

CSS vs. JS

Barcamp London
Christian Heilmann

TESCO

value

MOOD IMAGERY

**QUALITY
GUARANTEED**

Before the CSS revolution

- FONT tags, visual attributes, table hacks
- Far too much markup – large documents
- Redesigning pages was a PITA – even when using SSI
- Things had to change!

Alors, enfants de la cascade...

- CSS made it a lot easier:
 - No extra markup (pending an ID here and there)
 - Centralized look and feel in an external, cached document.
 - (Browser support permitting, of course)

The next target

- Bulky JavaScript rollovers with inline event handlers added unnecessarily to the document.
- Adding a rollover element (menu item) meant changing the HTML and the JavaScript (pre-caching script)
- Solution: Pure CSS rollovers (MSIE fix with background-position)

Taking it further

- Pure CSS rollovers with preloaded images proved CSS beats JavaScript.
- Next step:
 - Pure CSS popups
 - Pure CSS menus
 - Pure CSS “more links”

CSS, the one trick pony

- All of these are possible with pseudo selectors:
 - :hover for, well, hover effects
 - :focus for “keyboard support”
- The snag:
 - MSIE only supports :hover on links
 - Browsers only allow keyboard access to links and form elements

A Problem of Scope

- CSS solutions can only reach elements contained in other elements.
- A real bummer that A is an inline element

Semantics, Content, what?

- A lot of pure CSS solutions nest a lot of content in links to make them keyboard accessible
- This messes with assistive technology (links as lists) and generally is bad practice.

...and we're back

- Clever solutions use MSIE's conditional comments to add extra markup only when needed.
- However, this makes the HTML documents unnecessarily heavy and we are back to square one.

The Pure CSS Dilemma

- In order to have an almost accessible solution you need to:
 - Create invalid HTML
 - Add extra markup
 - Deal with browser issues by the truckload

The JavaScript Arsenal

- Very good browser support
- Conditional constructs for Progressive Enhancement
- DOM
- Timed execution
- Keyboard detection
- Events
- Ajax

The future is hybrid!

- Semantically valid markup
- JavaScript to provide the behaviour hooks
- CSS to style the effect / widget / page item

But... But... But...

JAVASCRIPT CAN BE TURNED OFF AND IS
NOT ACCESSIBLE!

- So can CSS.
- What do you want?
 - A non JS version that may work or
 - A JS version that can test if it will work and only applies itself when it can be used.