jQuery의 개요

* jQuery

- 전 세계에서 가장 많이 사용되는 JavaScript 라이브러리
- HTML 요소 제어
  => HTML 요소를 손쉽게 획득하고 제어할 수 있다.
- 손쉬운 이벤트 처리
  => 단 한번의 이벤트 정의로 크로스 브라우징 해결
- 요소의 탐색과 생성
  => DOM 요소를 제어하기 위한 간결한 기능 제공
- Ajax 통신처리
  => 페이지 이동이 없는 비동기 데이터 통신

※ 라이브러리 ?
자바스크립트로 만든 다양한 함수들의 집합

* jQuery 특징

- 크로스 브라우징
  => 한 번의 코딩으로 거의 모든 브라우저에게 적용된다.
- 간결한 문법
  => HTML 태그 (element)를 제어하거나 이벤트를 처리하는 부분 등 Javascript의 전반에 걸쳐서 구문이 짧아지기 때문에 개발 효율성이 높아진다.
- 익숙한 구문
  => CSS에서 사용하던 속성과 값을 그대로 Javascript 구문에 적용할 수 있어서 document 내장 객체가 제공하는 기능을 쉽게 사용할 수 있다.
- 다양한 플러그인
  => jQuery를 기반으로 하는 수 많은 플러그인들이 무료로 배포되고 있기 때문에, 갤러리, 메뉴 등의 구현에 대한 작업이 많이 단축된다.

jQuery 개발 환경

1. jQuery 라이브러리

- 사이트 : http://www.jquery.com
- production 버전 다운로드

  "Download the compressed, production jQuery 3.3.1"

2. jQuery 참조하기

① js 파일을 다운 받아서 사용

```
<script type="text/javascript" src="js/jquery-3.7.1.min.js"></script>
```

② http://code.jquery.com의 CDN 서비스를 사용

```
<script type="text/javascript" src="http://code.jquery.com/jquery-3.7.1.min.js"></script>
```

③ http://code.jquery.com의 CDN 서비스를 버전 명시 없이 사용

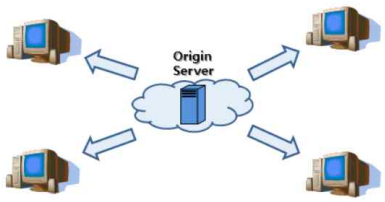
```
<script type="text/javascript" src="http://code.jquery.com/jquery.min.js"></script>
```

CDN(Contents Delivery Network) 이란?

- 지리,물리적으로 떨어져 있는 사용자에게 컨텐츠를 더 빠르게 제공할 수 있는 기술
- 느린 응답속도 / 다운로딩 타임 을 극복하기 위한 기술

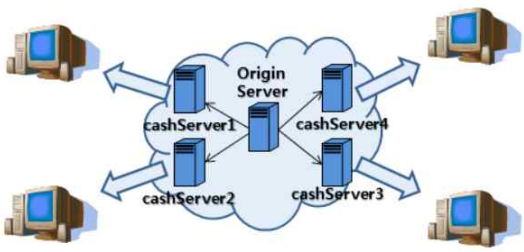
사용자가 원격지에 있는 서버(Origin Server)로 부터 Content(ex Web Object, Video, Music, Image, Document 등)를 다운로드 받을때 가까이 있는 서버에서 받는 것보다 시간이 오래 걸린다.
그러므로 사용자와 가까운 곳에 위치한 Cache Server에 해당 Content를 저장(캐싱)하고 Content 요청시에 Cache Server가 응답을 주는 기술이다.

▶ CDN을 사용하지 않는 경우



콘텐츠를 담고있는 서버(Origin Server)들은 모든 사용자의 요청에 일일이 응답해야 한다.
이는 막대한 트래픽을 유발하고, 트래픽이 과도하게 증가하거나 부하가 끊임없이 들어오는 경우 장애가 발생할 확률도 크다.

▶ CDN을 사용하는 경우



CDN을 사용함으로써 서버의 트래픽 부하 및 비용을 줄이고 사용자에게 빠른 서비스 제공도 가능하다. 장애 확률도 낮춰 줄 수 있다.

Hello jQuery

1. jQuery() 함수 사용

Javascript는 window 객체의 addEventListener() 함수와 attachEvent() 함수를 사용하여
onload 이벤트를 처리해야 하는데 이때 크로스 브라우징 처리를 하기 위해 if문으로 브라우저를 구별하는 분기문을 작성해야만 한다.
jQuery는 이러한 번거로움을 jQuery() 함수 하나로 해결할 수 있게 해준다.

예) jQuery() 함수에게 파라미터로 사용하려는 함수명을 전달하는 것으로 onload 처리가 완료된다.

```
function ex() {
    ......
}
jQuery(ex);
```

2. $ 객체의 사용

(1) jQuery() 함수로부터 전달되는 파라미터 받기

- jQuery의 모든 기능은 $ 라는 객체의 하위로 포함되어 있다.
- $ 는 jQuery의 모든 기능을 담고 있는 객체이자 함수이다.
- jQuery() 함수로 페이지가 열릴 때, 특정 함수를 호출하게 되면,
  이 특정함수는 파라미터로 $라는 객체를 받는다.
  이 $객체로 jQuery의 막강한 기능들을 사용할 수 있다.

예)

```
function ex($) {
    $ 객체 사용
}
jQuery(ex);
```

(2) HTML 요소를 획득하기

- Javascript에서 사용하던 document.getElementById("id") 구문이 jQuery에서는 $() 함수를 통해 작동하는데, $("CSS 셀렉터")의 축약된 형태로 객체 생성 표현을 제공한다.
- Javascript의 addEventListener() 함수를 사용하여 onload 이벤트를 처리하면서 HTML 요소를 객체로 생성하는 경우, 웹 브라우저에 의해 모든 HTML 요소가 로드된 후에 객체를 생성해야 하기 때문에 onload 이벤트에 의해서 호출되는 함수 안에서 객체획득이 가능하였다.
- jQuery 역시 이와같은 웹브라우저의 특성을 따라 $() 함수에 의한 객체생성은 jQuery() 함수에 의해서 호출되는 콜백함수 안에서 수행되어야 한다.

예)

```
var h1 = document.getElementById("hello");
=> var h1 = $("#hello");
```

(3) innerHTML의 간결화
- $() 함수를 사용하여 획득한 요소는 html() 함수를 내장하고 있는 객체가 된다.
- 이렇게 생성된 객체는 innerHTML 대신 html() 함수를 사용하여 요소안에 새로운 내용을 추가할 수 있다.

예)
h1.innerHTML = "Hello Javascript";
=> h1.html("Hello jQuery");

(4) 함수 이름 대신 함수 자체를 사용하기
① 함수 이름 사용
```
function ex() {
    ……
}
jQuery(ex);
```

② 함수 자체를 사용
```
jQuery(function() {
    ……
});
```

③ $객체를 전달받기 위해 파라미터 선언
```
$(function($) {
    $ 객체 사용
});
```

④ jQuery() 함수의 축약
```
$(function() {
    ……
});
```
=> $() 함수에 의해서 호출되는 함수는 $객체에 대한 파라미터를 명시하지 않아도,
자동으로 전달받게 된다.