Interactive Web Page with jQuery

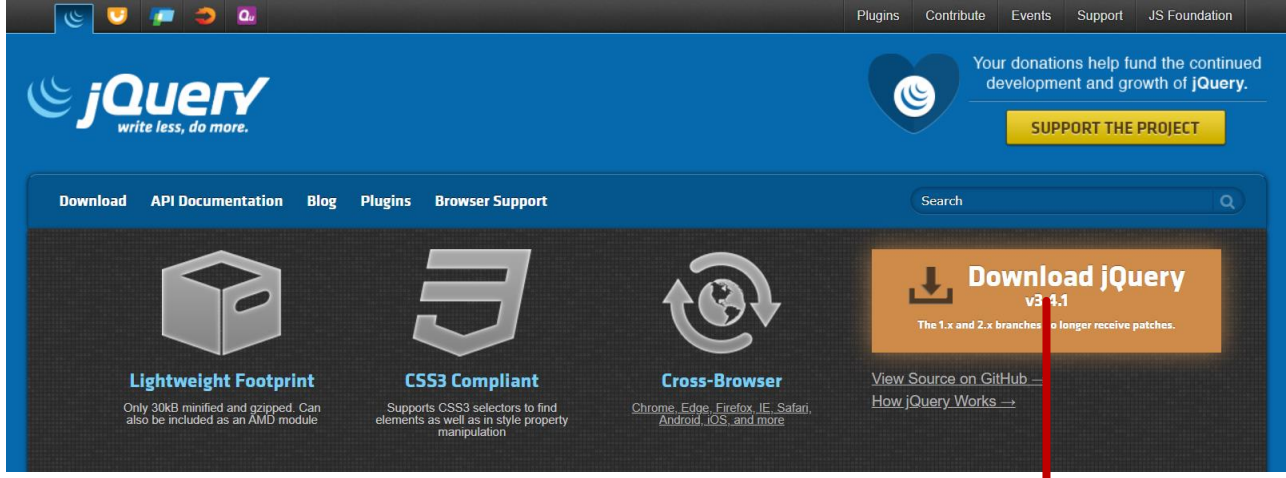
jQuery 소개

- jQuery
  - Mozilla의 JavaScript 관련 툴을 개발하던 존 레식이 작성, 2006년에 발표한 라이브러리
  - JavaScript와 HTML 사이의 상호작용을 강조하는 경량화된 웹 애플리케이션 프레임워크(FrontEnd)
  - MIT 라이선스와 GNU 일반 공중 사용 허가서 v2의 듀얼 라이선스를 가진 자유 오픈 소프트웨어

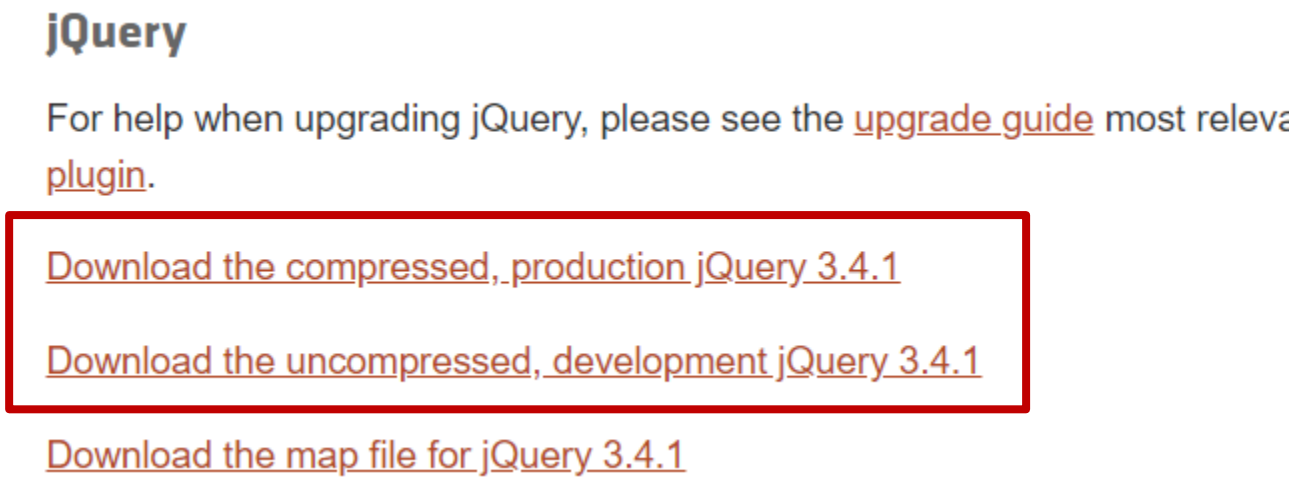
- jQuery의 특징
  - 대부분의 웹 브라우저(IE, Chrome, Opera, FireFox, Safari 등)를 지원하여 호환성 확보(크로스 브라우징)가 용이
  - 여러 개의 동작을 한 줄에 나열하여 임시 변수의 사용을 최소화하고 코드의 길이를 간략하게 함 (메서드 체인)
  - 클라이언트 객체를 통해 작업하므로, 웹 페이지 로딩 체감하기 힘들 만큼 빠르고 가벼움
  - 서버 사이트 코드와 클라이언트 사이트 코드를 효과적으로 분리, 분업 작업이 가능

jQuery 설치
: 다운로드 받아 설치하기

- 라이브러리 다운로드
  - https://jquery.com/ 에서 다운로드

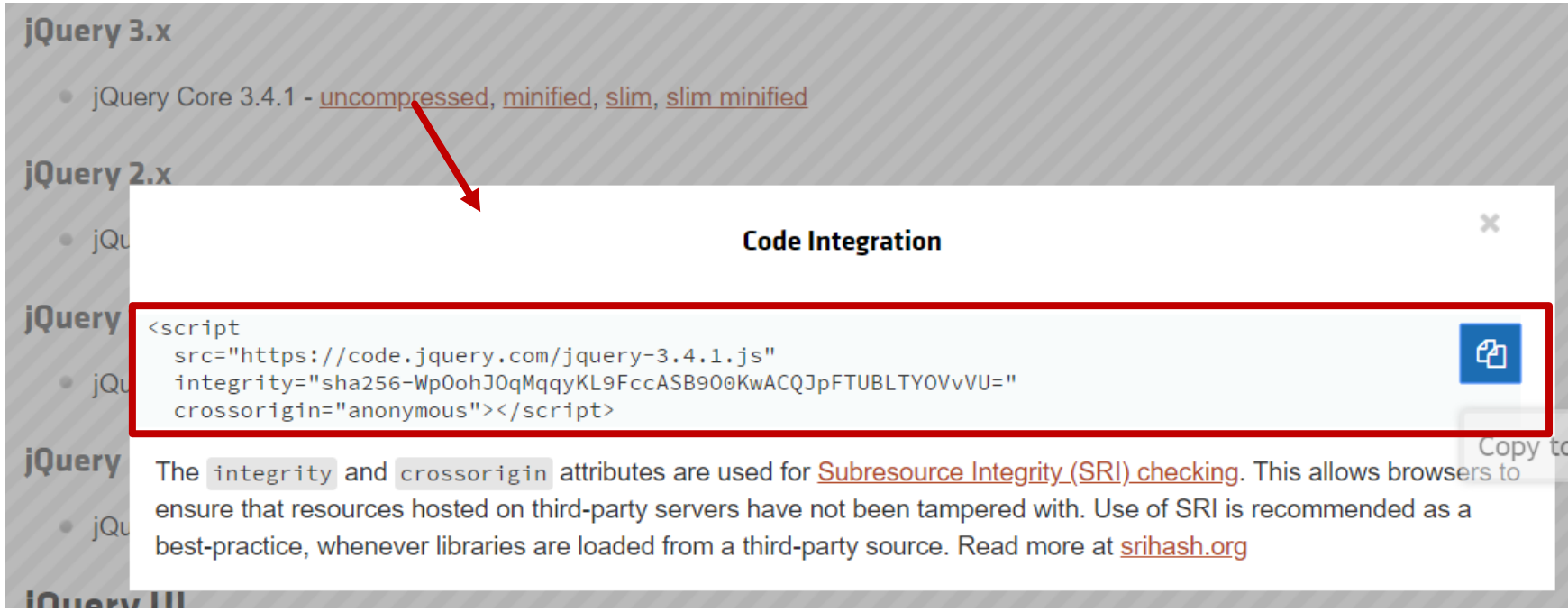


  - 원하는 jQuery 버전을 선택하여 다운로드
    - compressed 버전 : 소스 상 개행과 공백을 제거, 경량화한 버전
      실제 운영 중(Production)인 서비스에 적합
    - uncompressed 버전 : 사용자가 분석할 수 있도록 jQuery의 문서 내용을 보기 쉽게 해 둔 버전
      개발중(Development)인 서비스에 적합



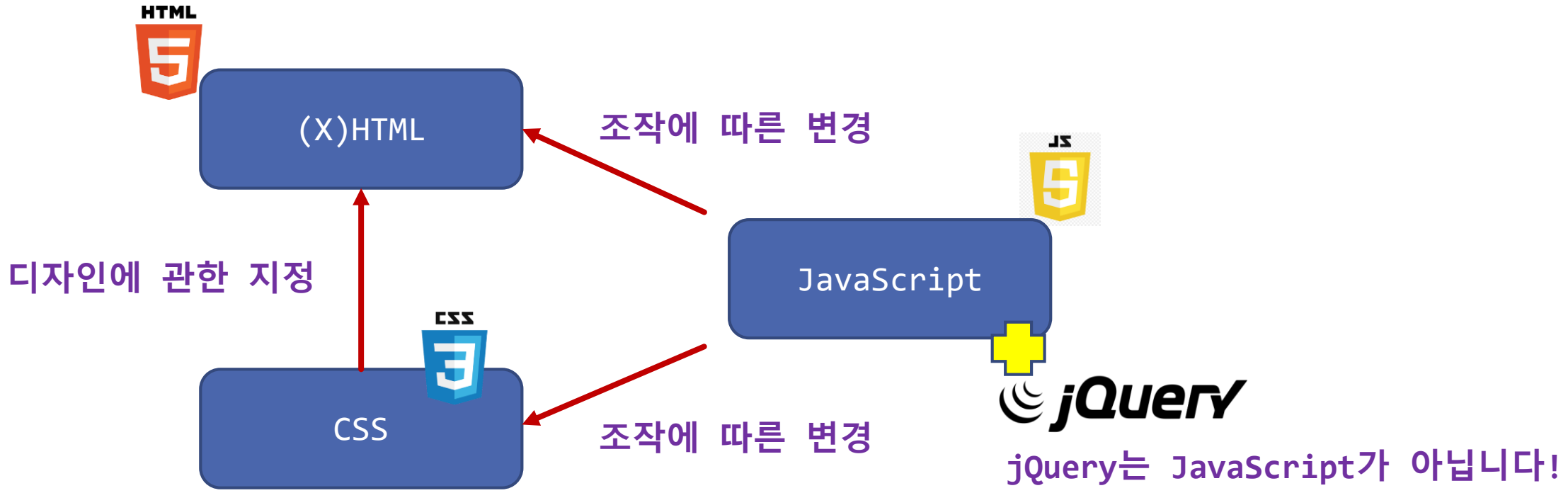
jQuery 설치
: CDN 버전 링크

- 원하는 CDN 사이트 선택
  - or https://code.jquery.com/ 에 접속
- 제공되는 script 태그 복사하여
  자신의 웹 페이지에 붙여 넣기



jQuery 시작하기
: 기본 지식

- (X)HTML + CSS + JavaScript(with jQuery)



jQuery 시작하기
: 기본 지식

- jQuery 라이브러리 사용
  1) Download한 .js 파일을 html 페이지 내 코드로 추가
  2) CDN(Content Delivery Network)을 이용한 HTTP 주소를 html 페이지 내에 추가

jQuery 기본 표현식

- 정식 명칭은 jQuery()로 표기하지만 통상적으로 $()로 축약하여 사용

```
jQuery("h1").css("color", "red");
```

jQuery 선택자 메서드 호출

Expression	Return Type	Arguments
$(htmlString)	Object	htmlString : HTML에 추가할 문자열
$(element)	Array<Object>	element : 검색할 element 요소
$(callback)	Object	callback : DOM이 모두 load된 후 실행할 함수

jQuery 시작하기
: 기본 지식

- jQuery 객체의 초기화
  - $(document).ready(function() { … }); 를 사용하여 jQuery 객체들의 초기화 작업을 수행할 수 있다

```
<script>
$(document).ready(function() {
    //  jQuery 객체의 초기화 코드 수행
    var p = $("p").get();
    console.log("P의 갯수", p.length);
});
</script>
<body>
    <p>홍길동</p>
    <p>장길산</p>
    <p>전우치</p>
</body>
```

jQuery 시작하기
: 기본 지식

- Wrapper
  - Wrapper란?
    - 무엇인가를 감싼다는 의미
      ($ 또는 jQuery로 감싼 것)
    - 엘리먼트 객체를 전달하거나 CSS 스타일 선택자를 전달
      : JavaScript의 document.getElementById(), document.getElementsByClassName()
      등에서 리턴되는 객체(혹은 객체의 배열)
    - 많이 활용되는 선택자는 id 셀렉터(#)와 class 셀렉터(.)
  - Wrapper의 안전한 사용
    - $(element)는 다른 JavaScript 라이브러리에서도 많이 사용하기 때문에 충돌 방지를 위해 명시적으로 jQuery(element)를 사용하기도 함
    - 다음과 같은 코드로 안전하게 jQuery 코드를 작성하기도 함

```
(function($) {
  //  $를 사용한 jQuery 코드 작성
})(jQuery);
```

jQuery 시작하기
: 기본 지식

- 메서드 체인(Method Chain)
  - jQuery의 강력한 장점 중 하나로 복잡한 기능을 간결한 코드로 작성할 수 있다
  - 인간 사고의 자연스러운 흐름과 일치하는 코드를 만들어낼 수 있다
  - jQuery의 메서드들은 자기 자신의 엘리먼트를 반환하기 때문에 체인처럼 꼬리에 꼬리를 무는 메서드 활용 코드를 작성할 수 있다

HTML

```
<body>
    <a href="#" id="foo">Some Link</a>
</body>
```

```
<script>
    jQuery("#foo")
        .attr("href", "http://www.naver.com")
        .text("네이버")
        .css("color", "green");
</script>
```

jQuery Code

Vanilla JS

```
<script>
    var foo = document.getElementById("foo");
    foo.setAttribute("href", "http://www.naver.com");
    foo.innerText = "네이버";
    foo.style.color = "green";
</script>
```

선택자와 탐색

선택자 기본

- 선택자(Selector)
  - CSS에서 조건에 맞는 엘리먼트(들)를 선택하기 위한 지시자
  - jQuery는 주어진 Selector에 따라 가장 적합한 객체(혹은 객체의 배열)를 반환한다
    - document.getElementById
    - document.getElementsByClassName
    - document.getElementsByTagName
    - document.querySelector

주어진 선택자에 따라 가장 적합한 메서드를 수행하고 결과를 반환

- 기본 선택자: 태그명, 클래스, ID를 이용한 선택

Selector	설명
*	모든 엘리먼트 선택
태그명	해당 태그로 작성된 모든 엘리먼트 선택
.class	엘리먼트의 클래스가 지정된 .class와 동일한 모든 엘리먼트 선택
#id	해당 id를 id 어트리뷰트로 가지고 있는 엘리먼트 선택

선택자 기본

- 속성 선택자
  - 특정 속성(Attribute)을 가진 엘리먼트 또는 특정 속성이 특정 값을 가진 요소를 선택
  - 입력 양식과 관련된 태그를 선택할 때 많이 활용

Selector	설명
[속성]	E1 엘리먼트의 모든 자손 E2 엘리먼트를 선택
[속성=값]	E1 엘리먼트의 모든 자식(직계) E2 엘리먼트를 선택
[속성!=값]	E1 엘리먼트의 바로 다음에 나오는 형제 요소 E2 엘리먼트 선택

탐색 기본

- 탐색(Traversing)이란?
  - 요소(Element)들의 집합에서 특정 개체를 찾거나 필터링, 추가하는 등의 작업을 위한 것
  - jQuery 기본 형식 안에 코딩에서 요소들에 대한 위치 기반으로 필터링할 수 있다
  - 선택자(Selector) 뒤의 콜론 다음에 기술하며 '선택자:필터' 형식으로 사용한다

- 인덱스(순서) 필터(Index Filter)

Filter	설명
:first	첫 번째 객체
:last	마지막 객체
:eq(n)	n번째 객체. 첫 번째 항목이 0(Zero-Based)
:gt(n)	n번째 초과 객체
:lt(n)	n번째 미만 객체
:odd	홀수 번째 객체
:even	짝수 번째 객체

탐색 기본

- 자식 필터(Child Filter)

Filter	설명
:first-child	첫 번째 자식 요소
:last-child	마지막 자식 요소
:first-of-type	첫 번째 일치하는 타입의 엘리먼트
:last-of-type	마지막 일치하는 타입의 엘리먼트
:nth-child(n)	n 수식을 만족하는 객체. 또는 even, odd로 짝홀번째 항목을 선택. 첫번째 항목이 1(One-Based)
:nth-of-type(n)	n 수식을 만족하는 타입의 엘리먼트
:only-child	유일한 자식 요소
:only-of-type	유일한 타입의 자식 요소

탐색 기본

- 내용 필터(Contents Filter)
  - 엘리먼트의 내용에 따라 특정 항목을 가진 엘리먼트만 추출
  - 하부의 모든 후손을 검색함

Filter	설명
:has(ex)	ex 태그를 가진 엘리먼트만 선택
:contains(str)	str 문자열을 가진 엘리먼트만 선택
:empty	내용이 비어있는 엘리먼트
:parent	자식을 가지는 요소

탐색 기본

- 입력 양식 필터(Form Filter)
  - 특정 입력 양식, 혹은 입력 양식의 상태에 따른 검색

Filter	설명
:button	입력 양식(input)의 타입이 button인 요소 선택
:checkbox	입력 양식(input)의 타입이 checkbox인 요소 선택
:text	입력 양식(input)의 타입이 text인 요소 선택
:image	입력 양식(input)의 타입이 image인 요소 선택
:submit	입력 양식(input)의 타입이 submit인 요소 선택
:hidden	입력 양식(input)의 타입이 hidden인 요소 선택
:focus	입력 양식이 활성화 되어 있는 요소를 선택
:disabled	입력 양식이 사용 불가 상태인 요소를 선택
:enabled	입력 양식이 사용 가능 상태인 요소를 선택
:selected	option 태그에서 선택된 요소를 선택

문서 조작

문서 객체 조작 기본

- 문서 객체 조작(Manipulation)
  - JavaScript만으로 문서 객체 모델(DOM)을 다루려면 복잡하지만, jQuery를 이용하면 손쉽게 다룰 수 있다
  - 조작이라 함은 엘리먼트에 값을 지정한다거나, 특정 엘리먼트의 값을 읽어들인다거나, 동적으로 엘리먼트를 생성, 추가, 복사, 제거하는 기능들을 말함
  - 조작과 관련된 메서드들은 성격별로 카테고리가 나누어져 있다
    - 내용 관련 메서드
    - 속성 관련 메서드
    - 스타일 관련 메서드

문서 객체 조작 기본

- 내용 관련 메서드
  - 문서 객체의 원하는 엘리먼트의 내용을 찾아 바꾸거나 읽어오는 것

Method	설명
html()	- 일치하여 반환된 엘리먼트 내부의 html을 읽어옴 - 엘리먼트의 innerHTML 속성과 동일한 내용 - 엘리먼트가 여러 개 리턴될 경우, 첫 번째 엘리먼트의 html을 읽어옴
html(value)	- 선택된 엘리먼트의 html 내용을 value로 치환 - 엘리먼트가 여러 개일 경우, 모두 적용
text()	- 선택된 엘리먼트의 내용을 단순 text로 읽어옴 - 엘리먼트의 innerText 속성과 동일한 내용
text(value)	- 선택된 엘리먼트의 내용을 단순 텍스트 value로 치환

문서 객체 조작 기본

- 속성 관련 메서드
  - 인수의 개수에 따라 동작이 달라지며, 속성의 이름만 주어지만 읽기, 이름과 값을 전달하면 변경
  - 속성이 없으면 새로 추가, 기존 값이 있는 경우는 변경

Method	설명
attr(name)	- 검색된 요소들 중, name에 해당하는 속성 명을 가진 엘리먼트의 속성 값을 리턴
attr(name, value)	- 검색된 요소들 중, name에 해당하는 속성의 값을 value로 치환
removeAttr(name)	- 검색된 요소들 중, name과 일치하는 모든 속성을 제거
val()	- 검색된 요소들 중 처음 일치하는 폼 요소의 value 값을 문자열로 반환
addClass(name)	- 검색된 요소들에 name 문자열을 class의 속성 값으로 적용
removeClass(name)	- 검색된 요소들에 name 문자열로 지정된 class를 모두 제거
toggleClass(name)	- 검색된 요소들에 name 문자열로 된 class가 있으면 제거, 없으면 적용

문서 객체 조작 기본

- 스타일 관련 메서드
  - CSS를 HTML 문서에 지정할 수 있는 메서드

Method	설명
css(name)	- 처음 검색된 요소의 name에 해당하는 스타일 속성 값을 반환
css(name, value)	- 일치하는 요소의 스타일 속성값을 설정
height(value)	- 일치하는 요소의 높이를 설정
width(value)	- 일치하는 요소의 너비를 설정

Event

Event 기본

- 이벤트(Event)란?
  - 어떤 액션에 의해서 발생하는 사건을 의미
  - 키보드 입력, 마우스 오버, 마우스 클릭 등
  - 이벤트가 수행되는 경우, 특정 메서드와 매칭하여 사용하며 이벤트 처리기라고도 함

- 이벤트 등록 메서드
  - jQuery에서는 사용자가 발생시키는 이벤트를 핸들링(처리)하는 메서드를 지원
  - 마우스 클릭, 키보드 입력 등 여러 가지 이벤트를 처리할 수 있음

Event 기본
: 이벤트 지원 메서드

Event	설명
.blur()	요소에서 포커스를 잃을 때 발생
.click()	요소를 클릭했을 때 발생
.focus()	요소가 포커스를 얻었을 때 발생
.hover()	마우스가 요소 위에 위치했을 때 발생
.keydown()	키 입력시 발생, 모든 키에 대해 적용
.keypress()	키 입력시 발생되지만, enter, tab 등의 특수 키에는 무시
.keyup()	키 입력 후 발생
.mousedown()	마우스 클릭시 발생
.mouseup()	마우스 클릭 후 발생
.ready()	DOM이 모두 준비 되었을 때 발생

- 이벤트 내에서의 this 키워드는 이벤트를 발생시킨 요소를 의미하며, 이때 전달되는 this는 DOM 객체이다
- 전체 이벤트는 https://api.jquery.com/category/events/ 에서 확인

Event 기본
: bind() 메서드 사용

- bind()

```
.bind(eventType, [eventData], handler(eventObject))
```

  - 객체와 이벤트를 연결해 주는 역할을 수행
  - bind()의 경우, 첫 번째 파라미터 값으로 이벤트의 이름을 문자열로 지정
  - 매우 간단히 이벤트를 동적으로 할당할 수 있는 장점

- 개별 이벤트 핸들러의 등록

```
$("h1").click(function() { console.log("Hello") });
```

- bind()를 이용한 이벤트 핸들러의 등록

```
$("h1").bind("click", function() { console.log("Hello") });
```

Event 기본
: bind() 메서드 사용

- unbind()

```
.unbind([eventType] [, data])
```

  - bind 된 이벤트를 요소에서 삭제
  - eventType 인자로는 제거할 이벤트 명을 문자로 기술

- trigger()

```
.trigger(eventType [,extraParameter])
```

  - eventType : JavaScript 이벤트 타입 문자열
  - extraParameter : 이벤트 핸들러에 전달할 파라미터

Event 기본
: trigger() 메서드 사용

- trigger()

```
.trigger(eventType [,extraParameter])
```

  - Trigger란 방아쇠라는 의미, 특정 이벤트를 강제 발생시키는 것
    - eventType : JavaScript 이벤트 타입 문자열
    - extraParameter : 이벤트 핸들러에 전달할 파라미터
  - 특정 이벤트 유형에 대해 선택된 요소에 연결된 모든 핸들러와 동작을 실행
  - 이벤트 발생시 실행될 함수나 bind()로 연결된 이벤트를 강제 실행

```
$("#foo").bind("click", function() {
    alert($(this).text())
});
$("#foo").trigger("click");
```

foo id를 가진 요소의 click 이벤트를 강제 발생

Event 기본
: 이벤트 동적 바인딩

- on()

```
.on(eventType [,selector] [,data] handler(event))
```

  - eventType : JavaScript 이벤트 타입 문자열
  - selector : 이벤트를 발생시킬 선택자(id, class, tag 등)
  - data : Event Handler에 전달할 데이터 목록
  - handler(event): 이벤트 발생시 기능을 처리할 함수

  - on 메서드는 기존 bind, delegate, live를 대체하는 메서드
  - Event Handler 바인딩에 대한 기존 메서드의 기능을 모두 제공
  - jQuery 1.7 이상 버전에서는 on() 메서드를 사용할 것을 권장

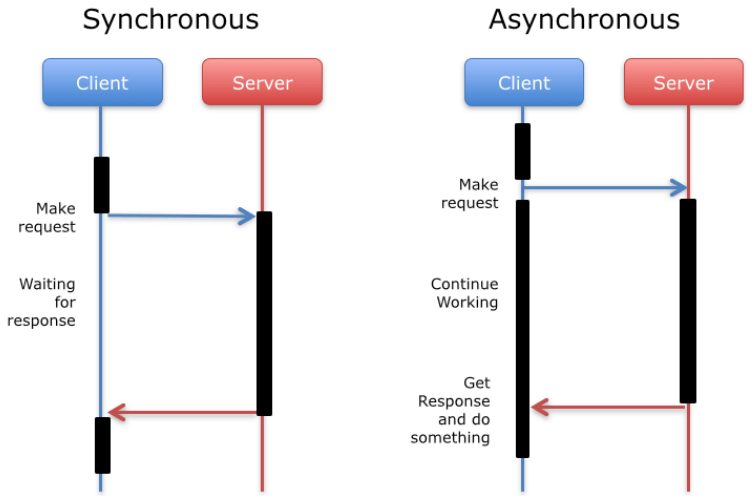
- off()
  - 이벤트 핸들러를 제거함

Ajax

Ajax 통신의 개념

- Ajax(Asynchronous JavaScript and XML)
  - 웹에서 화면을 갱신하여 데이터를 전달하지 않고, 서버로부터 **필요한 데이터만** 요청(Request)하여 받은 응답(Response)를 현재 화면에 갱신하는 방법
  - 브라우저에서는 새 페이지를 다시 로드하지 않고, 서버로 보낼 데이터를 Ajax Engine을 이용, 서버로 전송
  - Ajax Engine에서는 JavaScript를 통해 DOM을 사용하에 XMLHttpRequest 객체로 데이터를 전송
  - XMLHttpRequest 객체를 이용해서 비동기 방식으로 서버로부터 자료를 조회할 수 있다
  - 서버에서는 데이터를 전달할 때 전체 페이지의 HTML을 전달하지 않고 데이터(Text, XML, JSON 등), 혹은 화면의 일부에 해당하는 HTML 조각을 전달한다
  - 브라우저는 전달 받은 데이터 혹은 HTML 조각을 현재 화면에 수정 반영한다

Ajax 통신의 특징

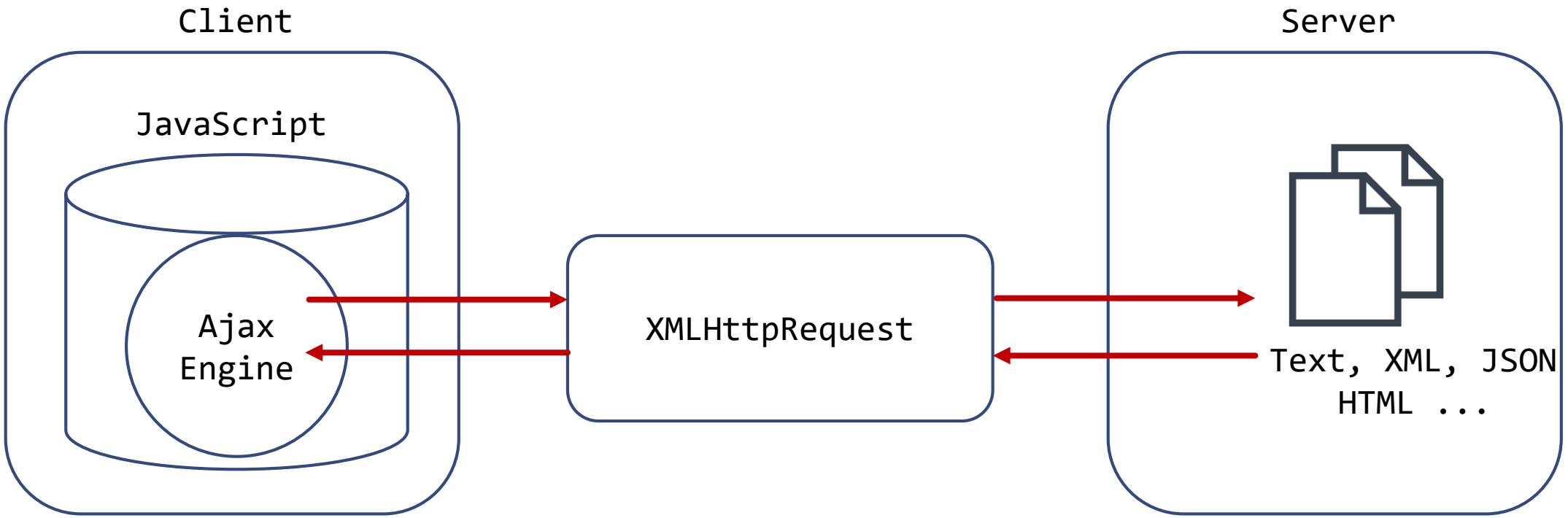


- Ajax 통신의 장단점

장점	단점
- Ajax 통신을 사용하지 않은 웹페이지에서는 서버로 요청을 보낼 때마다 화면 전체를 갱신하게 되지만, Ajax는 화면의 일부만 갱신하는 방법을 이용, 자원 낭비를 최소화 - 사용자가 화면 내에서 진행중인 작업을 초기화하지 않아도 된다(작업의 연속성 확보) - 서버에 핵심 데이터만 전송하므로 서버에 부담이나 속도 문제를 해결할 수 있다	- 브라우저의 버전이 낮거나 Ajax를 지원하지 않는 브라우저에서는 사용할 수 없다 - 화면의 이동 없이 데이터를 송/수신하므로 보안상의 문제를 일으킬 수 있다 - 디버깅이 쉽지 않다

Ajax 통신의 구조

- jQuery에서 Ajax 통신의 흐름도



Ajax 통신의 구조

- jQuery Ajax 통신 메서드

```
$.ajax(options)
```

  - options로 넘길 인자는 JavaScript Object 형식으로 작성, 인자는 가변적임

인자	설명
url	Ajax 요청을 전송할 주소 URL
type	서버로 데이터를 전송할 방식 지정(GET, POST)
data	서버로 전달할 데이터
dataType	서버가 반환하는 데이터 타입
success	Ajax 통신이 성공했을 시 실행할 콜백 함수 (함수의 첫 번째 인자는 서버가 리턴해주는 데이터의 결과)
error	Ajax 통신이 실패했을 시 실행할 콜백 함수 요청의 상태 코드와 에러 관련 정보를 확인할 수 있음

Ajax Example

- jQuery로 작성한 Ajax 통신 코드

```
//  jQuery로 작성한 ajax 통신
$("#foo").on("click", function(e) {
    $.ajax({
        url: "/ajaxJson.json",  //  URL
        type: "GET",            //  통신 방식
        data: "msg=" + $("#foo").val(), //  서버로 전송할 데이터
        dataType: "json",       //  서버로부터 전달 받을 데이터 형식
        success: function(result) {
            //  Ajax 통신에 성공했을 경우의 처리 콜백 함수
            console.log(result)
        },
        error: function(request, status, error) {
            //  Ajax 통신에 실패했을 경우의 처리 콜백 함수
            console.error(error);
        }
    })
});
```