

# jQuery

jQuery는 John Resig가 2006년에 만들었습니다. 브라우저 비호환성을 처리하고 HTML DOM 조작, 이벤트 처리, 애니메이션 및 Ajax를 단순화하도록 설계되었습니다.

jQuery는 JavaScript 라이브러리입니다.

라이브러리란 자주 사용되는 로직을 재사용하기 편리하도록 잘 정리한 일련의 코드들의 집합을 의미합니다.

- jQuery는 JavaScript 라이브러리입니다.
- jQuery는 JavaScript 프로그래밍을 크게 단순화합니다.
- jQuery는 배우기 쉽습니다.
- jQuery의 목적은 웹 사이트에서 JavaScript를 훨씬 쉽게 사용할 수 있도록 하는 것입니다.
- jQuery는 경량의 "적게 작성하고 더 많이 수행"하는 JavaScript 라이브러리입니다.
- jQuery의 목적은 웹 사이트에서 JavaScript를 훨씬 쉽게 사용할 수 있도록 하는 것입니다.
- jQuery는 수행하는 데 많은 JavaScript 코드가 필요한 많은 일반적인 작업을 수행하고 한 줄의 코드로 호출할 수 있는 메서드로 래핑합니다.
- jQuery는 또한 AJAX 호출 및 DOM 조작과 같은 JavaScript의 많은 복잡한 작업을 단순화합니다.

다운로드할 수 있는 두 가지 버전의 jQuery가 있습니다.

프로덕션 버전 - 축소 및 압축되었기 때문에 라이브 웹사이트용입니다.

개발 버전 - 테스트 및 개발용입니다(압축되지 않고 읽을 수 있는 코드).

두 버전 모두 jQuery.com에서 다운로드할 수 있습니다.

jQuery 라이브러리는 단일 JavaScript 파일이며 HTML `<script>` 태그로 참조합니다(`<script>` 태그는 `<head>` 섹션 안에 있어야 함).

jQuery CDN

```
<head>
<script src="jquery-3.6.1.min.js"></script>
</head>
```

ID로 HTML 요소 찾기

jQuery	myElement = \$("#id01");
JavaScript	myElement = document.getElementById("id01");

#### 태그 이름으로 HTML 요소 찾기

jQuery	myElements = \$("p");
JavaScript	myElements = document.getElementsByTagName("p");

#### 클래스 이름으로 HTML 요소 찾기

jQuery	myElements = \$(".intro");
JavaScript	myElements = document.getElementsByClassName("intro");

#### CSS 선택기로 HTML 요소 찾기

jQuery	myElements = \$("p.intro");
JavaScript	myElements = document.querySelectorAll("p.intro");

#### 텍스트 내용 설정

jQuery	myElement.text("Hello Sweden!");
JavaScript	myElement.textContent = "Hello Sweden!";

#### HTML 콘텐츠 설정

jQuery	myElement.html("<p>Hello World</p>");
JavaScript	myElement.innerHTML = "<p>Hello World</p>";

#### HTML 요소 숨기기

jQuery	myElement.hide();
JavaScript	myElement.style.display = "none";

#### HTML 요소 보이기

jQuery	myElement.show();
JavaScript	myElement.style.display = "";

#### HTML 요소 스타일링

jQuery	\$("#demo").css("font-size", "35px");
JavaScript	document.getElementById("demo").style.fontSize = "35px";

기타 예제: `$("#demo").css("font-size", "35px");` .css("속성명", "값")

"2개의 간격"은 `.css({속성명: "값", 속성명: "값"})`

## HTML 요소 제거

jQuery	<code>\$("#id02").remove();</code>
JavaScript	<code>document.getElementById("id02").remove();</code>

## 상위 요소 가져오기

HTML 요소의 부모를 반환합니다

jQuery	<code>myParent = \$("#02").parent().prop("nodeName"); ;</code>
JavaScript	<code>myParent = document.getElementById("02").parentNode.nodeName;</code>

## jQuery Syntax

jQuery 구문은 HTML 요소를 선택하고 요소에 대해 일부 작업을 수행하기 위해 맞춤 제작되었습니다.

기본 구문: `$(selector).action()`

jQuery 정의/액세스를 위한 \$ 기호

HTML 요소를 "조회(또는 찾기)"하는 선택기

요소에서 수행할 jQuery action()

The Document Ready Event(문서 준비 이벤트)

```
$(document).ready(function(){
```

```
    // jQuery methods go here...
```

```
});
```

문서 준비 이벤트를 위한 더 짧은 메서드도 있습니다.

```
$(function(){
```

```
    // jQuery methods go here...
```

```
});
```

이는 문서 로드가 완료(준비)되기 전에 jQuery 코드가 실행되는 것을 방지하기 위한 것입니다. 작업하기 전에 문서가 완전히 로드되고 준비될 때까지 기다리는 것이 좋습니다. 이것은 또한 헤드 섹션에서 문서 본문 앞의 JavaScript 코드를 가질 수 있게 합니다.

jQuery (문서)

jQuery Script  
문서 준비

## jQuery Selectors

jQuery 선택기를 사용하면 HTML 요소를 선택하고 조작할 수 있습니다.

jQuery 선택기는 이름, ID, 클래스, 유형, 속성, 속성 값 등을 기반으로 HTML 요소를 "찾는" (또는 선택하는) 데 사용됩니다. 기존 CSS 선택기를 기반으로 하며 추가로 자체 사용자 지정 선택기가 있습니다.

jQuery의 모든 선택기는 달러 기호와 괄호 `$()`로 시작합니다.

### The element Selector

jQuery 요소 선택기는 요소 이름을 기반으로 요소를 선택합니다.

ex) `$("p")`

### The #id Selector

jQuery #id 선택기는 HTML 태그의 id 속성을 사용하여 특정 요소를 찾습니다. ID는 페이지 내에서 고유해야 하므로 고유한 단일 요소를 찾으려면 #id 선택기를 사용해야 합니다. 특정 id를 가진 요소를 찾으려면 해시 문자를 쓰고 그 뒤에 HTML 요소의 id를 씁니다

ex) `$("#test")`

### The class Selector

jQuery class 선택기는 특정 클래스가 있는 요소를 찾습니다.

특정 클래스의 요소를 찾으려면 마침표와 클래스 이름을 씁니다.

ex) `$(".test")`

## More Examples of jQuery Selectors

Syntax	Description
<code>\$("*")</code>	모든 요소를 선택합니다.
<code>\$(this)</code>	현재 HTML 요소를 선택합니다.
<code>\$("p.intro")</code>	class="intro"인 모든 <p> 요소 선택
<code>\$("p:first")</code>	첫 번째 <p> 요소를 선택합니다.
<code>\$("ul li:first")</code>	<ul>의 첫 번째 <li> 요소를 선택합니다.
<code>\$("ul li:first-child")</code>	모든 <ul>의 첫 번째 <li> 요소를 선택합니다.
<code>\$("[href]")</code>	href 속성이 있는 모든 요소를 선택합니다.
<code>\$("a[target='_blank']")</code>	대상 속성 값이 "_blank"인 모든 <a> 요소를 선택합니다.
<code>\$("a[target!='_blank']")</code>	대상 속성 값이 "_blank"와 같지 않은 모든 <a> 요소를 선택합니다.
<code>\$(":button")</code>	type="button"의 모든 <button> 요소 및 <input> 요소를 선택합니다.
<code>\$("tr:even")</code>	모든 짝수 <tr> 요소를 선택합니다.
<code>\$("tr:odd")</code>	모든 홀수 <tr> 요소를 선택합니다.

Selector	Example	Selects
*	<code>\$("*")</code>	모든 요소 선택
#id	<code>\$("#lastname")</code>	id="lastname"인 요소
.class	<code>\$(".intro")</code>	class="intro"인 모든 요소
.class,.class	<code>\$(".intro,.demo")</code>	"intro" 또는 "demo" 클래스가 있는 모든 요소
element	<code>\$("p")</code>	모든 <p> 요소
el1,el2,el3	<code>\$("h1,div,p")</code>	모든 <h1>, <div> 및 <p> 요소
:first	<code>\$("p:first")</code>	첫 번째 <p> 요소
:last	<code>\$("p:last")</code>	마지막 <p> 요소
:even	<code>\$("tr:even")</code>	모든 짝수 <tr> 요소
:odd	<code>\$("tr:odd")</code>	모든 홀수 <tr> 요소

:first-child	\$("p:first-child")	부모의 첫 번째 자식인 모든 <p> 요소
:first-of-type	\$("p:first-of-type")	부모의 첫 번째 <p> 요소인 모든 <p> 요소
:last-child	\$("p:last-child")	부모의 마지막 자식인 모든 <p> 요소
:last-of-type	\$("p:last-of-type")	부모의 마지막 <p> 요소인 모든 <p> 요소
:nth-child(n)	\$("p:nth-child(2)")	부모의 두 번째 자식인 모든 <p> 요소
:nth-last-child(n)	\$("p:nth-last-child(2)")	마지막 자식부터 세어 부모의 두 번째 자식인 모든 <p> 요소
:nth-of-type(n)	\$("p:nth-of-type(2)")	부모의 두 번째 <p> 요소인 모든 <p> 요소
:nth-last-of-type(n)	\$("p:nth-last-of-type(2)")	마지막 자식부터 세어 부모의 두 번째 <p> 요소인 모든 <p> 요소
:only-child	\$("p:only-child")	부모의 유일한 자식인 모든 <p> 요소
:only-of-type	\$("p:only-of-type")	해당 유형의 부모의 유일한 자식인 모든 <p> 요소
parent > child	\$("div > p")	<div> 요소의 직계 자식인 모든 <p> 요소
parent descendant	\$("div p")	<div> 요소의 자손인 모든 <p> 요소
element + next	\$("div + p")	각 <div> 요소 옆에 있는 <p> 요소
element ~ siblings	\$("div ~ p")	<div> 요소 뒤에 나타나는 모든 <p> 요소
:eq(index)	\$("ul li:eq(3)")	목록의 네 번째 요소(인덱스는 0부터 시작)
:gt(no)	\$("ul li:gt(3)")	인덱스가 3보다 큰 요소 나열
:lt(no)	\$("ul li:lt(3)")	인덱스가 3 미만인 요소 나열
:not(selector)	\$("input:not(:empty)")	비어 있지 않은 모든 입력 요소
:header	\$(":header")	모든 헤더 요소 <h1>, <h2> ...
:animated	\$(":animated")	모든 애니메이션 요소
:focus	\$(":focus")	현재 포커스가 있는 요소
:contains(text)	\$(":contains('Hello'))	"Hello" 텍스트를 포함하는 모든 요소

:has(selector)	\$("div:has(p)")	<p> 요소가 있는 모든 <div> 요소
:empty	\$(":empty")	비어 있는 모든 요소
:parent	\$(":parent")	다른 요소의 부모인 모든 요소
:hidden	\$("p:hidden")	숨겨진 모든 <p> 요소
:visible	\$("table:visible")	보이는 모든 테이블
:root	\$(":root")	문서의 루트 요소
:lang(language)	\$("p:lang(de)")	lang 속성 값이 "de"로 시작하는 모든 <p> 요소
[attribute]	\$("[href]")	href 속성이 있는 모든 요소
[attribute=value]	\$("[href='default.htm']")	href 속성 값이 "default.htm"인 모든 요소
[attribute!=value]	\$("[href!='default.htm']")	href 속성 값이 "default.htm"이 아닌 모든 요소
[attribute\$=value]	\$("[href\$='.jpg']")	href 속성 값이 ".jpg"로 끝나는 모든 요소
[attribute =value]	\$("[title='Tomorrow']")	title 속성 값이 'Tomorrow'이거나 하이픈이 오는 'Tomorrow'로 시작하는 모든 요소
[attribute^=value]	\$("[title^='Tom']")	제목 속성 값이 "Tom"으로 시작하는 모든 요소
[attribute~=value]	\$("[title~='hello']")	특정 단어 "hello"를 포함하는 제목 속성 값이 있는 모든 요소
[attribute*=value]	\$("[title*='hello']")	"hello"라는 단어를 포함하는 제목 속성 값이 있는 모든 요소
:input	\$(":input")	모든 입력 요소
:text	\$(":text")	type="text"인 모든 입력 요소
:password	\$(":password")	type="password"인 모든 입력 요소
:radio	\$(":radio")	type="radio"인 모든 입력 요소
:checkbox	\$(":checkbox")	type="checkbox"인 모든 입력 요소
:submit	\$(":submit")	type="submit"인 모든 입력 요소

:reset	\$(":reset")	type="reset"인 모든 입력 요소
:button	\$(":button")	type="button"인 모든 입력 요소
:image	\$(":image")	type="image"인 모든 입력 요소
:file	\$(":file")	type="file"인 모든 입력 요소
:enabled	\$(":enabled")	활성화된 모든 입력 요소
:disabled	\$(":disabled")	비활성화된 모든 입력 요소
:selected	\$(":selected")	선택한 모든 입력 요소
:checked	\$(":checked")	모든 체크된 입력 요소

## jQuery Event Methods

웹 페이지가 응답할 수 있는 모든 방문자의 행동을 이벤트라고 합니다.

이벤트는 어떤 일이 발생하는 정확한 순간을 나타냅니다.

예)

요소 위로 마우스 이동

라디오 버튼 선택

요소 클릭

Mouse Events	Keyboard Events	Form Events	Document/ Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload



Method / Property	Description
bind()	버전 3.0에서 더 이상 사용되지 않습니다. 대신 on() 메서드를 사용하세요.
blur()	양식 필드가 포커스를 잃으면 함수가 실행됩니다.
change()	변경 이벤트를 연결/트리거합니다.
click()	클릭 이벤트 연결/트리거
dblclick()	더블 클릭 이벤트 연결/트리거
event.pageX	문서의 왼쪽 가장자리를 기준으로 마우스 위치를 반환합니다.
event.pageY	문서의 위쪽 가장자리를 기준으로 마우스 위치를 반환합니다.
event.preventDefault()	이벤트의 기본 동작 방지
focus()	포커스 이벤트를 연결/트리거합니다.
focusin()	focusin 이벤트에 이벤트 핸들러를 연결합니다.
focusout()	focusout 이벤트에 이벤트 핸들러를 연결합니다.
hover()	호버 이벤트에 두 개의 이벤트 핸들러를 연결합니다.
keydown()	keydown 이벤트를 연결/트리거합니다.
keypress()	키 누르기 이벤트를 연결/트리거합니다.
keyup()	keyup 이벤트를 연결/트리거합니다.
mousedown()	mousedown 이벤트를 연결/트리거합니다
mouseenter()	mouseenter 이벤트를 연결/트리거합니다.
mouseleave()	mouseleave 이벤트를 연결/트리거합니다.
mousemove()	mousemove 이벤트를 연결/트리거합니다.
mouseout()	mouseout 이벤트를 연결/트리거합니다.
mouseover()	mouseover 이벤트를 연결/트리거합니다.
mouseup()	mouseup 이벤트를 연결/트리거합니다.
off()	on() 메서드로 연결된 이벤트 핸들러를 제거합니다.
on()	요소에 이벤트 처리기를 연결합니다.

one()	선택한 요소에 하나 이상의 이벤트 핸들러를 추가합니다. 이 핸들러는 요소당 한 번만 트리거될 수 있습니다.
ready()	DOM이 완전히 로드되었을 때 실행할 함수를 지정합니다.
resize()	크기 조정 이벤트를 연결/트리거합니다.
scroll()	스크롤 이벤트를 연결/트리거합니다.
select()	선택 이벤트를 연결/트리거합니다.
submit()	선택 이벤트를 연결/트리거합니다.
trigger()	선택한 요소에 바인딩된 모든 이벤트를 트리거합니다.

## jQuery Syntax For Event Methods

click() 메서드는 이벤트 핸들러 함수를 HTML 요소에 연결합니다.

사용자가 HTML 요소를 클릭하면 함수가 실행됩니다.

```
$( "p" ).click(function(){
    // action goes here!!
});
```

dblclick() 메서드는 사용자가 HTML 요소를 두 번 클릭하면 함수가 실행됩니다.

```
$( "p" ).dblclick(function(){
    $(this).hide();
});
```

mouseenter() 메서드는 이벤트 핸들러 함수를 HTML 요소에 연결합니다.

마우스 포인터가 HTML 요소에 들어가면 함수가 실행됩니다.

```
$( "#p1" ).mouseenter(function(){
    alert("You entered p1!");
});
```

mouseleave() 메서드는 이벤트 핸들러 함수를 HTML 요소에 연결합니다.

마우스 포인터가 HTML 요소를 벗어나면 함수가 실행됩니다.

```
$( "#p1" ).mouseleave(function(){
    alert("Bye! You now leave p1!");
});
```

mousedown() 메서드는 이벤트 핸들러 함수를 HTML 요소에 연결합니다.

마우스가 HTML 요소 위에 있는 동안 왼쪽, 가운데 또는 오른쪽 마우스 버튼을 누르면 함수

가 실행됩니다.

```
$("#p1").mousedown(function(){  
    alert("Mouse down over p1!");  
});
```

mouseup() 메서드는 이벤트 핸들러 함수를 HTML 요소에 연결합니다.

마우스가 HTML 요소 위에 있는 동안 왼쪽, 가운데 또는 오른쪽 마우스 버튼을 놓으면 함수가 실행됩니다.

```
$("#p1").mouseup(function(){  
    alert("Mouse up over p1!");  
});
```

hover() 메서드는 두 가지 함수를 사용하며 mouseenter() 및 mouseleave() 메서드의 조합입니다.

첫 번째 함수는 마우스가 HTML 요소에 들어갈 때 실행되고 두 번째 함수는 마우스가 HTML 요소를 떠날 때 실행됩니다.

```
$("#p1").hover(function(){  
    alert("You entered p1!");  
},  
function(){  
    alert("Bye! You now leave p1!");  
});
```

focus() 메서드는 이벤트 핸들러 함수를 HTML 양식 필드에 연결합니다.

양식 필드가 포커스를 받으면 함수가 실행됩니다.

```
$("#input").focus(function(){  
    $(this).css("background-color", "#cccccc");  
});
```

blur() 메서드는 이벤트 핸들러 함수를 HTML 양식 필드에 연결합니다.

양식 필드가 포커스를 잃으면 함수가 실행됩니다.

```
$("#input").blur(function(){  
    $(this).css("background-color", "#ffffff");  
});
```

on() 메서드는 선택한 요소에 대해 하나 이상의 이벤트 핸들러를 연결합니다.

```
$("#p").on("click", function(){  
    $(this).hide();  
});
```

```
$("#p").on({
```

```

mouseenter: function(){
    $(this).css("background-color", "lightgray");
},
mouseleave: function(){
    $(this).css("background-color", "lightblue");
},
click: function(){
    $(this).css("background-color", "yellow");
}
});

```

## jQuery Effects

### Hide and Show

jQuery를 사용하면 hide() 및 show() 메서드를 사용하여 HTML 요소를 숨기거나 표시할 수 있습니다.

Syntax:

```
$(selector).hide(speed,callback);
```

```
$(selector).show(speed,callback);
```

speed parameter : "slow", "fast", or milliseconds.

callback parameter : hide() 또는 show() 메서드가 완료된 후 실행되는 함수입니다

ex)

```

$("#hide").click(function(){
    $("p").hide();
});

```

```

$("#show").click(function(){
    $("p").show();
});

```

```
<p>If you click on the "Hide" button, I will disappear.</p>
```

```
<button id="hide">Hide</button>
```

```
<button id="show">Show</button>
```

### jQuery toggle()

toggle() 메서드를 사용하여 요소 숨기기와 표시 사이를 전환할 수 있습니다.

표시된 요소는 숨겨지고 숨겨진 요소는 표시됩니다.

Syntax:

```
$(selector).toggle(speed,callback);
```

```

ex) $("button").click(function(){
    $("p").toggle();
});

```

## jQuery Effects – Fading

jQuery를 사용하면 요소를 페이드 인/아웃할 수 있습니다.

- `fadeIn()` : 숨겨진 요소를 페이드 인하는 데 사용됩니다.
- `fadeOut()` : 보이는 요소를 페이드 아웃하는 데 사용됩니다.
- `fadeToggle()` : `fadeIn()` 및 `fadeOut()` 메서드 간에 전환합니다.
- `fadeTo()` : 지정된 불투명도(0과 1 사이의 값)로 페이드할 수 있습니다.

### jQuery `fadeIn()` Method

Syntax: `$(selector).fadeIn(speed,callback);`

```
$("#button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
});
```

### jQuery `fadeOut()` Method

Syntax: `$(selector).fadeOut(speed,callback);`

```
$("#button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
});
```

### jQuery `fadeToggle()` Method

Syntax: `$(selector).fadeToggle(speed,callback);`

```
$("#button").click(function(){
    $("#div1").fadeToggle();
    $("#div2").fadeToggle("slow");
    $("#div3").fadeToggle(3000);
});
```

### jQuery `fadeTo()` Method

Syntax: `$(selector).fadeTo(speed,opacity,callback);`

```
$("#button").click(function(){
    $("#div1").fadeTo("slow", 0.15);
    $("#div2").fadeTo("slow", 0.4);
    $("#div3").fadeTo("slow", 0.7);
});
```

## jQuery Effects – Sliding

jQuery 슬라이드 메서드는 요소를 위아래로 슬라이드합니다.

- slideDown()
- slideUp()
- slideToggle()

### jQuery slideDown() Method

Syntax: \$(selector).slideDown(speed,callback);

```
$("#flip").click(function(){
    $("#panel").slideDown();
});
```

### jQuery slideUp() Method

Syntax: \$(selector).slideUp(speed,callback);

```
$("#flip").click(function(){
    $("#panel").slideUp();
});
```

### jQuery slideToggle() Method

Syntax: \$(selector).slideToggle(speed,callback);

```
$("#flip").click(function(){
    $("#panel").slideToggle();
});
```

## jQuery Effects – Animation

사용자 지정 애니메이션을 만들 수 있습니다.

### The animate() Method

jQuery animate() 메서드는 사용자 지정 애니메이션을 만드는 데 사용됩니다.

Syntax: \$(selector).animate({params},speed,callback);

```
$("#button").click(function(){
    $("#div").animate({left: '250px'});
});

$("#button").click(function(){
    $("#div").animate({
        left: '250px',
        opacity: '0.5',
        height: '150px',
        width: '150px'
    });
});
```

```
$("#button").click(function(){
```

```

$("div").animate({
    left: '250px',
    height: '+=150px',
    width: '+=150px'
});
});

```

모든 속성 이름은 animate() 메서드와 함께 사용할 때 카멜 케이스여야 합니다. padding-left 대신 paddingLeft를 작성하고 margin-right 대신 marginRight를 작성해야 합니다.

또한 컬러 애니메이션은 핵심 jQuery 라이브러리에 포함되어 있지 않습니다.

jQuery animate() - 미리 정의된 값 사용

속성의 애니메이션 값을 "show", "hide" 또는 "toggle"로 지정할 수도 있습니다.

```

$("button").click(function(){
    $("div").animate({
        height: 'toggle'
    });
});

```

여러 animate() 호출을 차례로 작성하면 jQuery가 이러한 메서드 호출로 "내부" 대기열을 생성합니다. 그런 다음 애니메이트 호출을 ONE by ONE으로 실행합니다.

```

$("button").click(function(){
    var div = $("div");
    div.animate({height: '300px', opacity: '0.4'}, "slow");
    div.animate({width: '300px', opacity: '0.8'}, "slow");
    div.animate({height: '100px', opacity: '0.4'}, "slow");
    div.animate({width: '100px', opacity: '0.8'}, "slow");
});

```

jQuery Stop Animations

jQuery stop() 메서드는 완료되기 전에 애니메이션이나 효과를 중지하는 데 사용됩니다. 애니메이션이나 효과가 완료되기 전에 중지하는 데 사용됩니다.

stop() 메서드는 슬라이딩, 페이딩 및 사용자 지정 애니메이션을 포함한 모든 jQuery 효과 함수에 대해 작동합니다.

Syntax: \$(selector).stop(stopAll,goToEnd);

선택적 stopAll 매개변수는 애니메이션 대기열도 지워야 하는지 여부를 지정합니다. 기본값은 false입니다. 즉, 활성 애니메이션만 중지되어 대기열에 있는 모든 애니메이션이 나중에 수행될 수 있습니다.

선택적 goToEnd 매개변수는 현재 애니메이션을 즉시 완료할지 여부를 지정합니다. 기본값은 false입니다.

#### jQuery Callback Functions

콜백 함수는 현재 효과가 100% 완료된 후에 실행됩니다.

JavaScript 문은 한 줄씩 실행됩니다. 그러나 효과를 사용하면 효과가 완료되지 않은 경우에도 다음 코드 줄을 실행할 수 있습니다. 이로 인해 오류가 발생할 수 있습니다.

이를 방지하기 위해 콜백 함수를 만들 수 있습니다.

콜백 함수는 현재 효과가 완료된 후에 실행됩니다.

```
$("#button").click(function(){
    $("#p").hide("slow", function(){
        alert("The paragraph is now hidden");
    });
});
```

#### jQuery – Chaining

jQuery를 사용하면 작업/메소드를 함께 연결할 수 있습니다.

체인을 사용하면 단일 문 내에서 여러 jQuery 메서드(동일한 요소에서)를 실행할 수 있습니다. 동일한 요소에서 여러 jQuery 명령을 차례로 실행할 수 있습니다. 이렇게 하면 브라우저가 동일한 요소를 두 번 이상 찾을 필요가 없습니다.

작업을 연결하려면 이전 작업에 작업을 추가하기만 하면 됩니다.

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

연결하면 코드 줄이 상당히 길어질 수 있습니다. 그러나 jQuery는 구문이 그다지 엄격하지 않습니다. 줄 바꿈 및 들여쓰기를 포함하여 원하는 대로 형식을 지정할 수 있습니다.

```
$("#p1").css("color", "red")
    .slideUp(2000)
    .slideDown(2000);
```

jQuery는 여분의 공백을 버리고 위의 줄을 하나의 긴 코드 줄로 실행합니다.

#### jQuery HTML

jQuery - Get Content and Attributes(콘텐츠 및 속성 가져오기)

jQuery에는 HTML 요소와 속성을 변경하고 조작하기 위한 강력한 메서드가 포함되어 있습니다.

#### jQuery DOM 조작

jQuery의 매우 중요한 부분 중 하나는 DOM을 조작할 수 있는 가능성입니다.

jQuery에는 요소와 속성에 쉽게 액세스하고 조작할 수 있는 DOM 관련 메서드가 함께 제공됩니다.



DOM = Document Object Model

DOM은 HTML 및 XML 문서에 액세스하기 위한 표준을 정의합니다.

W3C DOM(Document Object Model)은 프로그램과 스크립트가 문서의 내용, 구조 및 스타일을 동적으로 액세스하고 업데이트할 수 있도록 하는 플랫폼이자 언어 중립적인 인터페이스입니다.

콘텐츠 가져오기 - text(), html() 및 val()

DOM 조작을 위한 간단하지만 유용한 세 가지 jQuery 메서드입니다.

text() - 선택한 요소의 텍스트 내용을 설정하거나 반환합니다.

html() - 선택한 요소의 내용을 설정하거나 반환합니다(HTML 마크업 포함).

val() - 양식 필드의 값을 설정하거나 반환합니다.

```
$("#btn1").click(function(){
    alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
});
$("#btn1").click(function(){
    alert("Value: " + $("#test").val());
});
```

Get Attributes - attr()

jQuery attr() 메서드는 속성 값을 가져오는 데 사용됩니다.

```
$("#button").click(function(){
    alert($("#w3s").attr("href"));
});
```

jQuery - Add Elements

jQuery를 사용하면 새로운 요소/콘텐츠를 쉽게 추가할 수 있습니다.

append() - 선택한 요소의 끝에 내용을 삽입합니다.

prepend() - 선택한 요소의 시작 부분에 콘텐츠를 삽입합니다.

after() - 선택한 요소 뒤에 콘텐츠를 삽입합니다.

before() - 선택한 요소 뒤에 콘텐츠를 삽입합니다.

```
$("#p").append("Some appended text.");
$("#p").prepend("Some prepended text.");
```

<script>

```
function appendText() {
    var txt1 = "<p>Text.</p>";
    var txt2 = $("<p></p>").text("Text.");
```

```

var txt3 = document.createElement("p");
txt3.innerHTML = "Text.";
$("body").append(txt1, txt2, txt3);
}
</script>
<button onclick="appendText()">Append text</button>

```

```

$("img").after("Some text after");
$("img").before("Some text before");

```

#### jQuery - Remove Elements

요소와 콘텐츠를 제거하기 위한 jQuery 메서드

remove() - 선택한 요소(및 하위 요소)를 제거합니다.

empty() - 선택한 요소에서 하위 요소를 제거합니다.

```

$("#div1").remove();

```

```

$("#div1").empty();

```

jQuery remove() 메서드는 제거할 요소를 필터링할 수 있는 하나의 매개 변수도 허용합니다. 매개 변수는 jQuery 선택기 구문 중 하나일 수 있습니다.

다음 예제에서는 class="test"인 모든 <p> 요소를 제거합니다.

```

$("p").remove(".test");

```

class="test" 및 class="demo"인 모든 <p> 요소를 제거합니다.

```

$("p").remove(".test, .demo");

```

#### jQuery - Get and Set CSS Classes

jQuery를 사용하면 요소의 스타일을 쉽게 조작할 수 있습니다.

addClass() - 선택한 요소에 하나 이상의 클래스를 추가합니다.

removeClass() - 선택한 요소에서 하나 이상의 클래스를 제거합니다.

toggleClass() - 선택한 요소에서 클래스 추가/제거 간을 전환합니다.

css() - 스타일 속성을 설정하거나 반환합니다.

```

$("button").click(function(){
    $("h1, h2, p").addClass("blue");
    $("div").addClass("important");
});

```

```

$("button").click(function(){
    $("#div1").addClass("important blue");
});

```

```

$("button").click(function(){
    $("h1, h2, p").removeClass("blue");
});

```

```

$("button").click(function(){
    $("h1, h2, p").toggleClass("blue");
});

```

css() 메서드는 선택한 요소에 대해 하나 이상의 스타일 속성을 설정하거나 반환합니다.

```
css("propertyname","value");
```

```
$("p").css("background-color", "yellow");
```

Set Multiple CSS Properties

```
css({"propertyname":"value","propertyname":"value",...});
```

```
$("p").css({"background-color": "yellow", "font-size": "200%"});
```

## jQuery – Dimensions

jQuery를 사용하면 요소 및 브라우저 창의 크기로 작업하기가 쉽습니다.

- width() - 요소의 너비를 설정하거나 반환합니다(패딩, 테두리 및 여백 제외)
- height() - 요소의 높이를 설정하거나 반환합니다(패딩, 테두리 및 여백 제외)
- innerWidth() - 요소의 너비를 반환합니다(패딩 포함)
- innerHeight() - 요소의 높이를 반환합니다(패딩 포함)
- outerWidth() - 요소의 너비를 반환합니다(패딩 및 테두리 포함)
- outerHeight() - 요소의 높이를 반환합니다(패딩 및 테두리 포함)



## jQuery on() Method

on() 메서드는 선택한 요소 및 자식 요소에 대해 하나 이상의 이벤트 핸들러를 연결합니다. 이벤트 핸들러를 제거하려면 off() 메서드를 사용하십시오.

한 번만 실행된 후 자체적으로 제거되는 이벤트를 첨부하려면 one() 메소드를 사용하십시오.

### Syntax

`$(selector).on(event,childSelector,data,function,map)`

| Parameter     | Description  |
|---------------|--|
| event         | Required. 선택한 요소에 연결할 하나 이상의 이벤트 또는 네임스페이스를 지정합니다.<br>여러 이벤트 값은 공백으로 구분됩니다. 유효한 이벤트여야 합니다.             |
| childSelector | Optional. 이벤트 핸들러가 지정된 자식 요소에만 연결되어야 함을 지정합니다  |
| data          | Optional. 함수에 전달할 추가 데이터를 지정합니다.   |
| function      | Required. 이벤트가 발생할 때 실행할 함수를 지정합니다.  |
| map           | 선택한 요소에 첨부할 하나 이상의 이벤트를 포함하는 이벤트 맵 ({event:function, event:function, ...}) 및 이벤트가 발생할 때 실행할 함수를 지정합니다. |

### 여러 이벤트 첨부

요소에 여러 이벤트를 연결하는 방법.

```
$( "p" ).on( "mouseover mouseout", function() {  
    $( this ).toggleClass( "intro" );  
});
```

### map 매개변수를 사용하여 여러 이벤트 핸들러 연결

맵 매개변수를 사용하여 선택한 요소에 여러 이벤트 핸들러를 연결하는 방법입니다.

```
$( "p" ).on( {  
    mouseover: function() {  
        $( "body" ).css( "background-color", "lightgray" );  
    },  
    mouseout: function() {  
        $( "body" ).css( "background-color", "lightblue" );  
    },  
});
```

```

click: function(){
    $("body").css("background-color", "yellow");
}
});

```

#### 이벤트 핸들러 제거

off() 메서드를 사용하여 이벤트 핸들러를 제거하는 방법.

```

$("p").on("click", function(){
    $(this).css("background-color", "pink");
});
$("button").click(function(){
    $("p").off("click");
});

```

#### jQuery each() Method

each() 메서드는 일치하는 각 요소에 대해 실행할 함수를 지정합니다.

Syntax

\$(selector).each(function(index,element))

| Parameter               | Description   |
|-------------------------|---|
| function(index,element) | Required. 일치하는 각 요소에 대해 실행할 함수입니다.<br>index – selector의 인덱스 위치<br>element - 현재 요소("this" 선택자도 사용할 수 있음) |

```

$("button").click(function(){
    $("li").each(function(){
        alert($(this).text())
    });
});

```

#### jQuery data() Method

태그에는 지정 가능한 속성값들이 존재합니다.

예를 들어 a 태그에 href라는 속성을 지정하고 그 값으로 원하는 링크를 설정하는 것이 가능합니다. 그 외에 css 또는 스크립트를 사용하기 위해서는 id 또는 class를 사용할 수도 있습니다. 하지만 사용자의 필요에 의하여 설정을 하기도 하는데 이를 커스텀 속성이라 부를 수 있습니다. 이런 속성은 사용자의 필요에 의하여 별도로 정의하는데 제이쿼리의 내장된 속성과의 구분이 되지 않으면 혼란을 줄 수 있어 주의해야 합니다. 이런 이유로 data를 속성 앞에 접두어로 사용하는 것을 권고하며 이런 방식으로 설정하기 위해서 jQuery에는 data() 메소드가 존재합니다.

\$(선택요소).data(데이터속성에 사용된 이름)

```
<a href="#" data-no="523">Link to</a>
```

```
<script>
```

```
  var no = $("a").data("no");
```

```
  console.log(no);
```

```
</script>
```

위 예제는 data() 메소드에 인자로 no를 사용하여 data-no 속성에 사용된 값을 읽어오는 코드입니다. 이처럼 별도로 속성을 읽기 위해서 사용해야하는 attr() 또는 prop() 등의 메소드를 사용할 필요가 없으므로 더 편리합니다.