**<lecture6. user interfaces>**

**Single Page Applications**

Previously, if we wanted a website with multiple pages, we would accomplish that using different routes in our Django application. Now, we have the ability to load just a single page and then use JavaScript to manipulate the DOM. One major advantage of doing this is that we only need to modify the part of the page that is actually changing. For example, if we have a Nav Bar that doesn’t change based on your current page, we wouldn’t want to have to re-render that Nav Bar every time we switch to a new part of the page.

Let’s look at an example of how we could simulate page switching in JavaScript:



Notice in the HTML above that we have three buttons and three divs. At the moment, the divs contain only a small bit of text, but we could imagine each div containing the contents of one page on our site. Now, we’ll add some JavaScript that allows us to use the buttons to toggle between pages.



line9: forEach 돌면서 모든 div 태그 style을 display = 'none'으로 설정

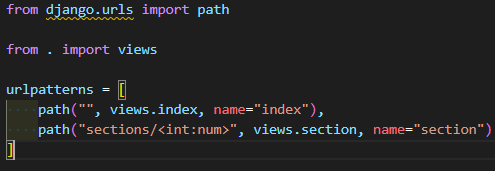
line13: showPage의 parameter 'page'를 받아서 style을 display = 'block'으로 변경.

line17: querySelectorAll과 forEach로 모든 button 태그 선택하고 button.onclick 하면 showPage의 parameter로 dataset의 page를 불러옴.

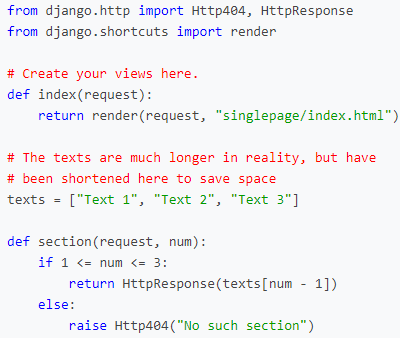
결과적으로 각 button을 클릭하면 showPage(page1), showPage(page2), showPage(page3)가 실행되어 id = page{$}가 매겨진 div 내의 내용들이 보이게 됨.

**Single Page Applications in Django**

**# singlepage/urls.py**



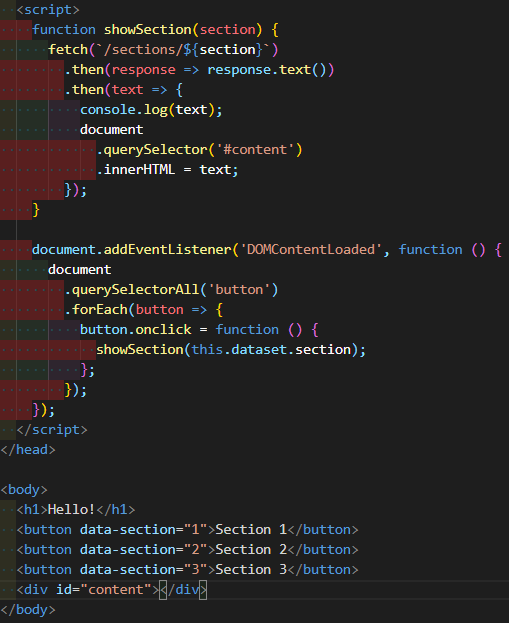
**# singlepage/views.py**

****

And two corresponding routes in views.py. Notice that the section route takes in an integer, and then returns a string of text based on that integer as an HTTP Response.

index.html 파일의 button 클릭을 통해 입력 받은 num값에 해당하는 array texts[num - 1]의 값을 반환.

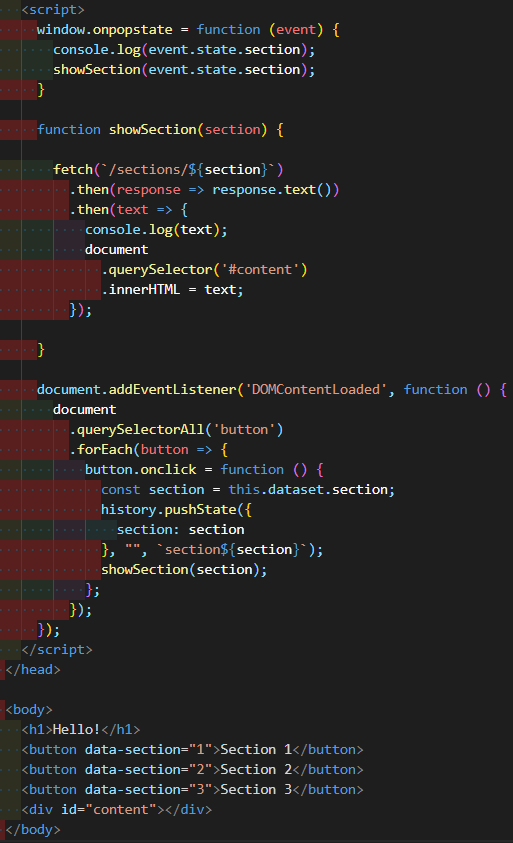
**# templates/singlepage/index.html**



within our index.html file, we’ll take advantage of AJAX, which we learned about last lecture, to make a request to the server to gain the text of a particular section and display it on the screen.

각 버튼을 클릭할 때마다 dataset의 data를 showSection의 parameter로 집어넣고, showSection의 parameter가 /sections/${section}에 성공적으로 입력되면 text형식의 data를 반환하고, text형식의 data를 성공적으로 반환하면, #content의 innerHTML로 text를 반환.

**JavaScript History API**



One disadvantage of our site though is that the URL is now less informative. You’ll notice in the video above that the URL remains the same even when we switch from section to section. We can solve this problem using the JavaScript History API. This API allows us to push information to our browser history and update the URL manually. Let’s take a look at how we can use this API. Imagine we have a Django project identical to the previous one, but this time we wish to alter our script to be employ the history API.

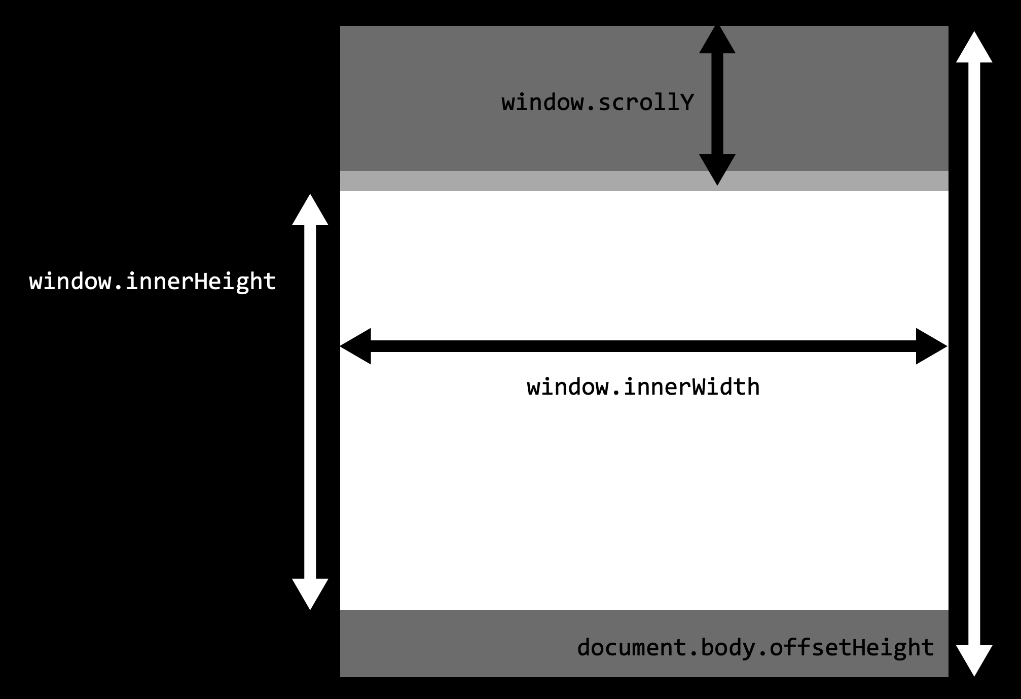
In the showSection function above, we employ the history.pushState function. This function adds a new element to our browsing history based on three arguments:

1. Any data associated with the state.
2. A title parameter ignored by most web browsers
3. What should be displayed in the URL
4. <https://developer.mozilla.org/ko/docs/Web/API/History/pushState>

The other change we make in the above JavaScript is in setting the onpopstate parameter, which specifies what we should do when the user clicks the back arrow. In this case, we want to show the previous section when the button is pressed. Now, the site looks a little more user-friendly:

history.pushState로써 세션 기록 항목에 state, title, url을 각각 저장함.

**Scroll**



* window.innerWidth: Width of window in pixels
* window.innerHeight: Height of window in pixels
* window.scrollY: How many pixels we have scrolled from the top of the page
* document.body.offsetHeight: The height in pixels of the entire document.

The following code, for example, will change the backgroud color to green when we reach the bottom of a page:

텍스트이(가) 표시된 사진

자동 생성된 설명