Matt Might
University of Utah
matt.might.net

HTML JavaScript CSS

Matt Might
University of Utah
matt.might.net

HTML

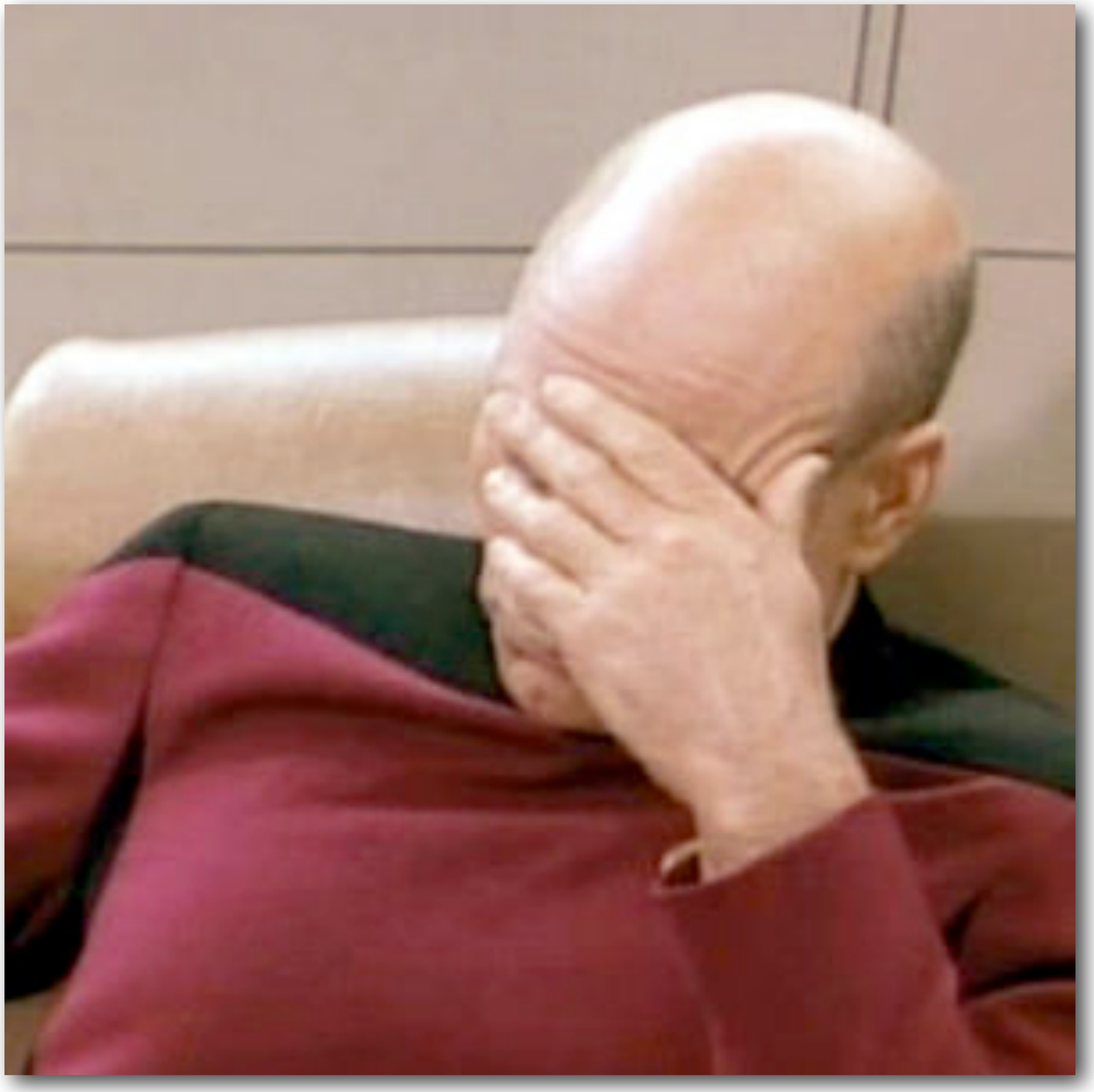
JavaScript

CSS

Today

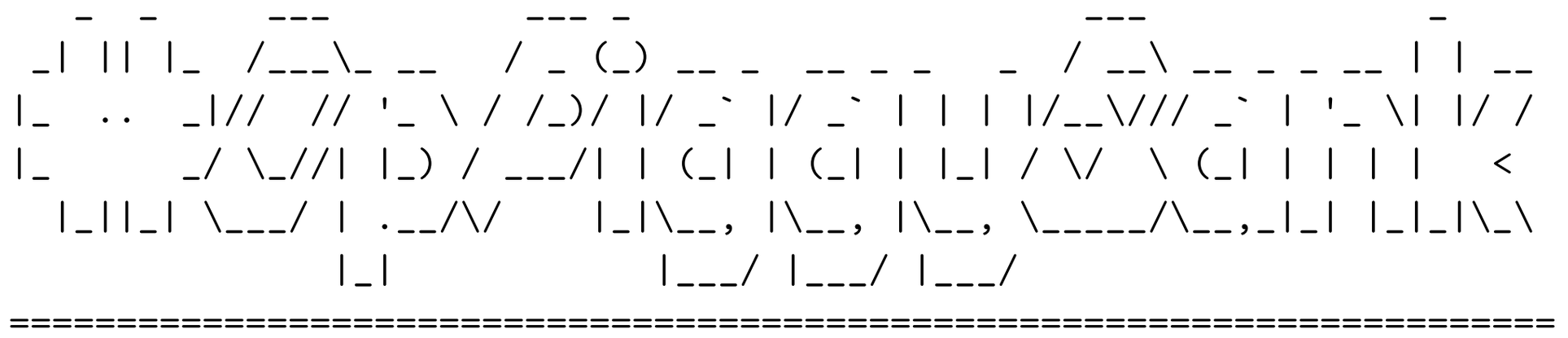
- HTML, XML, SXML
- JavaScript (DOM)

About Project 1 ...



Who could be *that* dumb?


SALT LAKE CITY
POLICE



#OpPiggyBank

@CabinCr3w & @ItsKahuna

id__username__hashed_password__first_name__last_name__employee_title__email__phone_number

3__jordan99PDsmith@slcgov99PDcom__db8a611bd35c926f17ccd2ab4d6ec6ecdd3dad06__Jordan__Smith__PR Specialist__jordan99PDsmith@slcgov99PDcom__801-707-6690
8__amarvel__feb051e448bb2c27f81b7b832c17806582183d8f__Aaron__Marvel__web developer II__aaron99PDmarvel@slcgov99PDcom__801-123-4567
9__ljones__7c01b8b7981eb1fe57817f2674f280c1eb9eb633__Lara__Jones____lara99PDjones@slcgov99PDcom__
10__jwatkins__109f4b3c50d7b0df729d299bc6f8e9ef9066971f__Jared__Watkins____jared99PDwatkins@slcgov99PDcom__
11__cwatkins__9cd3846f67e86c3ad877c7917baf36cce10e1d8e__Constance__Watkins____Constance99PDWatkins@slcgov99PDcom__

But the sergeant did say that all of the passwords obtained were all hashed – meaning they need to be figured out before the hacker could really know what all of the passwords are.

KUTV

2160

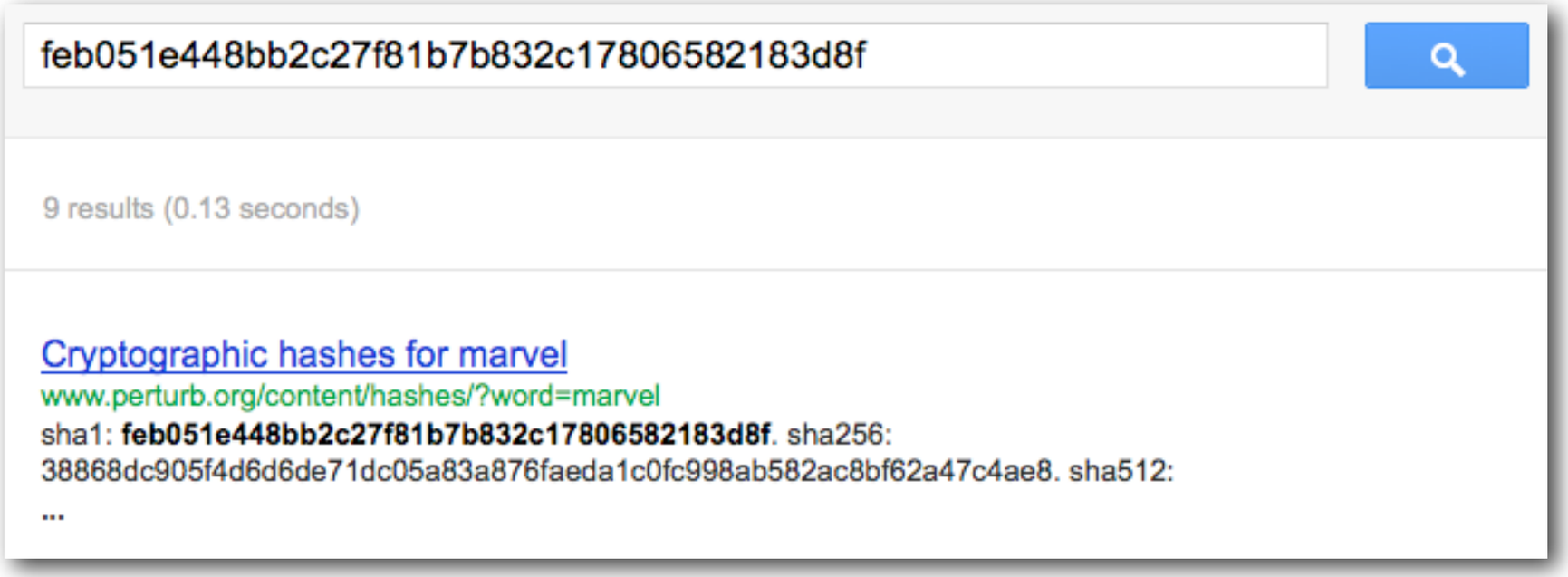
feb051e448bb2c27f81b7b832c17806582183d8f

Google Search I'm Feeling Lucky

ACHIEVEMENT UNLOCKED

I'm feeling lucky

Crack a password with Google.

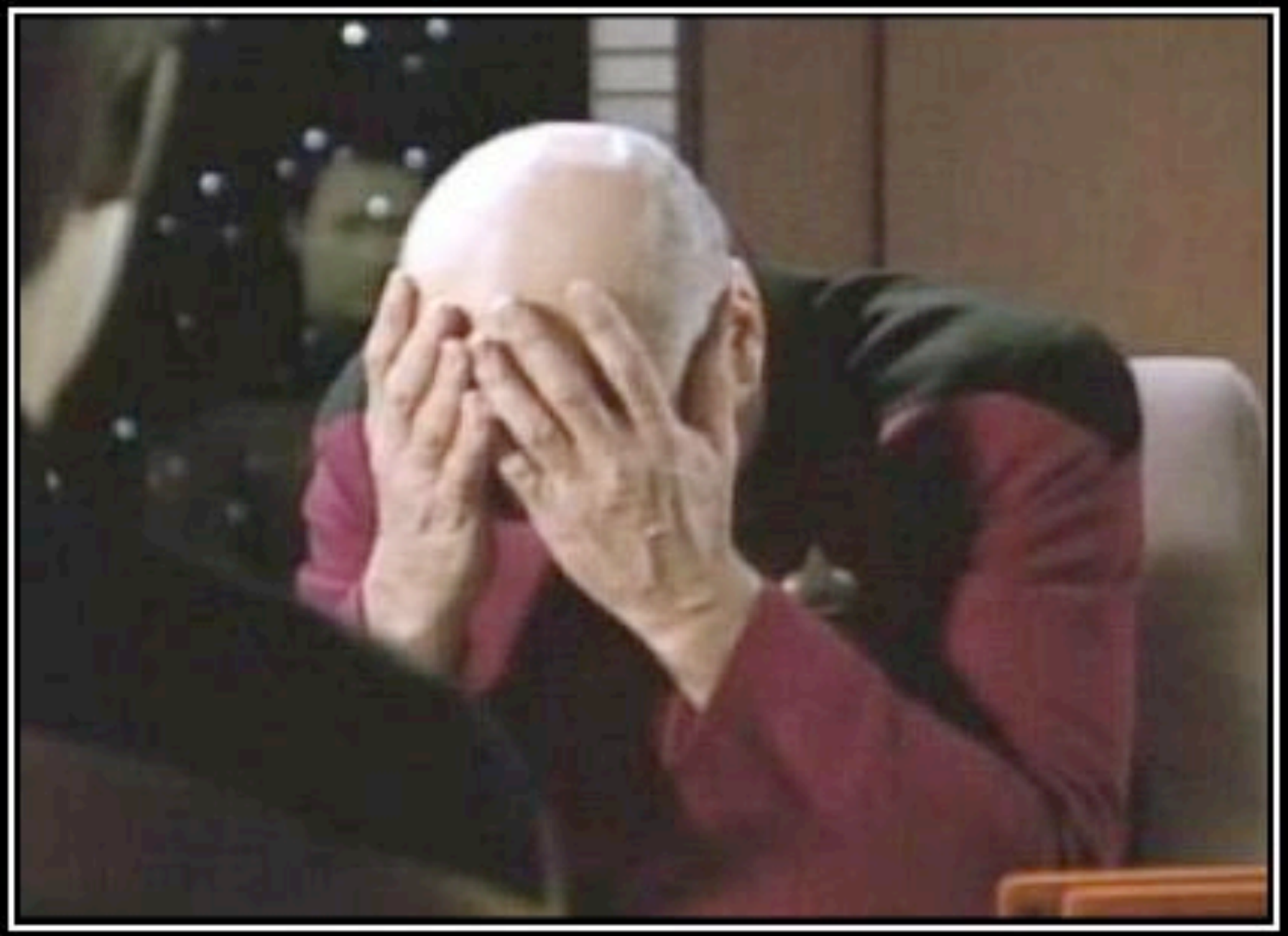

feb051e448bb2c27f81b7b832c17806582183d8f
9 results (0.13 seconds)
Cryptographic hashes for marvel
www.perturb.org/content/hashes/?word=marvel
sha1: feb051e448bb2c27f81b7b832c17806582183d8f. sha256:
38868dc905f4d6d6de71dc05a83a876faeda1c0fc998ab582ac8bf62a47c4ae8. sha512:
...

id__username__hashed_password__first_name__last_name__employee_title__email__phone_number

3__jordan99PDsmith@slcgov99PDcom__db8a611bd35c926f17ccd2ab4d6ec6ecdd3dad06__Jordan__Smith__PR Specialist__jordan99PDsmith@slcgov99PDcom__801-707-6690
8__amarvel__feb051e448bb2c27f81b7b832c17806582183d8f__Aaron__Marvel__web developer II__aaron99PDmarvel@slcgov99PDcom__801-123-4567
9__ljones__7c01b8b7981eb1fe57817f2674f280c1eb9eb633__Lara__Jones____lara99PDjones@slcgov99PDcom__
10__jwatkins__109f4b3c50d7b0df729d299bc6f8e9ef9066971f__Jared__Watkins____jared99PDwatkins@slcgov99PDcom__
11__cwatkins__9cd3846f67e86c3ad877c7917baf36cce10e1d8e__Constance__Watkins____Constance99PDWatkins@slcgov99PDcom__

```
ame__last_name__employee_title__email__phone_number

a611bd35c926f17ccd2ab4d6ec6ecdd3dad06__Jordan__Smith__PR Specialist__jordan99PDsmi
c17806582183d8f__Aaron__Marvel__web developer II__aaron99PDmarvel@slcgov99PDcom__8
f280c1eb9eb633__Lara__Jones____lara99PDjones@slcgov99PDcom__
0c6f8e9ef9066971f__Jared__Watkins____jared99PDwatkins@slcgov99PDcom__
17baf36cce10e1d8e__Constance__Watkins____Constance99PDWatkins@slcgov99PDcom__
```



Project 2

Too boring.

Check the web site!

HTML

Tag-oriented markup

GML (1960s)

```
:h1.Chapter 1:  Introduction
:p.GML supported hierarchical containers, such as
:ol
:li.Ordered lists (like this one),
:li.Unordered lists, and
:li.Definition lists
:eol.
as well as simple structures.
:p.Markup minimization (later generalized and formalized in SGML),
allowed the end-tags to be omitted for the "h1" and "p" elements.
```

SGML (1986)

```
<title>An SGML document</title>

<h1>Introduction</h1>

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)</li>
  <li>Unordered lists, and</li>
  <li>Definition lists</li>
 </ol>
 as well as simple structures.</p>
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

```
An SGML document</title>

<h1>Introduction</h1>

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)</li>
  <li>Unordered lists, and</li>
  <li>Definition lists</li>
 </ol>
 as well as simple structures.</p>
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

An SGML document

```
<h1>Introduction</h1>

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)</li>
  <li>Unordered lists, and</li>
  <li>Definition lists</li>
 </ol>
 as well as simple structures.</p>
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

An SGML document

```
<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)</li>
  <li>Unordered lists, and</li>
  <li>Definition lists</li>
 </ol>
 as well as simple structures.</p>
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

An SGML document

```
<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)</li>
  <li>Unordered lists, and</li>
  <li>Definition lists</li>
 </ol>
 as well as simple structures.
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

An SGML document

```
<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)
  <li>Unordered lists, and</li>
  <li>Definition lists</li>
 </ol>
 as well as simple structures.
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

An SGML document

```
<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)
  <li/Unordered lists, and/
  <li>Definition lists</li>
 </ol>
 as well as simple structures.
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

An SGML document

```
<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)
  <li/Unordered lists, and/
  <>Definition lists</>
 </ol>
 as well as simple structures.
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

An SGML document

```
<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)
  <li/Unordered lists, and/
  <>Definition lists
 </ol>
 as well as simple structures.
<p>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</p>
```

```
An SGML document

<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)
  <li/Unordered lists, and/
  <>Definition lists
 </ol>
 as well as simple structures.
<>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
</>
```

```
An SGML document

<h1>Introduction

<p>SGML supported hierarchical containers, such as
 <ol>
  <li>Ordered lists (like this one)
  <li/Unordered lists, and/
  <>Definition lists
 </ol>
 as well as simple structures.
<>
 Markup minimization (later generalized and formalized in SGML),
 allowed the end-tags to be omitted for the "h1" and "p" elements.
```

Design principle:

Inference is good.

Anti-pattern:

Human and machine disagree.

HTML (1989)

```
<html>
 <head>
  <title>A HTML document</title>
 </head>

 <body>
  <h1>Introduction</h1>

  <p>GML supported hierarchical containers, such as
   <ol>
    <li>Ordered lists (like this one),</li>
    <li>Unordered lists, and</li>
    <li>Definition lists</li>
   </ol>
   as well as simple structures.</p>
  <p>
   Markup minimization (later generalized and formalized in SGML),
   allowed the end-tags to be omitted for the "h1" and "p" elements.

   HTML added <a href="hyperlinks.html">hyperlinks</a>.
  </p>
 </body>
</html>
```

XML (1998)

XML: Tree-structured data

XML: End tags must match start tags.

<tag> data </tag>


<tag />

Design principle:

Both human and machine-readable.

Design flaw:

Neither human nor machine-readable.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This XML document was generated by RCCOBXML -->
- <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:apph=" http://www.redversconsulting.com/schema/apphead "
    xmlns:appb="http://www.redversconsulting.com/schema/appbody">
  - <env:Header>
    - <apph:reservation env:mustUnderstand="true">
       <apph:refNo>1234567</apph:refNo>
       <apph:timestamp>2010-11-30T14:26:59.125</apph:timestamp>
      </apph:reservation>
    </env:Header>
  - <env:Body>
    - <appb:resDetails>
      - <appb:train>
         <appb:date>12/24/2010</appb:date>
         <appb:time>09:30</appb:time>
         <appb:from>New York</appb:from>
         <appb:to>Chicago</appb:to>
         <appb:seat>57B</appb:seat>
        </appb:train>
      - <appb:passenger>
        <appb:name>John Smith</appb:name>
       </appb:passenger>
      - <appb:price>
         <appb:amount>$1,234.25</appb:amount>
         <appb:comments>Includes lunch & dinner.</appb:comments>
         <appb:comments>"Have a nice day!"</appb:comments>
        </appb:price>
     </appb:resDetails>
   </env:Body>
</env:Envelope>
```

Design flaw:

Subelement *v.* attribute

<tag name=value ...>

```
<tag>
  <name>value</name>
  data
</tag>
```

XML

- DTD: Structural type system
- XPath: Node selection model
- XQuery: “Like SQL for XML”
- XSLT: Define XML transformers

Design alternative: SXML

What's an s-expression?

John McCarthy

A number is an s-expression: `32`, `-13.2`, `3i`, `2/3`.

A string is an s-expression: `"foo"`

A symbol is an s-expression: foo

A boolean is an s-expression: `#t`, `#f`.

A list of s-expressions is an s-expression: $(s_1 \ \ldots \ s_n)$

SXML

```
<fleet>
 <ship id="1701">
   <name>Enterprise</name>
   <position x="10" y="20" />
   <speed>100</speed>
   <direction>0.13</direction>
 </ship>
 <ship id="1337">
   <name>Galactica</name>
   <name>The Bucket</name>
   <position x="10" y="20" />
   <speed>102</speed>
   <direction>0.12</direction>
 </ship>
</fleet>
```

```
(fleet
 (ship (@ [id 1701])
   (name "Enterprise")
   (position (@ [x 10] [y 20]))
   (speed 100)
   (direction 0.13))
```

```
(ship (@ [id 1337])
   (name "Galactica")
   (name "The Bucket")
   (position (@ [x 10] [y 20]))
   (speed 102)
   (direction 0.12)))
```

```
(fleet
 (ship (id 1701)
       (name "Enterprise")
       (position (x 10) (y 20))
       (speed 100)
       (direction 0.13))
 (ship (id 1337)
       (name "Galactica")
       (name "The Bucket")
       (position (x 10) (y 20))
       (speed 102)
       (direction 0.12)))
```

```
(fleet
 (ship [id 1701]
       (name "Enterprise")
       (position [x 10] [y 20])
       (speed 100)
       (direction 0.13))
 (ship [id 1337]
       (name "Galactica")
       (name "The Bucket")
       (position [x 10] [y 20])
       (speed 102)
       (direction 0.12)))
```

Modern HTML

<HTML>

</HTML>

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<TITLE></TITLE>
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head>
  <title></title>
  </head>
  <body>
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8" />
  </head>
  <body>
  </body>
</html>
```

Basic tags

- `a`
- `h`*n*
- `p`
- `br`
- `hr`
- `ol`, `ul`, `li`
- `title`
- `em`, `strong`
- `table`
- `img`

Interesting tags

- `script`
- `canvas`
- `style`
- `div`
- `span`
- `iframe`
- `object`
- `form`
- `input`
- `meta`

JavaScript

<script></script>

Looks like C; feels like Lisp.

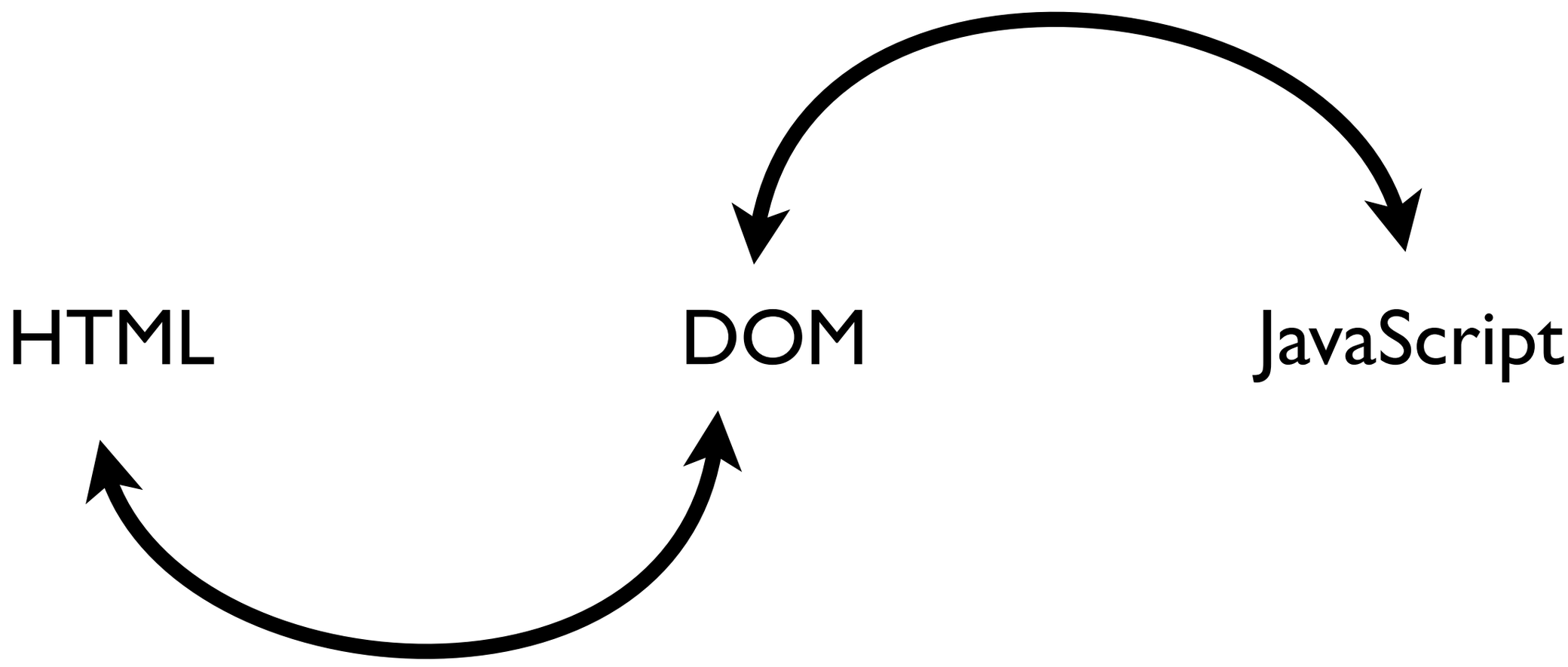
Nothing to do with Java.

Brendan Eich

JavaScript Tools

- Firebug debugger
- Web dev. toolbar
- Greasemonkey

DOM


HTML
DOM
JavaScript

document

document.write(*html*)

document.body

element.innerHTML

```
document.getElementById(id)
```

element.on*event*

`document.createTextNode(`*text*`)`

element$_1$`.appendChild(`*element*$_2$`)`

```
document.createElement(tagName)
```

parent.insertBefore(*newElement*,*beforeThis*)

Matt Might
University of Utah
matt.might.net

HTML JavaScript CSS

Matt Might
University of Utah
matt.might.net

HTML

JavaScript

CSS

JavaScript

Matt Might

University of Utah

matt.might.net

Grammatical forms

- Declarations
- Statements
- Expressions

Programs

$statement_1$

$statement_2$

$statement_3$

$statement_4$

...

Statements

declaration

block-statement

control-statement

with-statement

expression ;

Declarations

```
var id = expr ;
var id ;
```

function *name* () *block-statement*

Block statements

{ *statements* }

A word on scope

```
{
   int i = 0 ;
   {
      int i = 10 ;
   }
   printf("%i",i) ;
}
```

```
{
   var i = 0 ;
   {
      var i = 10 ;
   }
   console.log(i) ;
}
```

Only `function` and `with` can introduce new scope!

NO EXCEPTIONS!

```
x ;
```

```
x ; // Reference Error!
```

```
if (false) {
  var x ;
}
x ; // OK!
```

```
x ; // Also OK!
if (false) {
  var x ;
}
```

```
document.write(fact(3)) ;

function fact(n) {
 return (n == 0) ? 1 :  n*fact(n-1) ;
}
```

```
if (false) {
  function f(x) {
    return x + 1 ;
  }
}
f(10) ;
```

-compliant!

```
if (false) {
   function f(x) {
      return x + 1 ;
   }
}
f(10) ;
```

```
f(10) ;
if (false) {
  function f(x) {
    return x + 1 ;
  }
}
```

-compliant!

```
f(10) ;
if (false) {
   function f(x) {
     return x + 1 ;
   }
}
```

WHY!?

```
(begin
  expr1
  ...
  exprn)
```

(let ([*id expr*]) *exp*)

Control statements

```
if (expr) statement
if (expr) statement else statement
```

```
switch (expr) {
  case lit1:
  statements1

  ...

  case litn:
  statementsn

  default:
  statements
}
```

for (*declaration*; *expr*; *expr*)
statement

```
for (var id in expr)
    statement
```

```
for (id in expr)
    statement
```


```
for (id in expr)
  statement
```

```
do statement while (expr) ;
```

```
while (expr)
    statement
```

label: statement

```
break label ;
```

```
break ;
```

```
foo: while (true) {
   bar: while (true) {
      break foo ;
   }
}
```

```
foo: {
  bar: {
    break foo ;
  }
  console.log("eh")
}
```

```
continue ;
```

```
continue label ;
```

```
try {
  statements
} catch (id) {
  statements
}
```

```
try {
  statements
} catch (id) {
  statements
} finally {
  statements
}
```

```
try {
  statements
} finally {
  statements
}
```

Exceptional exceptions

```
function f() {
   try {
      return true ;
   } finally {
      console.log("boo") ;
   }
}
```

```
function f() {
  try {
    return true ;
  } finally {
    console.log("boo") ;
  }
  return false ;
}
```

```
function f() {
  try {
    return true ;
  } finally {
    console.log("boo") ;
    return false ;
  }
}
```

```
function f() {
  try {
    throw 10 ;
  } catch (e) {
    return e ;
  } finally {
    return 20 ;
  }
}
```

```
while (true)
  try {
    continue ;
  } finally {
    console.log(42);
  }
```

```
while (true)
   try {
      break ;
   } finally {
      console.log(42);
   }
```

`with` statements

```
with (expr) { statements }
```

Never use with!

Never*use `with`!

*Except to fake a let-form

```
{
 var i = 0 ;
 {
  var i = 10 ;
 }
 console.log(i) ;
}
```

```
{
 var i = 0 ;
 with ({i: 10}) {
  }
  console.log(i) ;
}
```

Expressions

num

id

expr binop expr

preop expr

expr postop

expr(*expr*,...)

expr[*expr*]

expr.*id*

this

null

"*string*"

'*string*'

{*field*$_1$: *expr*$_1$, ..., *field*$_n$: *expr*$_n$}

new *id* (*expr*,...)

[*expr*$_1$,...,*expr*$_n$]

/*regex*/

function (*vars*) *block-statement*

Anonymous functions

function (*vars*) *block-statement*

```
function map(f,object) {
   var res = {} ;
   for (k in object)
    res[k] = f(object[k]) ;
   return res ;
}

map(function (i) { return i+1; },
     { a: 0, b: 1 })
```

```
element.onwhatever =
 function (event) {
  respond
} ;
```

A taste of things to come...

```
(function(f){ f(f) })(function(g){ g(g) })
```

“Arrays”

$$[elem_1, ..., elem_n]$$

$$\texttt{new Array}(elem_1, ..., elem_n)$$

```
typeof [1,2,3]
```

Objects

```
{field1: expr1, ..., fieldn: exprn}
new id (expr,...)
```

```
{foo : 3, bar : [3,4]}
```

```
function Ship(x,y) {
  this.x = x ;
  this.y = y ;

  this.move = function (dx,dy) {
    this.x += dx ;
    this.y += dy ;
  }
}
```

Fields

```
obj['id'] = obj.id
```

```
o = {} ;
o.x = 3 ;
console.log(o.x) ; // prints 3
```

Prototypes

```
function F () {
}

F.prototype.foo = 3 ;

x = new F() ;

console.log(x.constructor);
console.log(x.foo);
```

```
function Ship(x,y) {
  this.x = x ;
  this.y = y ;
}
```

```
Ship.prototype.move = function (dx,dy) {
  this.x += dx ;
  this.y += dy ;
}
```

`object.__proto__`

`Object.getPrototypeOf(`*`object`*`)`

```
var x = {} ;
x.__proto__ = [].__proto__
```

A few bad angles

this

```
f(...)

o.f(...)

new f(...)

f.call(this,...)
```

Truthiness

```
'' == '0'
0 == ''
0 == '0'

false == 'false'
false == '0'
```

```
false == undefined
false == null
null == undefined

' \t\r\n ' == 0
```

What is `x`, so that

`x == x` is false

and `x !== x` is?

NaN

```
NaN == NaN
```

```
NaN === NaN
```

arguments

```
function f() { return arguments ; }
```

Object.prototype

```
Object.prototype.foo = 'Ooops!' ;
```

Matt Might
University of Utah
matt.might.net

HTML JavaScript CSS

Matt Might
University of Utah
matt.might.net

HTML

JavaScript

CSS

Matt Might

University of Utah

matt.might.net

CSS

Today

- CSS
- P2 - ?

SGML/HTML goal?

Separate style from content

<b>, <i>


<hr />

<font>

<table>

<blink>

<marquee>

awful.html

CSS: Let's try that again.

Cascading style sheets

,

style="*styles*"

styles ::=

{ *prop-name*: *value*; }

prop-name

prop-name

- background-*
- border-*
- {bottom,top,left,right}
- caption-side
- {clear,float,position,clip}
- color
- content
- counter-{increment,reset}
- cursor
- direction
- display
- empty-cells
- font-*
- {letter-spacing,line-height,word-spacing}
- {height,width}
- list-*
- margin-*
- {min,max}-{height,width}
- outline-*
- overflow
- padding-*
- page-break-*
- quotes
- table-layout
- text-*
- vertical-align
- white-space
- z-index

Examples

<style>

id="*name*"

```
#name { properties }
```

class="*name*"

```
.name { properties }
```

class="$name_1$ $name_2$ ..."

Selectors

- *
- *tag*
- [*attribute*]
- *tag tag*
- *tag* + *tag*
- *tag* > *tag*
- *tag* ~ *tag*
- :after, :before
- :checked
- :first-{child,letter,line}
- :first-of-type
- :hover
- :last-child
- :link
- :not()
- :only-child
- :only-of-type
- :visited

fixup.html

CSS + JavaScript

element.style

element.className

CSS Reset

JQuery

CSS3

- border-radius: *width*;
- border-shadow: *dx dy blur color*;
- text-shadow: *dx dy blur color*;
- display: box;
- transition: *property duration*;