

HTML: the language of the Web

 [Overview of all Lecture 2 materials](#)

Table of Contents

- [Learning goals](#)
- [Web sites vs. web applications vs. the web as a platform](#)
- [Electron](#)
- [Web design basics](#)
 - [Rule: Don't make me think](#)
 - [Rule: Minimize noise and clutter](#)
 - [Rule: If you cannot make it self-evident, make it self-explanatory](#)
 - [Expectations vs. reality: usability testing](#)
 - [Site navigation: the *trunk test*](#)
 - [Entry page checklist](#)
- [HTML5](#)
 - [The move towards HTML5](#)
 - [Who decides the HTML standard](#)
- [Self-check](#)

Learning goals

- Apply web design principles during the design stage of a web app.
- Explain the ideas behind usability testing and employ it.
- Employ HTML to create web pages.

Web sites vs. web applications vs. the web as a platform

The W3C has many working groups, among them (until 2016) the *Web Applications (WebApps) Working Group* whose [goal](#) was as follows:

As web browsers and the web engine components that power them become ubiquitous across a range of operating systems and devices, 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 reservation systems, online shopping / auction sites, games, multimedia applications, maps, enterprise-specific applications, interactive design applications, and PIM (email, calendar, etc) systems.

In 2017 this group was superseded by the [Web Platform Working Group](#) which has a similar goal (but not written up as nicely). The working group is responsible for a number of web technologies that move us closer

towards the vision of the *browser as the operating system*, including:

- client-side database and offline applications;
- file and filesystem APIs;
- WebSockets;
- Web Workers (enables web applications to spawn background processes);
- DOM & HTML;
- Canvas (for drawing);
- Web components (a component model for the web).

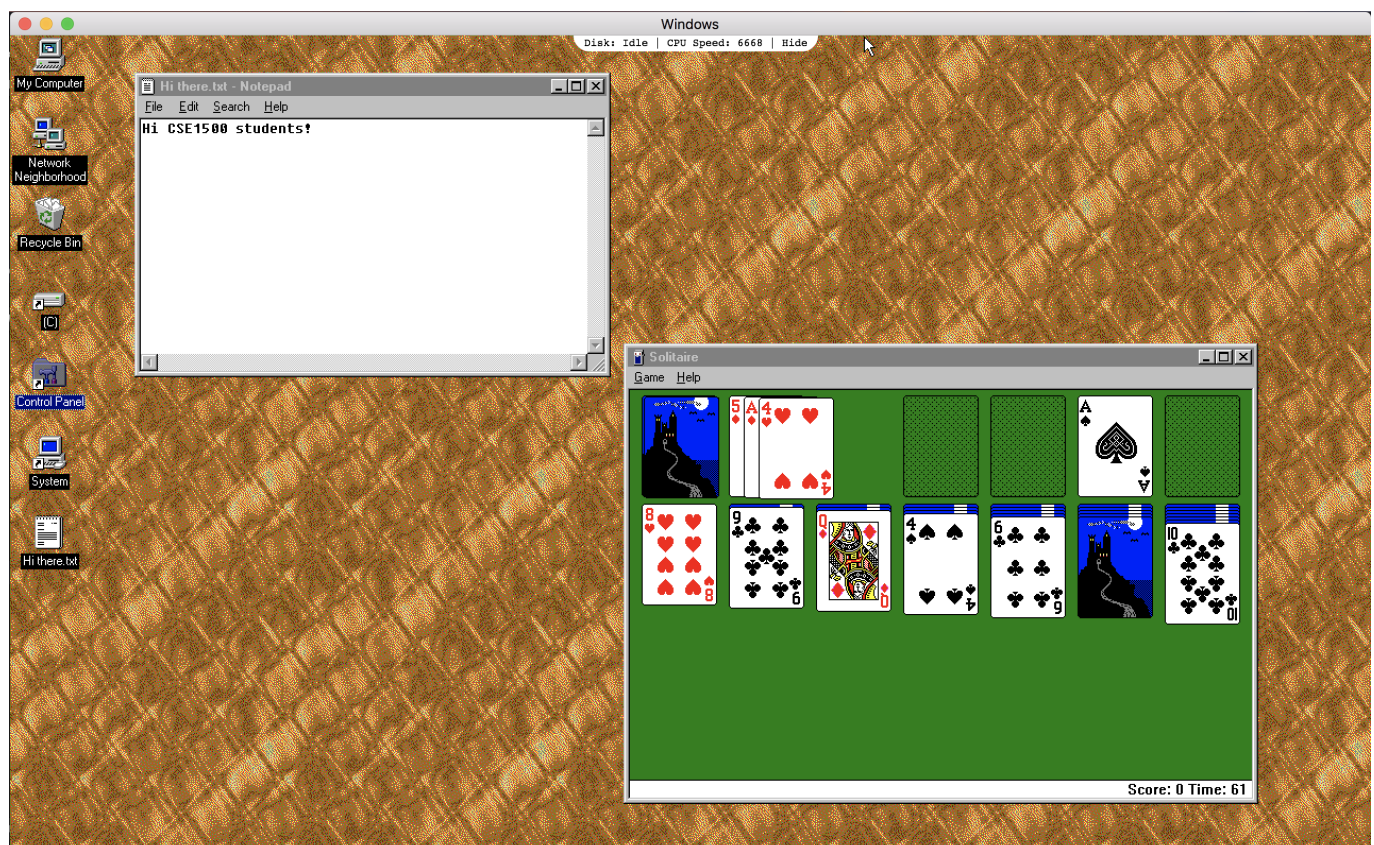
Electron

Do web technologies also help you to create desktop apps? Indeed, they do! [Electron](#) is an open-source project that enables you to build **cross-platform** desktop apps (for Windows, Mac and Linux) with HTML, JavaScript and CSS - the very technologies you learn about in this course.

Electron itself uses [Node.js](#), the server-side JavaScript runtime we cover in a later lecture together with [Chromium](#), an open-source web runtime (i.e. browser).

The major benefit of Electron should be clear: instead of writing three separate desktop variants (one for each operating system) you only have to write and maintain one. New features are integrated in one application instead of three, which reduces feature delivery time. For these reasons, many well-known applications today are built on Electron, including [Visual Studio Code](#) (the IDE we recommend you to use), the [Slack](#) app, [Atom](#) and [many, many more](#).

If you ever wanted to know how *Windows 95* looks like, there is an [Electron app for that as well](#). It looks like this:



On the downside, such a cross-platform approach usually brings with it considerable overhead - each application for instance bundles Chromium, which means that even if your app is not doing anything else besides **Hello World** it will be at least 30MB large. More concretely, the unzipped Windows95 Electron app is more than 454 MB in size, while the original Windows 95 operating system required about **40 MB of disk space**.

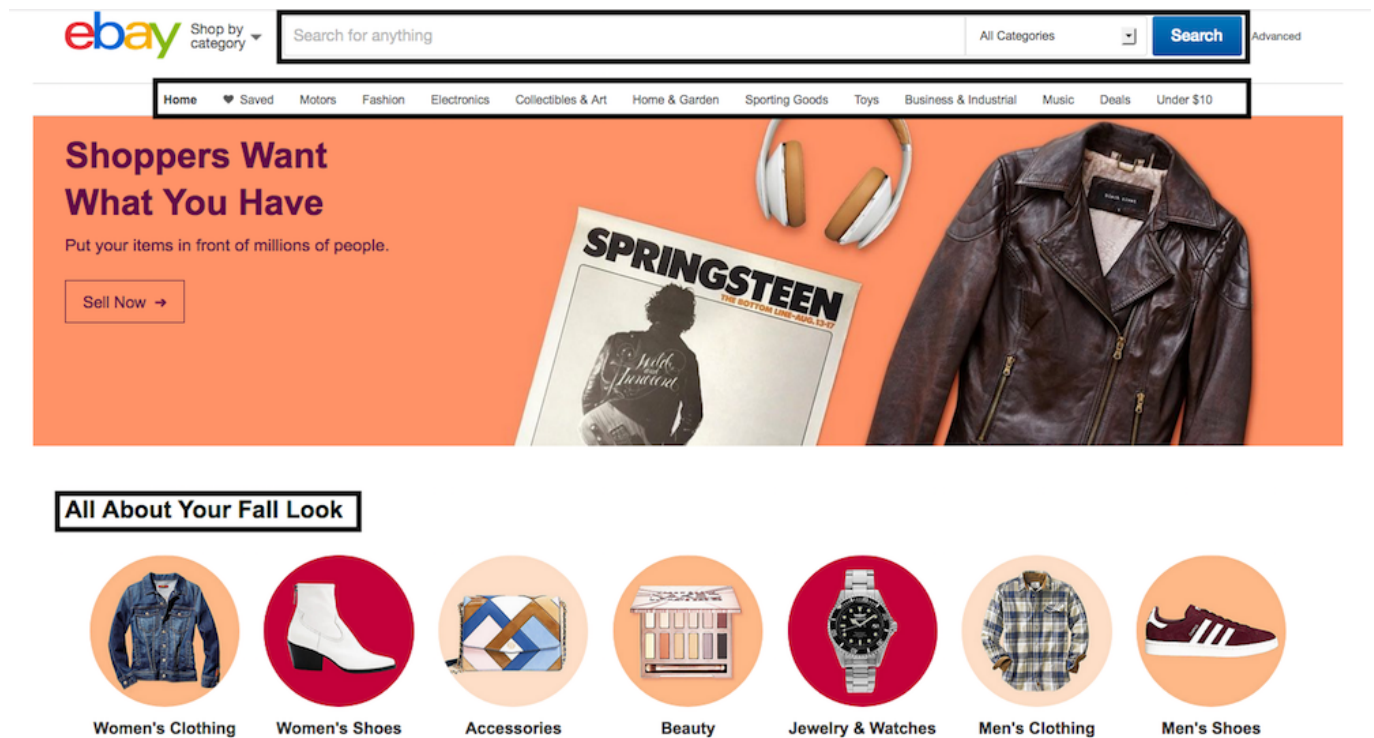
Web design basics

Web design is not trivial. However, a few basic rules go a long way. Most of these principles seem obvious, but in practice are often ignored. This lecture is based on the book **Don't Make Me Think, Revisited** by Steve Krug. It is very much worth a read. In the following sections we go over a number of Krug's rules.

Rule: Don't make me think

The way a web site or web application (I tend to use the terms interchangeably here; the rules apply to both apps and sites) works should be self-evident; the user should not have to expend **cognitive effort** to understand what she can do.

Consider this example of [ebay.com](https://www.ebay.com) 📌



Screenshot taken on September 10, 2018

📌 Here, it is very clear for the user what she can do: search for products, browse through the available items via product categories and shop for the upcoming fall season.

Contrast ebay with the following older example from [koopplein](https://www.koopplein.com) 📌



Screenshot taken February 8, 2014



It is not self-evident for the user how to act to achieve her goals and major questions are raised:

- How do I get to the offers?
- What if I want to look at offers from Delft **and** Rijswijk instead of one or the other?
- What is all this text about?

Good to know: In case you wonder how it is possible to go back in time and look at older versions of a web site (as today's [koopplein.nl presence](#) looks considerably better), head over to the [Wayback Machine](#), maintained by the Internet Archive whose mission is to **archive** the web. The Wayback Machine of course does not archive the entire web (this would be impossible), but it does take regular snapshots of more than **330 billion web pages**, including those of koopplein - [take a look](#).

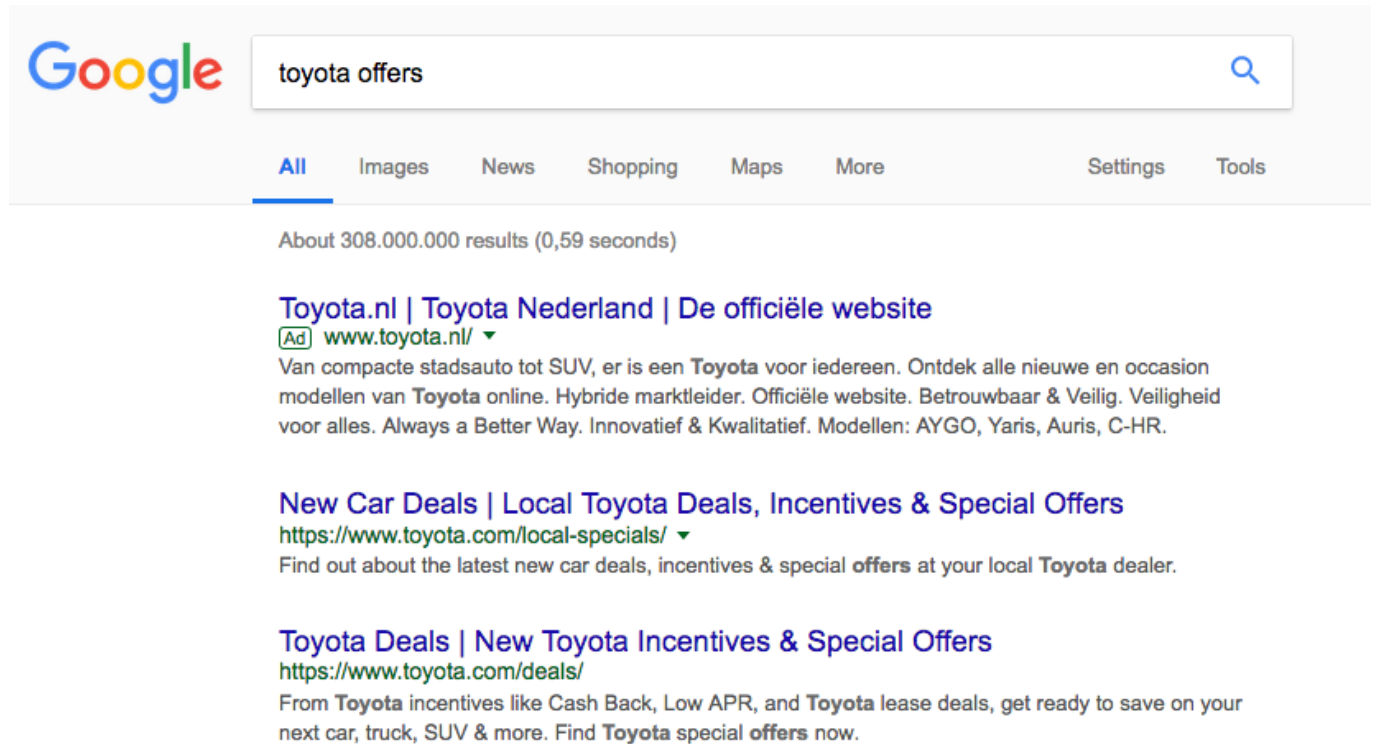
When naming and formatting links, buttons, section headers, etc. adhere to **established standards** and **be clear instead of clever**. For instance, a company's web site that has a link to its current job offers should use as link text **Jobs** or **Vacancies** (clear to the user what this link is about) instead of **Interested?** or **Join us!** (less clear).

Similarly, there are established style standards of how to format a link: in the early years of the web, blue underlined text was synonymous with a link and thus we are now stuck with the saying **10 blue links** as a synonym for web search results.


Users should also **not get lost within a web site**. A site should provide users with information on where they are and on how they arrived at that point. Ebay for instance leaves so-called **breadcrumbs**:

[Back to search results](#) | Listed in category: [Toys & Hobbies](#) > [Games](#) > [Chess](#) > [Vintage Chess](#)

Lastly, it should be easy for the user to **distinguish different parts of a site** such as advertisement vs. content. Here 📌 is an example from Google that does a poor job in this respect. It is not obvious on first sight that the first search result is indeed a paid advertisement instead of a so-called "organic" search result:



A few years ago ([August 15, 2013](#) to be precise) Google was a lot more forthcoming when it came to notifying its users about advertisements 📌

Google 

Search About 640,000,000 results

Web

Images

Videos

News

Shopping

More

Show search tools

Ads

[Toyota.com - New Toyota Incentives](#)
www.toyota.com/Deals
 Request a Quote, Explore Incentives at the Nationwide Clearance Event.

Local Specials Locate a Dealer
 Build & Price Request a Quote

[2013 Toyota - New Toyota Is Here](#)
shop.autonationtoyotanorthington.com/
 Browse And Get Our Internet Price.

[Toyota 1/2 Toyota - JeffHunterToyota.com](#)
www.jeffhuntertoyota.com/
 Find Photos, Specs & More Options Online. Find an Authorized Dealer!

[Toyota Cars, Trucks, SUVs & Hybrids | Toyota Official Site.](#)
www.toyota.com/ - Cached - Similar
 Explore the newest **Toyota** trucks, cars, SUVs, hybrids and minivans. See photos, compare models, get tips, calculate payments, and more.

[Current Toyota Vehicles](#)
 Find your perfect 2013 Toyota vehicle.
 Narrow current Toyota models down by ...

[Find a Toyota Dealer](#)
 Finding a Toyota dealership is easy.
 Our dealer locator provides the most ...

[Build](#)
 Build a custom Toyota vehicle package. Choose the model, select the engine, ...

[More results from toyota.com](#)


Ads


[Toyota Dallas](#)
www.cowboytoyota.com/
 Wide Selection of New Toyotas.
 Multiple Models. Shop Now. Dallas.


[2013 Toyota Clearance](#)
toyota-clearance-sale.autosite.com/
 Massive August **Toyota** Sale Now!
 Get Our Lowest **Toyota** Price & Save

[Toyota](#)
www.freemantoyota.com/
 2012 Tundra On Our Lot Now.
 Call Us For Info and Visit Today.


[2013 1/2 Toyota Closeout](#)
www.toyota.car.com/
 Special Aug. 2013 **Toyota** Discounts!
 Be Smart, Get Deals & Pay Less


It should be mentioned though that Google is not the only offender here, take this example  from Twitter which is similarly poorly designed in terms of content distinction between organic tweets and promoted (i.e. paid) ones:





See 2 new Tweets

 Pablo Retweeted


2Immerse @2Immerse · 1h

We're at #IBC2018 from Thursday with our object based media projects - do come and see us w @cwi_dis @IRTpresse @ChyronHego @btsport

IBC2018, RAI Am:





The world's mo:


media, entertain

technology sho


IBC launch for 2-IMMERSE open source software

2immerse.eu


 2
 






Chrom & Mass Spec @ChromSolutions · Sep 5


Separation Science, in collaboration with Thermo Fisher Scientific, has developed a learning hub focused on analytical solutions for biopharmaceutical characterization. Simply complete the form below to access the site. bit.ly/2wNqhla



SeparationScience

BioPharmaceutical Characterization Resource Center




 2
 

 Promoted

Rule: Minimize noise and clutter

The rule does not have to be explained, here are two examples that should make things clear 📌

ZDNet crawled on [March 31, 2001](#):


Where Technology Takes You

- Most Popular Products
- Find A Digital Camera
- Free Downloads

[Free Downloads](#) • [Product Reviews](#) • [Buying Guides](#) • [Faster Downloads](#) • [Free Updates](#) • [Hi-Tech Jobs](#)
[Smart Shopping](#) • [Domain Names](#) • [Top Notebooks](#) • [IT Resources](#) • [PC Check-Up](#) • [ZDNet onebox](#)

Search For:

[Search Tips](#)
[Power Search](#)

BREAKING NEWS

Mar 31, 2001 10:35 AM PT

- ▶ [Dell prods PC sales with freebies](#)
- ▶ [MarchFirst assets picked up by Divine](#)
- ▶ [New head for Sun storage unit](#)
- ▶ [Hackers' new craze: Worms](#)
- ▶ [Lawmakers seek domain pact review](#)
- ▶ [MS warns of IE security hole](#)
- ▶ [AnchorDesk: I'm not always right! Check out my worst tech predictions](#)
- ▶ [More news headlines...](#)

HOT PRODUCTS

Play that funky music!

Check out these portable MP3 players! See our five most popular picks and find a favorite of your own.

- 1. SONICblue Rio Volt**
[Read review](#) | [Check prices](#)
- 2. Intel Pocket Concert**
[Read review](#) | [Check prices](#)
- 3. Creative Labs Nomad II**
[Read review](#) | [Check prices](#)
- 4. Archos Jukebox 6000**
[Check prices](#)
- 5. SONICblue Rio 800**
[Read review](#) | [Check prices](#)

▶ [More product reviews...](#)


INTER@CTIVE INVESTOR

[My portfolio](#) | [Lookup Ticker](#)

SOUND OFF!

Windows 98 woes?

What do you do when your



Get great speakers for gaming

These speakers aren't solely designed for gaming, but their powerful sound will add depth to every punch, crash, and engine rev. [See them at ZDNet Reviews](#)

Reviews

[PCs](#), [Notebooks](#), [Cameras](#), [Handhelds](#), [Digital Audio](#)...

Shopping

[Hardware](#), [Software](#), [Auctions](#), [Computer Shopper](#), [e-centives](#)...

Business & Tech

[IT Resources](#), [Tech Jobs](#), [E-Commerce](#), [Small Biz](#)...

Help & How-To

[Viruses](#), [Fix It](#), [Experts](#), [How-To](#), [Free Updates](#), [Online Classes](#)...

Tech News

[Page One](#), [AnchorDesk](#), [Alerts](#), [Computing](#), [Rumors](#)...

Investing

[Stock Quotes](#), [Top Tech IPOs](#), [Headline News](#), [Portfolio](#)...

GameSpot

[Dreamcast](#), [PC](#), [PlayStation](#), [N64](#), [Previews](#), [Hints](#)...

Linux Center

[News](#), [Downloads](#), [Devices](#), [How-To's](#), [Hardware](#), [More](#)...

Downloads

[Free Software](#), [Top 20](#), [Games](#), [Exclusives](#), [Screensavers](#)...

Developer


[HTML](#), [Java](#), [Free Scripts](#), [Web Graphics](#), [Usability](#)...

Electronics

[Cameras](#), [TVs](#), [Digital Music](#), [Audio](#), [Theater](#), [Phones](#), [Gear](#)...

MyZDNet

[Free E-Mail](#), [eCircles](#), [Calendar](#), [Discussions](#), [ZDNet Rewards](#)...



TODAY ON ZDNET

Build your own Web site

What would you do with 15MB of free space on the Net? Find out how fast and easy it is to get set up online -- and it's free, too!
[Visit ZDNet SiteBuilder](#)

PC pranks for April fools

Play jokes on friends and coworkers from the comfort of your desk. Check out the fun you can have with e-mail tricks, computer sounds, and more. [Get the files at ZDNet Downloads](#)

Exclusive! Preview Windows XP

Check out this live, interactive preview of both the desktop and server versions of Microsoft's next-generation operating system. [Get details at ZDNet Business & Tech](#)

vs. ZDNet crawled on August 31, 2014 📌

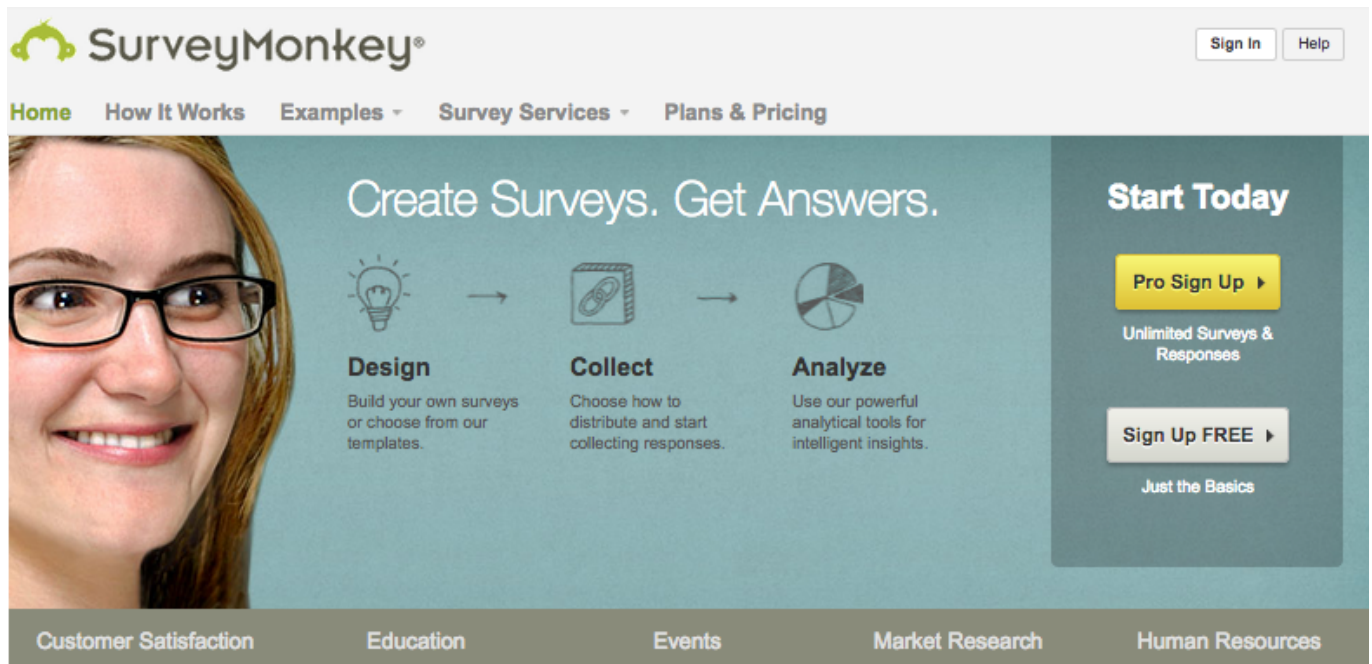


While it is not hard to go back to very old web designs and find faults in them, it should also be pointed out that in those times, every single HTTP request/response pair was time-consuming (the Internet was slow) and expensive. It made sense to push as much content as possible into a single web page which could then be sent to the client in a single HTTP response.

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

Self-explanatory sites require users to expend a small amount of cognitive effort. When a site is not self-explanatory, a small amount of explanatory text can go a long way. In today's mobile world where a lot of content is accessed *mobile first*, it is also vital to keep the mobile user in mind as well who has to deal with a small screen, a touch-based interface and possibly many distractions while surfing the web.

A positive example of this rule is the following SurveyMonkey ([December 1, 2013](#)) splash screen 📌



Lastly, **avoid happy talk**, that is text without any content for the sake of adding some text (e.g. a welcome message).

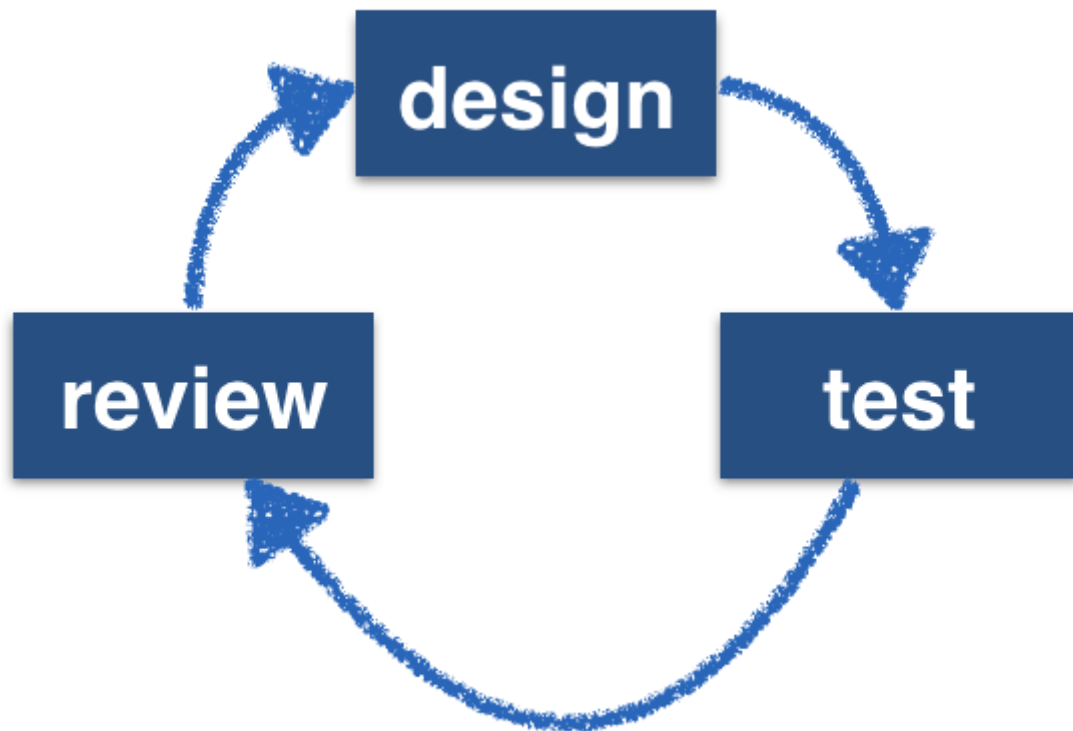
Expectations vs. reality: usability testing

Our expectations of web users are often not grounded in reality. We may expect users to be **rational**, **attentive** and having a **clear goal** in mind.

Instead, the average user:

- quickly scans a web page (not even reading it);
- decides within seconds whether or not a site is worth their attention (research on so-called *dwell time* has shown this [again](#) and [again](#));
- clicks on the first link they find;
- depends a lot on the browser's back button (and not all web applications can deal with it);
- does not read instructions.

A web application should be designed based on **user reality**. **Usability testing** is an important step to create a well-designed web application. The development cycle consists of *designing-testing-reviewing*:



In a **usability test**, a user is given a **typical task**, such as:

- Create a user account.
- Retrieve a lost password.
- Change the current credit card information.
- Delete a user account.
- Find an article in the archive;.
- Edit a posting made in a forum.
- Start a game with three players.

While the user is busy with the task, her actions towards completing the task are being recorded. These actions are then translated into performance metrics. Performance metrics depend on the task, it could be the **number of clicks** required to complete the task, the **time taken** or the **number of wrongly clicked elements**.

Usability testers should be a mix of target audience and average web users; 2-3 testers per iteration tend to be sufficient.

A typical usability setup has the following roles:

- The **participant** (our tester) sits in front of the device (laptop, mobile phone, tablet).
- The **facilitator** sits next to her and guides her through the test.
- The **observers** (developers of the app, managers, etc.) watch the usability test and discuss the tester's performance (and how to improve it) afterwards.

This [blog post](#) provides a good practical overview of usability testing with mobile devices, and includes imagery of the [participant and facilitator setup](#) and the [observer setup](#).

The result of a usability test are a set of issues. Each of those issues should be assigned a **priority** (low, medium, high) and the next iteration of the development should focus on the high priority problems. No new issues should be added to the list until the most severe issues are fixed.

Site navigation: the *trunk test*

In order to determine whether a web site's navigation scheme is useful, Krug developed the so-called *trunk test*. Given a web site, pick a random page in it, print it and give it to a user who has never seen the site.

As quickly as possible, the user should find:

- the name of the web site;
- the name of the page she currently views;
- major sections of the page;
- possible navigation options at this point; and
- *You are here* indicators.

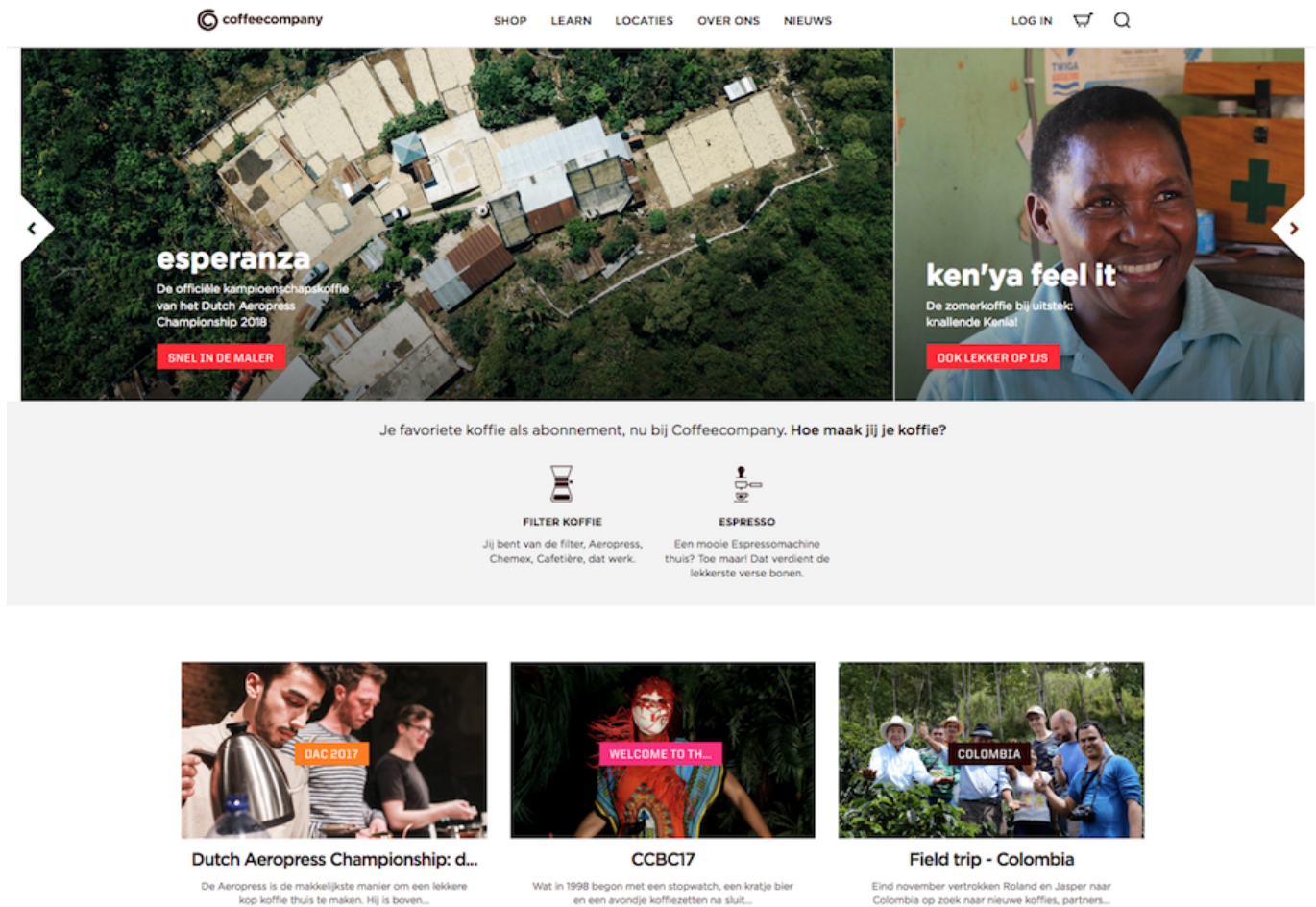
If the user is not able to find those elements, the site navigation scheme is considered sub-optimal.

Entry page checklist

The home page (or entry page) of a web application should answer a number of essential questions:

- What **is** this?
- What can I **do** here?
- Why **should** I be here?
- What do they **have** here?

Surprisingly many home pages are not able to answer these questions. Take for example this entry page:



Screenshot taken October 18, 2018

What is the core business of this company? This is the homepage of a Dutch cafe chain:
<https://coffeecompany.nl/> - not something easily guessable from the entry page.

Another serial offender of the entry page checklist are university home pages as immortalized in [this xkcd comic](#).

HTML5

HTML5 is 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;
- client-side **JavaScript**: manipulate the contents of HTML documents and respond to user interactions.

Modern web application development requires knowledge of all three technologies. In practice, it also requires a whole set of additional frameworks and tools to go from prototype code to production code, such as build tools, transpilers, code coverage tools and so on. Even for [frontend coding](#) alone. We will introduce a few of those tools throughout the practical assignments.

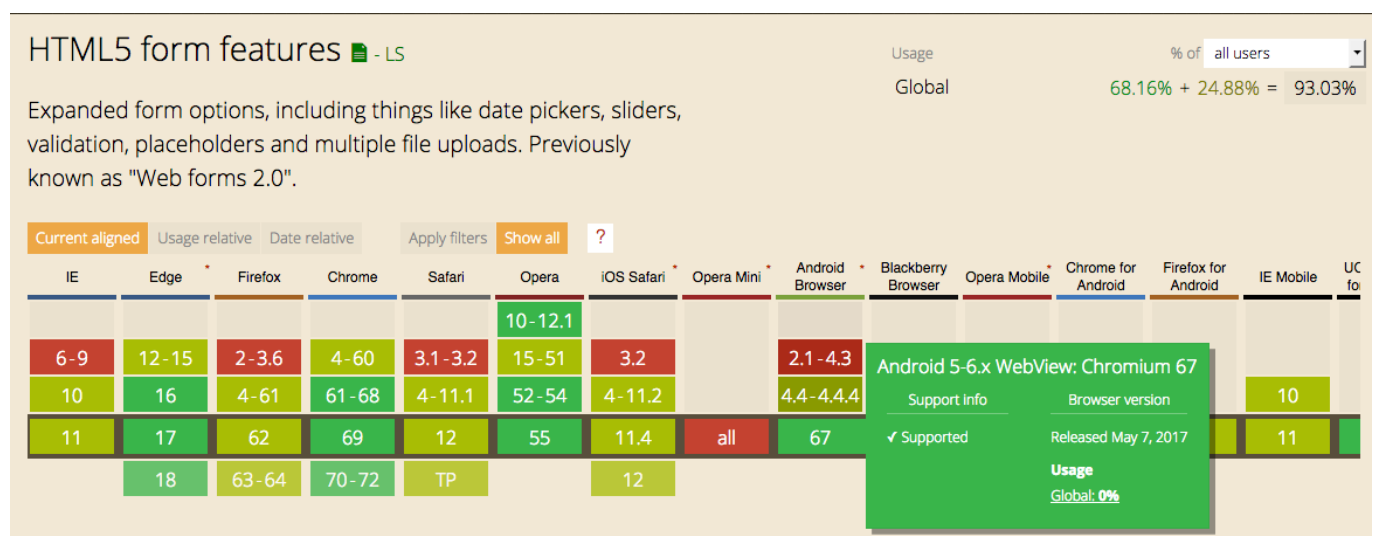
Before HTML5 we had **XHTML** and HTML 4.01. XHTML is a reformulation of HTML 4 as an XML 1.0 application and stands for **Extensible HyperText Markup Language**. It looks as follows (taken straight from the [W3C XHTML recommendation](#)):

```
<?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" xml:lang="en" lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://example.org/">example.org</a>.</p>
  </body>
</html>
```

XHTML was designed to make processing of web pages **easier for machines** by having a very strict set of rules. The problem though was that (X)HTML is written by developers, not machines and it turned out to be too much hassle to write valid XHTML. Moreover, browsers were and are able to render invalid XHTML pages properly (so why even try to write valid XHTML?) and thus XHTML was eventually abandoned in favor of HTML5, which is not only less strict but also added a host of new features to the language.

With this introduction of new features **browser compatibility** issues returned: some browser vendors are faster than others in implementing W3C standards (in addition to implementing their own non-standardized features). A good resource to check which browser versions support which HTML5 feature and to what extent is <https://caniuse.com/>.

As a concrete example, here is the browser support overview of HTML5 form features as provided by [caniuse](https://caniuse.com/):



We can thus easily find out which browser versions do (not) support these features or only partially support them. This enables application developers to make choices - depending on their target population (and the most popular browsers among the target population), certain HTML5 features should (not) be employed.

The move towards HTML5

The initial list of HTML tags (1991/92) was **static**: `<title>` `<a>` `<isindex>` `<plaintext>` `<listing>` `<p>` `<h1>` `<address>` `<hp1>` `<dl>` `<dt>` ``. JavaScript was created within 10 days (*which explains many JavaScript quirks*) in May 1995 by [Brendan Eich](#), a co-founder of Mozilla who today is behind the browser

Brave. At the time, Eich worked at Netscape, which offered Netscape Navigator, a dominant browser in those years. It was the beginning of client-side **dynamic** scripting for the browser.

Plugins were created to go beyond what at the time was possible with HTML. Probably the most famous plugin remains Adobe Flash, which was introduced in 1996. HTML5 is a drive to return rich content **directly** into the browser, without the need for plugins or addons.

HTML5 introduced a number of **semantic HTML elements** including `<article>` `<footer>` `<header>` `<main>` `<aside>` `<section>` `<output>`. As a guideline, when creating an HTML document, it is always best to select the **most specific** element to represent your content (instead of only using `<div>`'s). Semantic elements provide **meaning** but do not force a particular presentation. Older HTML elements (pre-HTML5) often do force a particular presentation, e.g. `` or `<i>`. At the same time, those heavily used HTML elements cannot be moved to an obsolete state - as this would inevitably break a large portion of the web. For the browser vendors, backwards compatibility is a necessity, not an option. It should be pointed out that **semantic HTML** is quite different from the grand vision of the [Semantic Web](#):

The Semantic Web is a Web of data – of dates and titles and part numbers and chemical properties and any other data one might conceive of.

Who decides the HTML standard

HTML is widely used, which makes standardisation a slow process. Many different stakeholders are part of W3C's [Web Platform Working Group](#) (Microsoft, Google, Mozilla, Nokia, Baidu, Yandex, etc.). The standardization process of the W3C is elaborate, as a wide variety of stakeholders have to build consensus. Confusingly, a **W3C recommendation** is the highest level of standardization possible, before achieving it, a number of steps leading up to the recommendation are required:

1. **Working Draft:** *a document that W3C has published for review by the community, including W3C Members, the public, and other technical organizations.*
2. **Candidate Recommendation:** *a document that W3C believes has been widely reviewed and satisfies the Working Group's technical requirements. W3C publishes a Candidate Recommendation to gather implementation experience.*
3. **Proposed Recommendation:** *a mature technical report that, after wide review for technical soundness and implementability, W3C has sent to the W3C Advisory Committee for final endorsement.*
4. **W3C Recommendation:** *a specification or set of guidelines that, after extensive consensus-building, has received the endorsement of W3C Members and the Director. W3C recommends the wide deployment of its Recommendations. Note: W3C Recommendations are similar to the standards published by other organizations.*

Source: [W3C Recommendation Track Process](#).

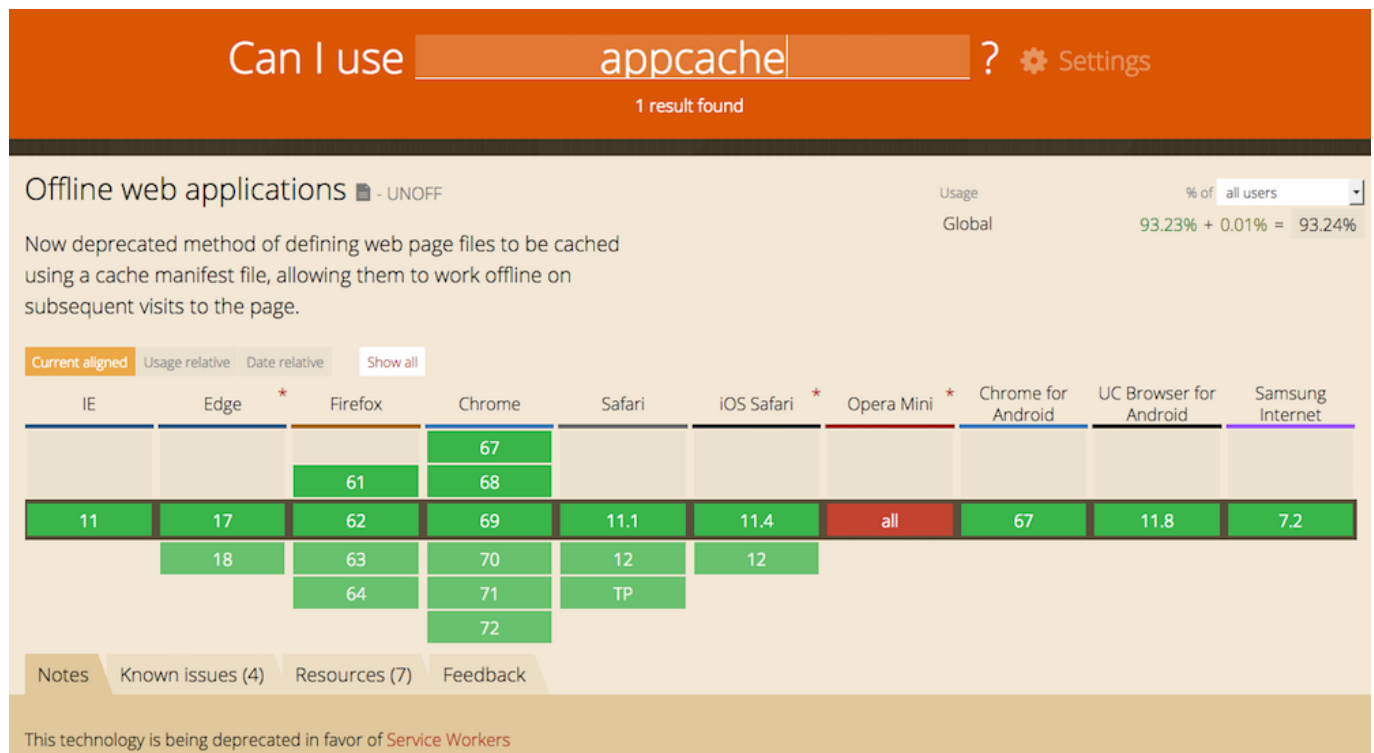
Informally, one could say that the W3C standardizes what the browser vendors have agreed upon and have chosen to implement or chosen to implement in the near future.

Consensus building takes time. HTML5 for instance was a candidate recommendation in Q4-2012 and became a W3C recommendation in Q4-2014. HTML5.1 was a candidate recommendation in Q1-2015 and became a

recommendation in Q4-2016.

As of August 2018, [HTML5.3](#) has a *Working Draft* status; if you look at the standard text you will find it to be very elaborate (this is a 1,000+ pages document!) and precise, sufficiently so that any browser vendor can take the text and implement the features described in it without ambiguity.

In rare cases, features added to a web standard can also be removed again, the [AppCache](#) is a prime example of this: it was developed as technology to enable offline web applications in a simple manner (by adding a manifest file to a site containing no more than a few lines of text), but turned out to have so many [pitfalls](#) that it was eventually abandoned in favor of another set of technologies ([Service Workers](#)). However, it is [still supported by all major browsers](#):



Screenshot taken on August 31, 2018.

The W3C writes the following about the AppCache: *This feature is in the process of being removed from the web platform. (This is a long process that takes many years.) Using the application cache feature at this time is highly discouraged. Use service workers instead.*

Self-check

Here are a few questions you should be able to answer after having followed the lecture and having worked through the required readings:

- Which of the following statements about HTML `<form>` methods `GET` and `POST` is true?
 - There is no difference, `GET` and `POST` can be used in exactly the same situations.
 - Through `GET` more data can be sent in an HTTP request than through `POST`.
 - Through `GET`, data is sent more securely than through `POST`.
 - Using `GET`, the data to be sent is visible in the URL which is not the case for `POST`.
- In the context of web design, what is the purpose of the trunk test?
- In a usability test, what is the task of the facilitator?

4. What happens if the **target** attribute is not defined in an HTML **<form>**?
 - The response received after submitting the form is displayed in the current frame or browsing context.
 - The HTML form lacks a required attribute and the browser will inform the user with a **405** error when the user tries to submit the form.
 - The response received after submitting the form is displayed in a new browser window.
 - The HTML form lacks a required attribute and the browser will inform the server with a **505** error that an invalid form request was made.
5. What does it mean for the W3C to produce a recommendation document?