Ask Company

리액트와 함께 장고 시작하기 / 웹 프론트엔드 기초

Overview

여러분의 파이썬/장고 페이스메이커가 되겠습니다.

웹 프론트엔드를 위한 3가지 언어

- HTML (Hyper Text Markup Language) : 웹페이지의 내용 및 구조
- CSS (Cascasding Style Sheet) : 웹페이지의 스타일
- JavaScript : 웹페이지의 로직

위 세 언어는 모든 웹브라우저에서 거의 유사하게 동작합니다.

여러 버전의 HTML/CSS/JavaScript가 있으며,
브라우저마다 지원하는/구현된 Features가 다를 수도 있습니다.

1) 한 HTML 파일에 CSS/Javascript를 모두 넣기도 하며

```
<!doctype html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>AskDjango Blog</title>
        <style>
            body { background-color: lightyellow; }
            #post_list .post { margin-bottom: 3px; }
        </style>
    </head>
    <body>
        <ul id="post_list">
            <li class="post">장고 기본편</li>
            <li class="post">서비스 배포하기</li>
        </ul>
        <script>
        /* CSS 만으로 충분하지만, Javascript 로도 스타일을 지정할 수 있습니다. */
        var posts = document.getElementsByClassName('post');
        for(var i=0; i<posts.length; i++) {
            var post = posts[i];
            post.style.backgroundColor = 'pink';
        }
        </script>
    </body>
</html>
```

2) 대개 별도 CSS/JavaScript 파일로 분리합니다.

```html
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>AskDjango Blog</title>
    <link href="blog.css" />
</head>
<body>
    <ul id="post_list">
        <li class="post">장고 기본편</li>
        <li class="post">서비스 배포하기</li>
    </ul>
    <script src="blog.js"></script>
</body>
</html>
```

blog.css

```css
body {
background-color: lightyellow;
}
#post_list .post {
margin-bottom: 3px;
}
```

blog.js

```javascript
var posts = document.getElementsByClassName('post');

for(var i=0; i<posts.length; i++) {
    var post = posts[i];
    post.style.backgroundColor = 'pink';
}
```

CSS/JavaScript 파일을 별도 파일로 분리하는 이유

1. HTML 응답 body 크기를 줄일 수 있습니다.
2. 여러 번 새로고침하더라도, 브라우저 캐싱기능을 통해 같은 파일을 서버로부터 다시 읽어들이지 않습니다.
3. 웹페이지 응답성을 높여줄 수 있습니다.

CSS/JavaScript의 슈퍼셋 언어

CSS 소스코드

- 처음에는 직접 CSS 날코딩을 하시고,
- 추후에는 성향에 따라 Sass, Less를 검토해보세요.
  - sass, less 문법으로 작성된 코드를 빌드하여, css파일을 만들어냅니다.

JavaScript

- 처음에는 직접 JavaScript 날코딩을 하시고,
- 추후에는 성향에 따라 TypeScript를 검토해보세요.
  - TypeScript 문법으로 작성되는 코드를 빌드하여, javascript파일을 만들어냅니다.

웹 프론트엔드와 백엔드

웹 프론트엔드와 백엔드

웹개발은 크게 백엔드와 프론트엔드 개발로 나눠집니다. 장고는 백엔드에 초점이 맞춰진 웹프레임워크입니다.

장고를 공부하실 때에는 백엔드에 포커스를 맞춰서 공부하시고, 웹프론트엔드는 최소화하세요.

- 2가지를 모두 한번에 잘할 수는 없습니다. 우선순위를 백엔드에 먼저 두세요. Android/iOS 앱도 일단 미뤄두세요.
- 단 번에 완전한(?) 웹서비스를 만들 순 없습니다. 점진적으로 개선해가세요.

개발 언어

프론트엔드 개발언어 (여러 클라이언트 단 중 하나)

- 브라우저 단에서 실행이 됩니다.
- HTML/CSS/JavaScript의 조합

보통의 브라우저, CLI 전용 브라우저, Android/iOS앱 등 HTTP요청을 보낼 수 있는 어떠한 프로그램이든지, 클라이언트가 될 수 있습니다.

백엔드 개발언어 (서버 단)

- 클라이언트 단의 요청을 처리/응답만 할 수 있으면 됩니다.
- 다양한 언어가 가능하며, 여러 언어/프레임워크를 섞어쓰실 수도 있습니다. (Python, NodeJS, Ruby, Java 등)

웹 요청 및 응답

웹은 HTTP(S) 프로토콜로 동작합니다.

하나의 요청은 클라이언트가 웹서버로 요청하며, 웹서버는 요청에 맞게 응답을 해야합니다.

- 응답은 HTML 코드 문자열, CSS 코드 문자열, JavaScript 코드 문자열, Zip, MP4 등 어떠한 포맷이라도 가능합니다.

웹서버에서 응답을 만들 때, 요청의 종류를 구분하는 기준

- URL (일반적), 요청헤더, 세션, 쿠키 등

웹서버 구성에 따라

- /static/flower.jpg : JPG파일 내용을 응답으로 내어주도록 설정했습니다.
- /blog/images/flower.jpg : 장고 View를 통해, JPG파일 내용을 응답으로 내어주도록 설정했습니다.
- /blog/images/flower/ : 장고 View를 통해, JPG파일 내용을 응답으로 내어주도록 설정했습니다.

웹서버 구성에 따라, 특정 요청에 대한 응답을 Apache/Nginx 웹서버에도 할 수도 있고 Django 뷰에서 응답을 할 수도 있습니다.

일반적인 장고 페이지의 예

뷰 : 포스팅 목록

```
# blog/urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('blog/', views.post_list, name='post_list'),
]

# blog/views.py
from django.shortcuts import render
from .models import Post

def post_list(request):
    return render(request, 'blog/post_list.html', {
        'post_list': Post.objects.all(),
    })
```

HTML 템플릿

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>AskDjango Blog</title>
</head>
<body>
    <h1>AskDjango Blog</h1>
    <ul>
        {% for post in post_list %}
            <li>
                {{ post.title }}
            </li>
        {% endfor %}
    </ul>
    <hr/>
    &copy; 2019, AskDjango.
</body>
</html>
```

단지 하나의 HTTP 요청에 대해, 하나의 응답을 받습니다.

1. 브라우저에서 서버로 HTTP 요청
2. 서버에서는 해당 HTTP요청에 대한 처리 : 장고에서는 관련 뷰 함수가 호출
3. 뷰 함수에서 리턴해야만 비로소 HTTP응답이 시작되며, 그 HTTP 응답을 받기 전까지는 **하얀 화면**만 보여집니다. 따라서 뷰 처리시간이 길어질수록 긴 화면이 보여지는 시간이 길어집니다.
4. 브라우저는 서버로부터 HTTP 문자열 응답을 1줄씩 해석하여, 그래픽적으로 표현합니다.

아직, HTML 문자열 응답에 추가 리소스 (CSS, JavaScript, Image 등) 가 없습니다.

만약 HTML 문자열 응답에 추가 리소스가 지정되어있다면?

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>AskDjango Blog</title>
    <link rel="stylesheet" href="/static/style.css" />
    <script src="/static/jquery-3.2.1.min.js"></script>
</head>
<body>
    <h1>Ask Django Blog</h1>
    <ul>
        {% for post in post_list %}
            <li>
                {{ post.title }}
            </li>
        {% endfor %}
    </ul>
    <hr/>
    &copy; 2019, Ask Company.
</body>
</html>
```

HTML 문자열은 1줄씩 처리되며, 외부 리소스는
해당 리소스가 로딩완료/실행될 때까지 대기합니다

HTML UI 응답성이 낮아지는 경우

과도한 JavaScript 로딩 및 계산

과도한 CSS 레이아웃 로딩 및 계산

잦은 시각적 개체 업데이트

HTML UI 응답성을 높이기 위해

실서비스시에 CSS/JavaScript파일은 Minify시켜서 다운로드 용량을 줄입니다.

대개 CSS를 HTML컨텐츠보다 앞에 위치시키고

- CSS가 컨텐츠보다 뒤에 위치한다면, 유저에게는 스타일이 적용되지 않은 HTML컨텐츠가 먼저 노출될 수 있습니다.

JS를 HTML컨텐츠보다 뒤에 위치시킵니다.

- JS는 스타일에 직접적인 영향을 끼치지 않기 때문에, HTML컨텐츠보다 나중에 로딩되어도 **대개** 괜찮습니다. 꼭 필요한 몇몇 JS는 HTML컨텐츠보다 앞에 위치시키기도 합니다.

구글맵이 나오면서부터,

문서의 시대 → 웹 애플리케이션의 세계 OPEN

표준 웹기술만으로 웹문서에서 탈피하여, 웹 애플리케이션 ~ !!!

웹 애플리케이션에서는 복잡한 UI 처리가 필요하므로, React/Vue/Angular 등을 많이 사용합니다.

표준 웹기술만으로 지도 서비스가 가능하다니 !!!

- 이전에는 ActiveX 혹은 Flash를 통해서만 구현가능한 것인줄 ...

Life is short.
You need Python and Django.

I will be your pacemaker.