API Design For the Masses



Robert O'Callahan Mozilla Corporation

About Me

- Longtime Mozilla contributor
- Long involvement in Web standards
- Contributions to CSS3, HTML5 and DOM APIs

Communication

- Formats
- Protocols
- APIs

Authors and Web servers on one side, browsers on the other

Old Way

- A few smart people define standards
- A few smart people work with them

New Way

- A few smart people define standards
- Millions of people work with them

Compare

• "Does my TCP/IP stack follow the RFCs?"

"My HTML page looks good, I'm done"

Web Exacerbations

- Multiple implementations on each side
- Intense competition on each side
- Different sides in different administrative domains

Without these, my conclusions may not apply ...

Error Recovery

- HTML4 has undefined behaviours
 - **Here** <i>**be** dragons</i>
- But authors haven't read HTML4
- It works in their browser
- Users prefer error recovery
- → Browsers must at least follow the dominant browser

Specify Error Recovery?

- Dominant browser is de-facto standard
- Add its behaviour to the de-jure standard?
- If not, all browser vendors (and some authors) must reverse-engineer behaviour
 - And the de-jure standard is a myth
- → Put error recovery in the spec

This is extremely controversial

Fatal Errors?

- XML: Going outside the spec is fatal
 - "Yellow screen of death"
- Fragile: implementations have bugs
 - E.g. Early SVG viewers didn't check namespaces
- → Pressure for implementations to match leniency of other implementations

Note On Aesthetics

- "Why should we specify what happens if you use negative word-spacing? It makes no sense!"
- Authors will try everything (accidentally?)
- They will expect consistent results

- Try to define behaviour for **all** inputs
- No "undefined" behaviours
- Simplify or eliminate "fatal" behaviours

Standards Versus Reality

- HTTP spec says to honour server's MIME type
- Authors send text/plain MIME type incorrectly
- Browsers must sniff when type is text/plain
- This cannot be realistically changed
- Change specs to match?
- → Yes! Spec is only useful if it matches reality!

Hugely controversial

- Specs that contradict entrenched practice must be changed to match it
- If following the spec harms users, the spec is worthless

Implement First

- Standards created in a vacuum are usually bad
- Need implementation experience
 - Simple?
 - Fast?
 - Completely specified?
- Need author experience
 - Easy to use?
 - Satisfies needs?
- Example: SVG

Standardize First

- Pre-standard implementations can poison the well
- Popular implementations become de-facto standards
- Implementors don't like revising their implementations and breaking stuff
 - They resist change during standards process

- Standardization and implementation must happen concurrently
- Need a fast feedback loop
- Need agile spec update ability

Vendor Prefixes

CSS and DOM specs use vendor prefixes

```
-moz-border-radius: 37px;
if (element.mozMatchesSelector)
{ ... element.mozMatchesSelector("div") ... }
```

- Lets people know they're using non-standard feature
- Lets standard behaviour diverge if necessary

- Provide non-standard/pre-standard extensions
- Ensure authors know they're non-standard

Evolution Vs Revolution

- Web standards evolved by accretion
- Temptation: sweep away, start over
 - XHTML2
- Almost impossible at Web scale
- Existing investments in content, tools, etc
- Even Flash, Silverlight struggle
- Evolution keeps winning

• Favour evolution over revolution if possible

Versioning

- Versioning with incompatible breaks is a form of revolution
 - Python 2.x vs Python 3
- Avoid!

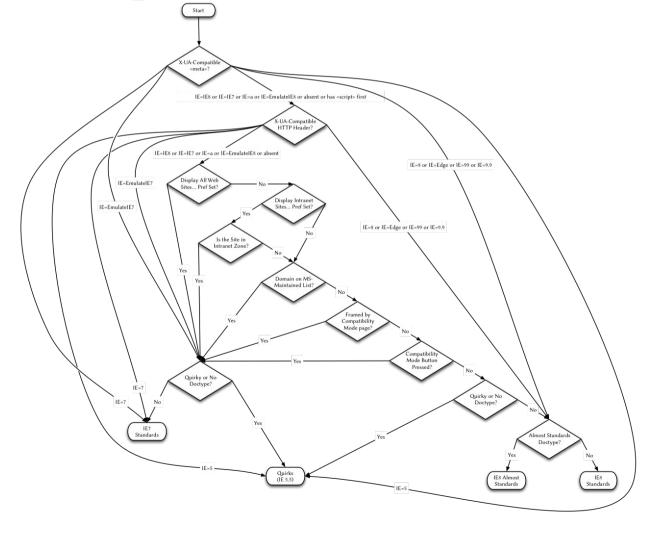
Versioning

- Versioning with back-compat is OK
- Doesn't work too well on the Web
- Content should work in old browsers with graceful degradation
 - Forward and backward compatibility
- Then versioning is unnecessary

Versioning

Downside: temptation to switch behaviour on

versions



- Aim for forward and backward compatibility (with graceful degradation)
- Avoid versioning

Feature Detection

- Authors like to condition on browser versions
- This is fragile and does not scale
- Encourage authors to use feature detection or CSS fallback

```
Opera/9.80 (Macintosh; Intel Mac OS X; U; en)
Presto/2.2.15 Version/10.00

if (document.getBoxObjectFor) { ... Mozilla path ... }
```

- Encourage content conditionals to use detection of features, not detection of implementations
- Provide mechanisms for such conditionals

Invisible Metadata

- Most Web pages are done when they look right in a browser
- Therefore authors won't add content that doesn't affect browser rendering
- If they do, they'll get it wrong
- Therefore metadata will not be added, or it will be added incorrectly
 - Example: longdesc used incorrectly 99% of the time

 Only add features where authors can see and correct their mistakes

One reason why the semantic Web failed

HTML5

- My "lessons" are just from some of the design assumptions of HTML5
- Ongoing battles over HTML5 within W3C due to these controversies

For Researchers

- Keep speculating about revolutions
 - So we know which direction to evolve
 - Or when it's time to revolt
- Explore feature design
 - You won't get enough users to poison the well
- Could use tool support for compatibility checking etc

Conclusions

- Mass adoption changes the rules
- Learn these lessons if you expect mass adoption
- Multi-vendor Web standards development is a real mess
- But it's better than the alternatives!!!