Lecture #12: Layout and Forms

CS106E Spring 2018, Young

In this lecture we look at how webpages are laid out. We begin by looking at three different historical methods used to layout webpages, two of which (float-based layout and flexbox-layout) are still in widespread usage. We then settle on a fourth method – grid-based layout — and we look at in some depth.

Next, we consider how to create forms on webpages. These allow the user to submit information to a web server. We learn how to place forms on a webpage and explore the various form elements available. We consider the difference between using the PUT method and the GET method for submitting information to a server. Correct choice of methods is important and incorrect use may have severe repercussions.

Website Layout

There are several different methods that have been used to layout webpages:

- The first webpages with complex layouts used HTML tables for layout. As you might imagine, you can use the rows and columns provided by tables in order to create rows and columns in a webpage. However, there were several important limitations:

    1) Using table-based layout very clearly placed the presentation information directly into HTML. As we've seen modern webpage design dictates that the HTML contain only semantic information. However, this technique preceded the development of CSS and when originally used, there were no other alternatives available.

    2) While laying out a simple table is not taxing for the web browser, laying out nested tables required more resources. In general, any layout requiring more than three levels of nesting (tables inside of tables inside of tables) was considered too computationally time consuming for use.

- Once availability of CSS became widespread, the float property was used for layout. As you may recall, using float we can place any block element on one side of the webpage and have other elements flowing alongside of it. If I take a set of blocks and float each of them left, they will stack up against each other along the left side of the webpage, forming columns.

  This float-based layout is probably still at this point the most widespread method for laying out webpages. Unfortunately as with the table-based method, it does require a fair amount of layout-related information to be placed within the HTML file. If you're interested in learning more about this method, let me know. I have handouts that I've written for some of my other classes that describe this method in depth, and I would be happy to send them to CS106E students.

- Recently developers have been using a fairly recent CSS addition called Flexbox to layout webpages. As of this writing, about 6% of web browsers in use in the United States do not fully

support Flexbox, so depending on the intended audience, this may cause issues. You can learn more about Flexbox here.

In fact, Flexbox was not really intended to be used as the primary method for laying out webpages. It was designed in conjunction with a second standard for defining Grids in CSS. The intention was that Grids were to be used for laying out the major sections of the webpage, whereas Flexbox could be used to layout smaller sections of a webpage. Unfortunately, Grids took quite some time for the web browser manufacturers to support.

- In this class, we will be using Grid-based layout. It is by far the most powerful and flexible of the layout methods. It's also probably the most easily understood. Unfortunately as of this writing, it's only supported by just over 85% of web browsers in use in the United States. So using Grid-based layout will lose you some visitors.

  I've got a separate handout up on Canvas that discusses Grid-based Layout. Go read it!!!

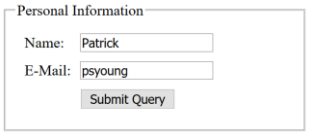
HTML Forms

- Forms on a webpage are used to submit information to web servers.

- I've yanked the chapter on forms from my CS105 Course Reader and put it as a handout up on Canvas. *Go read it!!!.*

Post vs. Get

Let's take a closer look at these two methods for submitting form information. The difference between the two determines how information in the form is submitted to the server. Suppose we have the following simple form allowing a user to submit their name and email:



- Using **GET** the information is encoded into the URL. If our form is defined like this:

```
<form
  action="example.php"
  method="get">
```

when we submit the form and look at the resulting URL we see the following:

```
example.php?name=Patrick&email=psyoung
```

That bolded section is the information I entered into the form. ***The information entered into the form is encoded directly into the URL when using GET.***

- In contrast, if my form uses:

```
<form
  action="example.php"
  method="put">
```

when we submit and look at the URL we just see the following:

```
example.php
```

The form input is not in the URL when using POST. Instead, it will be enclosed in the body of the request.

- So which method is better? The answer is "it depends?"
    - ***The most important characteristic of a GET request is that it is Idempotent***. This means that it should not change the state of the webserver and if the user makes the same request again, they should get the same answer back.
        - Think carefully about whether your submission should change the state of the web server when deciding which method to use. If I'm placing an order for something, that changes the state of the webserver as my order goes into the server database. If I post a message to a bulletin board, that changes the state of the server so others can see my message. These both need to be handled using POST.
        - In fact, if I make a second GET request to the server, and the web browser happens to remember what the response from the server for my previous request was (by storing the last response in the browser cache), it doesn't have to bother contacting the web server. It just redisplays the last result.

          Think about what would happen if I was trying to buy another item from a webstore and the web browser didn't bother making a request to the server, because the request used GET and the response webpage from my previous order was still in the cache. Incorrect use of GET would result in my store website completely failing.
    - If I want the user to be able to bookmark the result of a form submission and get the same results as before, then I need to use GET. Only GET will encode the form information in a way that can be stored in a bookmark or shared. With a POST bookmark, the web server won't know what information was entered into the form the last time the user visited the webpage.
    - Similarly, if I want the user to be able to email a results webpage to a friend, I need to use GET.
    - On the other hand, if the user is entering any kind of security related information, I don't want to put information like user names and passwords in the URL.
        - I do want to be clear though, **whether you use GET or POST unless you use HTTPS, which is a special secure version of HTTP, a hacker can intercept your IP packets and read any information submitted in the form.**
        - At the same time, using GET does add additional security concerns. If we encode any of the user's login information into the URL and they either email the resulting link to a friend or worse, post it to an online bulletin board, their login information is encoded into the link that they've just shared.