|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \*Dom [오른쪽 W3C 표준객체모델]  문서 객체 모델 ‘Dom’이란  문서 객체 모델(DOM, Document Object Model)은 XML이나 HTML 문서에 접근하기 위한 일종의 인터페이스로 이 객체 모델은 문서 내의 모든 요소를 정의하고, 각각의 요소에 접근하는 방법을 제공.  \*\*자바스크립트는  - 새로운 HTML 요소나 속성을 추가 가능.  - 존재하는 HTML 요소나 속성을 제거 가능  - HTML 문서의 모든 HTML 요소를 변경 가능  - HTML 문서의 모든 HTML 속성을 변경 가능  - HTML 문서에 새로운 HTML 이벤트를 추가 가능  - HTML 문서의 모든 HTML 이벤트에 반응  DOM의 종류 [W3C DOM 표준 세 가지 모델]  1. Core DOM : 모든 문서 타입을 위한 DOM 모델  2. HTML DOM : HTML 문서를 위한 DOM 모델  3. XML DOM : XML 문서를 위한 DOM 모델  **HTML DOM**  HTML DOM은 HTML 문서를 조작하고 접근하는 표준화된 방법을 정의합니다.  모든 HTML 요소는 HTML DOM를 통해 접근할 수 있습니다.  **XML DOM**  XML DOM은 XML 문서에 접근하여, 그 문서를 다루는 표준화된 방법을 정의합니다.  모든 XML 요소는 XML DOM를 통해 접근할 수 있습니다.  XML DOM에 대한 더 자세한 사항은 XML DOM 수업에서 확인할 수 있습니다. | | | | img_js_htmldom.png |
| script는 태그와 관련한 작업을 할 때 HTML태그와 CSS가 모두 로딩 된 후 시작되어야 script 소스가 인지가 됨.  **window.onload = function(){코드}** | | | | 윈도우 로드 후 기능은 코드 실행. |
| Jquery  **$(document).ready(function(){코드});**  외부 리소스. 이미지와 상관없이 브라우저가 Dom[document object model] 트리를 생성한 직후 실행.  **$(win dow).load()**  DOM의 standard 이벤트 / html의 로딩이 끝난 후에 시작 / 화면에 필요한 모든 요소[css/js/iframe 등등]이 웹 브라우저 메모리에 다 올려진 다음에 실행되어서 화면이 모두 나온 뒤에 실행되는 메시지나 애니메이션에 적합함.  Window.load와 같이 쓰이면 둘 중 하나만 적용.  단점.  전체페이지의 모든 외부 리소스가 브라우저에 다 불려오지 못했을 때는 더 기다려야 함  Body onload 이벤트와 같이 쓰이면 window.load()가 실행이 안되는 현상이 발생 | | | |  |
| **Script** | **Jquery** | | | **description** |
| document.getElementsByClassName() | $('.ClassName') | | | 클래스 선택 |
| document.getElementsByClassName(‘클래스명’)[순서]  document.getElementsByClassName(‘abc’)[0] | | | | 복수인 getElements 사용,  여러개 이기 때문에 몇번째에 있는 것인가를 선택 |
| document.getElementsByTagName() | $('TagName') | | | 태그요소 선택 |
| document.getElementsById | $('Id') | | | 아이디 선택 |
| document.querySelector() |  | | | CSS 선택자로 요소를 선택,  주의할 점  선택자에 해당하는 첫번째 요소만 선택 |
| document.querySelectorAll() |  | | | 특정 CSS 선택자를 가진 모든 요소를 배열로 가져오는 메서드 |
| <body>  <p class="abc">Lorem Ipsum Dolor</p>  <p class="abc">Lorem Ipsum Dolor</p>  <p class="abc">Lorem Ipsum Dolor</p>  </body | 1) document.querySeletor(‘abc’) 예제  <script>  document.querySeletor(‘abc’).style.color = ‘’red’  ;  </scirpt>  2) document.querySeletorAll() 예제  <script>  var ex = document.querySelector(‘.abc’);  ex[1].style.color =’red’  </script> | | | 1) ‘클래스 abc’를 가진 태그의 첫번째 요소만 선택하여 스타일 지정  2) ‘클래스 abc’를 가진 모든 태그요소 선택.  선언해 준 변수 “ex”의 배열[1]= 두 번째의 스타일.컬러는 빨간색이라고 정해 줌. |
| classList 들 add, remove, item, toggle, contains, replace …  **element 변수 설정**  **const element = document.getElementById(‘element’)** | **속성 메서드** | | |  |
| 클래스 하나 추가  element.classList.add(‘class name’)  여러 개 추가  element.classList.add(‘class name’, ‘class name2’) | addClass() | | | 클래스 추가 |
| 클래스 하나 삭제  element.classList.remove(‘class name’)  여러 개 삭제  element.classList. remove (‘class name’, ‘class name2’) | removeClass() | | | 클래스 삭제 |
| 클래스 교체  element.classList.replace(‘oldClass’, ‘newClass’);  클래스 토글  element.classList.toggle(‘Class’);  element.classList.toggle(‘Class’, ‘true’);  element.classList.toggle(‘Class’, ‘false); | toggleClass() | | | 스위치 역할: 클래스 추가 ,클래스 삭제  $(‘element’).toggleClass();  True 클래스 추가/ flase 클래스 삭제  $(‘element’).toggle(‘newclass’, true);  $(‘element’).toggle(‘newclass’, flase); |
| switchClass() | | | 기존 클래스 제거 후 새로운 클래스 추가; 클래스 교체  $(‘element’). switchClass(‘oldclass’, ‘newclass’);  =$(‘element’). removeClass(‘oldclass’).addClass(‘newclass’); |
| document.querySelector('선택자').classList.contains('클래스') | hasClass() | | | 클래스 이름이 있으면 true, false를 반환 |
| |  | | --- | | element.setAttribute('attribute\_name', 'attribute\_value'); |   document.getElementById('link' ).setAttribute('href','https://redmine.acego.net'); | attr() | | | 속성과 관련된 모든 기능을 수행  $(‘img’).attr(‘src’);  $(‘img’).attr(‘src’,속성값)  새로운 클래스 지정  $(‘element’).attr(‘class’, ‘newclass’); |
| element.value;  document.getElementById(‘선택요소’).value; | val() | | | val()은 양식(form)의 값을 가져오거나 값을 설정하는 메소드  [INPUT 태그에 정의된 value속성의 값을 확인하고자 할때 사용되는 함수] |
| element.innerHTML;  element.innerHTML = "<div style='color:red'>A</div>" | html() | | | 셀렉터 태그 내에 태그 추가 가능 |
| element.innerText;  element.innerText = "<div style='color:red'>A</div>"; | text() | | | 셀렉터 태그 내에 존재하는 자식태그들 중에 html태그는 모두 제외 한 채 문자열만 출력하고자 할 때 사용되는 함수[html태그까지 모두 문자로 인식시켜주는 함수]  $("셀렉터").text() |
|  | | 특정 태그내에 존재하는 태그가 문자열만 존재할경우, html() 함수나 text() 함수 아무거나 사용해도 무관 | | |
|  | **필터링 메서드** | |  | |
|  | frst() | | 선택된 요소들 중에 첫 번째 요소 | |
|  | last() | | 선택된 요소들 중에 마지막 요소 | |
| find()  findIndex() | eq() | | 인덱스 요소  $(‘li’),eq(0) : li의 첫 번째 | |
|  | filter() | | 조건에 일치하지 않는 요소 선택에서 제외, 일치하는 항목 반환 | |
|  |  | | $( "li" ).filter( ":nth-child(2n)" ).css( "background-color", "red" );  : li의 두 번째만 스타일 배경색 빨강  $( "li" ).filter( ".object”).css( "background-color", "red" );  :li의 .object 선택자를 가진 요소만  $( "li" )  .filter(**function**( index ) {  **return** $( "strong", this ).length === 1;  })  .css( "background-color", "red" );  : li의 두 번째 strong 태그만 값 반환  $( "li" )  .filter(**function**( index ) {  **return** index % 3 === 2;  })  .css( "background-color", "red" );  나머지가 2인 li 에 스타일 적용  Index = 4 >> 5 번째 li 요소 | |
|  | not() | | 조건에 일치하지 않는 모든 요소 반환 | |
|  |  | |  | |
|  | **형제,부모요소 메서드** | |  | |
|  | parent() | | 선택된 요소의 바로 위 부모 | |
|  | parents() | | 선택된 요소의 모든 조상요소 | |
|  | parentsUntil() | |  | |
|  | children() | | 선택된 요소의 자손 | |
|  | find() | | 선택된 요소의 모든 자손 요소들 | |
|  | siblings() | | 모든 형제 요소들 | |
|  | next() | | 바로 다음 형제 요소 1개 | |
|  | nextAll() | | 모든 다음 형제 요소들 | |
|  | nextUntil() | |  | |
|  | prev() | | 바로 이전 형제 요소 1개 | |
|  | prevAll() | | 바로 이전 모든 형제 요소들 | |
|  | prevUntil() | |  | |
|  | **마우스 이벤트 메서드** | |  | |
|  | click | |  | |
|  | dbclick | |  | |
|  | **mouseenter** | | 마우스 요소의 경계 외부에서 내부로 이동 할 때 발생  [자기자신만 이벤트 자식영역 노 이벤트] | |
|  | **mouseleave** | | 마우스 요소의 경계 내부에서 외부로 이동 할 때 발생  [자기자신만 이벤트 자식영역 노 이벤트] | |
|  | mouseover | | 마우스가 요소를 벗어날 때 발생.[자식영역까지 이벤트 발생] | |
|  | mouseout | | 마우스가 요소에 진입할 때 발생.[자식영역까지 이벤트 발생] | |
|  | mousemove | | 마우스가 움직일 때 이벤트 발생 | |
|  | mouseup | | 마우스 버튼을 뗄 때 | |
|  | mousedown | | 마우스 버튼을 누를 때 발생 | |
|  | hover | | mouseenter 와 mouseleave를 동시에 연결해주는 메서드 | |
|  | **키보드 이벤트** | |  | |
|  | keypress | | 글자가 입력될 때 | |
|  | keydown | | 키보드가 눌러질 때 | |
|  | keyup | | 키보드가 떼어질 때 | |
|  | **폼 이벤트** | |  | |
|  | submit | | Submit 버튼을 누르면 발생 | |
|  | select | | 입력 양식을 선택할 때 발생  [input[type=text]& input[type=textarea] 태그 제외] | |
|  | change | | 입력 양식의 내용을 변경할 때 발생함 | |
|  | focus | | Form 태그에 포커스가 되었을 때 실행 | |
|  | focusin | | 입력 양식에 초점이 맞춰지기 바로 전에 발생 | |
|  | focusout | | 입력 양식에 초점이 사라지기 바로 전에 발생. | |
|  | blur | | Form 태그에 포커스가 해제되었을 때 발생 | |
|  | **이펙트** | |  | |
|  | show() | | 보여 줌 | |
|  | hide() | | 사라지게 함. | |
|  | toggle() | | show() , hide() 번갈아 실행 | |
|  | slideDown() | | 슬라이드 효과와 함께 보여 줌 | |
|  | slideUp() | | 슬라이드 효과와 함께 사라지게 함 | |
|  | slideToggle() | | slideDown() , slideUp() 메서드 번갈아 실행 | |
|  | fadeIn() | | 선명하게 하며 보여 줌 | |
|  | fadeOut() | | 흐릿하게 하며 사라지게 함 | |
|  | animate() | | 사용자 지정 효과를 생성 함.  [opacity / height / width / top / bottom / left / right / margin / padding]  $(‘element’).animate(‘opacity’);  $(‘선택요소’).animate({‘속성’,’속성값’},속성실행 시간);  = $(‘element’’).animate({‘opacity’,’1’},500); | |
|  | animated() | |  | |
|  | stop() | | $(‘element’).stop();  $(‘element’).stop(clearQueue, finish);  clearQueue, finish는 true나 false의 값을 입력할 수 있다.(기본값 false) clearQueue가 true면 큐에서 대기 중인 애니메이션 모두 제거함. finish가 true면 진행 중인 애니메이션 강제 종료.  참고 자료 : https://simfairy.tistory.com/112 | |
|  | delay() | |  | |
|  | queue() | | queue()메서드는 큐에 적용된 애니메이션 함수를 반환하거나 큐에 지정한 함수를 추가함 | |
|  | dequeue() | | queue()메서드를 실행하면 실행 이후의 모든 애니메이션 취소,  dequeue() 메서드는 queue()메서드 실행 이후에 적용된 애니메이션 메서드가 취소되지 않게 연결해 줌  1. 큐(Queue)의 함수 반환  $("요소 선택").queue();  2. 큐(Queue)에 함수 추가  $("요소 선택").queue( function() {자바스크립트 코드} );  3. dequeue() 메서드  $("요소 선택").dequeue(); | |
|  | clearQueue() | | clearQueue() 메서드는 진행 중인(첫 번째) 애니메이션을 제외하고, 큐에서 대기하는 모든 애니메이션 함수를 제거 함. | |
|  |  | |  | |