2주 차 과제

1. **HTML의 form과 action의 개념**

* form: input 요소에 사용자가 입력한 정보를 서버로 넘기면서 요청하는 역할을 수행하는 태그.
* action: 폼 내에 입력한 값들을 전달할 서버에 대한 경로를 지정한다. 이 경로는 서버에서 폼 데이터를 처리하는 스크립트나 프로그램의 URL이다.

- method: 폼 데이터를 서버로 전송할 때 사용할 HTTP 메서드를 지정한다. Get이 기본값이며, get/post로 나뉜다.

1) GET: 요청 시 사용자가 입력한 값들이 url에 노출된다. 보안이 취약하지만, 북마크가 가능하다. 데이터가 URL의 쿼리 스트링으로 전송된다.

2) POST: 요청 시 사용자가 입력한 값들이 url에 노출되지 않는다. 보안에 더 안전하고, 큰 데이터를 전송할 수 있다. 데이터가 HTTP 요청 본문으로 전송된다.

- parameter (name, value): 폼이 제출되면, 각 입력 요소의 name과 value 쌍이 서버로 전송된다. name 속성은 입력 요소를 식별하는 이름을 지정하고, value 속성은 사용자가 입력한 값 또는 요소의 기본값을 나타낸다.

**2. Javascript**

- DOM Manipulation

DOM(Document Object Model)은 HTML, XML 등의 문서를 프로그래밍 언어에서 조작할 수 있도록 객체 형태로 표현한 것이다. 웹 페이지에서는 DOM을 이용하여 HTML 요소를 조작하고, 이벤트 처리 등을 할 수 있다.

- EventListener(onClick과 비교)

이벤트에 대한 처리를 위해 이벤트 핸들러를 등록해야 한다. 이벤트 핸들러는 해당 이벤트가 발생했을 때 실행될 함수를 말한다.

1) EventListener: ‘addEventListener’ 메서드를 사용하여 요소에 이벤트 핸들러를 추가한다. 같은 요소에 여러 이벤트 핸들러를 등록할 수 있다.

2) onClick: 요소의 ‘onclick’ 속성을 사용하여 이벤트 핸들러를 추가한다. 하나의 이벤트 핸들러만 등록할 수 있다.

- Event loop: 비동기 처리를 가능하게 하는 메커니즘으로, 호출 스택과 이벤트 큐를 사용하여 비동기 콜백함수를 처리한다. (여러 이벤트가 동시에 발생했을 때 어떤 순서로 콜백함수를 호출할 지 판단)

- getelementByID: 특정 ID를 가진 요소를 선택한다.

- getelementsByClassName: 특정 클래스를 가진 모든 요소를 선택한다. HTMLCollection 객체를 반환한다.

- queryselector vs queryselectorAll

1) queryselector: 주어진 CSS 선택자와 일치하는 첫 번째 요소를 반환한다.

2) queryselectorAll: 주어진 CSS 선택자와 일치하는 모든 요소를 반환한다.

- Node의 속성

- Node는 DOM의 기본 단위로, 다양한 속성과 메서드를 가진다.

- 종류: 문서 노드 / 요소 노드 / 속성 노드 / 텍스트 노드 / 주석 노드

- 관계: 부모 노드 / 자식 노드 리스트 / 첫 번째 자식 노드 / 마지막 자식 노드 / 다음 형제 노드 / 이전 형제 노드 /

**3. Javascript 문법**

- var와 let의 차이

1) var: 선언 이전에 변수를 사용할 수 있으며, 같은 스코프 내에서 여러 번 재선언할 수 있다.

|  |
| --- |
| var x = 1;  var x = 2; // 재선언 가능  console.log(x); // 2 |

2) let: 블록 스코프를 가지고, 변수의 재할당은 가능하나 재선언이 불가능하다. let으로 생성한 변수를 위에서 미리 참조할 수 없다.

|  |
| --- |
| let y = 1;  // let y = 2; // SyntaxError: Identifier 'y' has already been declared  y = 2; // 값 변경 가능  console.log(y); // 2 |

- const: 블록 스코프를 가지고, 변수의 재선언과 재할당이 불가능하다.

|  |
| --- |
| const z = 10;  // const z = 20; // SyntaxError: Identifier 'z' has already been declared  // z = 20; // TypeError: Assignment to constant variable |

- ES5의 Array 관련 메소드

1) forEach(): 배열의 각 요소에 대해 한 번씩 제공된 함수를 실행한다.

|  |
| --- |
| [1, 2, 3].forEach(element => {  console.log(element);  }); |

2) map(): 배열 내의 모든 요소에 대해 함수를 호출한 결과를 모아 새로운 배열을 반환한다.

|  |
| --- |
| const doubled = [1, 2, 3].map(element => element \* 2);  console.log(doubled); // [2, 4, 6] |

3) filter(): 주어진 함수의 테스트를 통과하는 모든 요소를 모아 새로운 배열을 반환한다.

|  |
| --- |
| const even = [1, 2, 3, 4].filter(element => element % 2 === 0);  console.log(even); // [2, 4] |

4) reduce(): 배열의 각 요소에 대해 제공된 reducer 함수를 실행하여 단일 출력 값을 반환한다.

|  |
| --- |
| const sum = [1, 2, 3, 4].reduce((accumulator, currentValue) => accumulator + currentValue, 0);  console.log(sum); // 10 |

- rest parameter: 함수의 인자 수를 가변적으로 설정할 수 있다. 함수의 매개변수 목록의 마지막 위치에만 사용된다.

|  |
| --- |
| function sum(...numbers) {  return numbers.reduce((acc, cur) => acc + cur, 0);  }  console.log(sum(1, 2, 3, 4)); // 10 |

- Arrow function: function 키워드 대신 화살표를 사용하여 간략한 방법으로 함수를 선언할 수 있다. 자신만의 ‘this’를 가지지 않는다.

|  |
| --- |
| const add = (a, b) => a + b;  console.log(add(2, 3)); // 5 |

- 1급 객체

1) 변수에 함수를 할당할 수 있다.

|  |
| --- |
| const greet = function(name) {  return `Hello, ${name}!`;  }; |

2) 함수의 인자로 함수를 전달할 수 있다.

|  |
| --- |
| function sayHello(greetingFunction, name) {  console.log(greetingFunction(name));  }  sayHello(greet, "Alice"); // Hello, Alice! |

3) 함수의 반환 값으로 함수를 사용할 수 있다.

|  |
| --- |
| function getGreeter() {  return function(name) {  return `Hi, ${name}!`;  };  }  const greeter = getGreeter();  console.log(greeter("Bob")); // Hi, Bob! |

- Object의 개념: key와 value로 이루어진 자료형으로, 키와 값이 한쌍으로 이루어져 있다는 것이 특징이다. Object(객체)를 선언하기 위해서는 중괄호를 이용해야 한다. 하나의 변수 안에 다양한 정보들을 저장할 수 있고 유용하게 관리할 수 있다.

|  |
| --- |
| const person = {  name: "John",  age: 30,  greet: function() {  console.log(`Hello, my name is ${this.name}`);  }  };  console.log(person.name); // John  person.greet(); // Hello, my name is John |