CSSになり損ねた言語たち

1991

HTML誕生

Tim Berners-LeeによってHTMLが発表された1991年には、ページのスタイルを設定する方法はありませんでした。

HTMLタグがどのように処理されるかはブラウザ次第

そうした事情から、ページがどのようなスタイルで処理されるかを”提案”するような標準的な方法を求める声が上がるようになりました。

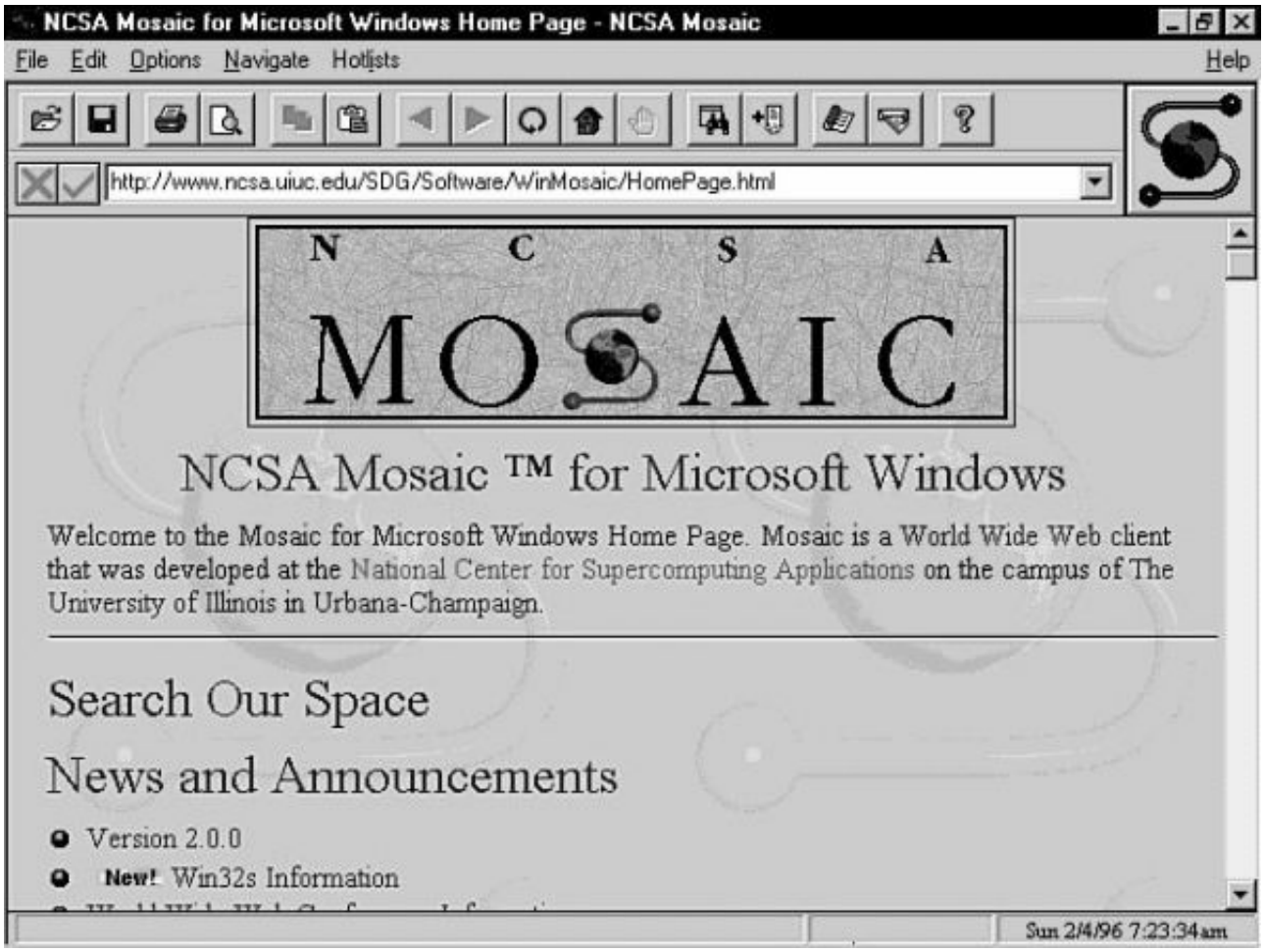
しかし、CSSが導入されるのは5年先で、完全に実用化されるには10年の歳月を待たねばなりません。

もしかしたら標準化されていたかもしれない競合する多くのスタイリング方法がこの時期に誕生しました。

それにこうした言語の多くには、意外にも現代の開発者たちがぜひともCSSに加えたいと思えるような機能が含まれていることもあるのです。

1993

NCSA Mosaic



Mosaicブラウザのバージョンはまだ1.0に届いておらず、当時流通していたその他のブラウザも対応していたのはHTMLのみでした。HTMLのスタイルを指定する方法はなく、 `<h1>` の表示もブラウザの解釈次第というのが当時の状況です。

最初の提案

6月に、Robert Raischは “Web文書に連動してスタイル情報を伝え、簡単に解析ができるフォーマット” を作成する提案をwww-talkメーリングリストで行いました。

RRP

```
@BODY fo(fa=he,si=18)
```

この表記で、フォントファミリー（fa）はHelvetica（he）、フォントサイズ（si）は18ポイントの設定となります。

［豆知識］ - RRPには、2011年までCSSにもなかったカラムレイアウトを指定する方法が含まれています。例えば3列、幅が”80単位”の時は次のようになります。

```
@P co(nu=3,wi=80)
```

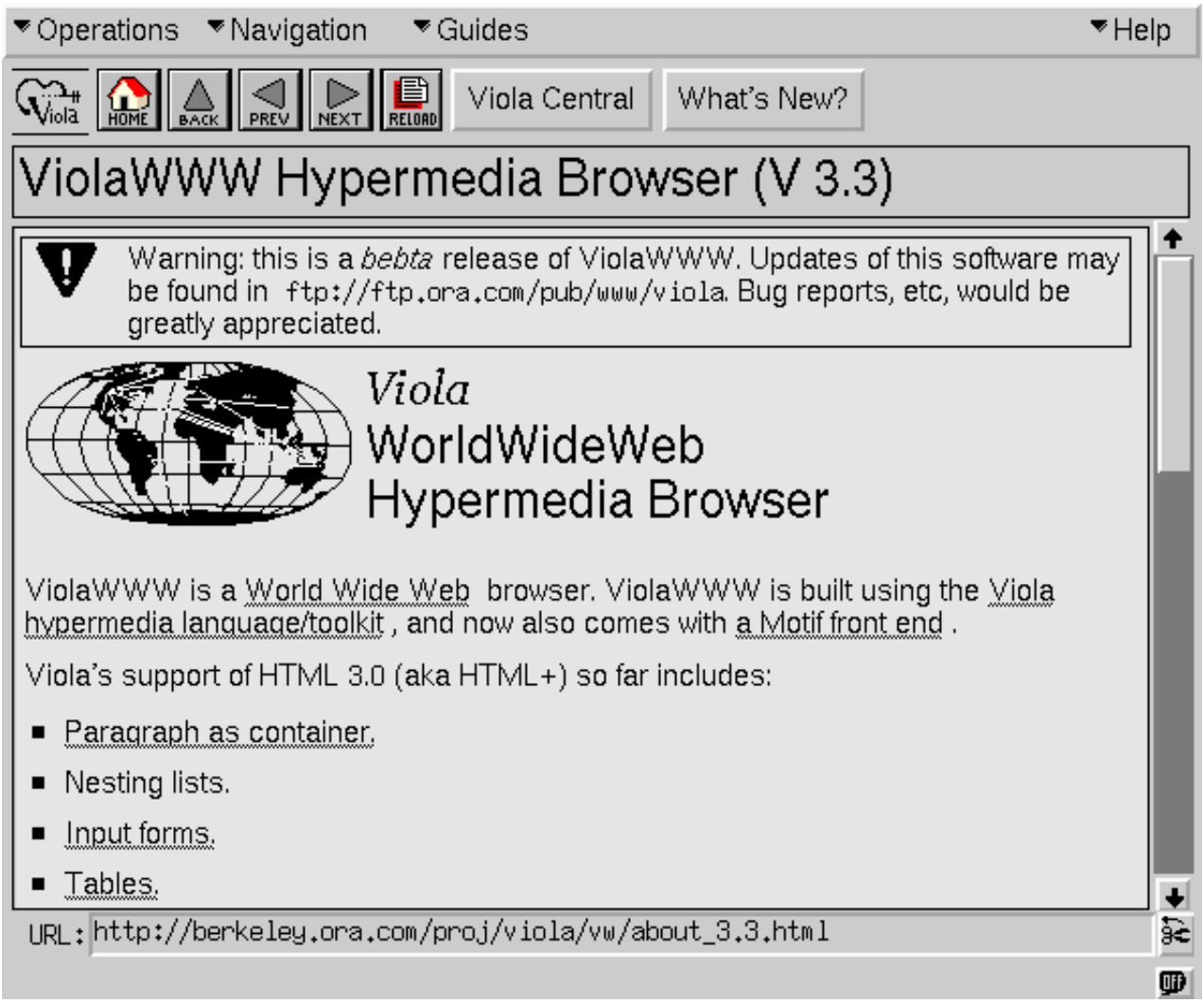
Marc Andreessen（後に人気を博したブラウザであるMosaicの生みの親）はRRPの提案を認識してはいましたが、Mosaicに実装することはありませんでした。その代わりスタイルを定義するために `<FONT>` や `<CENTER>` などを導入し、HTMLタグを使う道を急速に（いささか悲劇的に）歩んだのです。

1993

Viola、そしてブラウザ黎明期の争い

一般的な認識に反して、Mosaicは最初のグラフィックベースのブラウザではありません。グラフィカルなブラウザとして先陣を切っていたのは、Pei-Yuan Weiによって4日間で開発されたViolaWWWです。

ViolaWWW



Pei-Yuan（上記ブラウザ開発者）は現代のCSSでなじみ深い入れ子構造の形式をサポートしたスタイルシート言語を考案しました。

```
(BODY fontSize=normal
      BGColor=white
      FGColor=black
  (H1   fontSize=largest
        BGColor=red
        FGColor=white)
)
```

StylusやSASSなどの言語で使われているインデントシステムに似た括弧システムが使われます。これによりそのシンタックスは、少なくともある点では、最終的にWebの共通言語となったCSSよりも優れている可能性があると言えるのではないでしょうか。

［豆知識］ - Pei-Yuan Weiの提案はまた、今日でも使われている外部スタイルシート参照の方法を導入したことでも有名です。

```
<LINK REL="STYLE" HREF="URL_to_a_stylesheet">
```

Web以前のスタイルシート

> HTMLはコンピュータ科学者のみに愛される類いの代物だ。確かに文書の基本構造は表現できるかもしれない。しかし文書は構造化されたテキストデータベース以上のものであり、視覚的なインパクトも備わっている。HTMLは紙面デザイナーが持つであろう視覚的な創造性を完全に排除している。 – Roy Smith 1993年

1987

文書のスタイルを表現するための言語の必要性はインターネットの時代よりもはるかに前からありました。

- 元々HTMLはSGMLというインターネット前夜の言語をベースとしたものです。
- アメリカ国防総省は彼らが取り扱う大量の文書をより簡単に保管および送信するためにSGMLが使えないかを検討することにしました。
    - その検討するチームを、CALSチームと呼びます。
- CALSチームはSGML文書をスタイリングするための言語を開発しFOSIと名付けました。

1993

Pei-Yuanの提案のわずか4日後に、Steven HeaneyはWebにおけるスタイリングに関して “一から新しいものを作る” のではなく、FOSIから派生したものを使うのが最適だという提案をしました。

FOSI

FOSIの文書自体はSGMLで書かれており、Web開発者がSGMLの派生形であるHTMLに精通していること

を考えると、これは多少なりとも理にかなった動きではあります。文書例は次のような感じです。

```
<outspec>
  <docdesc>
    <charlist>
      <font size="12pt" bckcol="white" fontcol="black">
    </charlist>
  </docdesc>
  <e-i-c gi="h1"><font size="24pt" bckcol="red", fontcol="white"></e-i-c>
  <e-i-c gi="h2"><font size="20pt" bckcol="red", fgcol="white"></e-i-c>
  <e-i-c gi="a"><font fgcol="red"></e-i-c>
  <e-i-c gi="cmd kbd screen listing example"><font style="monoser"></e-i-c>
</outspec>
```

［豆知識］ - FOSIが導入したem単位は非常に優れており、今ではCSSでのスタイリングを熟知する人々にとって推奨の方法となっています。

プログラム言語なスタイルシート

チューリング完全

> たとえばC言語やJavaなどをはじめとする、一般的なプログラミング言語（の背景にある計算モデル）は全てチューリング完全である。
>
> コンピュータ言語のうち、少なくともチューリング完全でなければプログラミング言語とは呼ばれない。

JavaScriptも［プログラミング言語、つまりチューリング完全］である。

DSSSL

> DSSSLをスクリプト言語と同じカテゴリに入れるのは間違っている。DSSSLはチューリング完全なプログラミング言語だ。一方、スクリプト言語は（少なくとも私のその用語の使い方では）手続き型であり、DSSSLとはその性格を異にする。DSSSLは完全に関数型であり副作用もない。DSSSLのスタイルシートでは何も起こらない。スタイルシートは巨大な関数であり、その値はスタイルが適用された文書におけるデバイスに依存しない抽象的な非手続き型の記述で、それが表示領域の仕様（宣言と言ってもいい）として下流のレンダリングプロセスに送られる。 – Jon Bosak

```
(element H1
  (make paragraph
    font-size: 14pt
    font-weight: 'bold))
```

プログラミング言語なので、関数を定義することもできます。

```
(define (create-heading heading-font-size)
  (make paragraph
    font-size: heading-font-size
    font-weight: 'bold))

(element h1 (create-heading 24pt))
(element h2 (create-heading 18pt))
```

［豆知識］ - この機能には少し嫉妬をするかもしれませんが、DSSSLは継承された値を変数として扱い、それらで計算することができます。

```
(element H1
  (make paragraph
    font-size: (+ 4pt (inherited-font-size))))
```

その他の可能性

実際にCSSへと発展した言語のことを話す前に、もう1つ別の言語の提案にも触れておきたいと思います。

PSL96

PSLの核となる部分はCSSに似ています。

```
H1 {
  fontSize: 20;
}
```

要素の位置を指定されたサイズ（Width）だけではなく、ブラウザにレンダリングされた実際の（Actual Width）サイズに基づいて次のように表すことができるのです。

```
LI {
  VertPos: Top = LeftSib . Actual Bottom;
}
```

論理式をスタイルに追加することも可能です。

```
A {
  if (getAttribute(self, "href") != "") then
    fgColor = "blue";
    underlineNumber = 1;
  endif
}
```

1994

CSSの過去の痕跡

少なくとも名前の上において直接CSSにつながる言語は、1994年にHåkon W Lieが提案したCHSS (Cascading HTML Style Sheets、カスケーディング・HTML・スタイル・シート)と呼ばれるものです

良いアイデアのほとんどがそうであるように、原案はひどいものでした。

```
h1.font.size = 24pt 100%
h2.font.size = 20pt 40%
```

式の最後にある%を見てください。この%は、現在のスタイルシートがこの値に関して、どの程度”所有権”を持つかを表しています。例えば、前のスタイルシートがh2をフォントサイズ30ptと60%の所有権で定義し、このスタイルシートが、h2を20pt 40%としたとします。すると、2つの値が所有権の割合に基づいて組み合わされ、26pt程度の値になります。

何十年経ってもCSSに盛り込まれなかったであろう機能も含まれていました。その例の1つとして、ユーザの環境に基づいて論理式を書くことが可能でした。

```
AGE > 3d ? background.color = pale_yellow : background.color = white
DISPLAY_HEIGHT > 30cm ? http://NYT.com/style : http://LeMonde.fr/style
```

次に起こったこと

> Microsoftは、絶対にインターネットでスタンダードを切り開く。—John Ludeman 1994年

1996

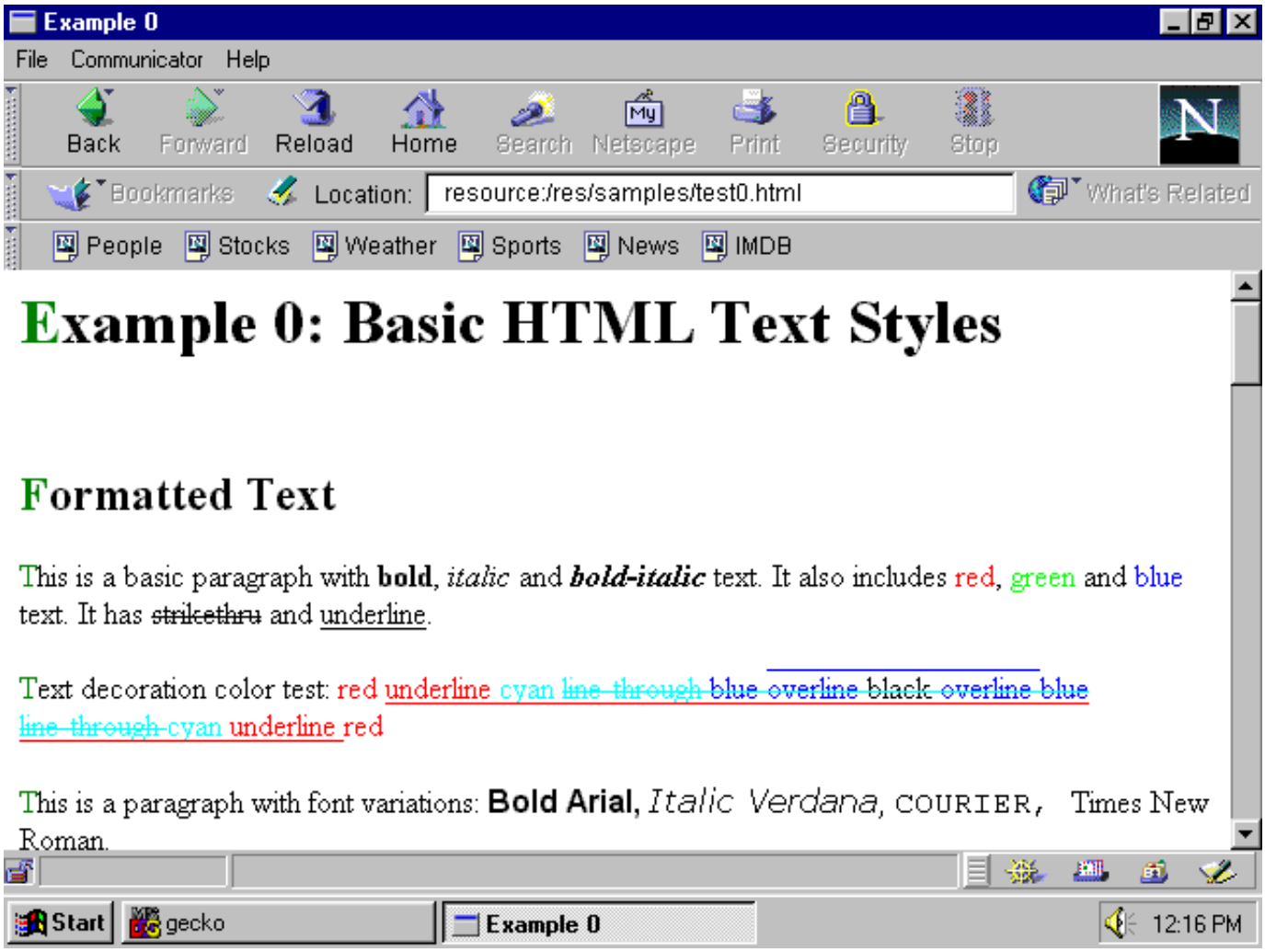
Håkon Lieは、自分の提案を簡素化し続け、Bert Bosとの共同作業の成果として、1996年12月、CSSの仕様書の初版を公開しました。

他の提案の多くと比較して、CSSの注目すべき事実の1つは、その簡潔さです。簡単に解析でき、簡単に書け、簡単に読めます。インターネット史上他の多くの例でもそうであるように、勝利に輝くのは、エキスパートが高性能と感じる技術ではなく、初心者が習得しやすい技術なのです。

CSSは1997年まで使い物にはならず、完全にサポートしているブラウザは2000年3月まで1つもありませんでした。ブラウザのサポートがほんの数年前まで規格準拠からは程遠い状態だったことを、開発者なら誰でも知っています。数年前とは、CSSリリースから15年以上も後のことです。

最後の問題

> もし、Netscape 4が要素に適用されたCSSの規則を無視し、ランダムに空白をページのあらゆる構造的要素に追加したならば、またもしIE4がを正しく理解したのにパディングし損ねたとしたら、どんなCSSを書くのが無難なのだろうか？　CSSを全く書かないという選択をする開発者もいる。IE4の欠陥をカバーするようなスタイルシートとNetscape 4の重大ミスをカバーするようなスタイルシートを個別に書く開発者もいる。— Jeffrey Zeldman



JSSS

Internet Explorer 3が（いささかひどい）CSSサポート付きで公開されたことはよく知られています。これに対抗するために、Netscape 4でもCSSのサポートが必要と決定されました。しかし、この（HTMLとJavaScriptを考えると）第三の言語に必要以上の期待をかけるよりも、CSSの実装は、JavaScriptに変換して実行することによって対応すべきと決定されました。

シンタックスは、JavaScriptそのままで、スタイリング専用のAPIが追加されています。

```
tags.H1.color = "blue";
tags.p.fontSize = "14pt";
with (tags.H3) {
  color = "green";
}

classes.punk.all.color = "#00FF00"
ids.z098y.letterSpacing = "0.3em"
```

スタイルとスクリプトの境界を簡素化するというアイデアは、確かに合理的で、現在、Reactコミュニティで、それなりに復活しているように感じられます。

JavaScriptは、それ自体が当時非常に新しい言語でしたが、リバースエンジニアリングによって、既にIE3にそのサポートが「JScript」として追加されていました。それより大きな問題は、人々は既にCSSを評価していて、Netscapeが、当時多くの規格団体から不当な扱いを受けていたことです。NetscapeはJSSSを規格委員会に確かに提出したのですが、受理されませんでした。3年後、Netscape 6はJSSSのサポートをやめ、JSSSは（ほとんど）静かに消えました。

2000

Internet Explorer 5.5が2000年、ほぼ完全にCSS1をサポートして公開されました。もちろん、今では知られている通り、ブラウザのCSS実装は、見事にバグだらけで、少なくともその後10年間は扱いづらいものでした。今日、状況は幸運にも劇的に改善し、開発者が一度コードを書けば、どのブラウザでもほぼ同じように機能すると期待できるという夢がようやく実現しました。

引用
http://postd.cc/the-languages-which-almost-became-css/