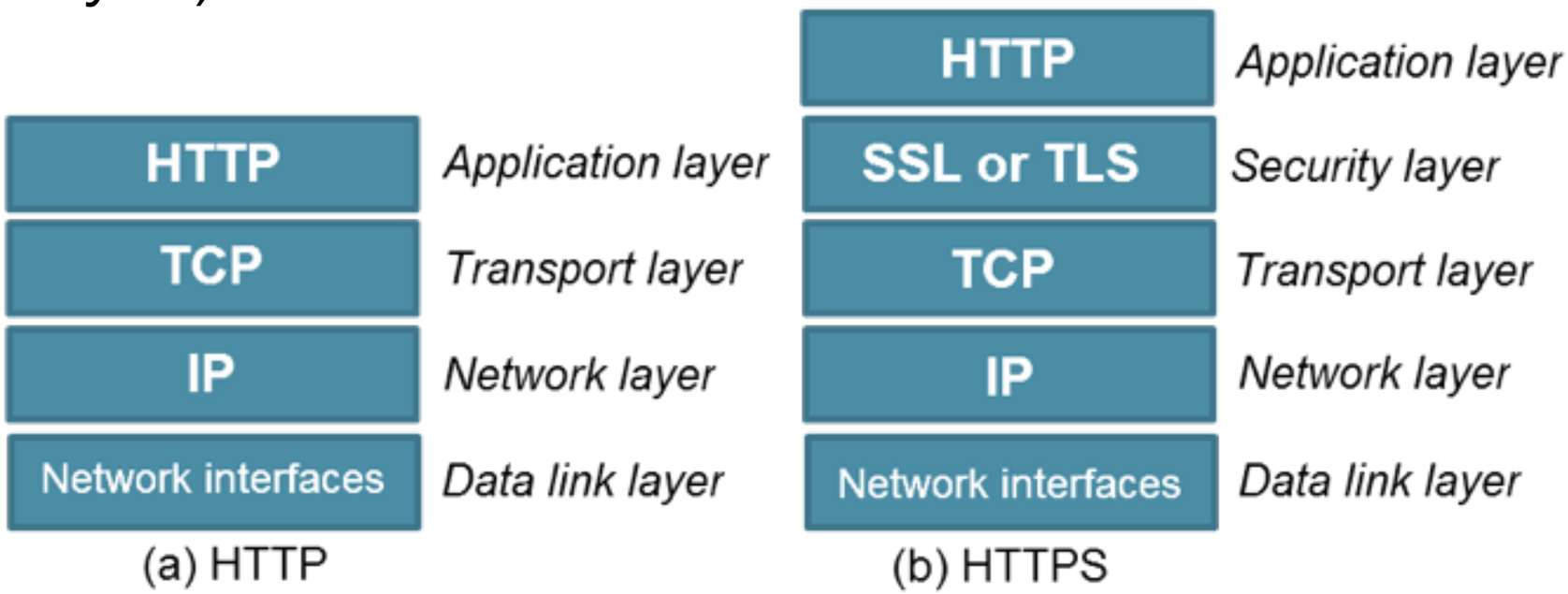
- So far: lightweight authentication
  - Not useful for purchasing, bank transactions or confidential data
- Secure HTTP should provide

A. Server authentication (client is sure to talk to the right server)
B. Client authentication (server is sure to talk to the right client)
C. Integrity (client and server are sure their data is intact)
D. Encryption
E. Efficiency
F. Adaptability (to the current state of the art in encryption)

Secure HTTP: HTTPS

- HTTPS is the most popular secure form of HTTP
- URL scheme is `https://` instead of `http://`

- Request and response data are **encrypted** before being sent across the network (SSL: Secure Socket Layer)



Client & server **negotiate** the cryptographic protocol to use.

Demo: openssl & https

- SSL is a complicated binary protocol

- **OpenSSL** is the most popular open-source toolkit to implement both SSL and TLS (Transport Layer Security) and a host of cryptographic algorithms

Pointer: http://httpbin.org/

- A **HTTP request & response service**

- A very useful site to play around with the different features of the http protocol

- You will use it in Assignment 1

Authentication summary

HTTP ensures that content

A. can be correctly identified.

B. can be unpacked properly.

C. is fresh.

D. meets the user’s needs.

E. arrives complete and untampered with.

HTTPS ensures that content

A. is transported securely.

HTML: the language of the Web

TI1506: Web and Database Technology
Claudia Hauff

Lecture 2 [Web], 2014/15

Course overview [Web]

1. http: the language of Web communication
2. **Web (app) design & HTML5**
3. JavaScript: interactions in the browser
4. node.js: JavaScript on the server
5. CSS: Lets make things pretty
6. Ajax: asynchronous JavaScript
7. Personalization: Cookies & sessions
8. Securing your application

At the end of this lecture, you should be able to …

- **Apply** Web design principles during the design stage of a Web app
- **Explain** the ideas behind usability testing and **employ** it
- **Create** basic HTML documents containing forms
- **Create** Web apps that can function offline

Lectures vs. book chapters

- Each lecture **covers a single topic**

- Each book chapter **focuses** on one topic but also covers other topics

The material introduced in the required-reading book chapters is relevant for this course.

Web sites vs. Web applications

"As Web browsers and the Web engine components that power them become ubiquitous […], developers are increasingly **using Web technologies** to build **applications** and are relying on Web engines as **application runtime environments**.

Examples of applications now commonly built using Web technologies include […] **games**, **multimedia applications, maps**, […] **interactive design applications**, and PIM (**email, calendar,** etc) systems.

W3C: Web Applications Working Group

There is more: Firefox OS

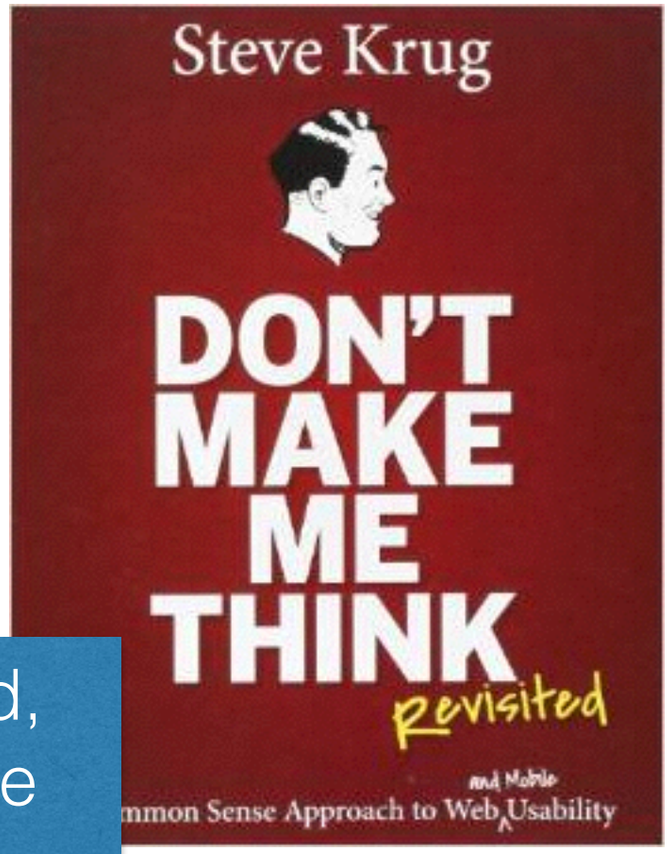
- Mozilla's open source mobile operating system based on Linux, developed for smartphones
- All hardware devices and services are accessible through **standard Web technologies**
- **Web developers use HTML/CSS/JavaScript to write apps for Firefox OS**
- Web pages can be easily converted to Web applications for Firefox OS

Web design basics

Disclaimer

- Web design is not trivial
- I am not a Web designer
- Most of you will become computer scientists, not Web designers

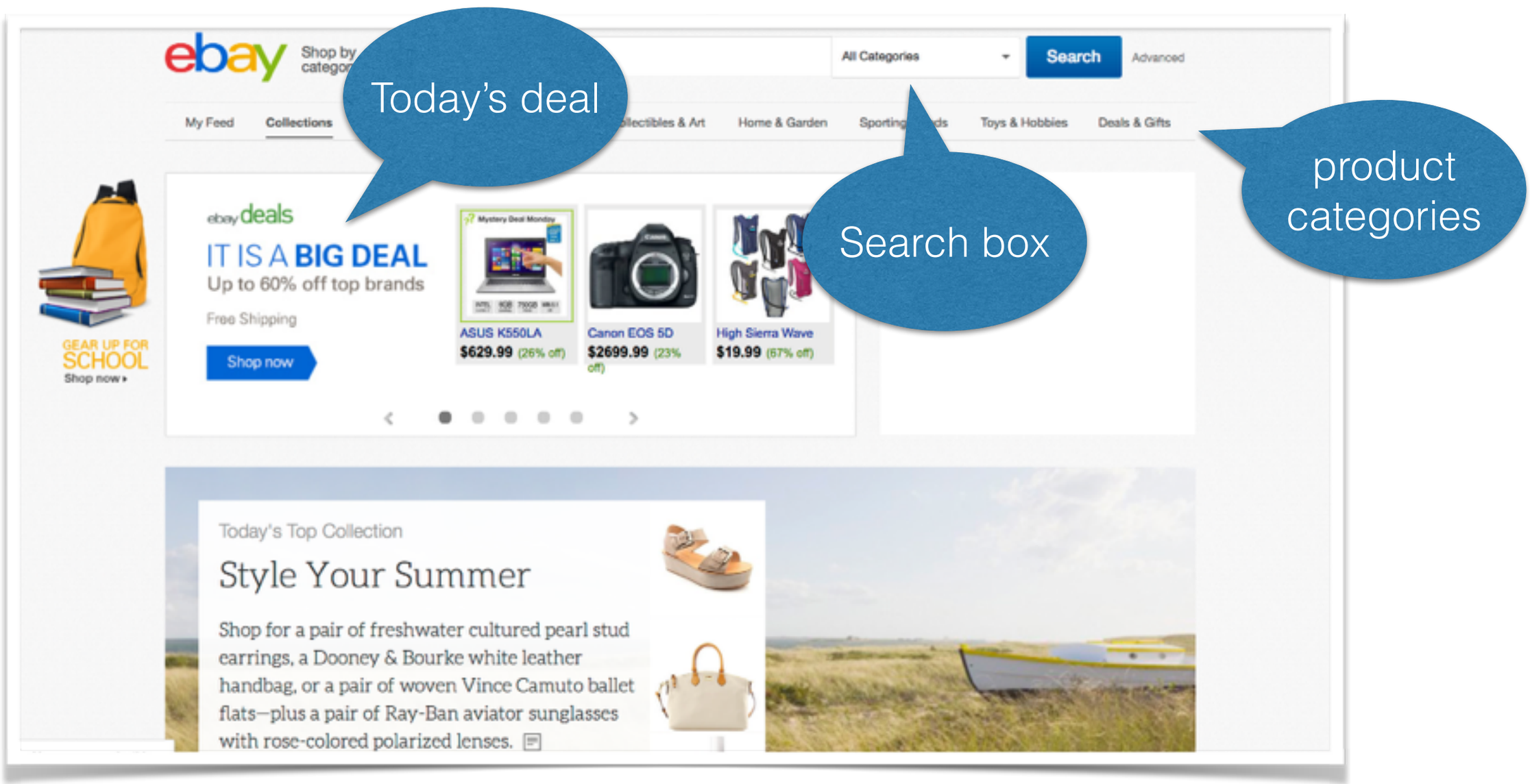
- But: a few basic rules go a long way
- Most basic principles seem obvious, but are often ignored
- We follow Steve Krug's design advice



A recommended read, available online on the campus network

Rule#1: Don't make me think

- The way a web site/app "works", should be self-evident
- Users' cognitive effort should be minimal



Rule#1: Don't make me think

- The way a web site/app "works", should be self-evident
- Users' cognitive effort should be minimal

How do I get to the products?

What if I want to look at Delft **and** Den Haag?

That is a lot of text …

Minimising cognitive effort

- Name links, buttons, sections, etc. **clearly, not cleverly**

Jobs | Vacancy offers | Join us!

- Adhere to established style standards

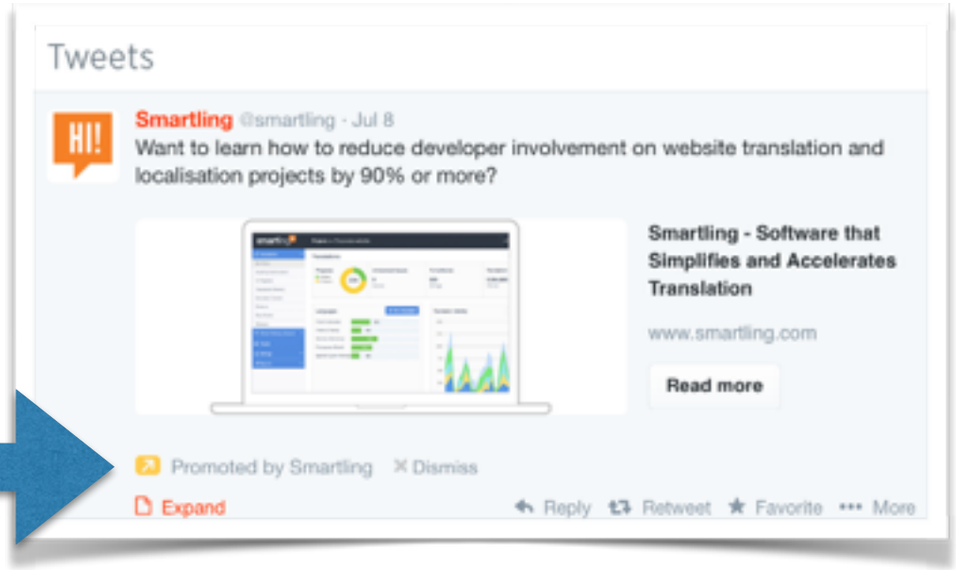
this is a link in 1999 this is a link is this a link?

- Avoid users getting lost within the web site; tell users where they are and how they arrived (leave “breadcrumbs”)

buggy ruska sprinter

Delft / Auto & Toebehoren / Oldtimers/ Klassiekers / **buggy ruska sprinter**

- Clearly divide the different parts of a site (e.g. advertisements vs. content)



Rule#2: minimize noise & clutter

Rule#3: If you cannot make it self-evident, make it self-explanatory

- Not everything can be made self-evident
- Self-explanatory sites require users to expend a **small** amount of cognitive effort
- A **small** amount of explanatory text can go a long way
- Keep the mobile user in mind (scrolling is expensive)
- Avoid “happy talk” - text without any content for the sake of adding text (e.g. welcome message)

Rule#3: If you cannot make it self-evident, make it self-explanatory

- Not everything can be made self-evident
- Self-explanatory sites require users to expend a **small** amount of cognitive effort
- A **small** amount of explanatory text can go a long way
- Keep the mobile user in mind (scrolling is expensive)
- Avoid “happy talk” - text without any content for the sake of adding text (e.g. welcome message)

Rule#3: If you cannot make it self-evident, make it self-explanatory

- Not everything can be made self-evident
- Self-explanatory sites require users to expend a **small** amount of cognitive effort

Web users: idea vs. reality

Idea: Web users are rational, attentive with a clear goal in mind

- **Reality**: Web users …
- quickly **scan** (not read!) a Web page
- often decide **within seconds** whether or not a site is worth their effort
- click on the **first link** that looks reasonable
- depend a lot on the browser's back button
- do **not read instructions** or take the time to learn how things work, instead learning-by-doing

A Web site's design should be based on user reality!
Usability testing is important to make a site a success.

Site navigation: the “trunk test”

Setup:

- Print a random page from a (your) Web site
- Find a user who has not visited the Web site before
- Hand over the printout to the user

As quickly as possible the user should circle:

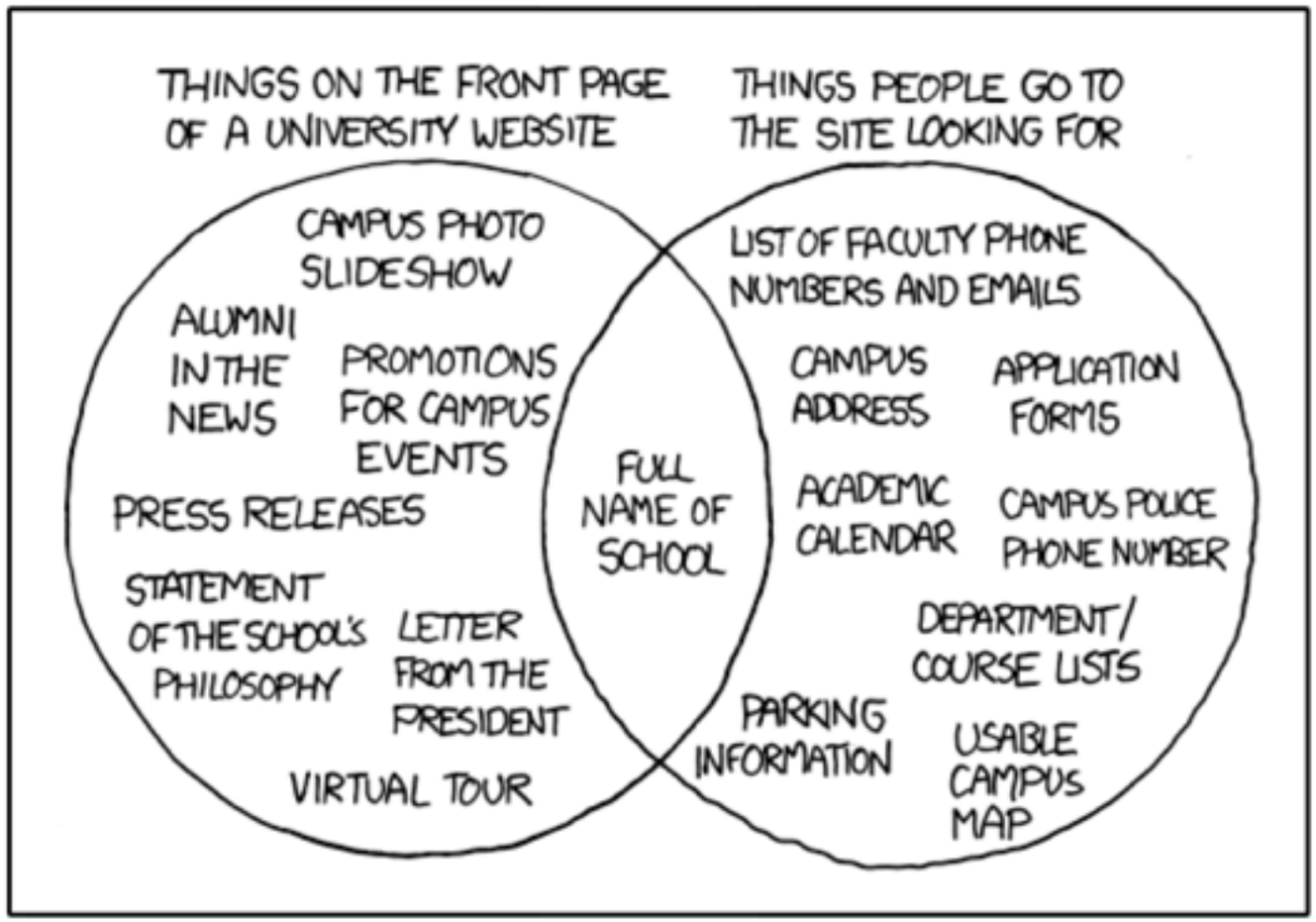
- Name of the Web site
- Name of the page she currently views
- Major sections of the page
- Possible navigation options at this point
- “You are here” indicators

The home page

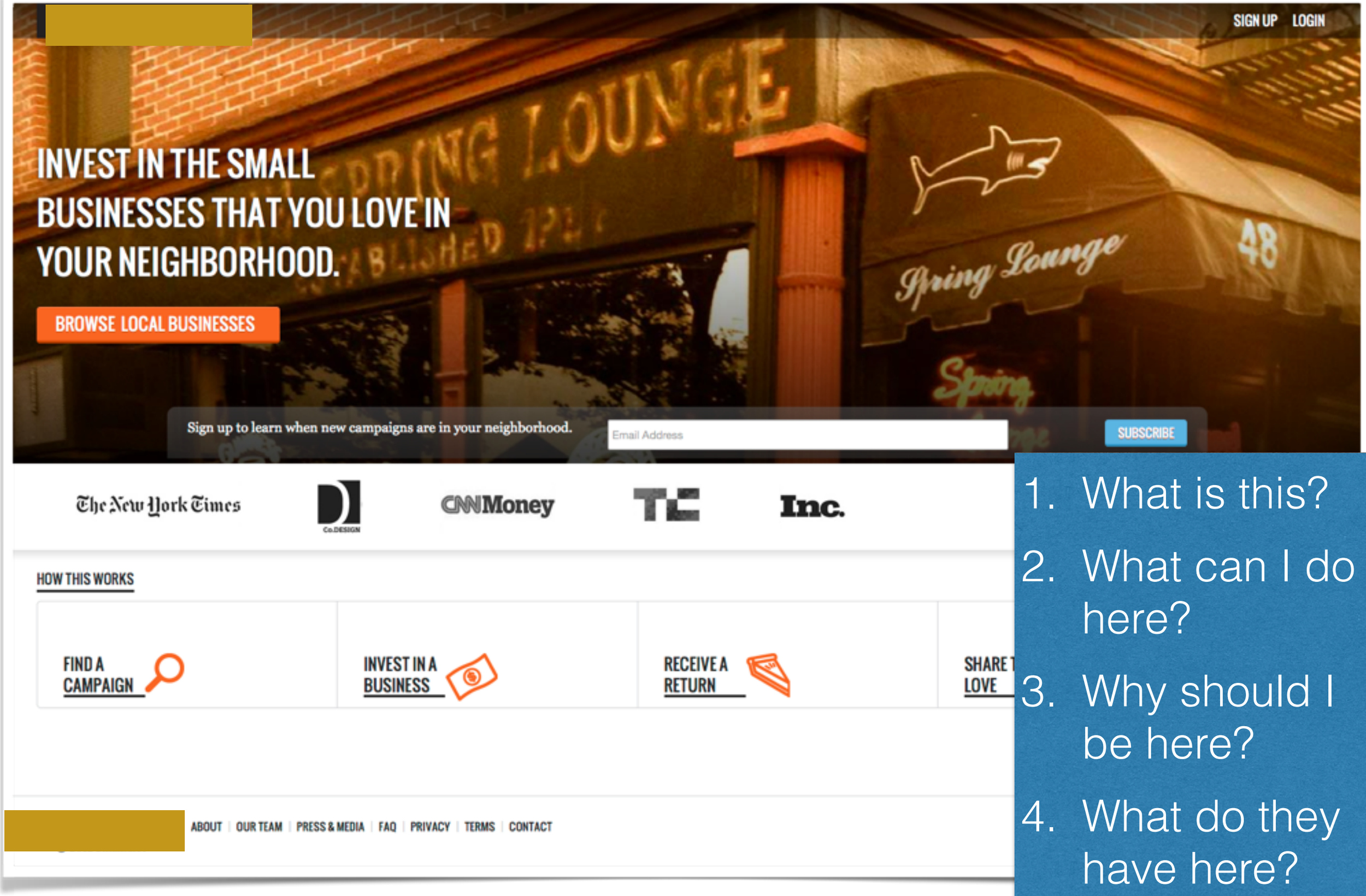
- A Web site's home page: the point of entry for most users
- Focus on the main questions:

http://xkcd.com/773/

1. What is this?
2. What can I do here?
3. Why should I be here?
4. What do they have here?



Question: why are you mainly heading to tudelft.nl ?



1. What is this?
2. What can I do here?
3. Why should I be here?
4. What do they have here?

Question: why should I be here?

SHOP LEARN LOCATIES OVER ONS LOG IN

NIEUWS

Je favoriete koffie als abonnement, nu bij Coffeecompany. **Hoe maak jij je koffie?**

FILTER KOFFIE

Jij bent van de filter, Aeropress, Chemex, Cafetière, dat werk.

ESPRESSO

Een mooie Espressomachine thuis? Toe maar! Dat verdient de lekkerste verse bonen.

Meet the Makers: La Marzocco

CC captain Leslie ging samen met brander Tom en technicus Havo langs bij de makers

How-to Chemex

Niet alleen een esthetisch fraaie zetmethode, hij maakt - met wat liefde en aandacht - ook fa...

Barista Belle bakt Banana Bread

We waren op bezoek bij de mannen van American Baking Company en kregen het

Question: what does this business do?

Rule#4: test often and early

- **Usability test:** give a user a **typical task** and observe **how well** she is able to perform it

  e.g. number of clicks, time required, number of wrong clicks

- Design - test - review cycle
- Testers should be a mix of target audience and average Web users; 2-3 testers per iteration are enough

Setup:
- **Participant** (tester) sits in front of the laptop
- **Facilitator** sits next to her and guides her through the test (keeps her focused, ensures think-aloud)
- **Observers** (developers, stakeholders, managers, etc.) watch and note usability problems

What can you test?

- How to create a user account
- How to retrieve a lost password
- How to change the credit card information
- How to delete a user account
- How to find an article in the archive
- How to edit a posting made on the forum
- ……

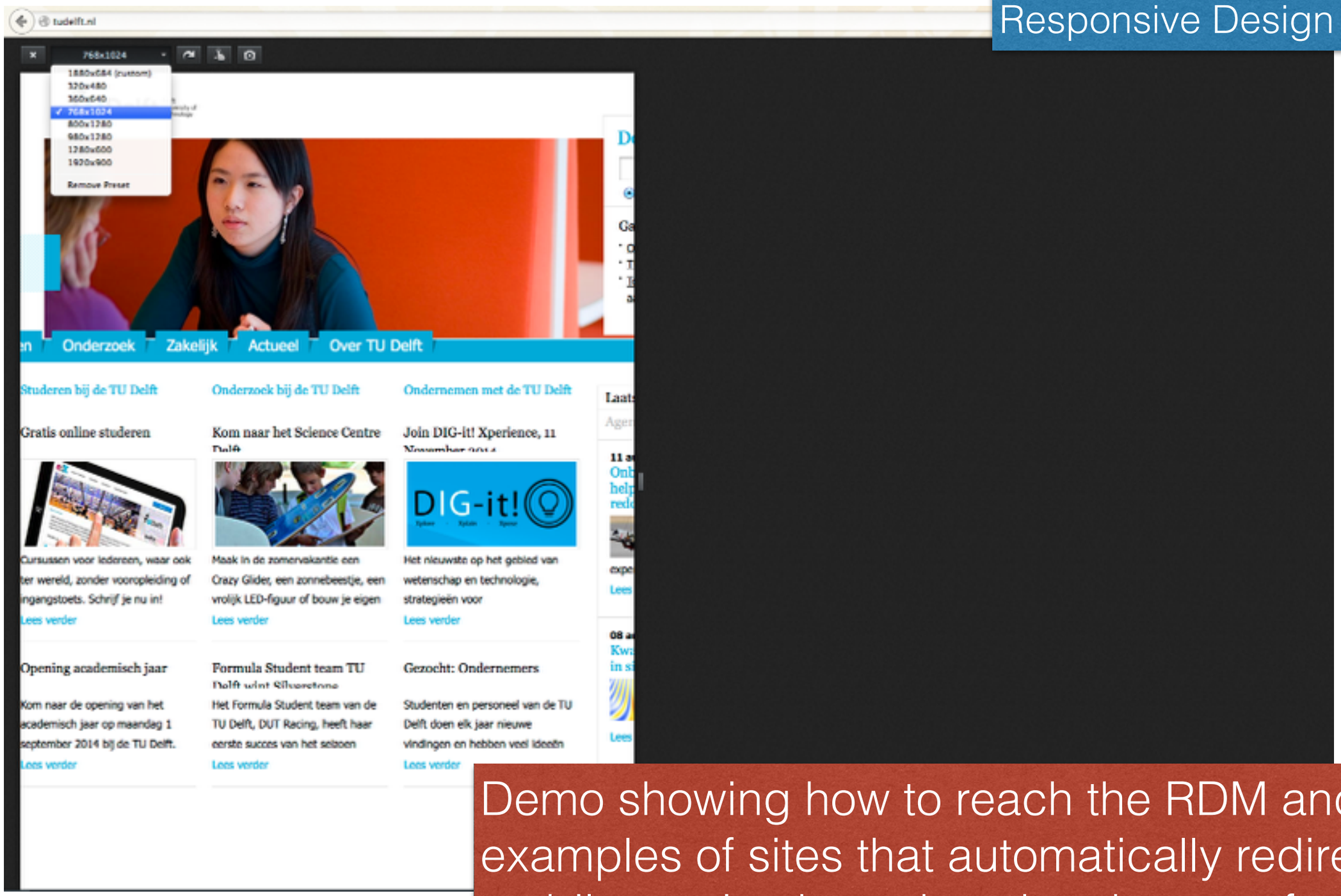
Usability testing: the aftermath

- Assign each found problem to a priority (low, medium, high)
- Focus on the high priority problems
- Create an ordered list of high priority problems and start fixing the most severe one

- Not every problem can be fixed at once
- Do not add new problems to the list until you fixed everything

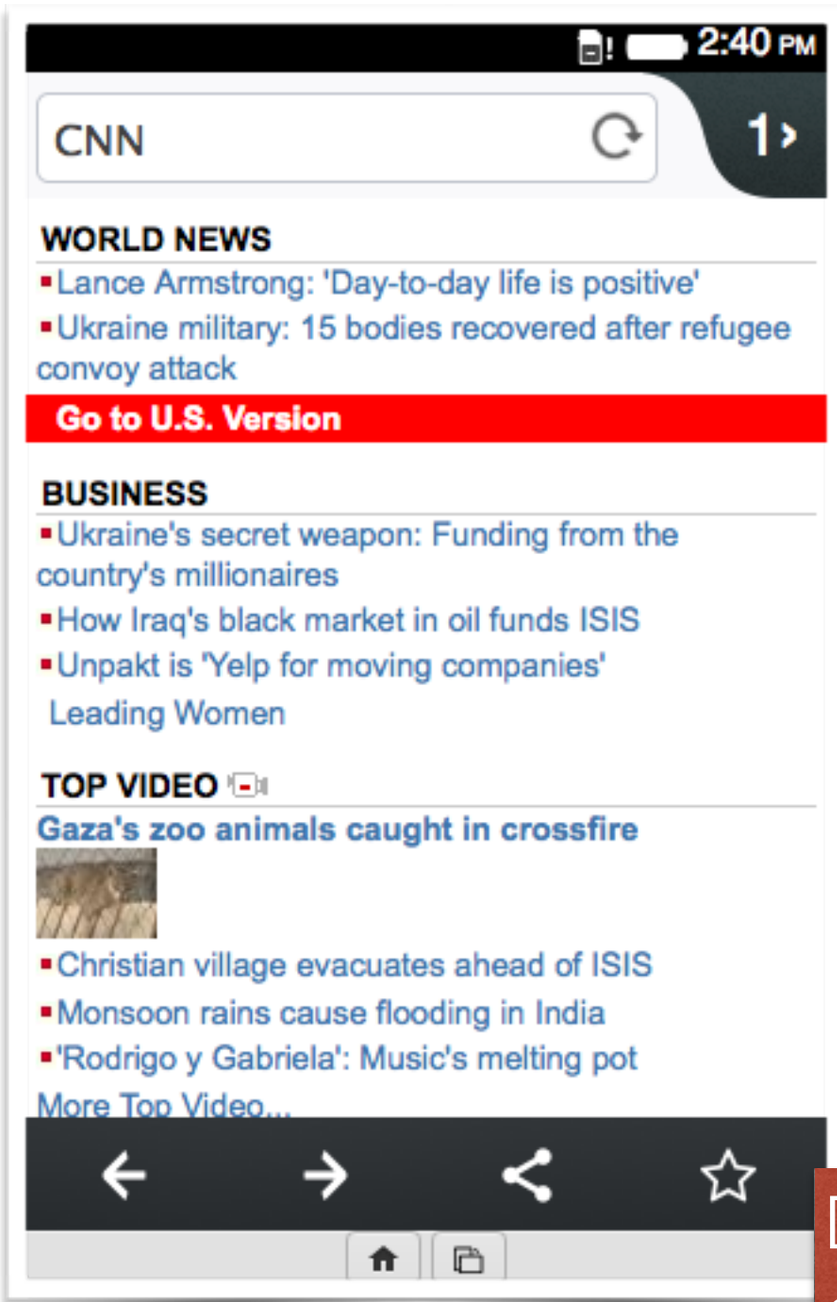
Demo: the mobile experience

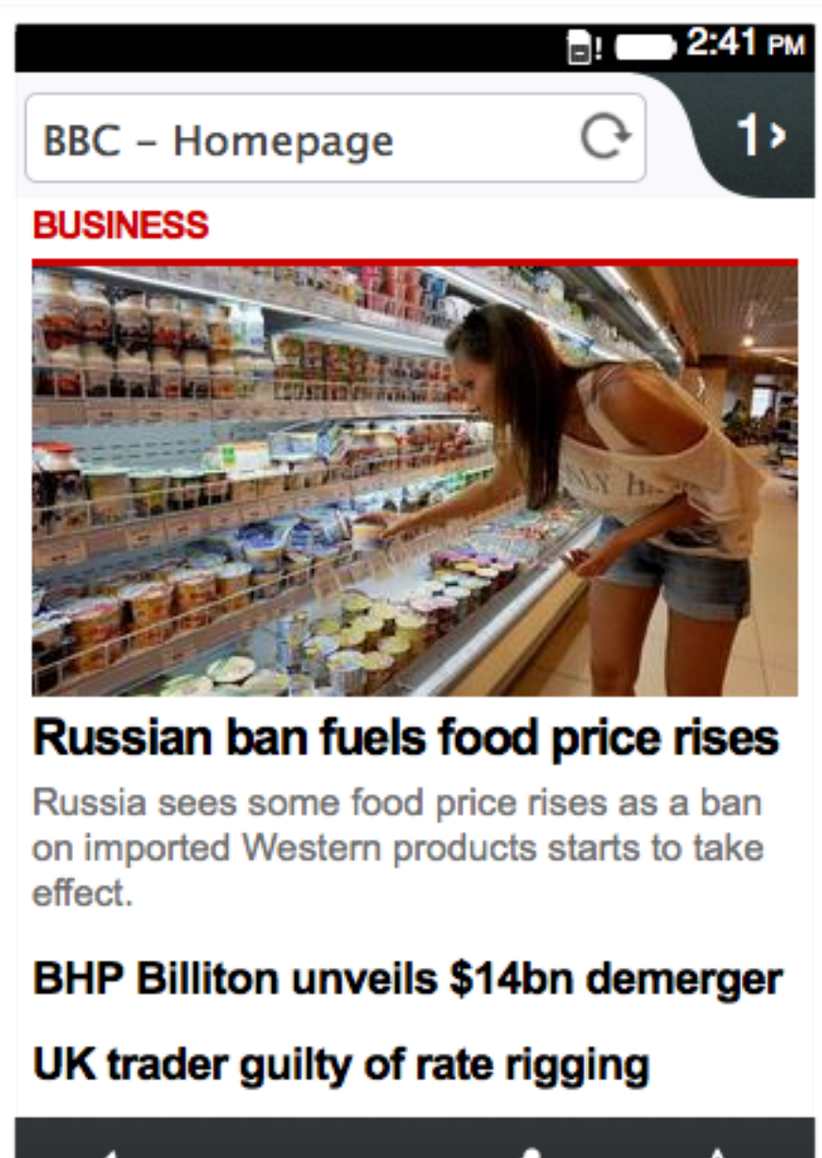
tudelft.nl viewed from the Firefox's Responsive Design Mode



Demo showing how to reach the RDM and a few examples of sites that automatically redirect to the mobile version based on the change of screen size.

Demo: the mobile experience





Demo showing how to reach the RDM and a few examples of sites that automatically redirect to the mobile version based on the change of screen size.

Designing for a mobile device

Mobile screens are small.
Users move faster and read less.

- Decide whether the mobile version should offer a subset or all features of the Web site
- Pages visited often should not be more than 1-2 taps away
- No cursor (and no hover), just touch
- Very little space for instructions — a problem for complex Web applications
- Navigation options need to be obvious or memorable

- Usability testing is similar to the standard setup

HTML5: lets take a closer look

Having read ch. 2 of the course book, you should be able to …

- Create a basic HTML5 page that validates
- Use the `img, a, ul, p, div`, … tags appropriately
- Explain the different purposes of HTML and CSS
- Identify the structure of a given HTML page
- Create a DOM tree from a given structure

Question: what does “DOM” stand for?

Chapter 2 in one slide

```
<!doctype html>
<html>
  <head>
    <title>My First Web App</title>
  </head>

  <body>
    <h1>Hello World!</h1>
    <p>Nice to meet you.</p>
  </body>
</html>
```

- An HTML document contains tags and content
- Tags are **metadata**
- Tags **structure** the content of the document

Chapter 2 in one slide

informs the browser about the HTML version

```
<!doctype html>
<html>
  <head>
    <title>My First Web App</title>
  </head>
```

header describes the document

```
  <body>
    <h1>Hello World!</h1>
    <p>Nice to meet you.</p>
  </body>
</html>
```

body contains the document content

Tags

Content

- An HTML document contains tags and content
- Tags are **metadata**
- Tags **structure** the content of th

The rendered page does not show the tags, just the content.

HTML5 overview

- A set of related technologies (core HTML5, CSS, JavaScript) that together enable rich Web content
- Core **HTML5**: mark up content
- **CSS**: control the appearance of marked-up content
- **JavaScript**: manipulate the contents of HTML documents & respond to user interactions & program for the `<canvas>` element
- Modern Web (app) development requires knowledge of all three technologies
- Before HTML5: XHTML and HTML 4.01

Not all browsers support all features.

http://caniuse.com/

The move towards HTML5

- Initial list of HTML tags (1991/92) was simple & **static**:
  `<title> <a> <isindex> <plaintext> <listing> <p> <h1> <address> <hp1> <dl> <dt> <ul>`
- JavaScript appeared in 1995, developed by Netscape - beginning of client-side **dynamic** scripting for the browser

JavaScript is not part of HTML, but HTML5 assumes JavaScript to be available.

- Plugins (e.g. Adobe Flash, 1996) were created to go beyond what is possible with HTML

HTML5: a drive to return rich content *directly* into the browser

- *Semantic* HTML became a popular wish to enable large-scale automated processing of Web content

`<div>` vs. `<footer>`

A good read: the discussion on the img tag

Who decides the HTML standard?

W3C recommendation: features are stable & implemented in multiple (2+) browsers

- HTML is widely used, making standardisation a slow process
- Many different stakeholders are part of W3C's **HTML Working Group** (Microsoft, Apple, Google, Mozilla, Nokia, Adobe, Intel, Baidu, etc.)
- HTML5: candidate recommendation in Q4-2012, **W3C recommendation in Q4-2014**
- HTML5.1: candidate recommendation in Q1-2015, **W3C recommendation in Q4-2016**
- In practice: W3C standardizes what the browser vendors have chosen to implement and agree upon

Who decides the HTML standard?

W3C recommendation: features are stable & implemented in multiple (2+) browsers

- HTML is widely used, making standardisation a slow process
- Many different stakeholders are part of W3C's **HTMI**
  Working Group (Mic
  Adobe, Intel, Baidu, e
- HTML5: candidate re
  recommendation in
- HTML5.1: candidate r
  recommendation in
- In practice: W3C stan
  have chosen to imple

W3C

HTML5

A vocabulary and associated APIs for HTML and XHTML

W3C Recommendation 28 October 2014

This Version:
http://www.w3.org/TR/2014/REC-html5-20141028/
Latest Published Version:
http://www.w3.org/TR/html5/
Latest Version of HTML:
http://www.w3.org/TR/html/
Latest Editor's Draft of HTML:
http://www.w3.org/html/wg/drafts/html/master/
Previous Version:
http://www.w3.org/TR/2014/PR-html5-20140916/
Previous Recommendation:
http://www.w3.org/TR/1999/REC-html401-19991224/

HTML5 is modular and complex

- there is more than just Core HTML5
- **Web Workers**: Web applications can spawn background workers to run processes (scripts) running in parallel to their main page
- **Web Storage**: client-side storage
- **WebSocket**: bidirectional communication with server-side processes (e.g. Rumpetroll demo)
- **WebRTC**: real-time communication between browsers (for videoconferencing, etc.)
- **HTML Media Capture**: enables user access to a device's media capture mechanism (e.g webcamtoy demo)
- …

HTML Media Capture: webcamtoy.com
WebSocket: http://rumpetroll.com/

Semantics vs. presentation

- HTML5 introduced a number of semantic HTML elements including: `<article>` `<footer>` `<header>` `<main>` `<aside>` `<section>` `<output>`
- Semantic elements provide meaning but do not force a particular presentation
- Older HTML elements (pre-HTML5) do force a particular presentation, e.g. <b> or <i>
- Heavily used HTML elements cannot easily be moved to an “obsolete” state

When creating an HTML document, always select the most specific element to represent your content.

Many HTML elements exist

- Usually, only a small number is used
- HTML5 introduced a large amount of new elements
- Many are still in an experimental state, i.e. not supported by most main browser vendors

HTML forms

Sending data to the server: via JavaScript or pure HTML. The HTML-based solution requires the use of **forms**: simple HTML elements that provide an easy way to capture data on the client and send it to the server.

HTML is best presented with HTML slides !

Today we covered

- Authentication (the end of lecture 1)
- Basic rules of Web design
- Usability testing
- HTML5 in context
- HTML forms
- The application cache

Readings

- **Required reading:**
  Chapter 2 of the course book [Web]

- Recommended:
  Dive into HTML5, Chapters 1, 5 and 8 [available online]
  The Definite Guide to HTML5, Chapters 3 and 40 [available online]

End of Lecture