SWCP Portal
November 2013

It's not supposed to be rocket science

Web Design Without Tears

The ongoing debacle of the rollout of the new federal health-care website provides a vivid example of the problems involved in developing and putting a site online.

Supposed to be up and running over a month ago, **HealthCare.gov** still doesn't work for the millions of people who've been busily trying to access it. In fact, just *6* people were actually able to sign up the first day! The problems have seemingly overwhelmed both CGI Federal, the company responsible, and the government. It certainly hasn't been **cheap**, either; so far going from the *$92.7 million* allotted up to *$232 million,* and the price tag is still climbing.

Taking far too long, way over-budget, and not working right – such headaches may seem sadly familiar indeed to many people who've struggled with their own websites. Is it because the Web was invented by egghead scientists that site creation is often so hard, time-consuming, and expensive? Is there any way for a regular person to get the job done without going crazy or broke or both?

Without trying to justify the botched way the project has been handled, the short answer is that there are *lots* of reasons why developing websites can be difficult. But it *is* possible nowadays to make and maintain even the most complicated portals much more simply than in the past. To understand how, it's necessary to know a little about how the World Wide Web works – knowledge that will undoubtedly help if you ever decide to go for it yourself.

Going beyond text

Basically a webpage is an electronic document written in a language called **HTML**. Each one is a program that contains information to be displayed and instructions on how to do it. It is in essence so simple that a webpage can be written on a notepad, and the **first exercise** still given to students shows how it's done. The code looks like this:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

Which, when saved as an HTML file and opened in a Web browser, would appear onscreen something like this:

My First Heading

My first paragraph.

HTML means "*HyperText Mark-up Language*" and it's easy to see from this how it cleverly works. The text in a pair of angle brackets <>, called a **tag**, is not seen by the reader, but is "hyper", above, that is, surrounding the visible content, which it "marks up" with directions for display. The tags first tell what kind of document it is (the `doctype`), and then how to show the parts. If a link was included in the page, tags would give the address to go to; if a photo, tags would tell where to get it from and where to place it on the page.

HTML was invented by atomic scientists at **CERN** to share the results of their experiments. And it worked great for that. However, they made a few initial decisions that would greatly complicate their simple idea – and which haunt us still.

The geeks were chiefly concerned with **legibility**. They wanted HTML to be readable on *any* kind of computer or screen, but not be dependent on the original look. This was difficult back then, but it is complicated further by the large variety of operating systems and the dazzling range of screen sizes nowadays, ranging from handhelds to full tabletops.

A closely related challenge designers have had to face is that the scientists didn't care about prettiness, so tags for styling and placing text were basic, few, and thrown in with the rest.

Once they got hold of HTML, designers did the best they could. There was only *one* item flexible enough to build complicated layouts, and that was the **table**. Artists quickly realized they could stick anything in these arrays of cells, not just rows of numbers, and make individual cells any size needed. Before long, webpages went crazy with tables, content filling invisible boxes inside other boxes inside still others.

To fix this mess, **CSS**, "*Cascading Style Sheets*" were created. Basically, CSS treats *every* single element on a webpage as a box that can be precisely placed and styled. Sometimes too accurately, because the elements and their properties interact in many different ways and are dependent on each other. A single missing bracket in the code can lead to nightmarish illegible cascades of ugly confusion.

By separating style rules from actual content, CSS is almost as powerful as HTML. Yet resulting webpages are still static. **Scripts** in other languages were put into the code to make pages interactive, modify content or the display. Out of all these ingredients grew our complex online world today.

Continued on back

Continued from front

Design by description

HTML is a living language with internally consistent rules, and is constantly evolving. While much has been added, some commands have been quietly dumped, like the `<blink>` tag, which made things flash in and out of sight. It seemed a cool idea at first, but soon proved to be really irritating.

Styling is being shifted completely to CSS along with some scripting duties. HTML version 5 is still being developed to work with CSS3. Spin-offs of HTML like **XHTML** have been created for many other applications than the Web, such as **ebooks**, and there is no end in sight to the power of tags.

Yet the implementation of HTML has been rocky. One thing that has plagued the Web since the early days are the use of **proprietary standards** by competing systems – much like with ebooks today. During the "browser wars" between Microsoft and Netscape, designers would often have to break webpages and insert clumsy workaround hacks to cobble together something that would look much the same in both browsers. Things are somewhat better all around today, but the problem persists because different products incorporate new features in their own way and time.

Having to deal with varying size screens on machines with different capabilities, with code that didn't always work the same, and with very little control over the ultimate appearance anyway probably drove a lot of talented designers nuts. Plus webpages are not just lovely displays, they are small **computer programs**, with all that implies.

There aren't many people who combine an artist's sensitivity with a programmer's patience. We've listed a few of the best pros in town in the **Epromenade** at our website.

For really big projects, teams working together are necessary, but it's anything but quick, easy, and cheap. The most massive enterprises like the healthcare exchanges have huge numbers of people trying to use them all at once, incoming datastreams that must be manipulated on the go with tricky interacting math problems and a slew of immense databases – while keeping all that personal information private and safe.

Doing it yourself

However you go about making a website, the most important thing is to know your message and your audience. There are lots of beautiful packaged sites for restaurants you can buy, for instance, but they are all of necessity somewhat generic. A really good website will not only be nice-looking and totally functional, it will be individually distinctive. Usually a balance between uniqueness and budget must be struck. This is where having a clue about how it works can really help.

The dual nature of webpages is clearly reflected in the tools used to build them. Unlike other forms of digital design, there are *no* drawing tools at all in the Web designer's kit. The closest things are sites and programs that try to turn drawings of webpages directly into CSS formulae. Otherwise, creating a webpage is much like trying to paint a picture by writing a careful description of how to do it.

Though most of the high-power programs, such as the leader, Adobe's **Dreamweaver**, promise *WYSIWYG* – "what you see is what you get": that the resulting page will look just like what is being designed as it is developed – all have trouble delivering. The program includes tools to simplify tag creation, but like all tools and code repositories, require some knowledge of HTML and CSS to use effectively.

At the top end, Dreamweaver is so expensive you can rent the use of it from Adobe for *$50/month*. Other good programs, both for purchase, like **Serif WebPlus**, or free, like **Kompozer**, are somewhat similar in looks and function, but they all have their individual quirks and limitations.

As one might expect, there are a host of resources on the Web, both tools and tutorials, many free. You can get an entire free education in Web design through the excellent hands-on examples at the **W3schools.com** site, with thorough, authoritative refererences, plus pages where code can be submitted to check that it works properly. To teach kids, perhaps adopting **Web Literacy Standards** will help.

For people wanting publish right now, the good news is that it's not necessary to reinvent anything. The most powerful websites today are run by "*Content Management Systems*" – **CMS** – and come in two parts. The "front end" that users see is the display of content and navigation, but behind it, the "back end" contains controls for administrators to add, arrange, and style new material, or even quickly modify the look of the entire site. Scripts, sometimes in add-on modules or **widgets**, are not only used to assemble pages on the fly. They can add or modify functionality, so it's possible for even a newbie to customize a website, adding media galleries, response forms, calendars and so on, and keep it running.

Literally thousands of **CMS platforms** are available with a wide range of capabilities. Some are quite expensive, but many powerful systems are **open-source** and free, such as **Drupal** or **Joomla**. These can easily manage corporate presences, full shopping sites and extensive wikis.

For people with somewhat more modest needs and ambitions, there is **WordPress**. Starting as a blogging platform, WordPress has become possibly the most widely used and versatile CMS in the world, and one that SWCP strongly and enthusiastically recommends.

WordPress, too, is open-source, with thousands of free, easily-changed themes to govern the look and way the site works, and almost as many add-on widgets. But since CMS relies on scripts and databases, all such sites use more resources. So they cost more to run than the old-fashioned static kind. At SWCP, WordPress requires **Basic Web Hosting** (*$15/month*), while Drupal or Joomla need **Professional** (*$25/month*). And both require your own domain (starting at *$20/year*).

We are so keen on WordPress that our own websites are built with it. Plus we hold regular **talks** and work-alongs for users by experts in our **Ideas and Coffee** coworking space, and are working to make updating WordPress sites even easier. Because the World Wide Web is here to stay, and everyone should be able to build a home in it.

Southwest Cyberport

New Mexico's Expert Internet Service Provider since 1994

505-243-SWCP (7927) ◉ **SWCP.com** ◉ **Help@swcp.com**

5021 Indian School NE, Suite 600, Albuquerque, NM 87110

Portal *editor/chief writer, Jay Nelson* **jnelson@swcp.com**