HTML 5

The Next Generation of Markup on the Web

What?

- New version of HTML
- Initial work by the WHATWG (2004)
- Work-in-progress recently adopted by the W3C (2007)

Who?

- WHATWG
  - Apple
  - Mozilla
  - Opera
  - Anyone who joins the mailing list (you?)
- W3C HTML-WG
  - Apple
  - Mozilla
  - Microsoft
  - Opera
  - Others
  - Public Invited Experts (you?)

Why?

- There's a lot of information on the web
- Most of that information is text/html
- Most (>95%) of that HTML is syntactically invalid

Conclusion:

- HTML is important and isn't going away any time soon
- We should evolve HTML rather than ditch it
- But the vast legacy creates problems

Why (continued)?

- HTML 4 – 1997
  - Underspecified
  - Inconsistent
  - Does not match reality
  - Missing features needed to compete with propriety technologies (Flash, Silverlight, etc.)
- XHTML 2
  - Requires XML
  - Not backward compatible (cannot be implemented in current browsers)

How?

- Identify use cases
- Identify possible solutions
- Pick the best solution according to constraints like compatibility, accessibility, etc

- Important to collect as many different inputs on use cases as possible
  - Need input!

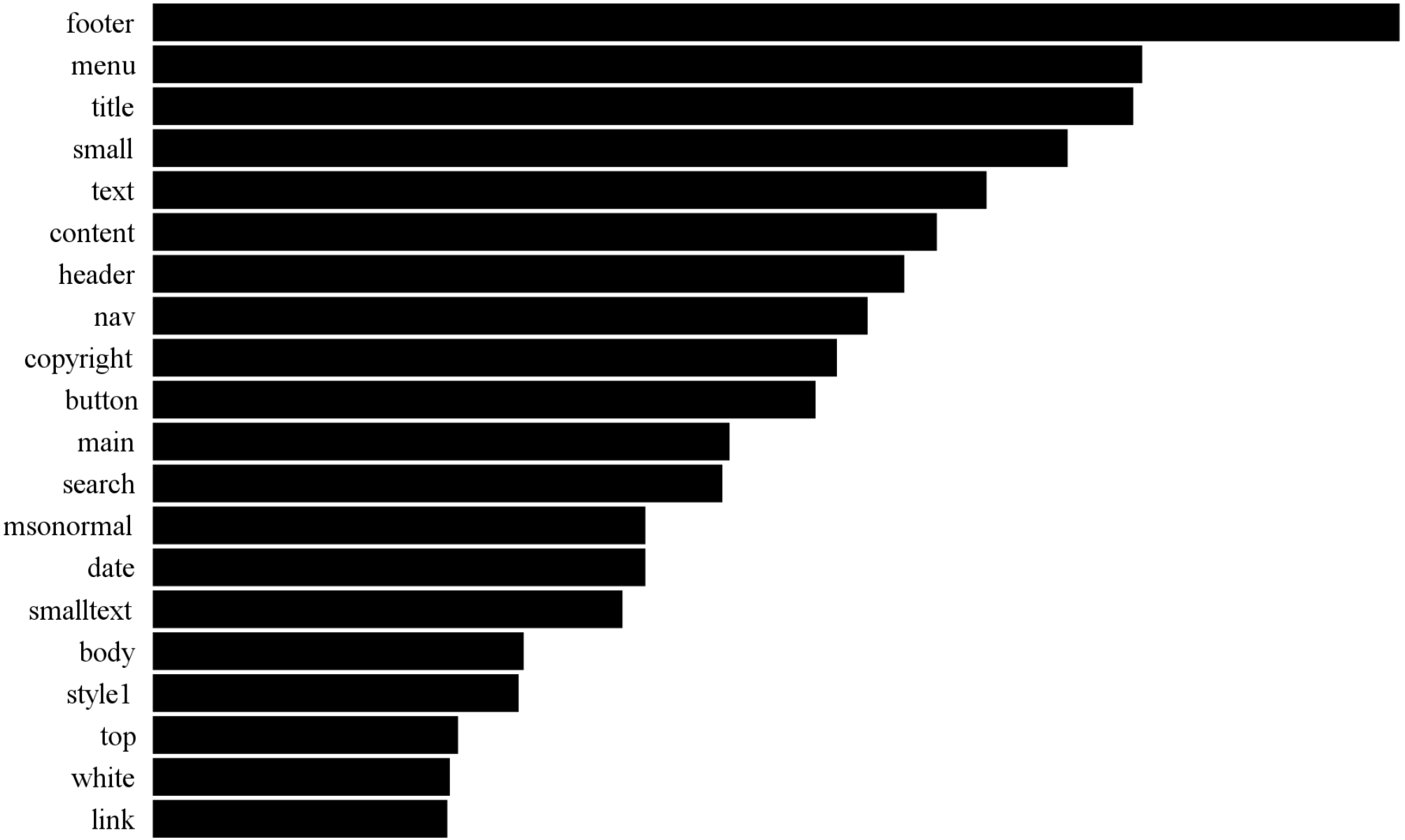
A HTML 5 Document

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello World!</title>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is my first HTML 5
document</p>
  </body>
</html>
```

New Structural Elements

```
<nav>
<article>
    <header>
    <section>
        <aside>
     <footer>
```

Popular HTML4 Classes



Heading Algorithm

```
<h1>Page Title</h1>
<section>
  <h2>Section Heading</h2>
    <section>
      <h1>Embedded content</h1>
    </section>
</section>
```

- Page Title
  - Section Heading
    - Embedded Content

Figures

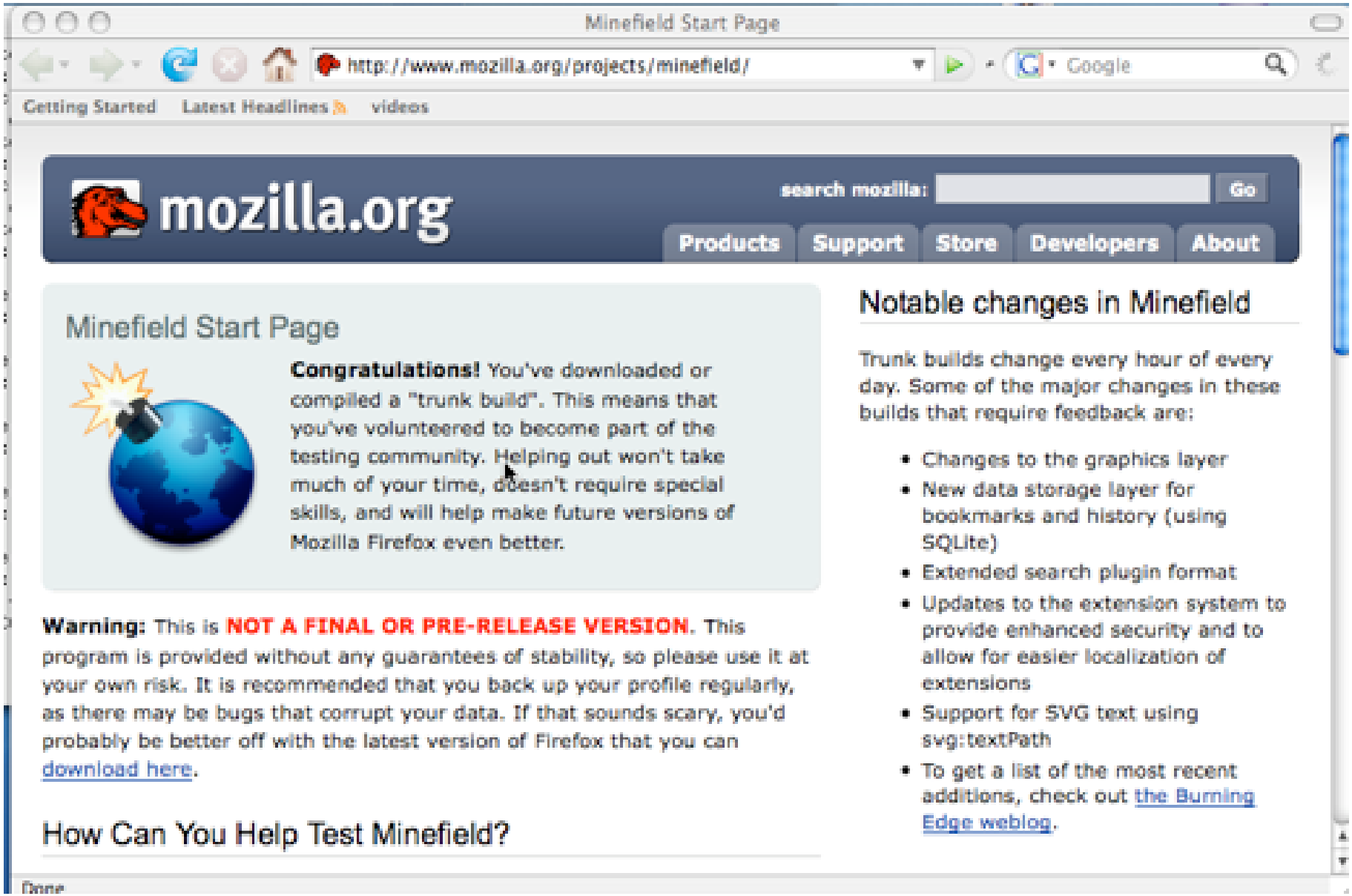
```
<figure>
    <img src=”myimage”>
    <legend>
        A caption for
        my image
    </legend>
</figure>
```

Audio and Video

```
<video>
  <source src="myvideo.ogg"
          type="video/x-ogg">
  <source src="myvideo.mpg"
          type="video/mpeg">
  Fallback content
</video>

<audio src="myaudio.ogg"
type="audio/x-ogg"
autoplay>
```

Firefox <video>



New Inline Elements

```
You searched for <m>marker</m>

BarCamb is on <time datetime="2007-08-
24">August 24th 2007</time>

<progress value="128"
max="1024">12.5%</progress>

Rating: <meter min="0" max="5"
value="4"><img
src="4stars.png"></meter>
```

Forms

- Web Forms 2 specification adds lots of features for authoring forms
  - New input types
  - Basic client side validation
  - Repetition blocks
  - Remote data (for e.g. select elements)
- Opera already has a native implementation of much of the spec
- Several JavaScript implementations are under development

Web Forms 2 Example

```
<input type="email” value="a@b">

<input pattern="[1-9]{10}"
value="1234567891">

<input type="number" min="7" max="25"
step="2"></label>

<input type="date” required>
```

Other Stuff

- Drawing: `<canvas>`
- Rich text editing: `@contentEditable`
- Tree/table-like controls: `<datagrid>`
- Drag and drop
- Server sent events
- Lots more...

Parsing

- HTML 0 (1991) – Not SGML
- HTML 2 (1995) – SGML
- HTML 3.2 (1997) – SGML
- HTML 4 (1997) – SGML
- XHTML 1 (2000) – XML

The Real World

```
<b>foo <i>bar</b> baz</i>
```

foo *bar* *baz*

Parsing

- HTML 5 defines 2 Serializations:
  - A custom "classic" syntax
  - An XML syntax (XHTML 5)
- The classic syntax has a fully specified parsing algorithm
- The parsing algorithm is designed (and being refined) for compatibility with deployed content

Parsing Implementations

- html5lib (Python, Ruby)
- Validator.nu HTML Parser (Java)
- ph5p (PHP)

html5lib example

```
import html5lib
from html5lib import treebuilders
import lxml.etree

p = html5lib.HTMLParser(
      tree=treebuilders.getTreebuilder(
                              “etree”, lxml.etree))
f = open(“mydocument.html”)
t = p.parse(f)

#Now we can use ordinary lxml.etree methods
tables = t.xpath(“//table”)
```

html5lib example 2

```
import html5lib
from html5lib import sanitizer

p = html5lib.HTMLParser(
      tokenizer=sanitizer.HTMLSanitizer())
in_file = open(“mydocument_unsafe.html”)
t = p.parse(in_file)

out_file = open(“mydocument_safe.html”, “w”)
out_file.write(t)
```

Get Involved

- whatwg.org
- w3.org/html