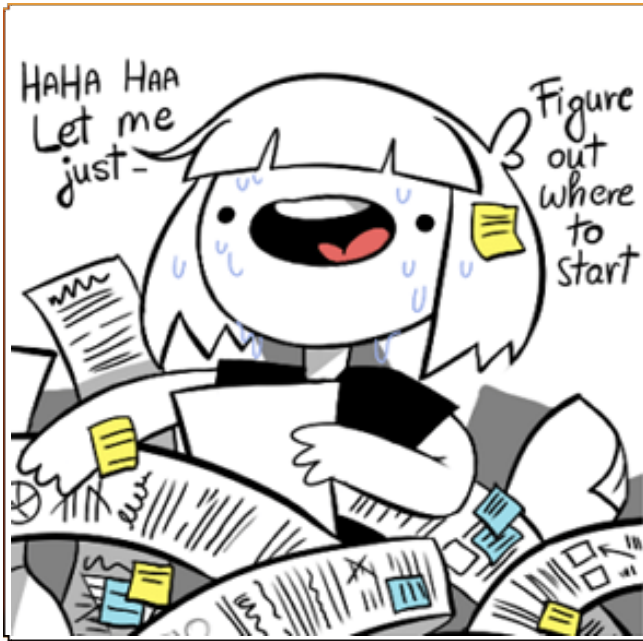
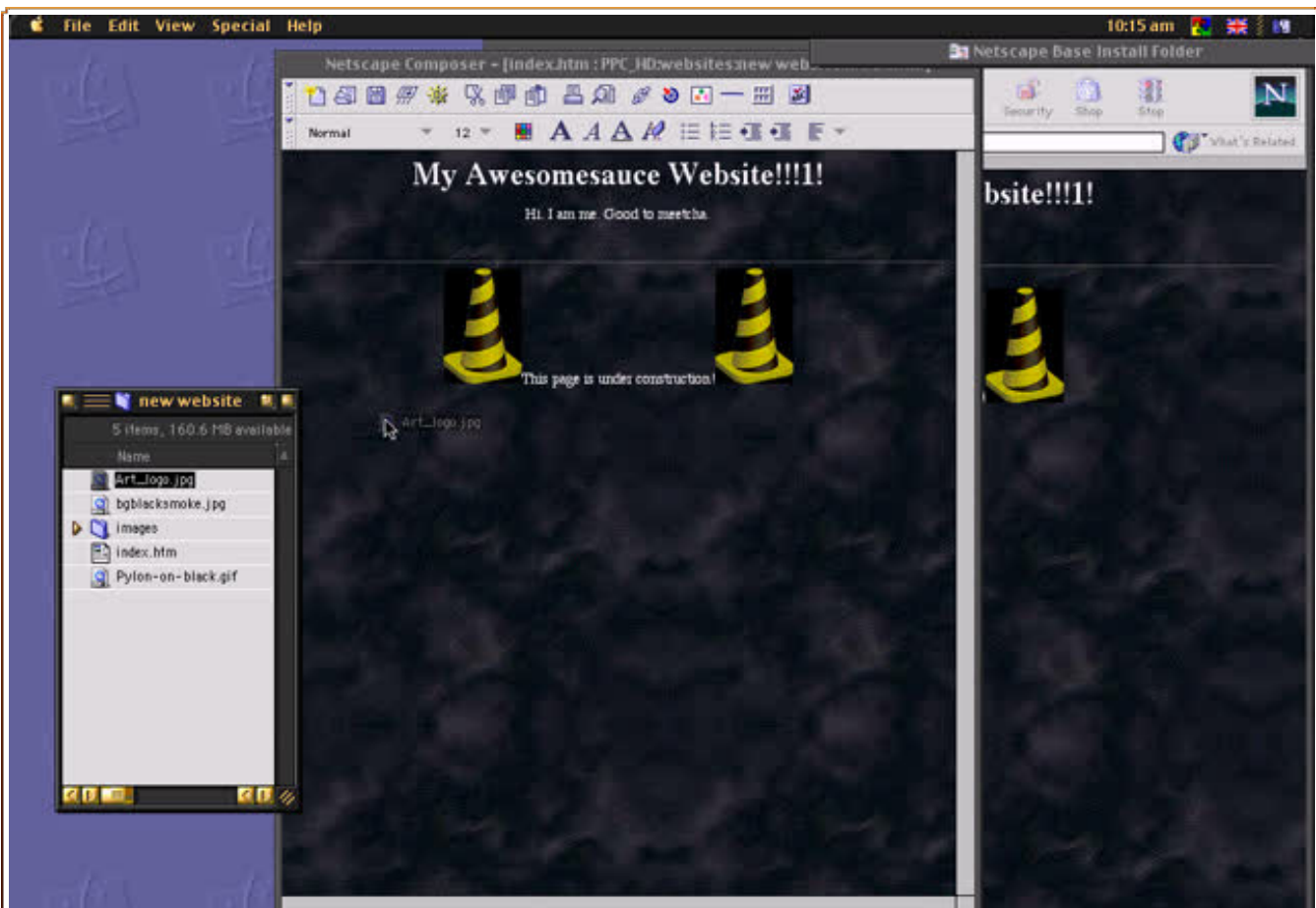


Amazing Tools From a Bygone Era



I often tell people it's easy to make a website. Why do I believe that?

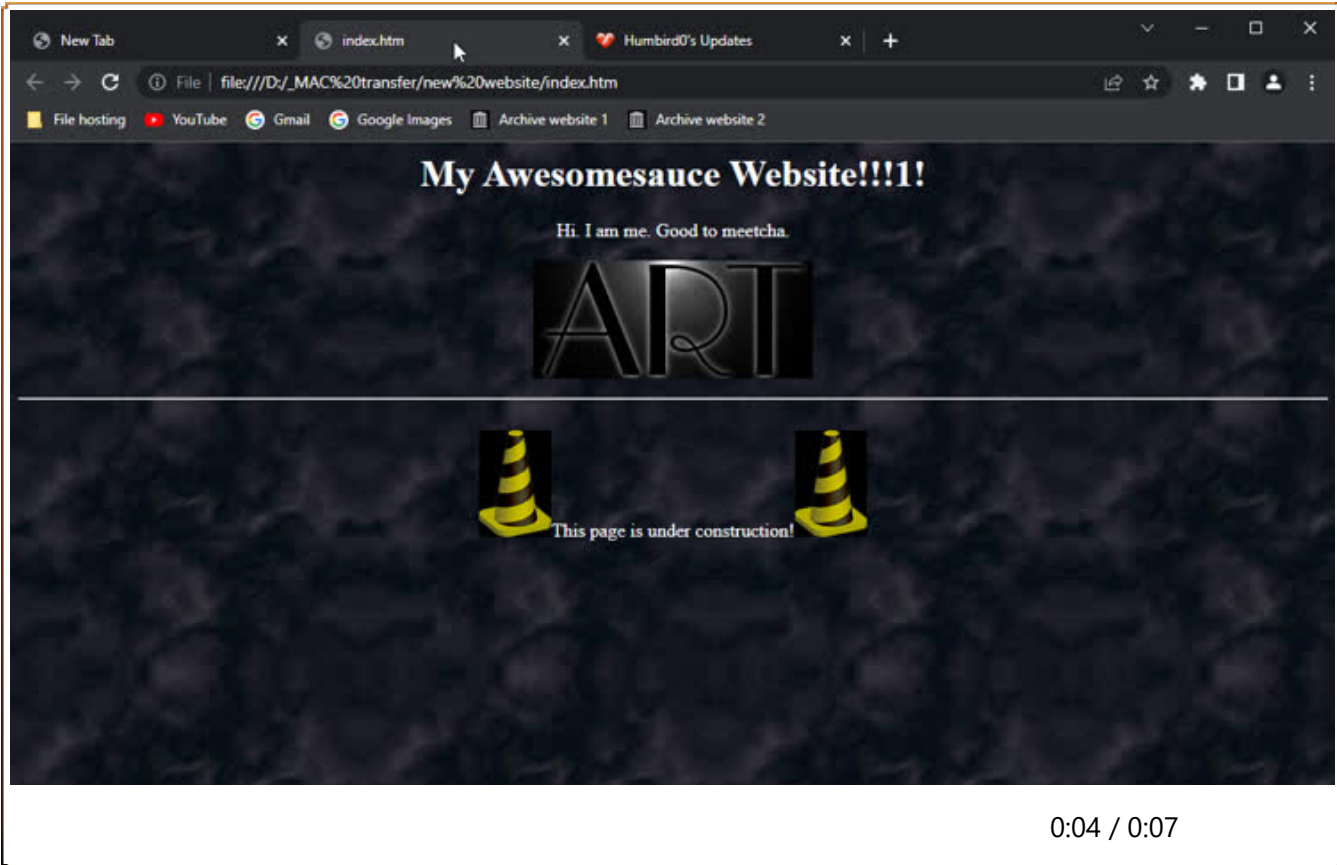
Because I grew up in a time when web browsers like [Netscape Communicator](#) came with GRAPHICAL website editors that worked without you having to know any code. They worked like word processors. They were called [WYSIWYG editors](#) (What You See Is What You Get)



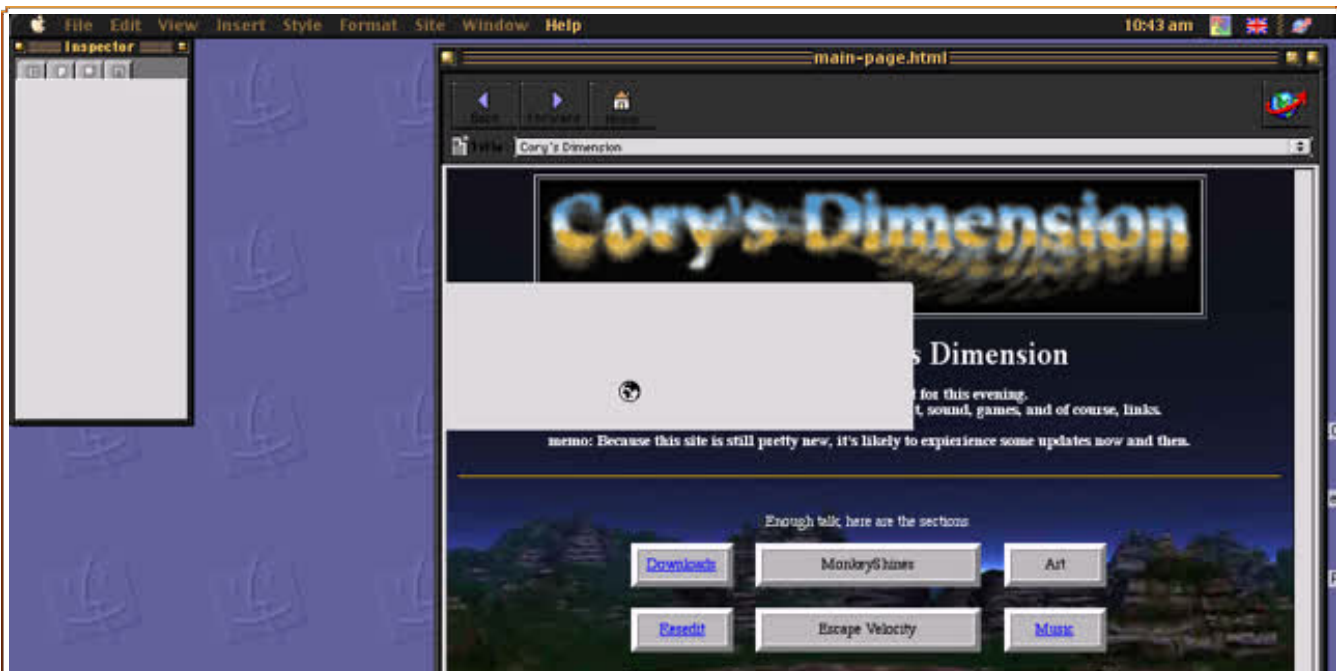
0:04 / 0:29

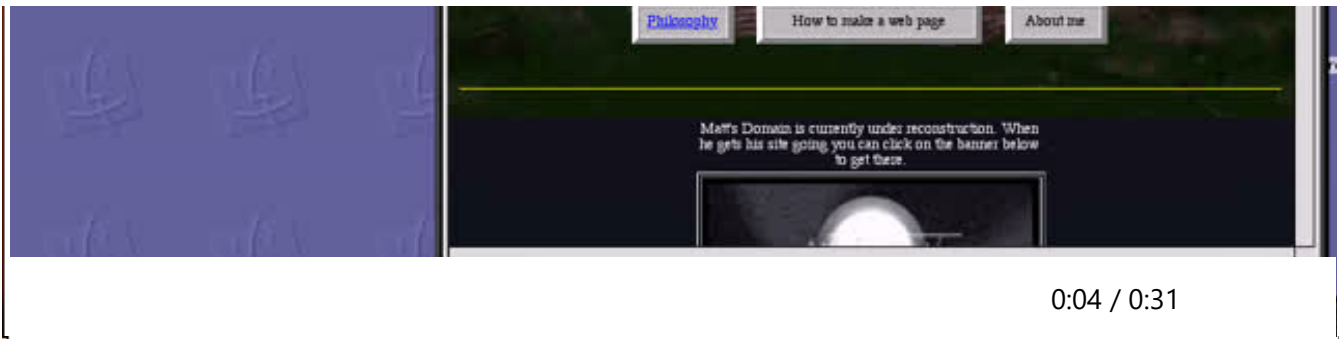
I take it for granted that easy tools like these exist. But they're old, so why would anybody want to use ancient software from 25 years ago? ... because they *STILL WORK*.

New features were gradually added to HTML, but the original features did not go away. You can still build a website "the old way" and with very few exceptions it will still work perfectly. Granted these methods are now "the wrong way" because... tables or something. But more importantly these methods are easy. It's a good way to start.

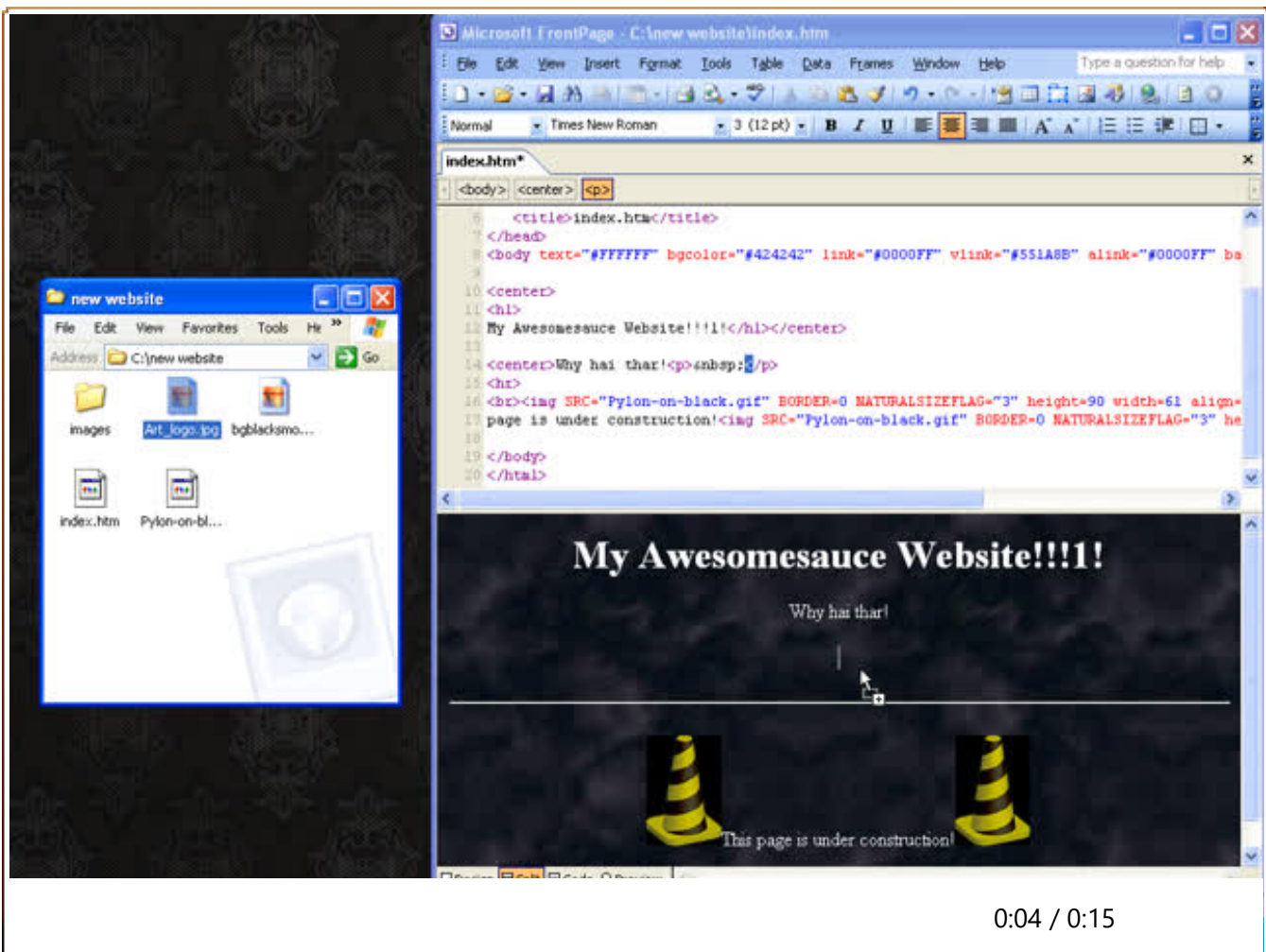


I built my first website in [Adobe Pagemill](#) in the year 2000. Look how easy this is!

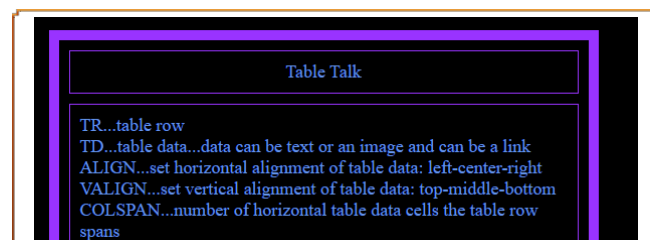


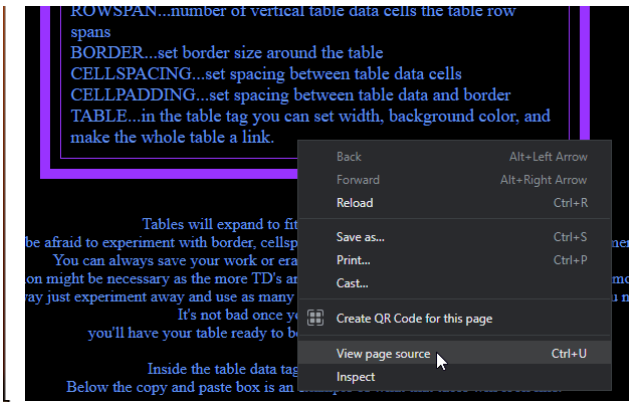


And Microsoft made a great one called **Frontpage**. All of these are visual editors. All drag-and-drop. All easy to use. And if you had any curiosity at all, you would naturally peek behind the curtain at the text that it's writing.



If a **GeoCities** website did something cool, you could just right-click their page and “View Source” to see how they did it. Then just copy-and-paste it into your own page. That’s not stealing. It’s learning. Nobody owns HTML code. The web was designed like this on purpose.





... or you can just look up the main HTML tags. You only really need about 5 of them most of the time. [](#), [<A HREF>](#), [<DIU>](#), [](#), [
](#)

<https://wvrocker.tripod.com/codes.html>

<https://wvrocker.tripod.com/tables.html>

Rick's Cafe

Copy and Paste the codes to your page...For information on how to Copy and Paste [Click Here](#)
 Put your own text where it says text here.
 Replace URL with the http:// address you want that link to go to.

Basic Page

```

<html>
<head>
<title>Title of your page here</title>
</head>
<body>
All the stuff you want on your page goes between the body
tags.
```


The Old Approach

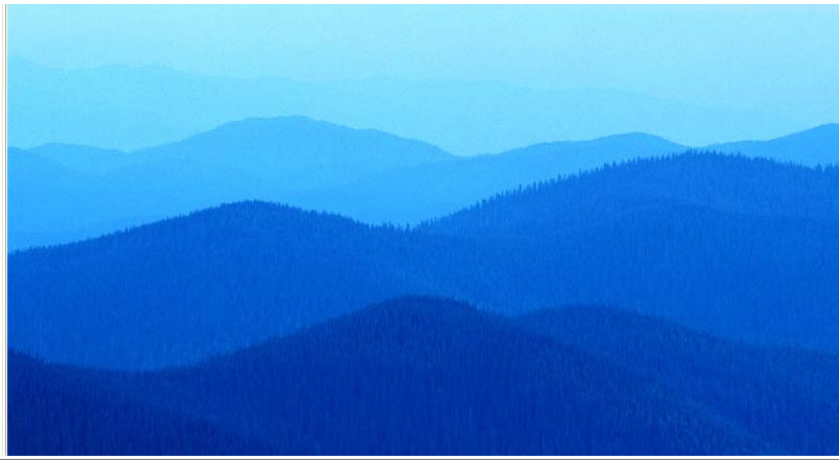
But fine. Big deal if you can make an ugly GeoCities page. Everything is just centered and stacked on top of each other. How do you make it, you know... good?

Tables! [<TABLE>](#), [<TR>](#), [<TD>](#)

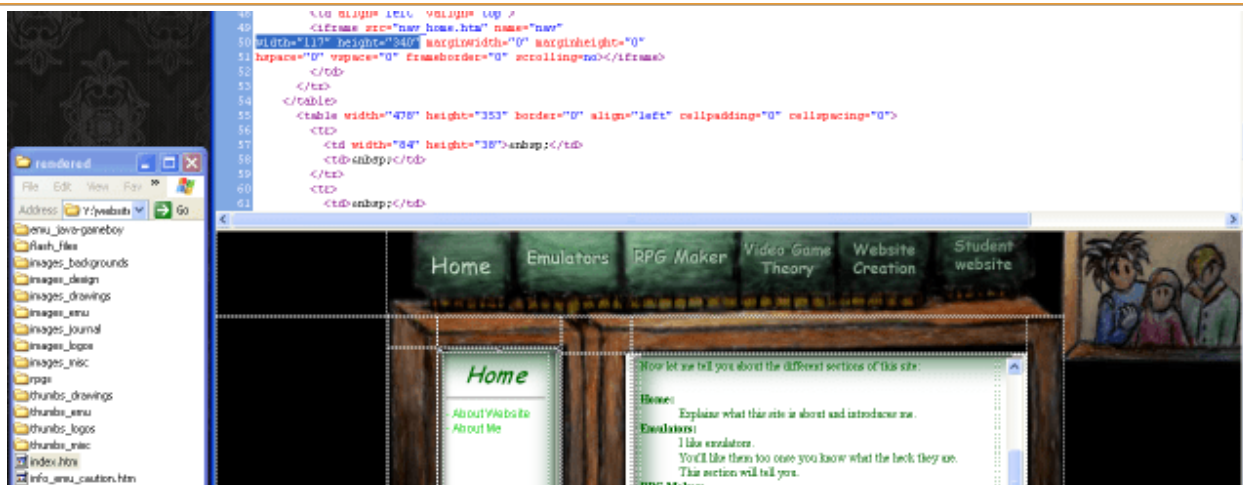
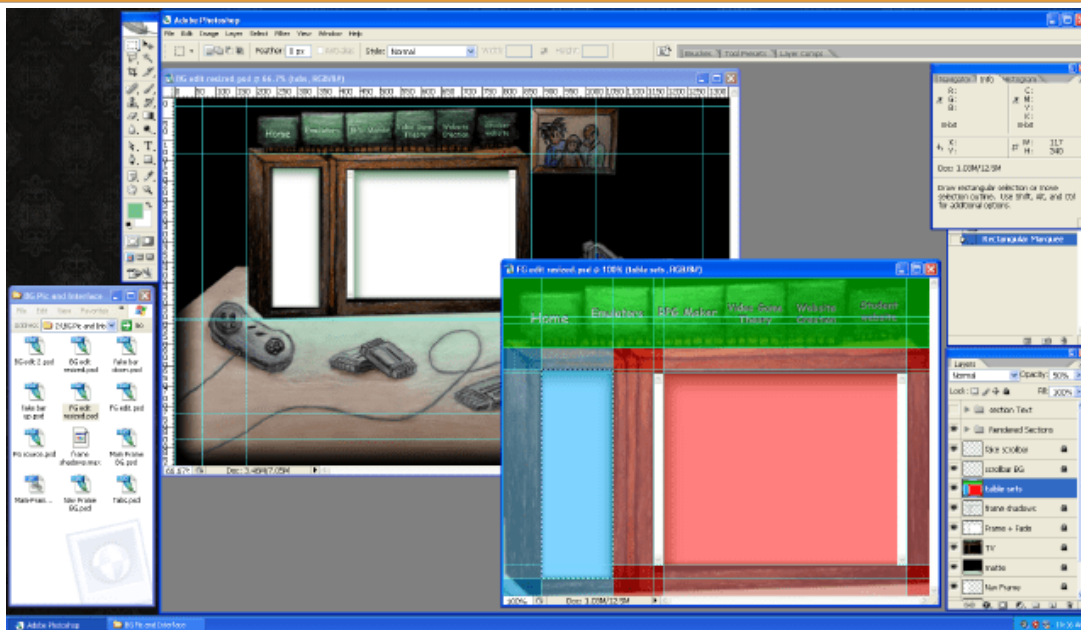
Back then that was the only way to put things next to each other in any controlled way.

<https://wvrocker.tripod.com/tables.html>

	<p>Why hai thar!</p> <p style="background-color: #4a90e2; color: white; padding: 2px 5px; border-radius: 5px; display: inline-block;">Click this!</p>
<p>About me</p> <p>My Art</p> <p>Favorite things</p> <p>Downloadz</p>	



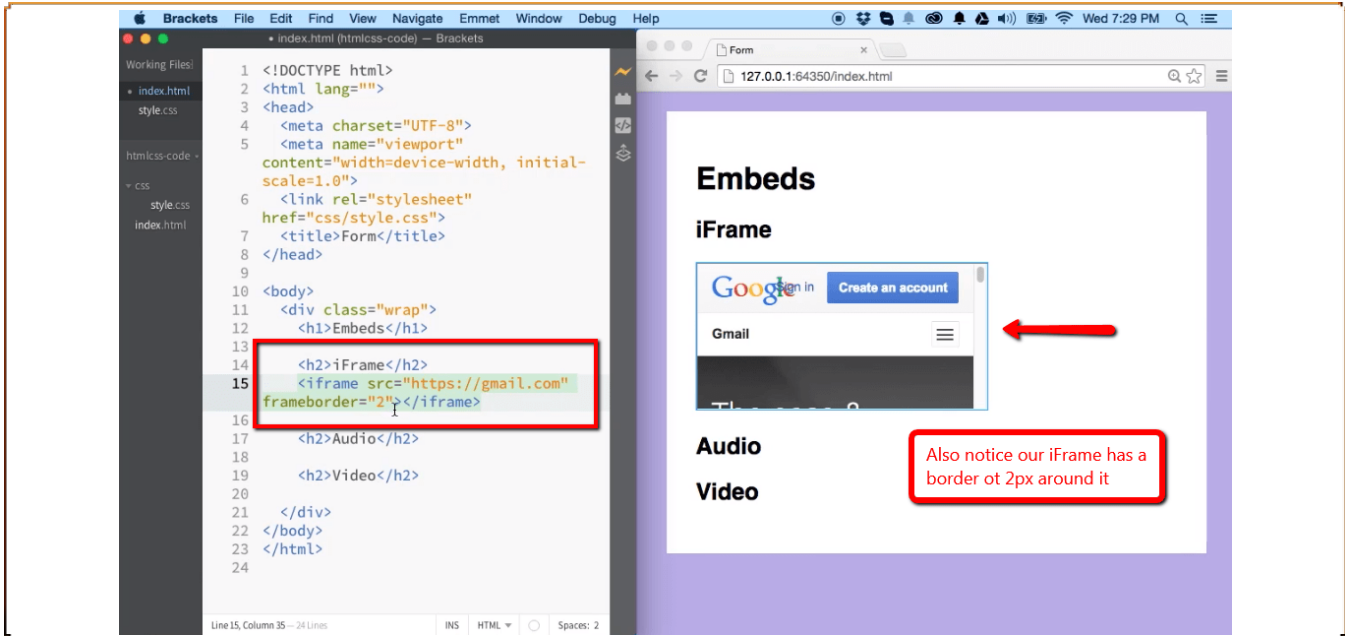
So here's how the "good looking" websites were made back then. First you draw a mock-up of your page in Photoshop... yes really. Then export a picture of your design as a JPEG and set that as your *website's background* so you can trace over it in a graphical website editor! Then you use tables to position things down to the exact pixel. Photoshop even has tools for slicing up pictures for this exact purpose, just so you can put all the pieces inside a table to surround your content.



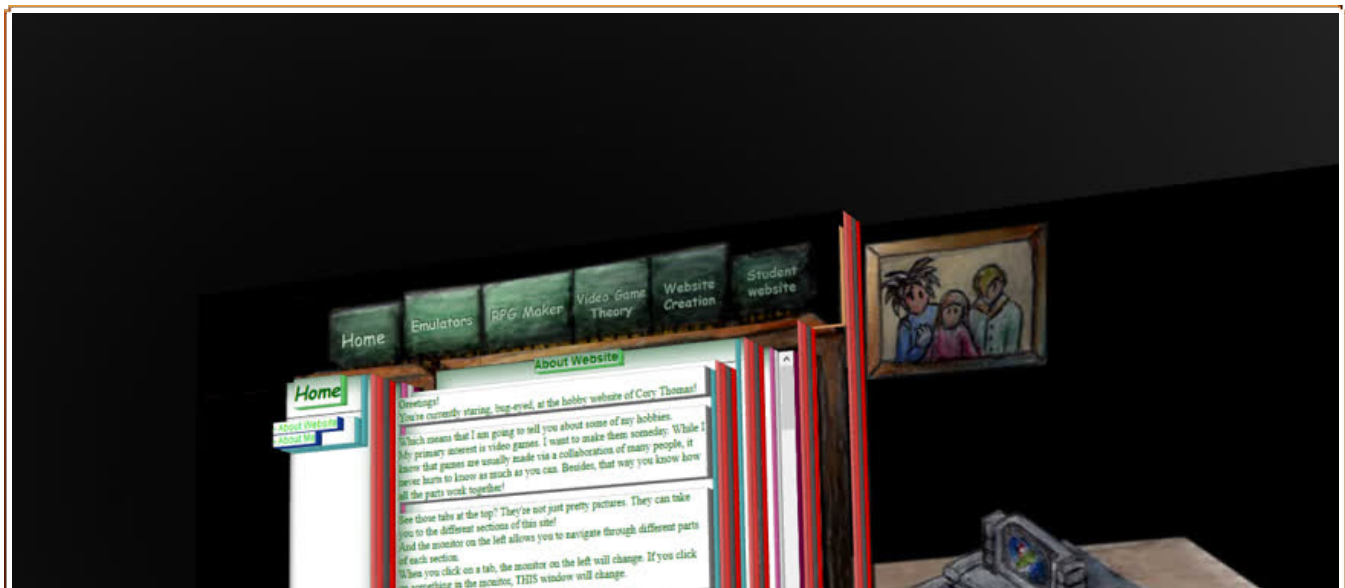


Tables are very convenient, but also very rigid. Normally you would have to create a new layout for every single page. But you can re-use parts of a page using `<IFRAME>` tags which let you keep your navigation buttons on-screen, and only change part of the page when you click on links. Basically you can display a page inside of another page.

<https://ilovecoding.org/lessons/embeds-video-audio-and-iframe-elements>



That was how I built this website back in 2004. It doesn't even use CSS. Hell it barely even uses pictures. It's mostly just one giant background picture with areas you can click. Behind the scenes it's a table and a couple of invisible iFrames. When I click the buttons, the pages get loaded inside those areas.





0:04 / 0:05



0:04 / 1:01

Learning New Things

So if this is so easy, why is nobody using these anymore?

Because of 2 reasons:

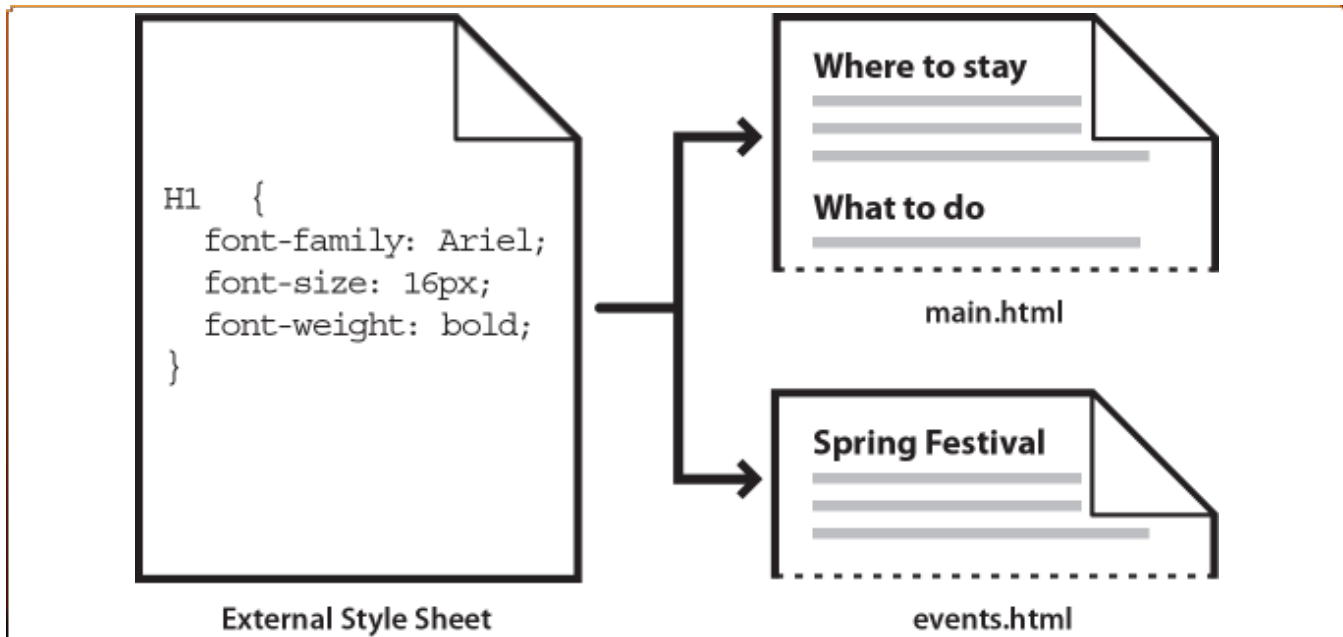
1. It's hard to change old websites.
2. Smartphones.



It's hard to change old websites.

It's easy to make a website this way. But it's hard to *change the way it looks* afterwards,

because you would have to modify every... single... page! One by one. You could use Dreamweaver templates to rewrite all the pages for you, but then you would have to pay Adobe forever. You can avoid all of this by using a [CSS Style Sheet](#), which is a separate file that describes how all the pages should look. And because it's a separate file, you can just [change that one file](#) to completely transform how the rest of your website looks.




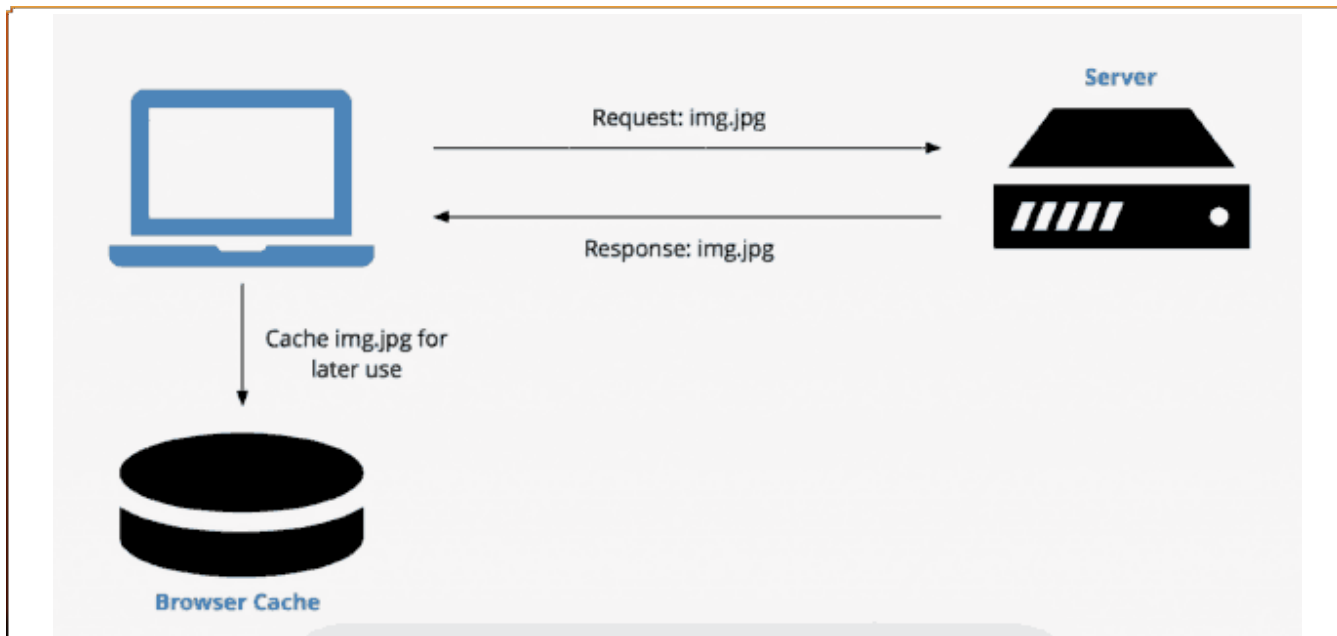
Smartphones


Smartphones also threw a monkey-wrench into things. In the early 2000's you could safely assume everybody had a computer monitor with at least a 800x600 resolution that was at least a foot wide (and 20 pounds). But smartphones are 2 inch screens with completely random resolutions. Even if you can fit all the pixels, everything will look too tiny to read or tap on with our giant clumsy sausage fingers. So you end up needing to stack things when they don't fit side-by-side. Tables can't really do that.

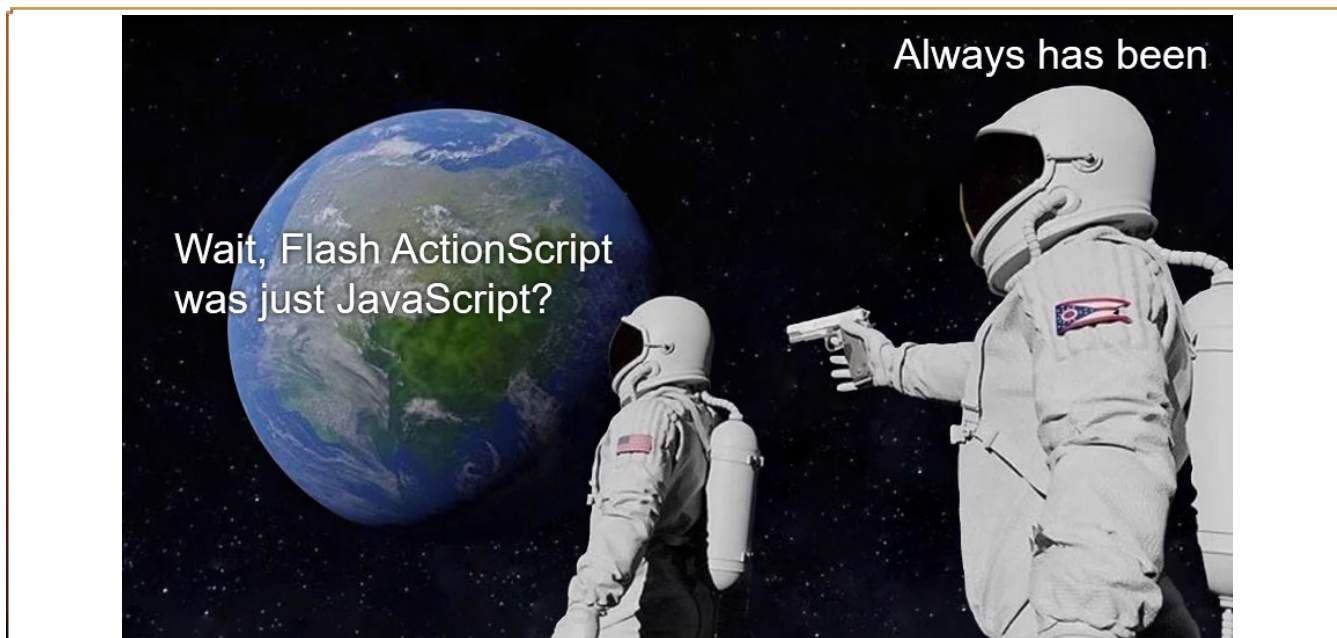


I actually dragged my feet a little with CSS. I didn't start learning it until about 2011. I threw

But that has drawbacks too. My  **Pokemon hentai game** was in full-swing and all the downloads were stressing my free domain, so I switched to a paid host for \$10 bucks a month and looked for ways to make a more efficient website. I heard about this new thing called “Single-page JavaScript websites” so I thought I’d try it. It seemed to make sense. In theory if everything is rendered and **cached** on the visitor’s computer, and I later add a new project then the only new thing people have to download is a single JSON file and a tiny thumbnail picture.



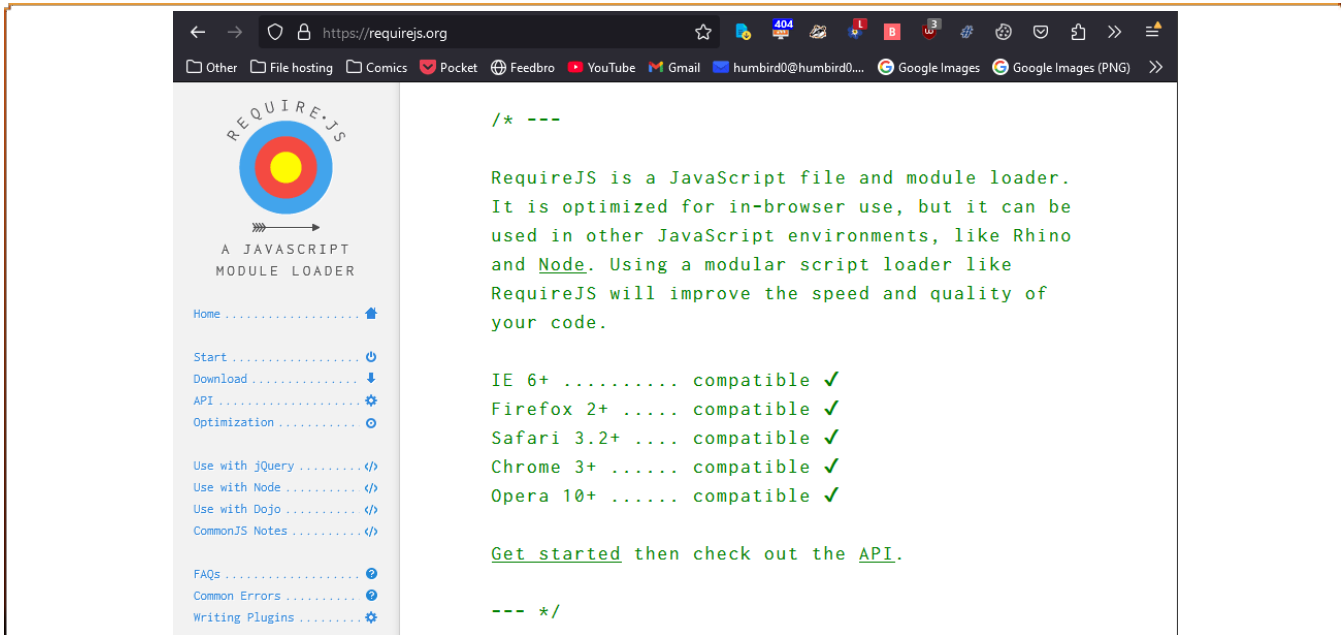
And I had already learned **JavaScript**... completely by accident! It turned out that Flash’s **ActionScript** language was basically just JavaScript in a trench coat. I had already created my custom  **RPG Maker** by this point, so I was pretty damn good at it.



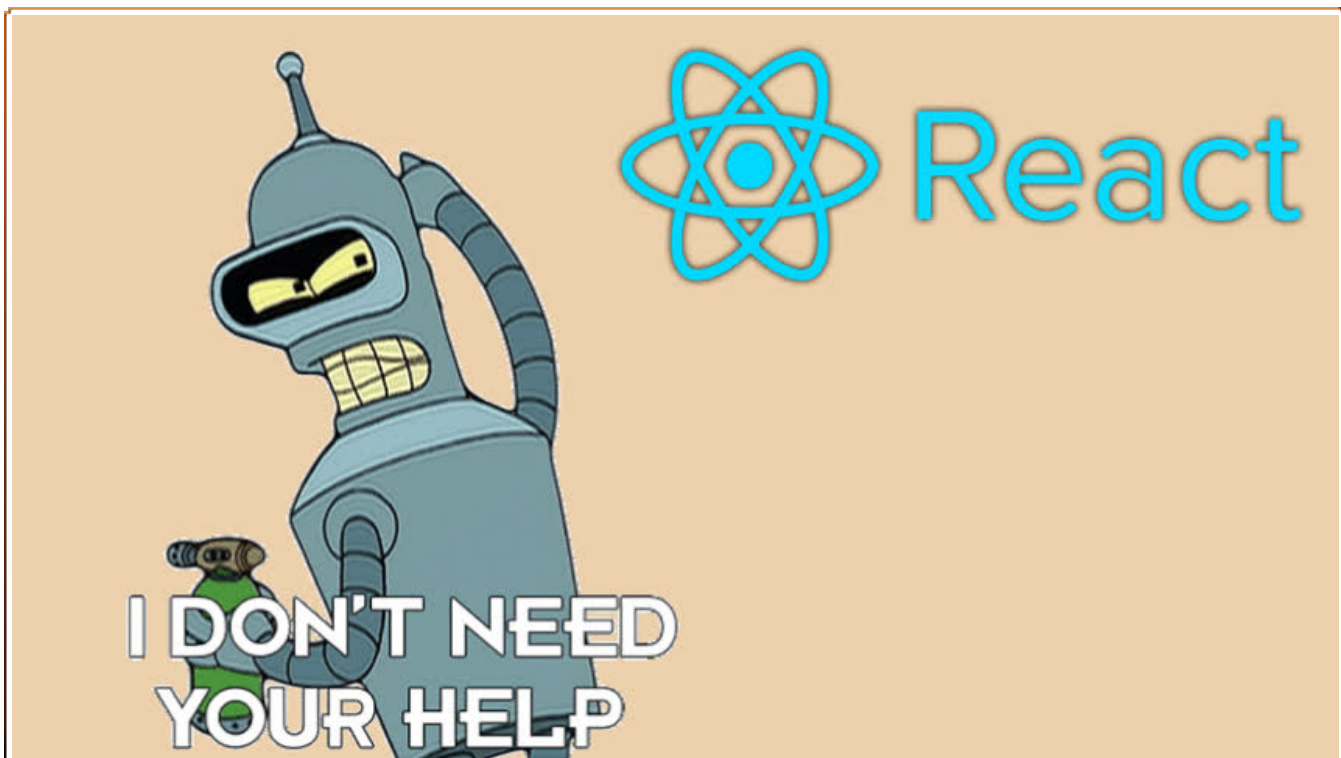
I briefly tried using **React.js** but it seemed like extreme overkill, so instead I wrote my own

“model-view controller” from scratch. Originally as a single long JS file. Then I discovered [require.js](#). This thing is glorious! Modular programming baby! The concept itself even revolutionized the way I program my Flash games.

<https://requirejs.org/>



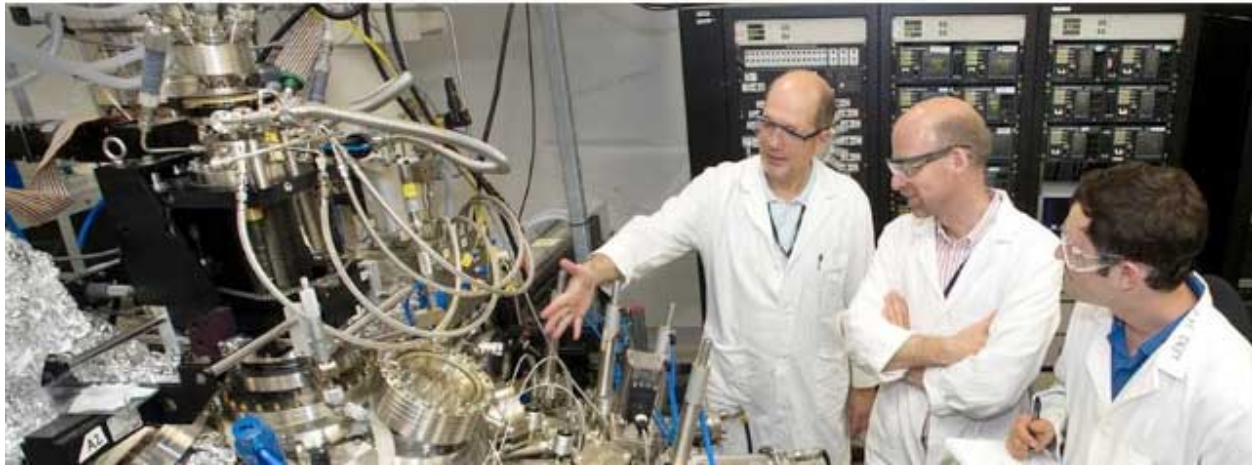
I didn't download a bunch of “helper” JavaScript libraries. I mostly just used [jQuery](#) to tame Internet Explorer's quirks and [require.js](#) to organize stuff, and then wrote the rest myself. It all worked so well that I never felt any need to add “helper” code.



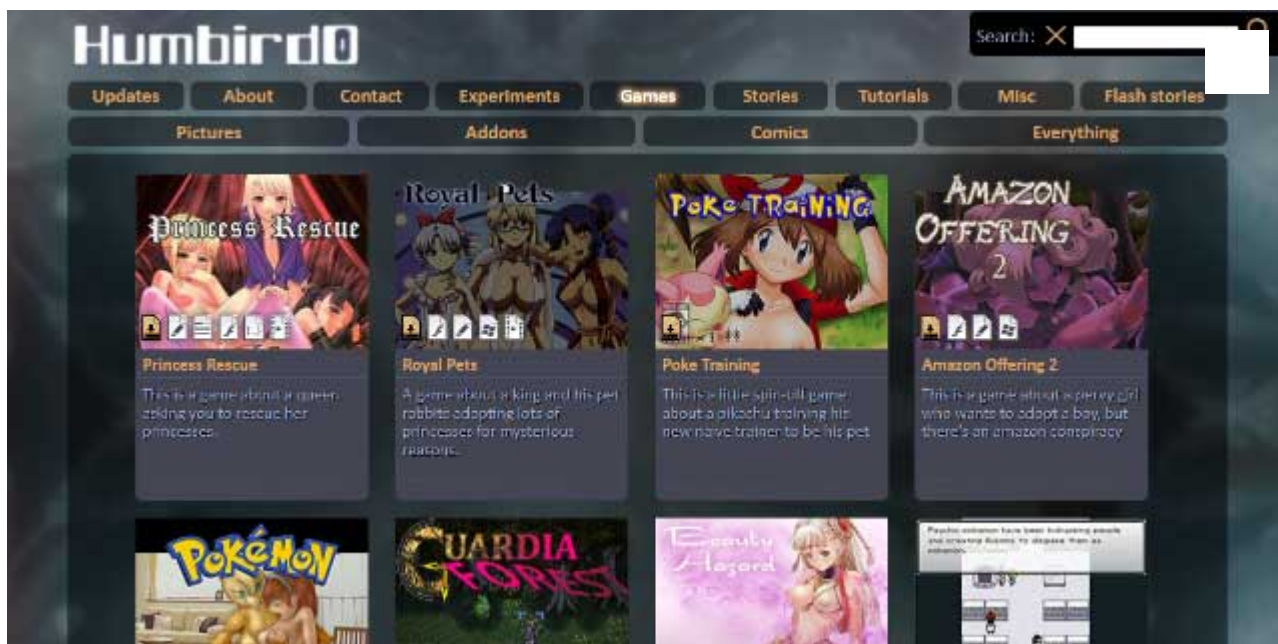
Avoiding Modern Mistakes

As a side-effect I mostly ignored the React.js craze, CoffeeScript, SASS, and the avalanche of other JavaScript helper libraries and build tools that were released over the years. I simply didn't need them. That turned out to be a very good thing. Most "professional" websites struggle to even display a simple picture if JavaScript is turned off.

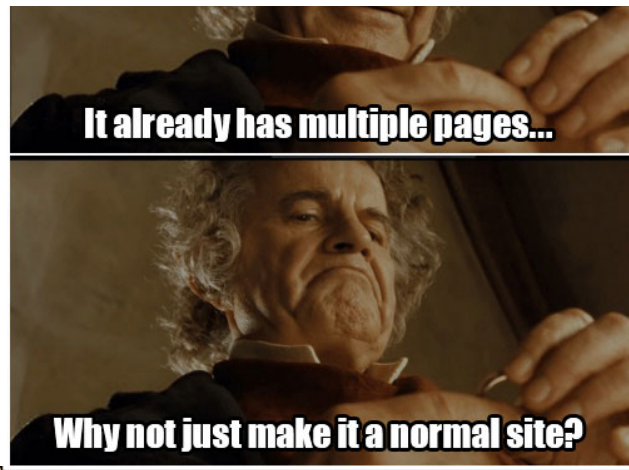
JavaScript developers deciding on the best way to render static HTML



Meanwhile my old JavaScript website could display an entire fully usable thumbnail gallery if JavaScript is off. After learning about [The Wayback Machine](#), I became obsessed with [semantic HTML](#), machine-readable [metadata](#), and [digital preservation](#), so I modified my JavaScript website to work perfectly even on the WayBack Machine. Go ahead and [try it](#). Even the old search engine still works!



Eventually I realized that my JavaScript website had long since given up the efficiency of a single JSON file in exchange for preservability. It also took over 2 seconds to load even over a blazing fast internet



to load even over a blazing fast internet connection. Meanwhile a normal “static” website could do all of these things better. And since each of my projects already had its own HTML page for their “readme” descriptions, it was almost a static website already, so why did I need the JavaScript? I also wanted to write my website using Markdown to make its content more **future-proof**. So in 2023 I completely remade my website using a static site generator called HUGO.

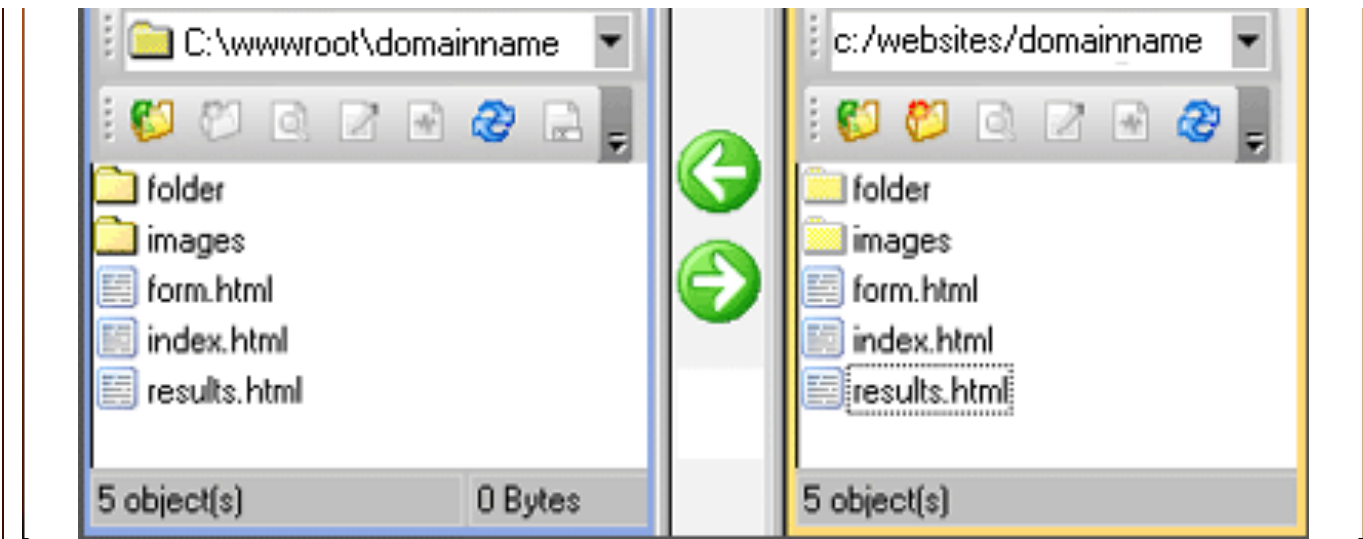
So I’m back to using templates. But this time I’m not relying on Dreamweaver and Adobe. I’m using a program called **HUGO**, which is free, open-source, and multi-platform. I can just write my pages as simple Markdown text files, and HUGO converts them into web pages for me. But since I already know all about HTML and CSS I wrote my own templates for HUGO to make my pages exactly the way I want them. But you could just use a pre-existing **HUGO theme** instead.



Your Turn

So when I tell people that making websites is easy, that’s because I remember using simple visual editors, making simple websites with tables, signing up for **cheap hosting (which only costs about \$3 a month)**, and **using FTP** to drag-and-drop files onto the internet. I watched HTML grow from something simple to something with a bunch of optional complexity, so I already know which parts are optional. But I forget that other people *don’t*.





These days it's kind of hard for people to figure out what matters next to the mountain of advertisements for JavaScript frameworks and “helper” tools.

Spoiler alert: ALL of them are optional. Most of them make things *more complicated*. And they do that on purpose so they can sell you their “solution”. Most of them should be avoided. The truth is that HTML itself is actually simple. Just focus on **HTML and CSS**. Then experiment with other things later.

<https://webguide.neocities.org/css/>

