HTML.Next
Panel, W3C TPAC Plenary

Larry Masinter

(DISCLAIMER: Personal opinions Not Adobe or W3C TAG)

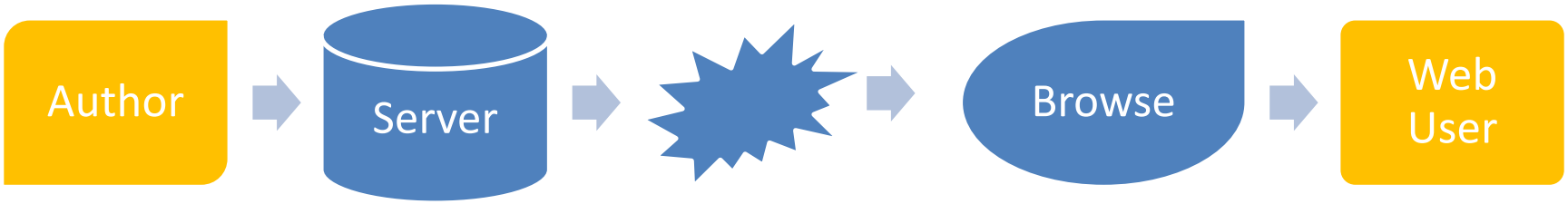
Lyon, France, 11/3/2010

New directions for HTML.next technology

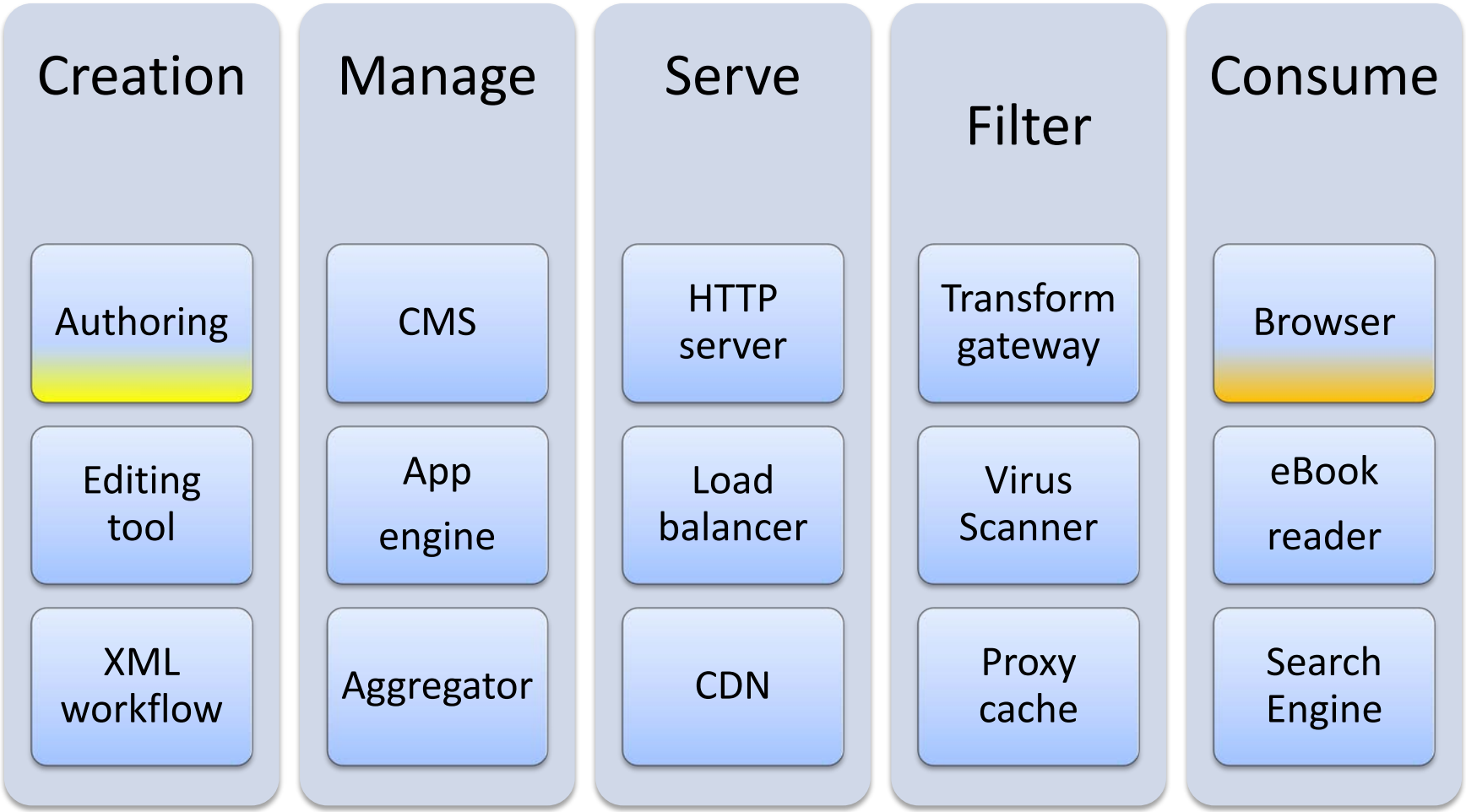
What I said I would talk about…

(not speaking to following slides, read them later)

Simple Model of The Web



More complicated model of The Web



Some different requirements: High quality publications

- Finer grain control of layout through CSS enhancements
- Integration of XML toolchain
- Making an archival profile of HTML/CSS

Some different requirements: Authoring tools

- Graphics model to match Authoring Model
  - Key frames, scrubbing, timelines, synchronization
- Ability to target versions
  - Identify modules, versions, deployment
- Asynchronous environment and animation
  - Look at synchronization
- Ability to extract API from document
  - Forms, declarative vs. imperative

Requirements: HTML in Email & other environments

- No Scripting?
  - (Canvas requires scripting)
- No content negotiation
  - Sniffing rules different
  - Security model different

HTML.Next: Polarized Opinions

- **Set of choices**
  - Capture the ways in which opinions are polarized
  - Give a handy label to the issue
- **Rough cut**
  - Label may not match how you think of it
  - You probably hate both
- **Which one do you agree with more?**
  - Which one do you hate less?
  - Just for fun, informal show of hands

1. Match Reality

a) **Standards should match reality**
   the standard should follow what some [all, most, the important, the open source] implementations have implemented [are willing to implement in the very near future]
b) **Standards should try to lead reality**
   try to move things in directions that improve simplicity, modularity, reliability, and other values.

2. Reverse Engineering

a) **Standards should reduce "reverse engineering"**
   Say enough to reduce costs of anyone else having to reverse engineer how things (content, common readers) work in the world today…
b) **Matching behavior isn't important**
   Matching existing behavior is only of short term interest; such guidelines might appear in a "current implementation guide" but don't belong in a standard of long-term value

3. Precision

a) **Standards should precisely specify behavior**
   Give detail of how to implement something compatible with the what is currently deployed, sufficiently that no user will complain that some implementation doesn't work "the same". Such behavior should be mandated by the standard.
b) **Standards be loose as possible**
   Minimize conformance requirements to allow widest possible range of implementations; even if it means that not all existing (badly written) content works uniformly. Conformance ("MUST") should be used rarely.

4. Leading

a) **Standards should lead the community**
   Standards should add exciting new features. New features should ideally appear first in the standard.
b) **Standards follow innovation**
   *Sample implementations should be widely reviewed and tested and only **after** wide **experience** with technology be added to the standard.*

5. Extensibility

a) Non-standard extensions should be avoided
   Ideally, we should eliminate any non-standard extensions and technologies; everyone's experience should be the same.
b) Non-standard extensions are valuable
   Innovations have come (will continue to come) from competing (non-standard) extensions; such extensions (and plug-ins) should be encouraged, even though particular extensions are not universally deployed.

6. Modularity

a) **Modularity is valuable**
   Specifying technology in smaller separate parts is beneficial: the ability to choose subsets extends the range of applications; modules can evolve independently
b) **Modularity is disruptive**
   Independent evolution leads to divergence and confusion. Subsets just mean unwanted choices, chaos.

7. Timely

a) **Standards take too long, move faster**
   Shipping the latest proposed features is a good way to validate proposed standards and get technology in the hands of users; standards that take years aren't interesting.
b) **Taking years to finish a standard is OK**
   Encouraging users to deploy experimental extensions before they are completed will cause fragmentation; not all experiments will (should) succeed.

8. Authors Ignore Standards

a) **Web authors don't care about standards**
   Most individual authors, designers, developers and content providers ignore standards anyway, so any efforts based on assuming authors will change isn't helpful.
b) **Influencing authors is possible**
   Authors can and will stick to standards if at least some popular browsers agree to tie new features to standards-conforming content.

9. Always-On Committee

a) **The Web should continue to grow**
   Web standards committee should be always on, to allow for rapid evolution. The notion of version numbers for standards is obsolete in a world where there are continual improvements.

b) **Standards should be stable**
   Continual innovation may be good for technology suppliers, but is bad for standards; evolution should be handled by allowing individual technology providers to innovate, and then to bring these innovations into standards in specific versions.

10. Open source

a) **Open source implementation is crucial**
   Allowing any company or software developer to provide their own private extensions is harmful; a content standard should be managed by the group of major (or major open source) implementors, so that any "standard" extension is available to all.

b) **Open source is unnecessary**
   Proprietary extensions and capabilities (from a single source or a consortium) have benefited the web in the past and will continue to be sources of innovation

11. Browsers and the Web

a) **The web is first and foremost "what browsers do"**
   The web is an application dominated by browsers primarily, and secondarily web applications (browser technology used for installable applications)
b) **Other needs can dominate browser needs**
   Web technologies extend to the widest range of Internet applications, including email, instant messaging, news distribution, syndication and aggregation, help systems, electronic publishing; requirements of these applications should have equal weight, even when they are meaningless for what "browsers" are used for.

12. Royalty Free

a) **Avoid all patented technology**
   Every component of a browser MUST be implementable without any restriction based on patents or copyright (although creation tools, search engines, analysis, translation gateways, traffic analysis may not be)
b) **Patented technology has a place**
   In some cases, patented technology cannot be avoided, or is so widespread that "royalty free" is just one more requirement among many tradeoffs

13. Forking

a) **Forking the spec allows innovation**
   Having multiple specifications which offer different definitions same thing (HTML) allows leading features to be widely known and implemented, and allows group to work around organizational bottlenecks.

b) **Forking the spec is harmful**
   Multiple specifications which claim to define the same thing is a power trip, causing confusion.

14. Accessibility

a) **Accessibility is one of many requirements**
Accessibility is an important requirement for the web platform, but only one of many sets of requirements, to be traded off against the requirements of other user communities when developing standards

b) **Accessibility is not an option**
Insuring that those who deploy products implementing W3C standards allow building accessible content is necessary before W3C can endorse or recommend that standard.

15. Architecture

a) **Architecture is mainly theoretical**
   "Architecture" is primarily a theoretical and not very useful way, mainly of adding requirements that aren't very useful.
b) **Architecture and consistency is crucial**
   Consistency between components of the web architecture and guidelines for consistency and orthogonality are important enough that existing work should slow down to insure architectural consistency.

More topics

- DRM: DRM is Evil? DRIM is Important Feature?
- Privacy: Up to browsers? Mandated in specs?
- Voice: Integrated? Separate spec?
- Applications: Great? Misuse: use Browser?
- JavaScript: Essential, stable?
  Fundamentally broken?

Summary

- **None of these are HTML5 issues**
  … all of them are HTML.next issues
- **None of these differences are irreconcilable**
  … but ‘compromise’ may not be the answer
- **Multiple specs, co-existence, reconciliation**

Can’t solve the problems unless
you acknowledge them