DOM

|  |  |
| --- | --- |
| 1 | 다음 설명 틀린 것은?    (1)HTML과 Javascript는 모두 프로그래밍 언어이다    (2)HTML은 프로그래밍 언어가 아니므로, Javascript와 같은 응용프로그램들이 직접 제어할 수 없다.    (3)(2)번과 같은 이유로 w3c 에서는 DOM API을 제안하였다    (4)DOM이란 (Document Object Mode) 의 약자로서, 문서내의 모든 태그를 1:1 대응하는 객체로 모델화 시켜 놓은 기술을 일컫는다  (5)사실상 Javascript와 같은 응용프로그램들은 문서를 직접제어하는 것이 아니라, DOM을 통해 간접적으로 제어할 수 있다. |
| 2 | 다음 설명 틀린 것은?  (1)DOM의 여러 객체 중 <HTML> 태그에 대응하는 객체는 document 객체이다  (2)DOM은 w3c에서 지정한 표준이므로, DOM에 대한 사용법을 알기 위해서는 설명서(API Document)를 참조하면 된다  (3)HTML 태그가 다른 태그를 포함할때 부모, 자식 관계가 성립하듯 DOM간에도 부모자식 관계가 그대로 적용된다.  (4)위(3)번의 논리대로 라면, <body> 태그는 <html> 태그보다 자식요소 이므로 document.body 라는 접근 방식이 허용될 것이다.  (5)메모리에 올라온 수많은 DOM 객체를 접근하는 방법 중 하나는, 태그들간 구분을 위해 사용했던 id를 이용하는 방법이다.  (6)document.getElementById() 메서드는 document 하위에 생성된 자식 DOM 들 중 하나를 id 를 통해 접근할 수 있는 메서드이다. |
| 3 | 다음 설명 중 틀린 것은?  (1)Javascript 가 DOM을 사용하려면 적어도 tag에 대응하는 DOM이 메모리에 생성된 이후에나 가능하다  (2)위(1)번의 이유 때문에 개발자는 언제나 문서가 모두 읽혀진 시점 , 즉 문서의 종료 태그인 닫는 </html> 뒤에 <script>태그를 추가해야  한다.  (3)문서의 모든 태그가 모두 DOM화 되어 메모리에 올라온 시점을 알려주는 이벤트가 바로 load 이벤트이며 <body>태그에 onLoad 이벤트  핸들러를 적용하면 된다  (4)문서에 태그를 기재하여 생성하는 방식이 아니라, 자바스크립트에 코드에서 DOM을 생성할 수 있는 방법도 제공하는데, 이때 사용하는  메서드가 바로 document.createElement() 이다.  (5)document.createElement() 에 의해 생성된 DOM 객체는 문서에 tag를 명시하여 생성한 객체와는 틀리기 때문에 id 나 스타일을 적용 할  수 없다. |