

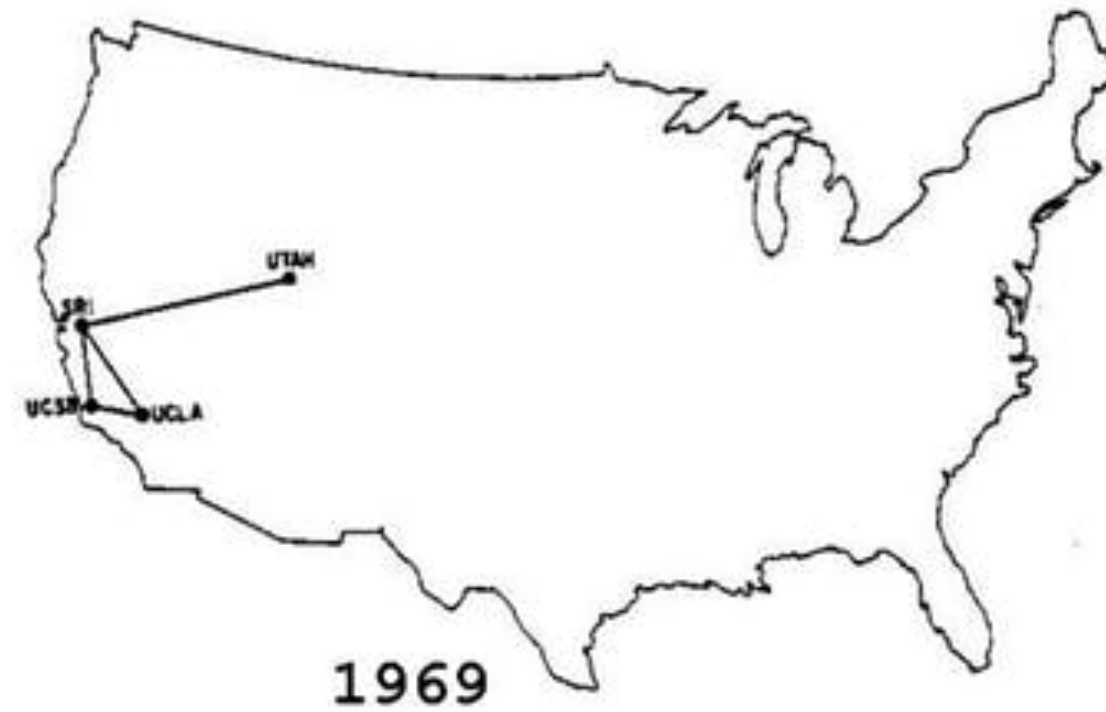
# HTML PRINCIPLES

Getting started with web design the right way.

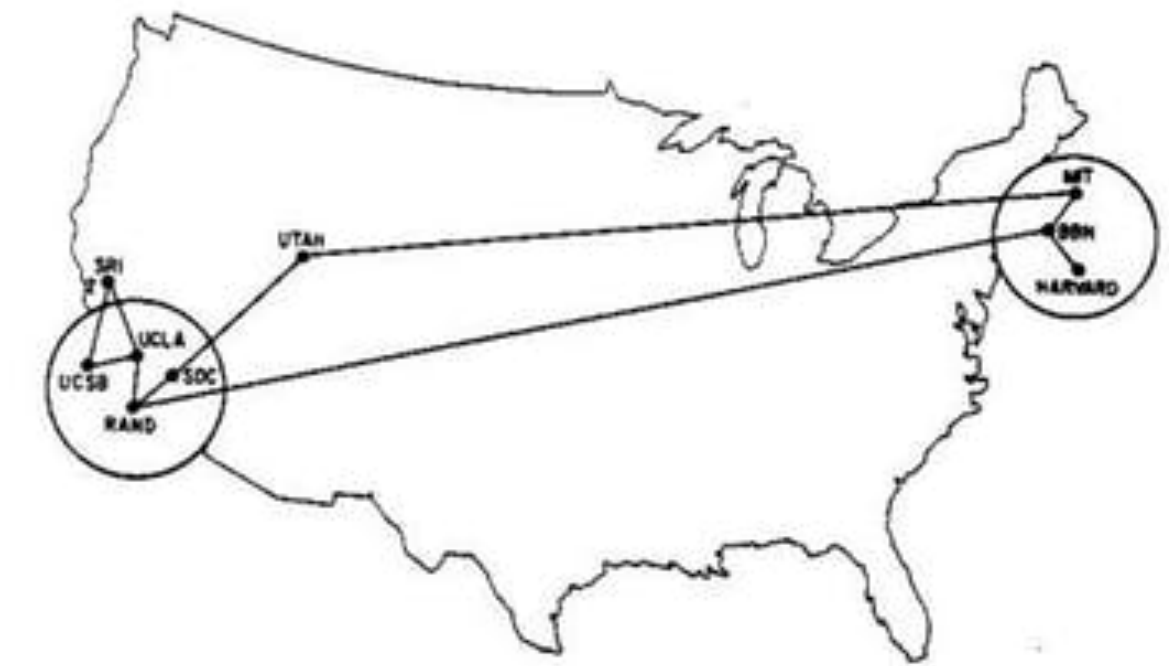
# 1960s ARPANET

**ARPANET is developed.** It is the first “packet-switching” network using TCP/IP protocol and is a precursor to World Wide Web.

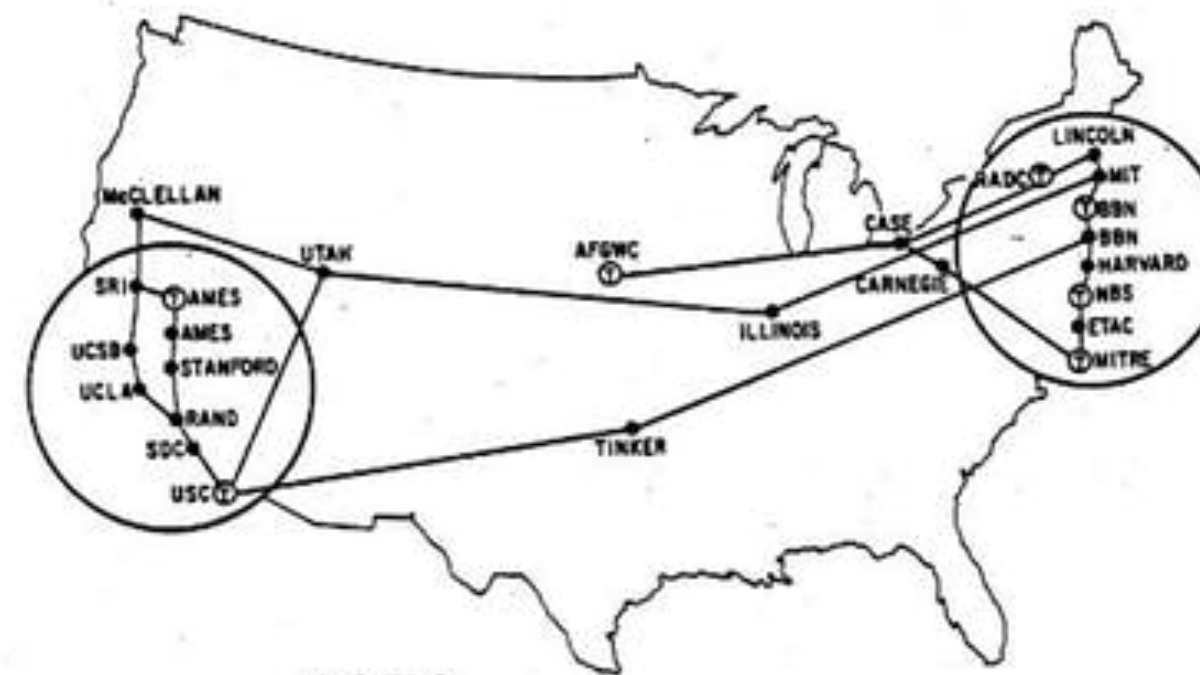
<http://en.wikipedia.org/wiki/ARPANET>



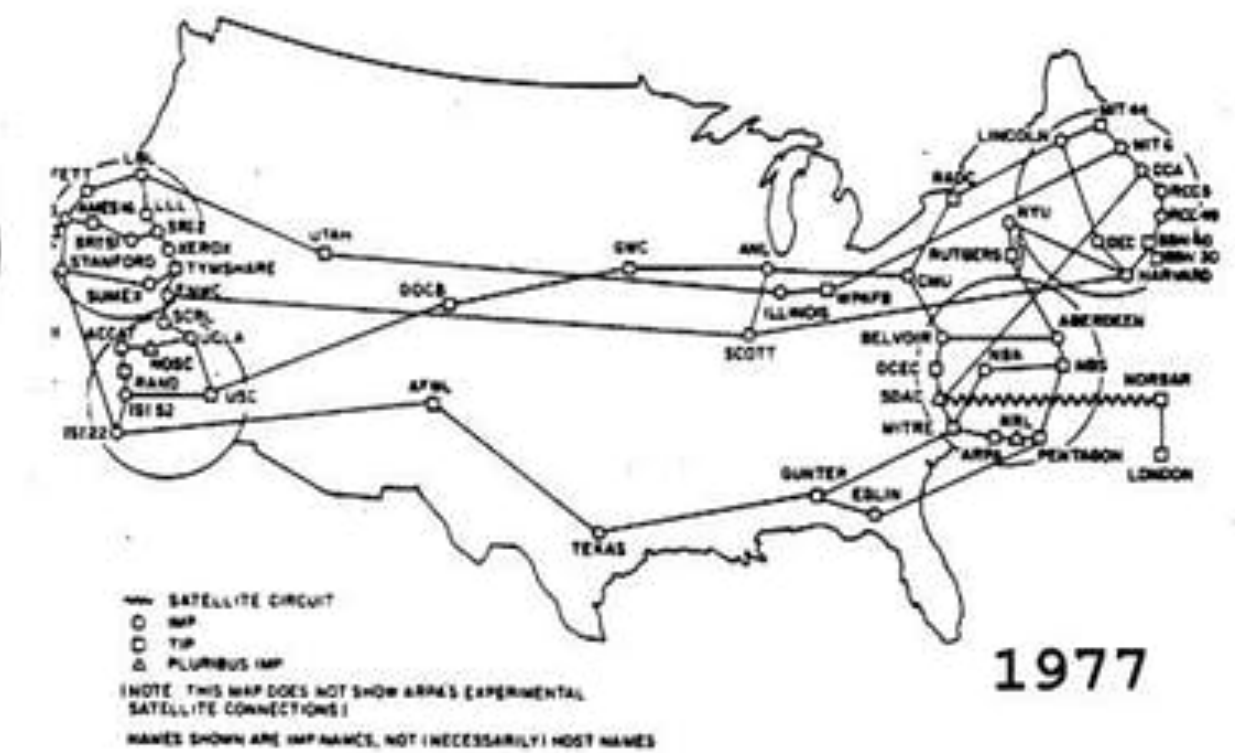
1969



1970



1972

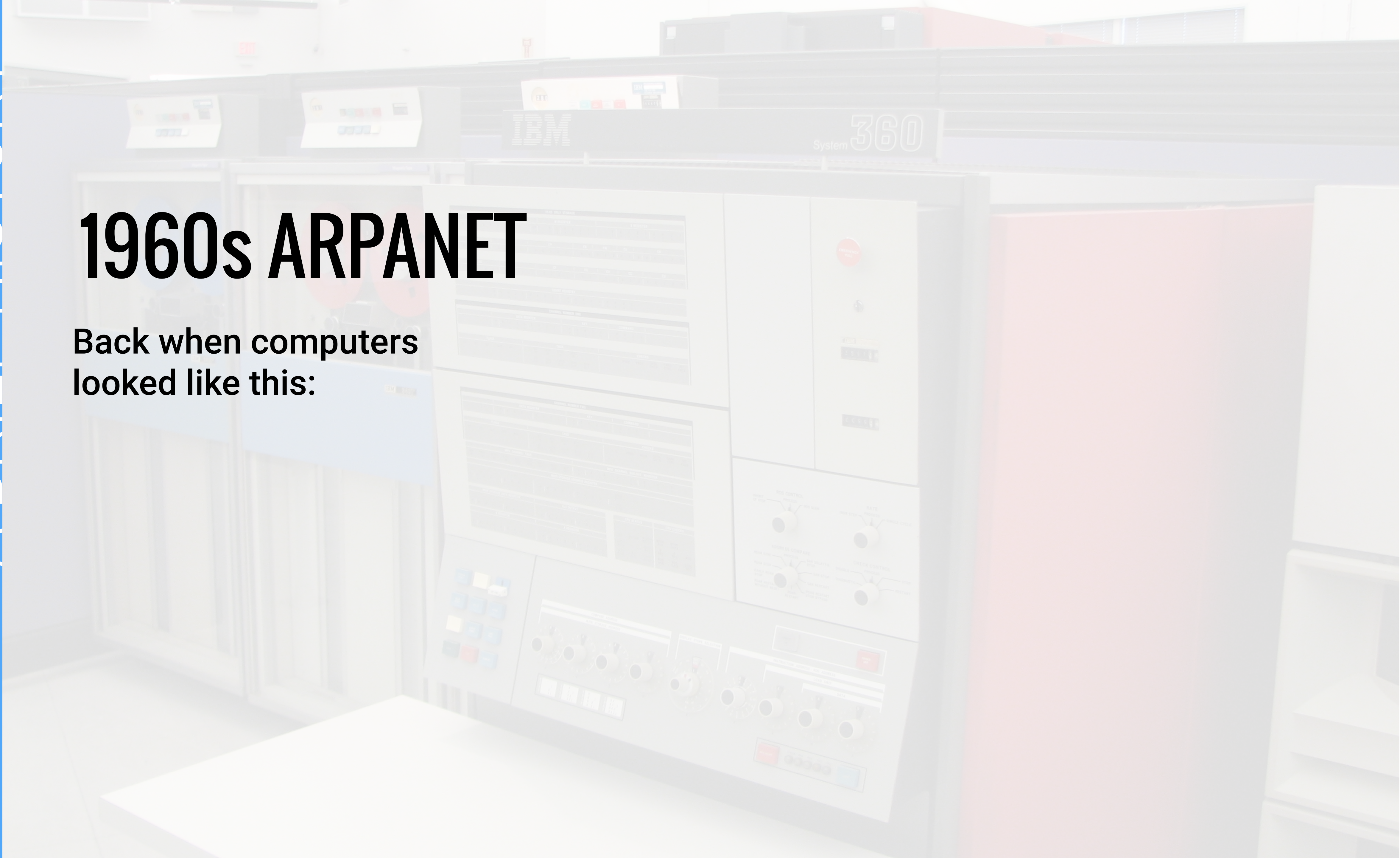


1977



# 1960s ARPANET

Back when computers looked like this:





# A BRIEF HISTORY





# COMPUTERS IN THE 1970s + 1980s

The 70s and 80s saw a large increase of consumer and business 'personal' computers, but most people still didn't have a simple way of connecting to one another.

## A BALANCE OF FEATURES

The APPLE-1 SYSTEM is a fully assembled, tested & burned-in microprocessor board using the 6502 microprocessor. The board contains processor & support hardware; complete video electronics for a 40 character/line, 24 line video display; on-board RAM capacity of 8K BYTES; software system monitor in PROM; and fully regulated power supplies. The Apple attaches directly to an ASCII encoded keyboard and a video monitor, allowing the efficient entry and examination of programs in hexadecimal notation. The use of the new 16-pin 4K RAM chips results in low power and high density memory, which can be upgraded to the 16K chips when they become available (32K bytes on-board RAM!!)

A fast (1 kilobaud) cassette interface is available and includes a tape of Apple Basic. And ... Yes, Folks. Apple Basic is Free!



**APPLE-1** **\$666.66**  
\*includes 4K bytes RAM

- Micro** • 6502 Microprocessor
- Interface** • Full video display electronics - 40 char/line, 24 line. Outputs composite video.
- Has ASCII keyboard interface on-board.
- Cassette interface board available. FAST - 1 Kilobaud.
- Memory** • Uses 16-pin 4K Dynamic RAMs.
- 8K BYTE RAM capacity on-board!
- Upgradable to 16K RAM chips.
- Software system monitor in PROM
- Basic** • Apple Basic ... pseudo-compiled. FAST, FREE.
- Power** • Fully regulated power supplies on-board.

DEALER INQUIRIES INVITED

**APPLE COMPUTER COMPANY**

770 Welch Road, Suite 154  
Palo Alto, California 94304

Phone: (415) 326-4248

CIRCLE NO. 42 ON INQUIRY CARD

JULY 1976



# COMPUTERS IN THE 1970s + 1980s

And without a large network of users and common platforms, the cost of these machines didn't make sense for most consumers and small businesses until...

## the 10-Megabyte Computer System



Only  
**\$5995**  
COMPLETE

*New From IMSAI®*

- 10-Megabyte Hard Disk
  - 5¼" Dual-Density Floppy Disk Back-up
  - 8-Bit Microprocessor  
(Optional 16-bit Microprocessor)
  - Memory-Mapped Video Display Board
  - Disk Controller
  - Standard 64K RAM  
(Optional 256K RAM)
  - 10-Slot S-100 Motherboard
  - 28-Amp Power Supply
  - 12" Monitor
  - Standard Intelligent 62-Key ASCII Keyboard (Optional Intelligent 86-Key ASCII Extended Keyboard)
  - 132-Column Dot-Matrix Printer
  - CP/M\* Operating System
- You Read It Right ...  
All for \$5995!*

**IMSAI**® ... Thinking ahead for the 80's

415/635-7615

Computer Division of the Fischer-Freitas Corporation

910 81st Avenue, Bldg. 14 • Oakland, CA 94621

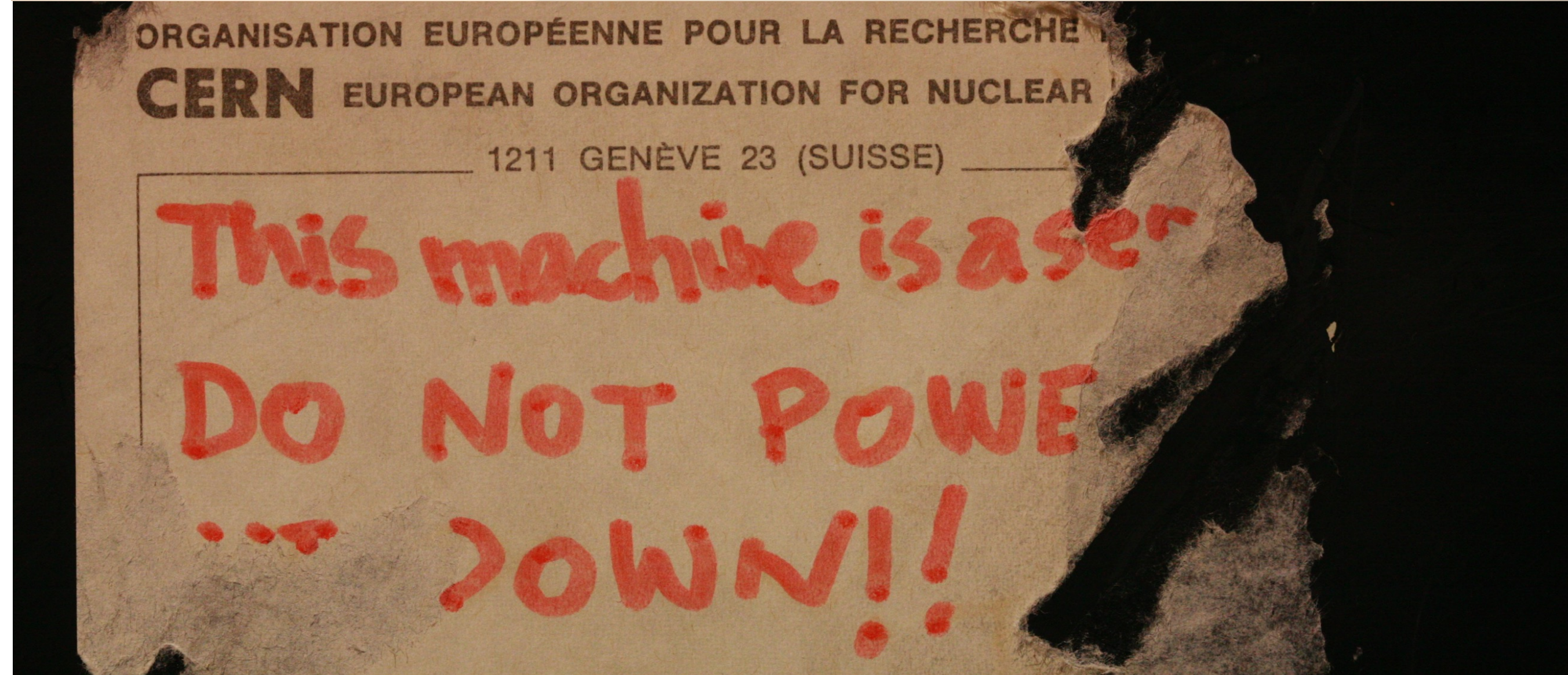
\*CP/M is a trademark of Digital Research. Im Sai is a trademark of the Fischer-Freitas Corporation



# WORLD WIDE WEB CONCEIVED 1990

**Tim Berners-Lee** created what we now know as the internet, or "world wide web," in 1990, although he proposed it in 1989.

*right: the first web server*

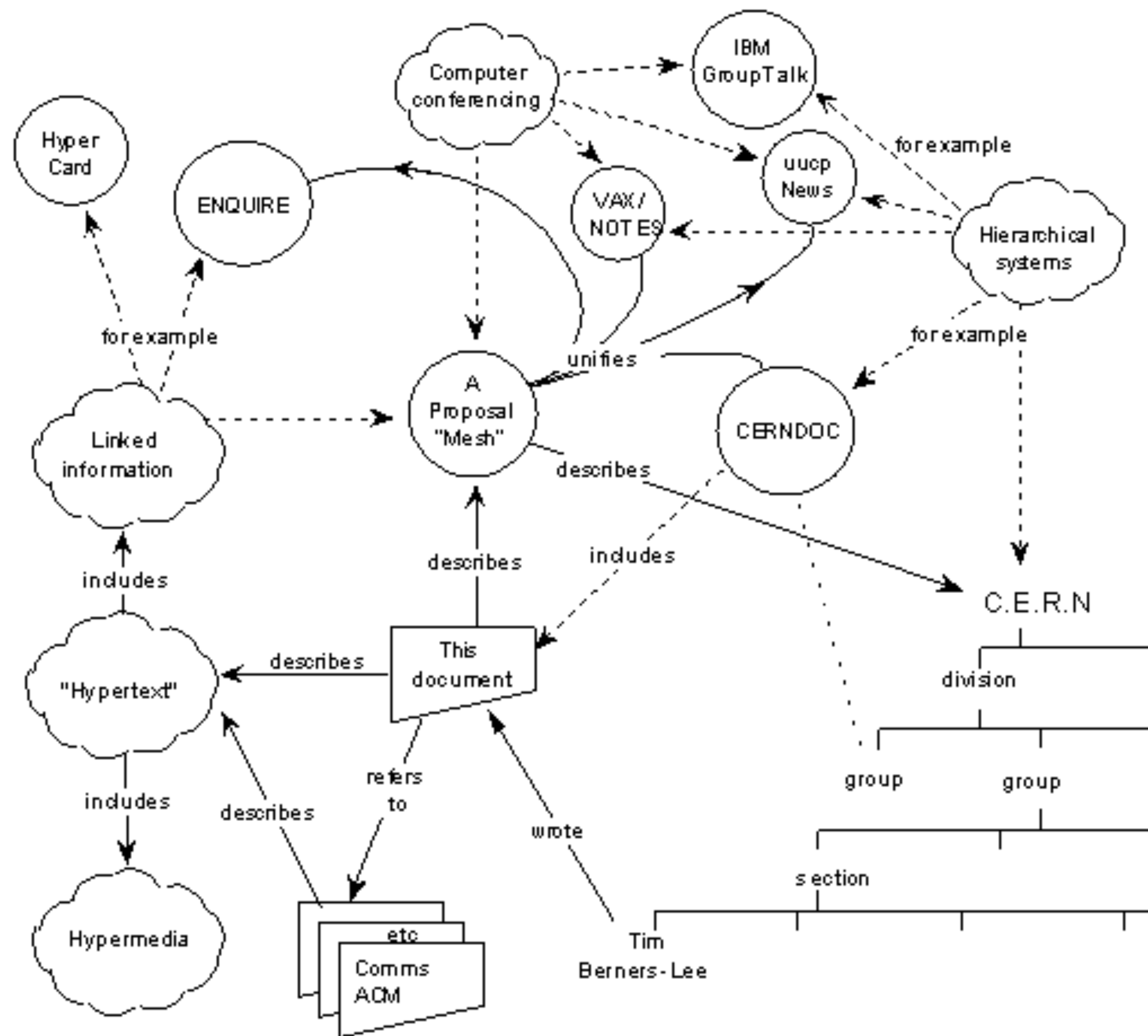




# HTTP:// PROTOCOL

Tim Berners-Lee's HyperText Transfer Protocol (HTTP) is what allows computers to deliver and receive web traffic.

*right: HTTP idea*

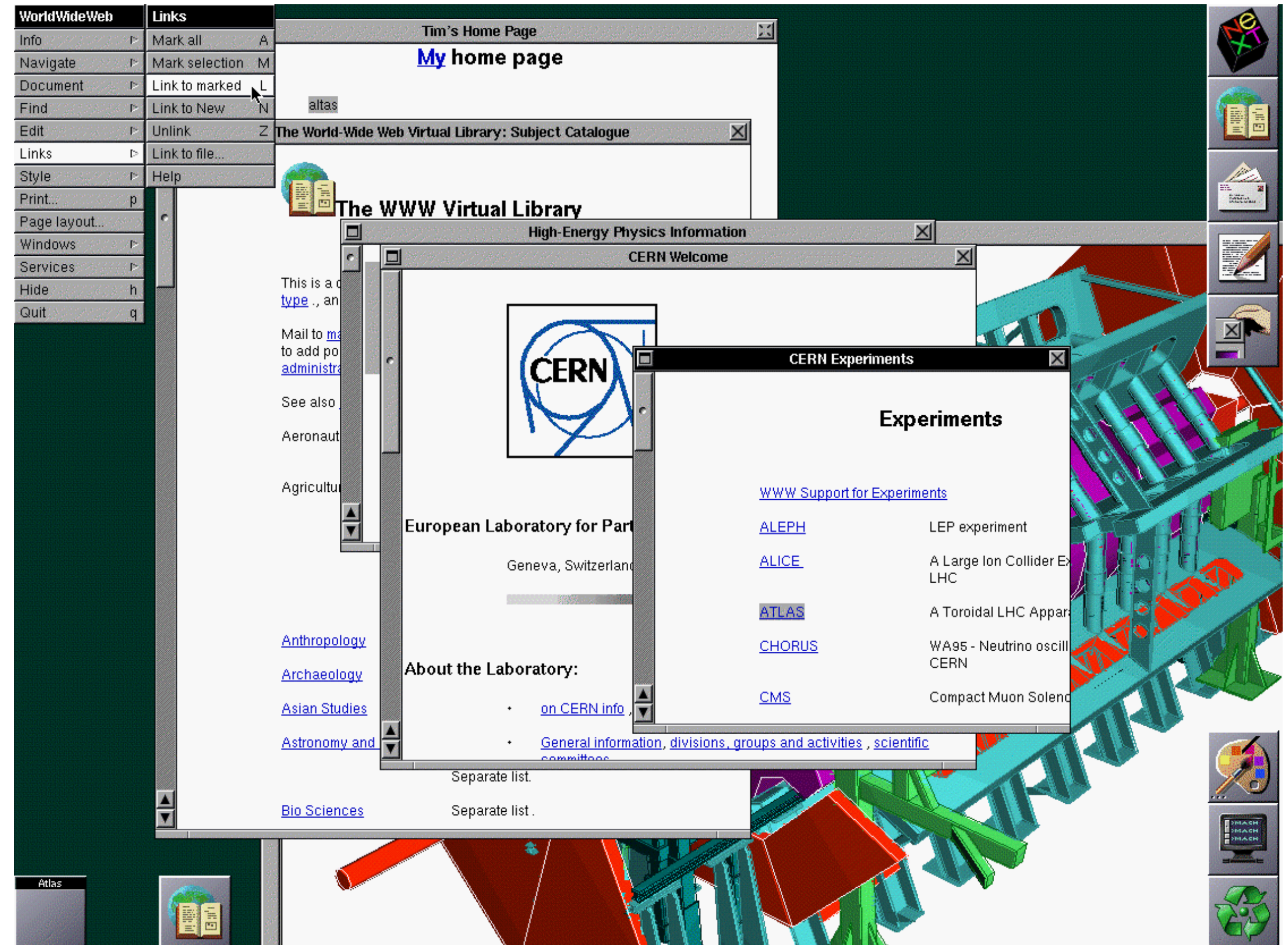




# HTML LANGUAGE, FIRST WEB BROWSER

Tim Berners-Lee also created HTML (Hypertext Markup Language), which is still the underlying structure of all web documents today. To view web documents, he also created the first web browser in 1990.

*right: first web browser*

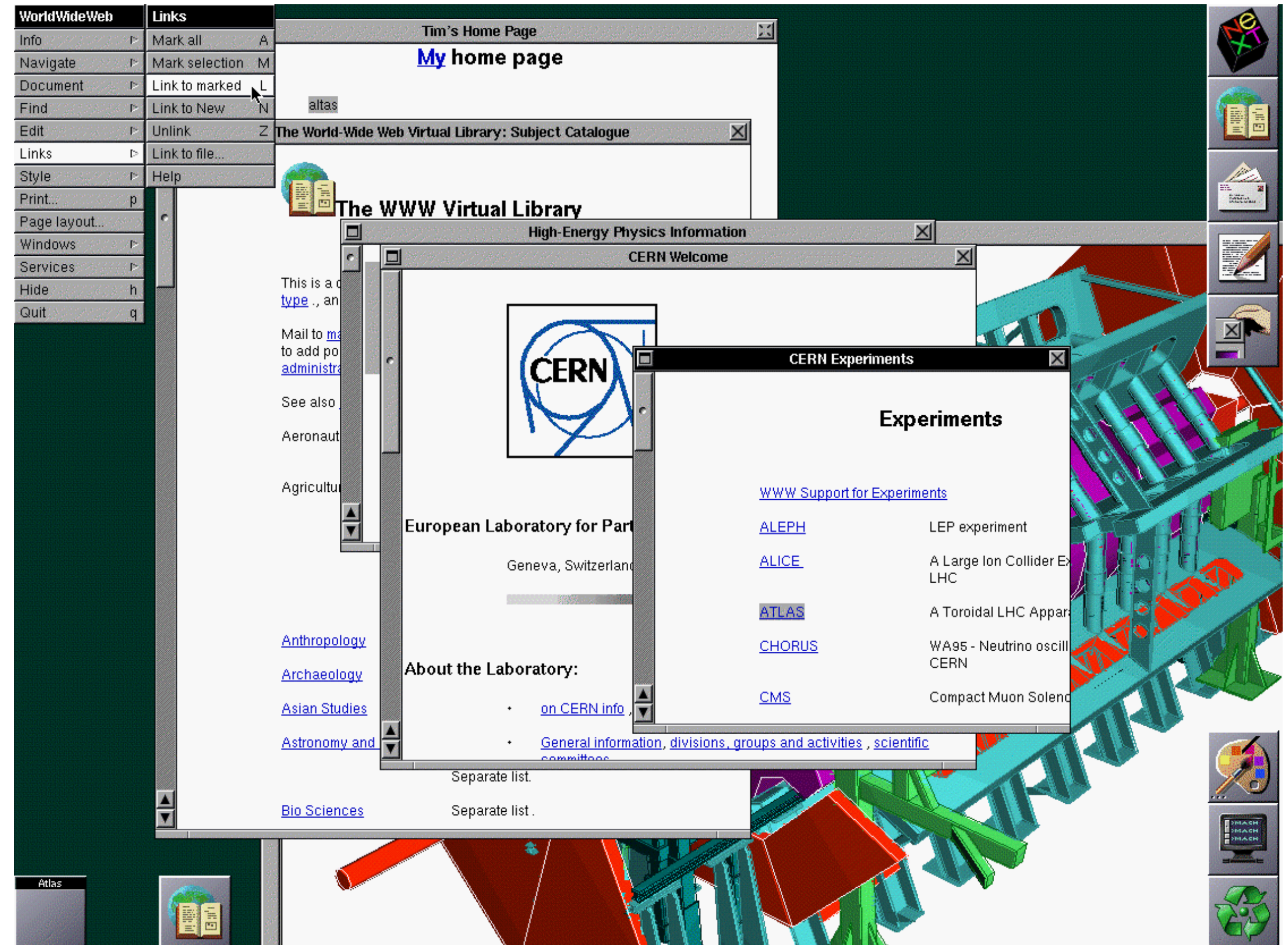




# 1990s, EXPLOSION WEB OF USERS

With a standardized platform and language for **communication**, web traffic became within reach for normal consumers, especially as computer technologies advanced to make devices more affordable and practical.

*right: first web browser*





# FIRST E-COMMERCE TRANSACTION, 1994

Pizza Hut sold the first web-purchased product through its transaction service, "NetMarket." Check out the lovely site on the right.

<https://www.pizzahut.com/assets/pizzanet/home.html>

Document Title: Welcome to PizzaNet!  
Document URL: http://www.pizzahut.com/



## Welcome to PizzaNet!

PizzaNet is Pizza Hut's Electronic Storefront and is brought to you by Pizza Hut® and The Santa Cruz Operation®. You may click on the Pizza Hut logo on any page to submit comments regarding PizzaNet to [webmaster@Pizzahut.COM](mailto:webmaster@Pizzahut.COM)

If you would like to order a pizza to be delivered, please provide the following information:

Name:

Street Address:

Voice Phone ###-###-####  
 (where we can reach you)





# 2002, XHTML

As commercial uses for the web took off, increasing demands for standards were becoming necessary. HTML became stricter so it could incorporate XML (Extensible Markup Language) docs and be more **SEMANTIC** and **ADA Accessible**.





## ~2004-2007, FIRST PRACTICAL INTERNET CELL PHONES

With the increase in cell phone and PDA use, internet use became the next step in creating universal access.

Because Apple refused to support Flash Player on iPhones, javascript because crucial for rich animated content. **Enter HTML5.**



iPhone 2007, the game changer.



Nokia 2004



## 2007, HTML5

The HTML5 specification was adopted as the starting point for the new HTML working group in 2007. New markup was created to offer better database integration, semantic markup, javascript integration, and better video capabilities. It created better usability on all devices.

# HTML





# SO MANY DEVICES

After the introduction of the **first iPhone**, device sizes and types exploded, making it more important than ever for designers to rethink their standard way of writing code and designing for the web.





# NOW: UNIVERSAL DESIGN

Now we are at a place where HTML, CSS, and JavaScript are used in practically any internet-connected device, both big and small.





# Hypertext Markup Language

**Current version of HTML is HTML5.**

Prior to HTML5 the standard was XHTML, which was the first to offer strict semantic markup and a focus on separating “presentation from structure.”

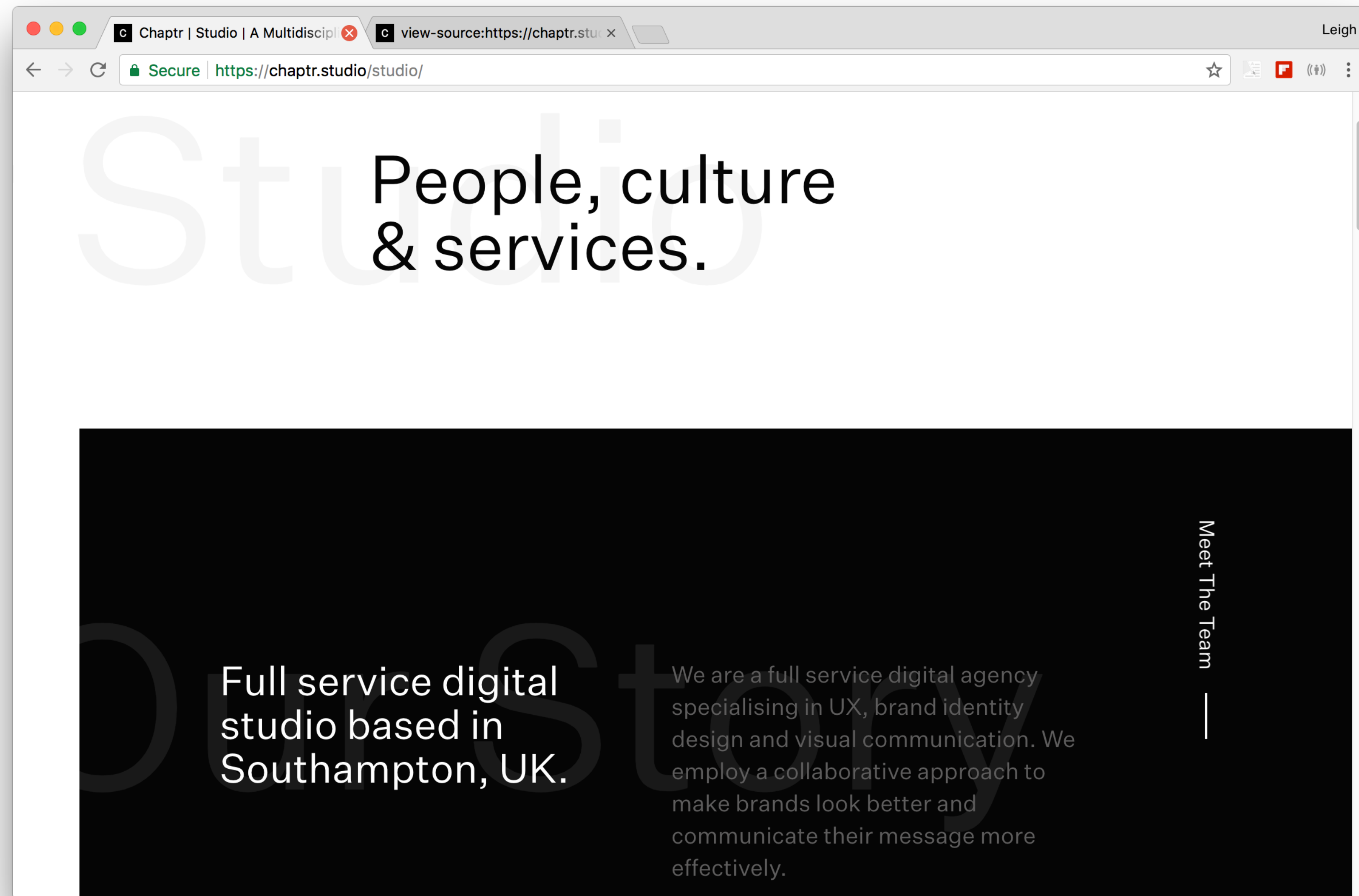


# What HTML looks like

Display View vs. Code View...

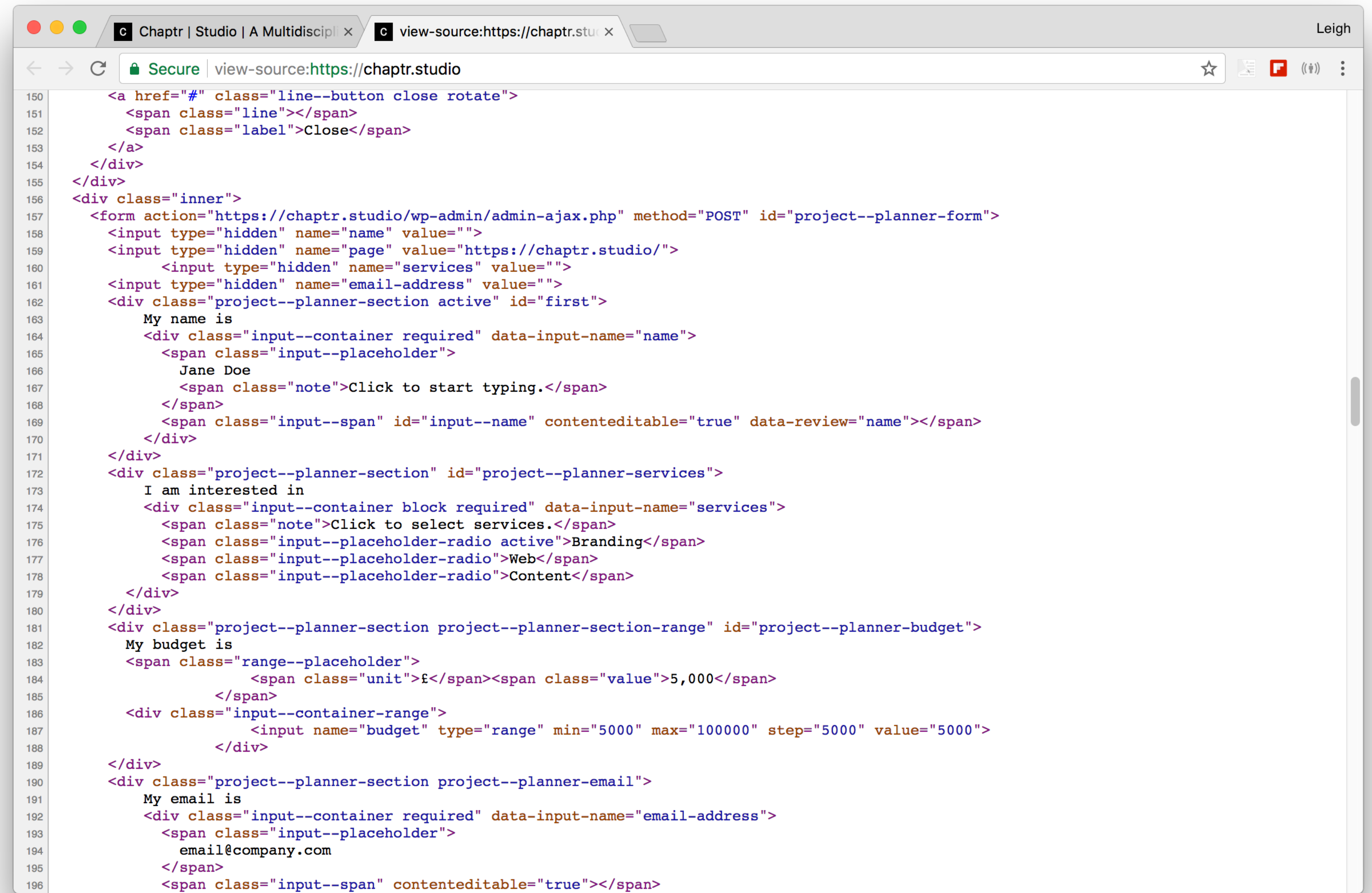


Here's what we see as a viewing audience:





Here's what we do as web designers and developers:



```
150 <a href="#" class="line--button close rotate">
151   <span class="line"></span>
152   <span class="label">Close</span>
153 </a>
154 </div>
155 </div>
156 <div class="inner">
157   <form action="https://chaptr.studio/wp-admin/admin-ajax.php" method="POST" id="project--planner-form">
158     <input type="hidden" name="name" value="">
159     <input type="hidden" name="page" value="https://chaptr.studio/">
160     <input type="hidden" name="services" value="">
161     <input type="hidden" name="email-address" value="">
162     <div class="project--planner-section active" id="first">
163       My name is
164       <div class="input--container required" data-input-name="name">
165         <span class="input--placeholder">
166           Jane Doe
167           <span class="note">Click to start typing.</span>
168         </span>
169         <span class="input--span" id="input--name" contenteditable="true" data-review="name"></span>
170       </div>
171     </div>
172     <div class="project--planner-section" id="project--planner-services">
173       I am interested in
174       <div class="input--container block required" data-input-name="services">
175         <span class="note">Click to select services.</span>
176         <span class="input--placeholder-radio active">Branding</span>
177         <span class="input--placeholder-radio">Web</span>
178         <span class="input--placeholder-radio">Content</span>
179       </div>
180     </div>
181     <div class="project--planner-section project--planner-section-range" id="project--planner-budget">
182       My budget is
183       <span class="range--placeholder">
184         <span class="unit">f</span><span class="value">5,000</span>
185       </span>
186       <div class="input--container-range">
187         <input name="budget" type="range" min="5000" max="100000" step="5000" value="5000">
188       </div>
189     </div>
190     <div class="project--planner-section project--planner-email">
191       My email is
192       <div class="input--container required" data-input-name="email-address">
193         <span class="input--placeholder">
194           email@company.com
195         </span>
196         <span class="input--span" contenteditable="true"></span>
```



# Tags

**HTML is based on the use of TAGs.**

TAGs are the beginnings and endings of element declarations. They tell the browser how to process the given data.



# Tags

The basic anatomy of a TAG:

**<tagname>**Blah blah blah**</tagname>**

(BTW, this is a dummy, not-real example)



# Tags

It could also look like this:

```
<tagname>
```

```
    Blah blah blah
```

```
</tagname>
```

- The excess whitespace outside of tags does not matter.
- The "Blah blah blah" portion will only display a SINGLE space character in between words, regardless of how many consecutive spaces or tabs you type.



# Tags

Almost all tags have an **opening**...

**<tagname>**

Blah blah blah

**</tagname>**



# Tags

Almost all tags have an opening and **closing tag**.

**<tagname>**

Blah blah blah

**</tagname>**

The only difference between an opening and closing tag is that the **closing tag starts with a “/” forward-slash inside the angle brackets**.



# Tags

A real example of a "paragraph" tag:

`<p>`

This is where you type the paragraph text you want to show up on the screen.

`</p>`



# Tags

These are real examples of “**heading level**” tags:

**<h1>**I’m the most important heading.**</h1>**

**<h2>**I’m important heading level #2.**</h2>**

...

**<h6>**I’m the least important heading level #6.**</h6>**

(Heading levels have six levels, #1 being the highest in hierarchy.)



# Tags

These are real examples of “**unordered list**” and “**list item**” tags:

```
<ul>
```

```
  <li>I'm a list item inside of a list.</li>
```

```
  <li>I'm a list item inside of a list.</li>
```

```
</ul>
```

(Unordered lists show up with bulleted list items.)

# Tags

These are real examples of “**ordered list**” and “**list item**” tags:

```
<ol>
```

```
  <li>I'm a list item inside of a list.</li>
```

```
  <li>I'm a list item inside of a list.</li>
```

```
</ol>
```

(Ordered lists have numbers by default, indicating sequence.)



# Tags

But there are some tags that are self-closing (also called "empty elements"). Below are all valid ways of writing these example empty elements in HTML5:

An image file:

```

```

```

```

A line break:

```
<br />
```

```
<br>
```

Horizontal rule (a line):

```
<hr />
```

```
<hr>
```

# Tags

...and the **TAG** list goes on:

<html>  
<head>  
<title>  
<base>  
<link>  
<meta>  
<style>  
<body>  
<section>  
<nav>  
<article>

<aside>  
<header>  
<footer>  
<address>  
<p>  
<hr>  
<pre>  
<blockquote>  
<ol>  
<ul>  
<li>

<dl>  
<dt>  
<dd>  
<figure>  
<figcaption>  
<div>  
<a>  
<em>  
<strong>  
<small>

<cite>  
<q>  
<abbr>  
<data>  
<code>  
<img>

...AND MANY  
MANY MORE.



# Tags

A **TAG** can also be referred to as an **ELEMENT**.

Here is a GREAT resource list of HTML5 elements with explanations of each one is used:

[https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5\\_element\\_list](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5/HTML5_element_list)

# Tags and Their Parts

A **TAG** can also have **ATTRIBUTES**.

`<a href="http://google.com">Look something up here!</a>`



# Tag Attributes

A **TAG** can also have **ATTRIBUTES**.

`<a href="http://google.com">Look something up here!</a>`



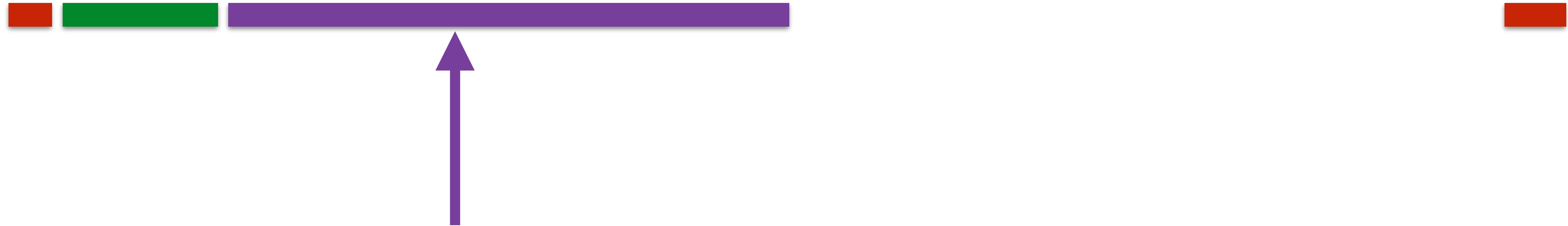
**ATTRIBUTES** are information within the tag, defining the tag.

Different tags have different attribute options. This example is a "hyper reference" telling us where the "anchor" link will take us when clicked.

# Attribute Values

A **TAG** **ATTRIBUTES** have **VALUES**.

`<a href="http://google.com">Look something up here!</a>`



The diagram shows the HTML code `<a href="http://google.com">Look something up here!</a>` with color-coded components: `<a` is red, `href="` is green, `http://google.com"` is purple, and `>` is red. A purple arrow points from the text below to the purple value. Small red bars are under the opening and closing tags.

**VALUES** provide pinpoint detail about the attribute. In this example the value of "http://google.com" is WHERE the "hyper reference" will take us when clicked. "Look something up here!" is the visible, clickable link on the web page.



# Nesting Tags

To make pages, we have to “nest” tags within each other to develop page structure and hierarchy. An example of nesting:

```
<article>
```

```
  <header>
```

```
    <h1>Print vs. Web</h1>
```

```
  </header>
```

```
    <p>Although web delivery is efficient and inexpensive, printed materials can give a sense of....</p>
```

```
</article>
```

The "article" is the **parent** of the "header" and "paragraph" tags. The "heading level 1" is the **child** of the "header". H1, HEADER, and PARAGRAPH are all **descendants** of the "ARTICLE, and conversely, the ARTICLE is the **ancestor** of all other tags above.

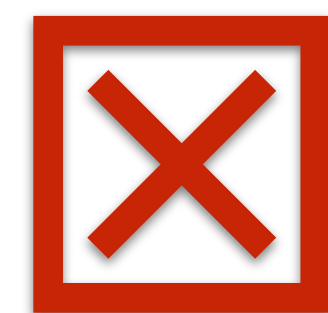
# Nesting Tags

You have to nest tags PROPERLY for reliable and valid results:

```
<p><a href="#part1">Go to Part 1</a></p>
```



```
<p><a href="#part1">Go to Part 1</p></a>
```



It is extremely important to end tag elements in the proper order when nesting. The first example above is correct, nesting the anchor link entirely within the open and closing paragraph tags. The second example is WRONG, ending the tags out of order.



# **SPEAKING OF PROPER NESTING...**

Let's look at the basic, required parts of any HTML page...

# Document Requirements

Minimum HTML5 requirements are:

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <title>Title goes here</title>  
  </head>  
  
  <body>  
  </body>  
</html>
```



# Document Requirements

Only one instance of each following element can and must be in a page.

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
  <head>
```

```
    <title>Title goes here</title>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```

# Requirements : <!DOCTYPE ...>

What is !DOCTYPE ????

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
  <head>
```

```
    <title>Title goes here</title>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```



# Requirements : <!DOCTYPE ...>

What is !DOCTYPE ????

## <!DOCTYPE html>

- The “doctype” announces to a browser which version of HTML it is using. This helps the browser know what to expect and how to behave.
- Without a “doctype”, browsers go into “quirks mode” and can interpret code unpredictably.
- Analogy:** If we look at HTML page elements as if it’s a person, the DOCTYPE would be like the social standards a person is expected to follow. The browser interprets the HTML based on that DOCTYPE condition, sort of like how you’d interpret what is acceptable behavior at home vs. work. People might respond unpredictably if you showed up to work in pajamas, but perhaps no one would even notice at home.

# Requirements : <html>

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
  <head>
```

```
    <title>Title goes here</title>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```



# Requirements : <html>

```
<html lang="en"> ... </html>
```

The “html” tag represents the root of an HTML document. All other elements (except “doctype”) must be descendants of this element.

**Analogy:** If compared to a human being, the <html> element is like meeting the “whole person.”

**NOTE:** The “language” attribute is now a requirement for the <html> tag. Without it, your browser will ask if you want a translation of the page.

# Requirements : <head>

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
  <head>
```

```
    <title>Title goes here</title>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```



# Requirements : <head>

<head>...</head>

The "head" tag contains a collection of metadata about the document, including links to, or definitions of, scripts and style sheets. **At a minimum, it MUST contain a "title" tag.**

**Analogy:** The head tag is like the brain of the page that pre-processes information so that it can understand how to style and interpret what it encounters within the page before interacting with you.

# Requirements : <title>

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
  <head>
```

```
    <title>Title goes here</title>
```

```
  </head>
```

```
  <body>
```

```
  </body>
```

```
</html>
```



# Requirements : <title>

<title>Title goes here</title>

- This tag defines the title of the document, shown in a browser's title bar or on the page's tab at the top of the window.
- It can only contain text. Nested HTML tags will not be interpreted.
- **Analogy:** this page identity is is akin to a person telling you his/her name.... "Hi, my name is Such-and-Such."

# Requirements : **<body>**

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
  <head>
```

```
    <title>Title goes here</title>
```

```
  </head>
```

```
    <body>
```

```
    </body>
```

```
</html>
```

# Requirements : <body>

<body> ... </body>

- Contains the content of an HTML document that is **visually displayed** in the browser's viewport/window.
- **Analogy**: This is like the outward appearance, gestures, speech, and behavior of a person. It's what we get to see and experience as an audience.



# Requirements : Technical Recap

```
□ <!DOCTYPE html>  
┌ <html lang="en">  
│   ┌ <head>  
│   │   □ <title>Title goes here</title>  
│   └ </head>  
│   ┌ <body>  
│   └ </body>  
└ </html>
```

# Requirements : Metaphorical Recap

HTML tags are fairly intuitive. An easy way to remember the basic parts is to humanize a page:

**Doctype** = laws governing behavior

**<html>** = the whole person

**<head>** = the brain that gathers information

**<title>** = name / identity a person calls one's self

**<body>** = the physical representation and behaviors of the person

# **SPEAKING OF INTUITIVE...**

Let's talk about "SEMANTIC" html markup.



# Semantic

***adjective*** \si-'man-tik\  
\si-'man-tik\

: of or relating to the meanings of words and  
phrases

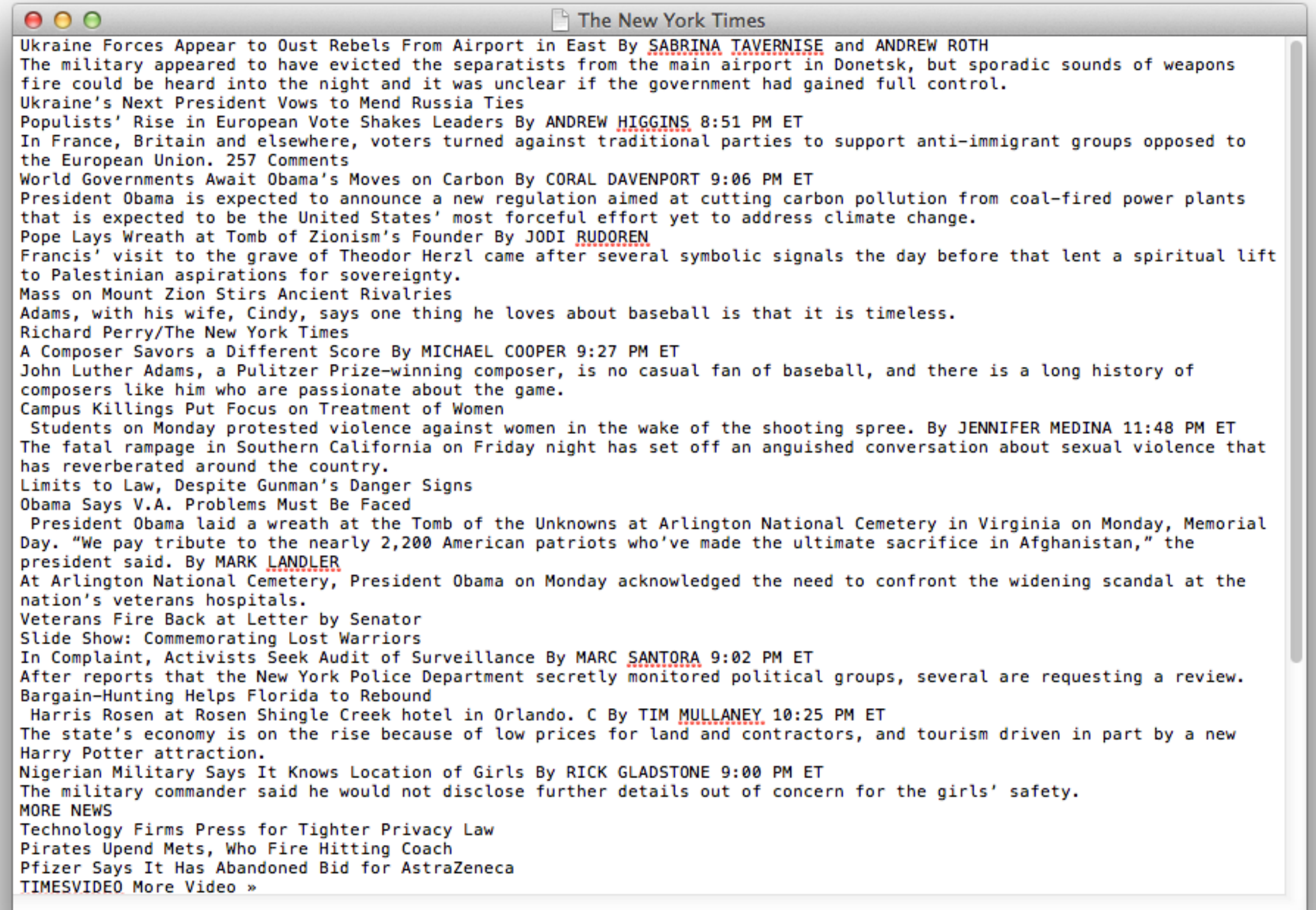
# Semantic Tags in HTML

1. HTML tags are imbued with meaning.
2. They should not be arbitrarily assigned.
3. They have meaning outside of visual representation.
4. Visually impaired people can still understand the meaning of sections in a page with semantic markup.
5. Search engines can also understand meaning imbued in semantic markup.

# A "semantic" Case Study

Imagine that you are reading a newspaper but there are no headings, no margins, no indents, no bold, no italics, no paragraph separations, etc.

...It's **JUST TEXT**:

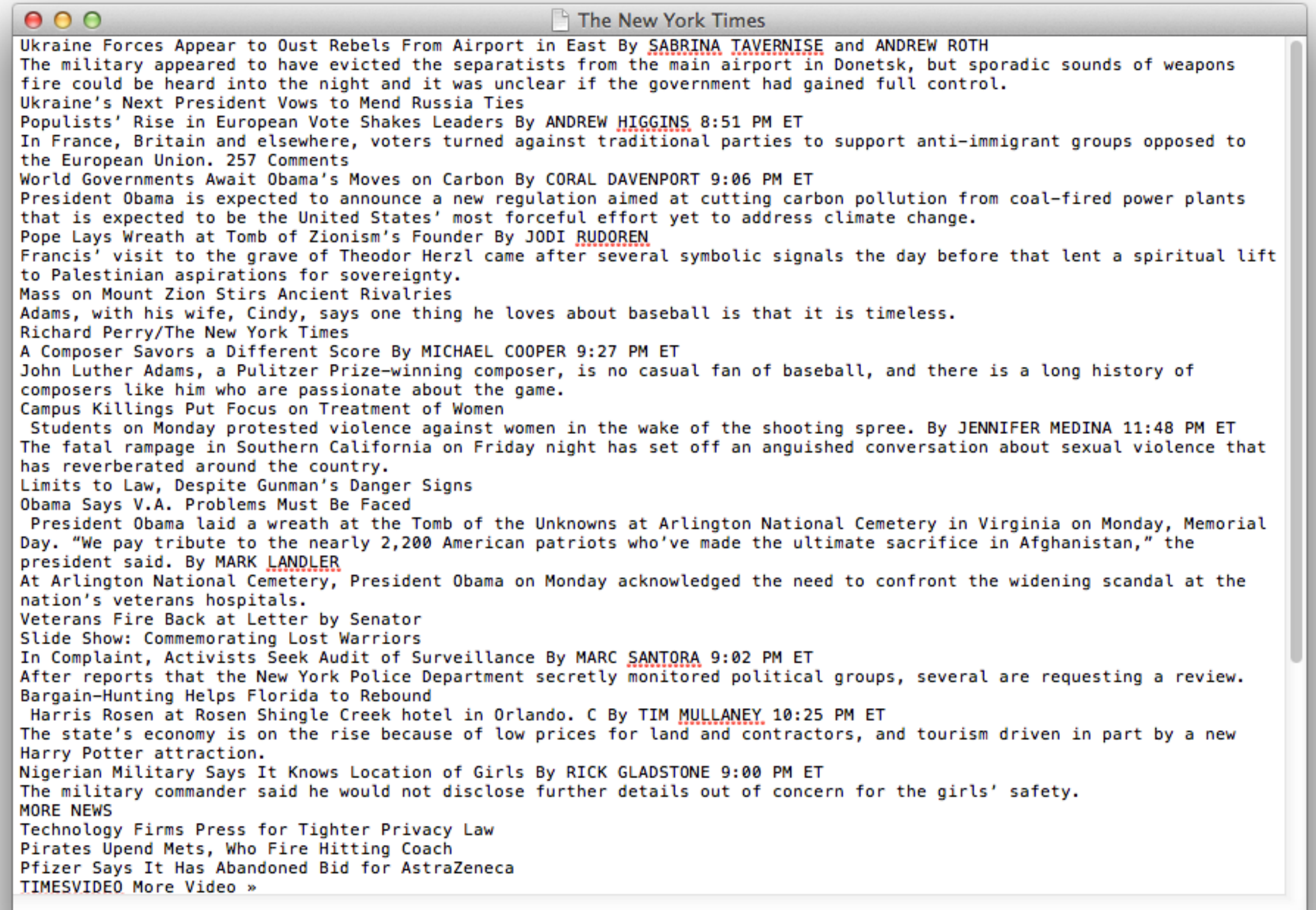




# A "semantic" Case Study

Without meaningful hierarchy, no one would want to read through that page because it would be a laborious experience.

**This is how search engines and blind people experience pages with poor semantic markup.**





# A "semantic" Case Study

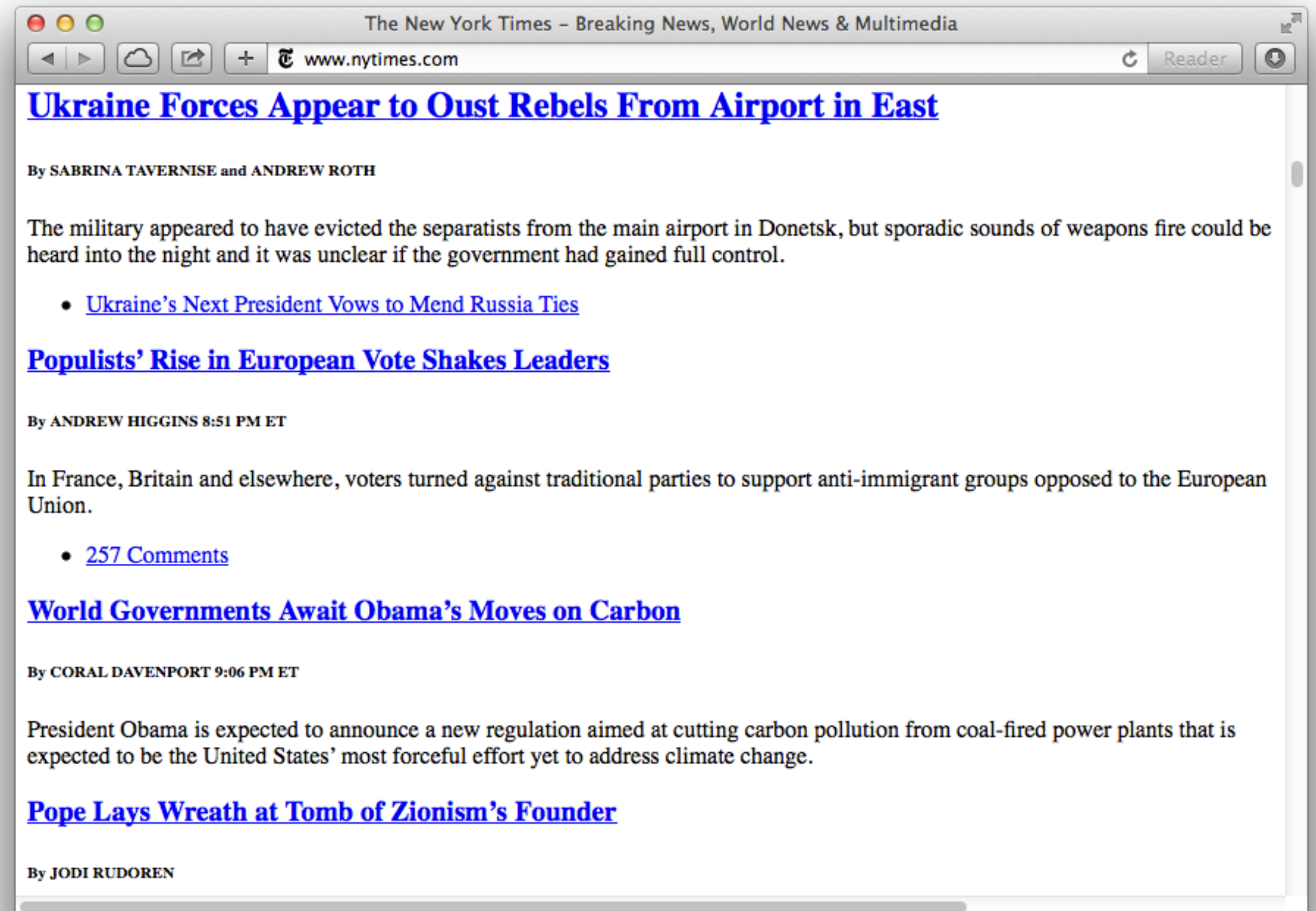
Because we have writing conventions to help organize our thoughts, our **HTML structure** should also resemble that structure for clarity.

**<h1>** = most important heading

**<p>** = separates complete thoughts into paragraphs

**<li>** = an item in a list

**AND SO ON...**





# A "semantic" Case Study

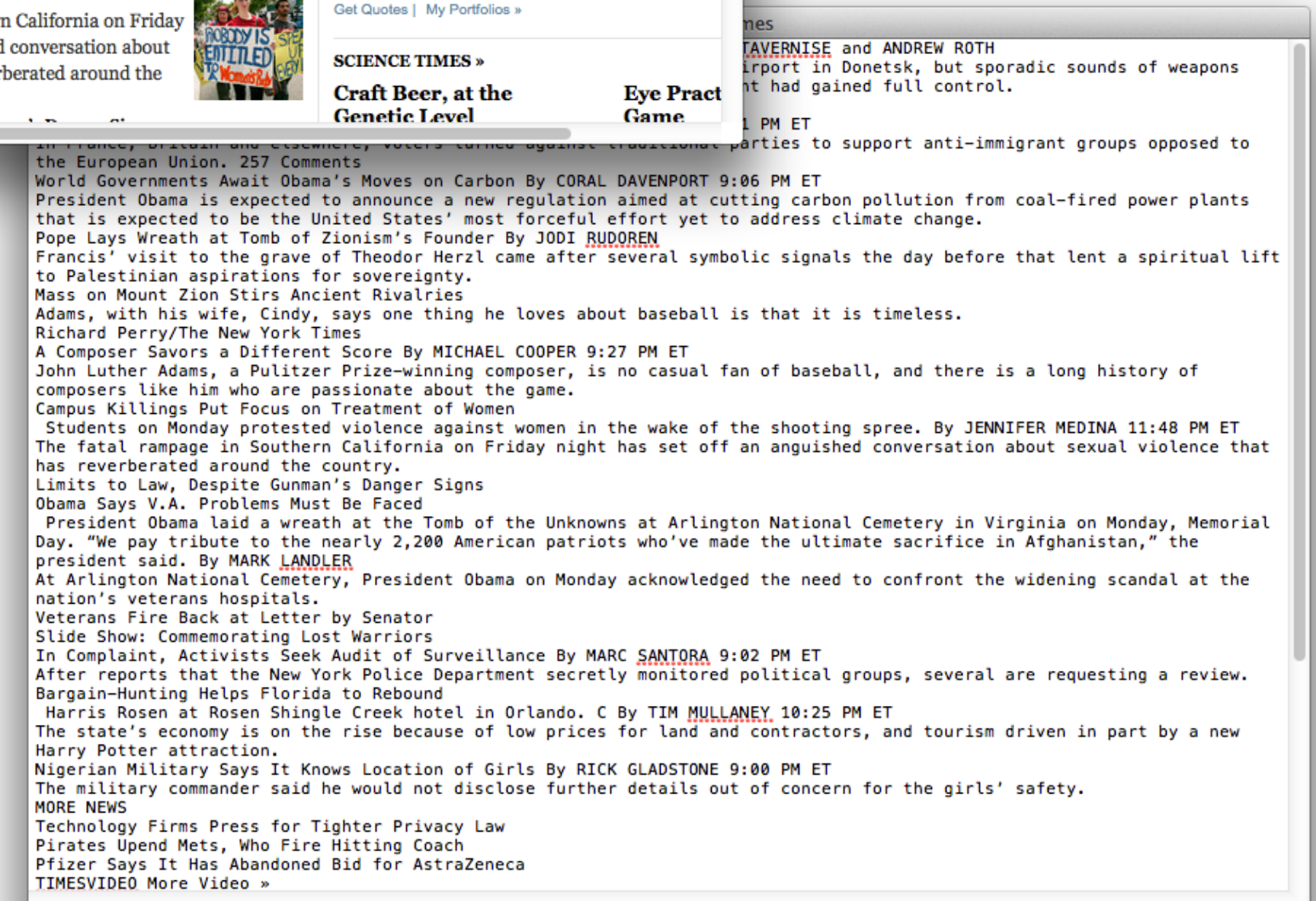
This might seem intuitive, but instead of semantic tags, some designers have terrible habits of using almost all **non-semantic markup**, such as `<div>` tags and `<span>` tags and simply styling them with CSS to look attractive.

But that doesn't mean screen readers can interpret ANY meaning from them whatsoever.



Looks great with CSS....

But this could be the same non-semantic site viewed by a screen reader.





That brings us to the topic of

# **STRUCTURE vs. PRESENTATION**

# HTML Structure

The **STRUCTURE** of an HTML document should be a well-crafted page that **can be understood semantically** by search engines and assistive devices like screen readers (aids for the blind).

# HTML Structure

Additionally, **STRUCTURE** also incorporates the actual **CONTENT of the document** (e.g. headings, article text, links, pictures, video, etc).



# CSS Presentation

The **PRESENTATION** half of the coin uses **VISUAL STYLING** to communicate the semantics to people not using screen readers.

This is accomplished by pairing **CSS** with the structure.

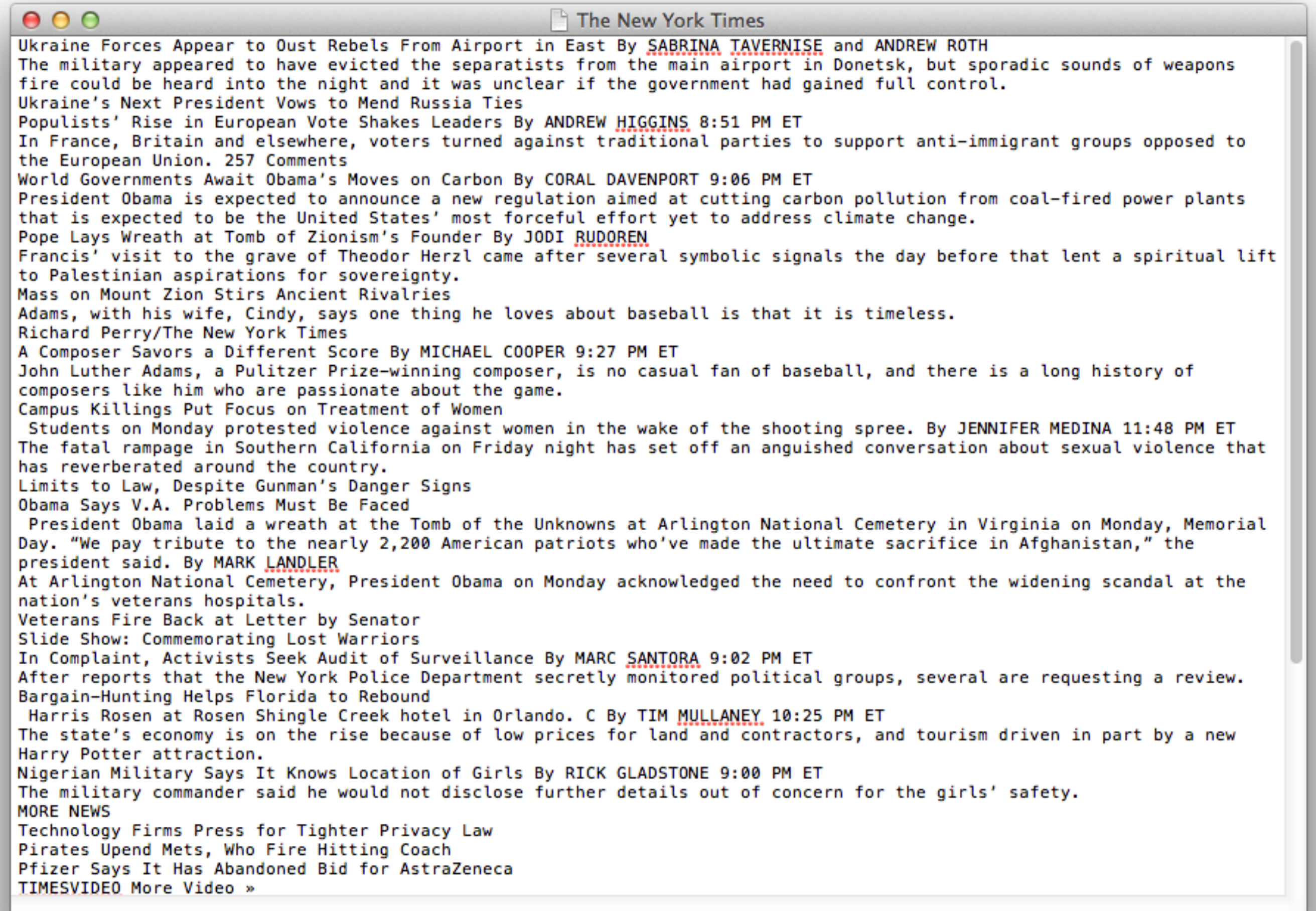
# Structural Semantics + Presentation

To make a point about STRUCTURE vs. PRESENTATION, let's revisit our former example....

# No Semantic Markup, No CSS

This page is an example of a page with no semantic markup and thus poor structure.

Furthermore, it has no styling.

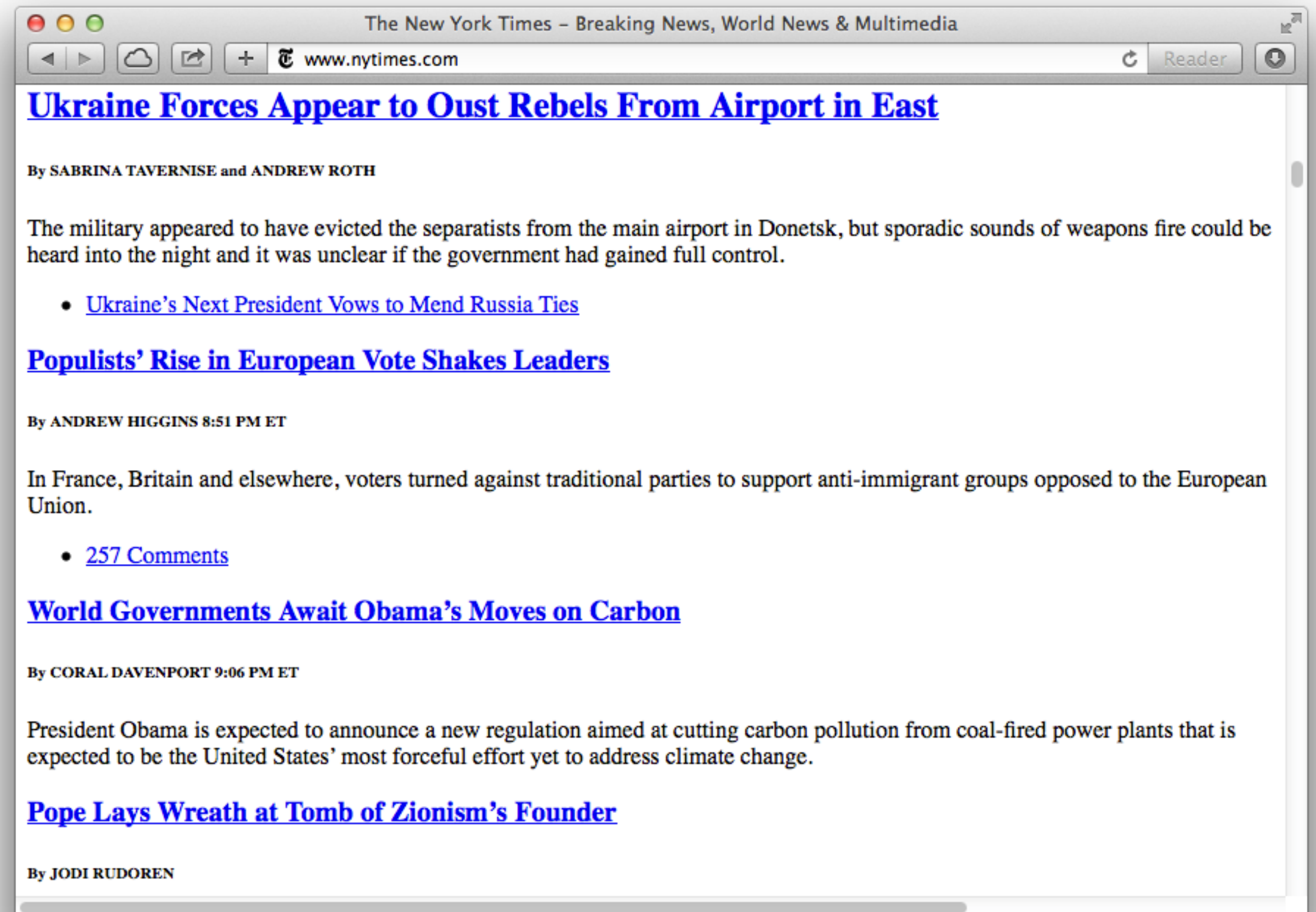




# Reorganized with Semantic Markup

The next step is figuring out a logical organization of the content and marking it up with all semantic tags.

Even without adding CSS styling yet, we can have a more meaningful page technically readable by search engines, screen readers, and sighted people.





# Now Add the Presentation Layer

This is the same site but with the **addition of CSS**, which adds more visual hierarchy to the underlying semantic markup.

This is clearly the best experience for ALL 'users':

- **sight-impaired (screen readers)**
- **seeing people**
- **search engines**

The screenshot shows a browser window displaying the New York Times website. The browser title is "The New York Times - Breaking News, World News & Multimedia" and the address bar shows "www.nytimes.com". The website features a navigation menu with categories like WORLD, U.S., NEW YORK, OPINION, BUSINESS, TECHNOLOGY, SCIENCE, HEALTH, SPORTS, ARTS, FASHION & STYLE, and VIDEO. The main content area includes several articles:

- Ukraine Forces Appear to Oust Rebels From Airport in East** by Sabrina Tavernise and Andrew Roth. The article text states: "The military appeared to have evicted the separatists from the main airport in Donetsk, but sporadic sounds of weapons fire could be heard into the night and it was unclear if the government had gained full control." Below the article is a link: "Ukraine's Next President Vows to Mend Russia Ties".
- Populists' Rise in European Vote Shakes Leaders** by Andrew Higgins. The article text states: "In France, Britain and elsewhere, voters turned against traditional parties to support anti-immigrant".
- A Composer Savors a Different Score** by Michael Cooper. The article text states: "John Luther Adams, a Pulitzer Prize-winning composer, is no casual fan of baseball, and there is a long history of composers like him who are passionate about the game." Below the article is a link: "Campus Killings Put Focus on Treatment of Women".

On the right side of the page, there are sections for "The Opinion Pages" with links to "A Cable Merger Too Far" and "Today's Times Insider". Below that is a "MARKETS" section showing stock indices for Japan (Nikkei), HangSeng, and China (Shanghai) with their respective values and percentage changes. At the bottom right, there is a "SCIENCE TIMES" section with a link to "Craft Beer, at the Genetic Level" and "Eye Pract Game".

# Non-Semantic Tags

So, you might ask:

*Is there ever a time to use non-semantic tags?*

**SURE:**

**<div>...</div>**

**<span>...</span>**



# Non-Semantic Tags

There are occasions when you need to use generic tags for **PRESENTATION purposes** and really don't want a tag container to have any special meaning.

*That is when you use either `<div>` or `<span>`.*



# A good use of the <div> tag ...

Let's say that you have your page marked up perfectly with semantic tags but you simply want the existing content (see image on right) to be centered instead of left justified.





# A good use of the <div> tag ...

You can wrap a <div> around something without changing any of the meaning so that you can visually center it. First, nest the stuff you want to center inside something like: **<div id="wrapper"> ... </div>**

The image shows a screenshot of a website header and main content area. The header contains navigation links: METHODS, SOLUTIONS, CABEDGE, and CONTACT. A red ribbon icon is positioned between SOLUTIONS and CABEDGE. In the top right corner, there is an orange button labeled "REQUEST A PROPOSAL" with "QUICK & EASY" below it and a right-pointing arrow. The main content area features the heading "DOING GOOD WORK..." in a large, bold, serif font. Below the heading, there are four service categories: "BRAND EXPERIENCE", "ORGANIC / PAID SEARCH | SOCIAL MEDIA", and "STRATEGIC WEB DESIGN". A "LEARN MORE" button is centered below these categories. At the bottom of the main content area, there are three icons: a whisk, a bowl of food, and a measuring cup. The footer contains the "cabedge" logo, which includes the text "An Atiba Company". To the right of the logo, contact information is provided: "cabedge", "615.942.9937", "on the horn", "615.942.9976fax", and "sayhowdy@cabedge.com". On the far right of the footer, there are three input fields: "Name", "Email address", and "Phone number", followed by a "Tell us about it" button.



# A good use of the <div> tag ...

`<div id="wrapper"> ... </div>`

Then you can write a CSS style targeting the "wrapper" ID so that it moves it to center.





# A good use of the <div> tag ...

`<div id="wrapper"> ... </div>`

Now the content is visually centered so it is visually pleasing without altering the meaning for screen readers.





# A good use of the <div> tag ...

```
<div id="wrapper"> ... </div>
```

Now the content is visually centered so it is visually pleasing without altering the meaning for screen readers.





# Difference between `<div>` & `<span>`

## `<div>`

- A “**block-level**” container tag.
- “Block-level” elements take up the entire width of a line unless otherwise specified in the CSS.

## `<span>`

- An “**inline**” container tag.
- “Inline” elements take up only as much space as necessary and can sit side-by-side with other “inline” elements within a line.

# <div> "block" example

```
<div>div example 1</div>
```

```
<div>div example 2</div>
```

div example 1

div example 2





# <div> "inline" example

<span>span example 1</span>

<span>span example 2</span>

