[11/16] 십이지로 알아보는 연애유형(1) 톺아보기

1) HTML 반응형 웹 구현 설정

- <head> : Bootstrap > 문서 > 시작하기 > 1, 2번
  - <meta name="viewport" content="width=device-width, initial-scale=1">
  - <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9">

- <body>
  - main, qna, result 등의 class에 mx-auto 설정

```
<section id="main" class="mx-auto my-5 py-5 px-5">
```

  - img 에 img-fluid 설정

```
<img src="img/main.png" width="100%" class="mx-auto img-fluid">
```

2) button : Bootstrap > 문서 > 컴포넌트 > 버튼에서 원하는 속성을 class에 부여

```
<button type="button" class="btn btn-outline-primary mt-3"
onclick="js:begin()">시작하기</button>
```

```
<!-- 여기까지 html 종료, 이후 js  -->
```

3) font : fonts.google.com > 원하는 폰트 클릭 > 우측 상단 장바구니 메뉴 클릭

- <head> : selected family 의 link 삽입
  - <link rel="preconnect" href="https://fonts.googleapis.com">
  - <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  - <link href="https://fonts.googleapis.com/css2?family=Nanum+Pen+Script&display=swap" rel="stylesheet">
- css : font-family: 'Nanum Pen Script', cursive;

```
*{font-family: 'Noto Sans KR', sans-serif;}
```

4) Animation Effect

- fade in/out : 밝기 증가, 밝기 감소

```
function begin(){
    main.style.webkitAnimation = "fadeOut 1s";
    main.style.animation = "fadeOut 1s";
    setTimeout(() => {
        qna.style.webkitAnimation = "fadeIn 1s";
        qna.style.animation = "faddIn 1s";
        setTimeout(() => {
        main.style.display = "none";
        qna.style.display = "block";     }, 500)
        let qIdx = 0;
        goNext(qIdx);}, 500)}
```

- 과정
  - 버튼에 의해 begin() 실행
  - main 과 qna 에 fade 효과 부여
  - main 은 none 으로 사라지고, qna 가 등장
  - setTimeout 에 의해 Effect delay 부여

5) setTimeout

- 기본 구조 : setTimeout(func|code, [delay], [arg1], [arg2], ...)

```
function sayHi() {
    console.log("안녕하세요.");
}
setTimeout(sayHi, 1000);
```

안녕하세요.

  - func|code : 위 이미지에서는 sayHi 에 해당
  - delay : 1000 이면 1초, 2000이면 2초후 출력

- redirection 구조 : setTimeout(() => { }, 500)

```
setTimeout(() => {
    qna.style.display = "none";
    result.style.display = "block";}, 500)
```

```
setTimeout(()=>{
    var target = qnaList[qIdx].a[idx].type;
    for(let i=0; i<target.length; i++){
        select[target[i]] += 1;
    }
    for(let i=0; i<children.length; i++){
        children[i].style.display='none';
    }
    goNext(++qIdx);},400)
```

  - { } 에는 변수설정, 기능, if문등 다양한 구문이 가능

6) Array 의 구조(data.js)

```
const qnaList = [
  {q: '1. 이성 사이에 진정한 친구는 있다, 없다?',
    a: [{ answer: 'a. 이성 사이에 친구가 어딨어? 절대 없어', type: [1, 2, 4, 9] },
        { answer: 'b. 친구 있지, 절대 이성으로만 안 보일뿐', type: [0, 3, 6, 5, 10, 8] },
        { answer: 'c. 난 잘 모르겠어..', type: [7, 11 ] },]},
  {q: '2. 좋아하는 사람이 생겨 연락하고자 한다 ',
    a: [{ answer: 'a. 바로 먼저 연락한다.', type: [0, 3, 2, 8] },
        { answer: 'b. 언제 연락할지, 어떻게 말할지 심사숙고하여 연락한다.', type: [1, 6, 5, 10] },
        { answer: 'c. 상대방에게 먼저 연락 올 때까지 기다린다.', type: [7, 4, 9, 11] },]},
```

- const qnaList =[
  { q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }
  { q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }
  …
  ]
  - { q , a } 가 총 12개로 이루어짐

- 배열의 선언
  - 배열의 선언은 let 으로 하여 재시작시 새 배열을 받아야 데이터가 쌓이지 않음

```
let select = [0,0,0,0,0,0,0,0,0,0,0,0];
```

    - const 로 선언하면 새로 select 를 선언할 수 없음
  - 만약, const 로 선언한다면 select.fill(0) 로 새 배열로 시작을 명령

```
select.fill(0);
```

- goNext 에서의 Array 호출

```
function goNext(qIdx){
    var q = document.querySelector('.qBox');
    q.innerHTML = qnaList[qIdx].q;
    for(let i in qnaList[qIdx].a){
        addAnswer(qnaList[qIdx].a[i].answer, qIdx, i); }}
```

  - 설정 상황
    - function begin() 에서 qIdx =0; 설정이 된 상황
    - goNext() 의 파라미터로 qIdx 를 사용

  - qnaList[qIdx] : qnaList = [ ]
    - [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]

  - qnaList[qIdx].q : qnaList = [{ q , a }]
    - [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]

  - qnaList[qIdx].a : qnaList = [{ q , a }]
    - [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]

  - qnaList[qIdx].a[i] : qnaList = [{ q , a [i] }]
    - a[0] : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]
    - a[1] : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]
    - a[2] : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]

  - qnaList[qIdx].a[i].answer : qnaList = [{ q, a[i].answer}]
    - a[0].answer : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]
    - a[1].answer : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]
    - a[2].answer : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]

- addAnswer 의 Array 호출

```
var target = qnaList[qIdx].a[idx].type;
for(let i=0; i<target.length; i++){
    select[target[i]] += 1}
```

  - 설정상황
    - button 을 클릭시(이벤트리스너에 의해) 위 이미지 부분이 실행됨

  - qnaList[qIdx].a[idx].type : qnaList = [{ q, a[i].type}]
    - a[0].type : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]
    - a[0].type : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]
    - a[0].type : [{ q, a [{ answer, type[ ] } , { answer, type[ ] } , { answer, type[ ] }] }]
    - 해당 type 에는 [0~12] 의 다양한 숫자가 여러개 존재함

```
type : [1, 3, 2, 10, 8]
type : [7, 9, 11]
type : [0, 6, 5, 4]
```

- 다시시작 function infor()
  - 기존의 배열을 리셋

```
select.fill(0);
```

  - 기존의 결과값을 리셋

```
result.style.display = "none";
main.style.display = "block";
```

결과값을 *none,* 보이고자하는 첫화면을 *block* 으로 호출

  - 기존의 innerHTML 값을 리셋

```
resultImg.innerHTML = "";
```

img 가 계속 쌓이므로 해당하는 *resultImg* 를 *blank* 처리

- 특정 파라미터의 증감 : qIdx 는 질문지의 갯수를 의미, ++qIdx 로 증감을 표현
  - begin() : let qIdx = 0 으로 초기 값 설정, goNext(qIdx) 파라미터에 부여

```
function begin(){
    let qIdx = 0;
    goNext(qIdx);}
```

  - goNext(qIdx) : qnaList[qIdx] 의 배열 값으로 설정

```
function goNext(qIdx){
    var q = document.querySelector('.qBox');
    q.innerHTML = qnaList[qIdx].q;
    for(let i in qnaList[qIdx].a){
        addAnswer(qnaList[qIdx].a[i].answer, qIdx, i); }}
```

    - qnaList[qIdx] : qnaList의 배열 값 [qIdx]로 설정
    - addAnswer(,qIdx) : 함수 addAnswer 의 두번째 파라미터
      - 두 개가 동시에 파라미터로 사용됨

  - addAnswer(,aIdx,) : 두번째 파라미터로 지정

```
function addAnswer(answerText, qIdx, idx){
    // ...중간 코드 생략...
    goNext(++qIdx);}
```

    - ++qIdx : 파라미터가 수치적 증감이 표현가능

  - 정리 : ++qIdx 로 인하여 증감을 표현이 가능하다는 것을 알게됨
    - 즉, [배열] 구조이외에 (파라미터)에서도 증가가 가능

7) calResult() : select 의 최빈값??을 도출

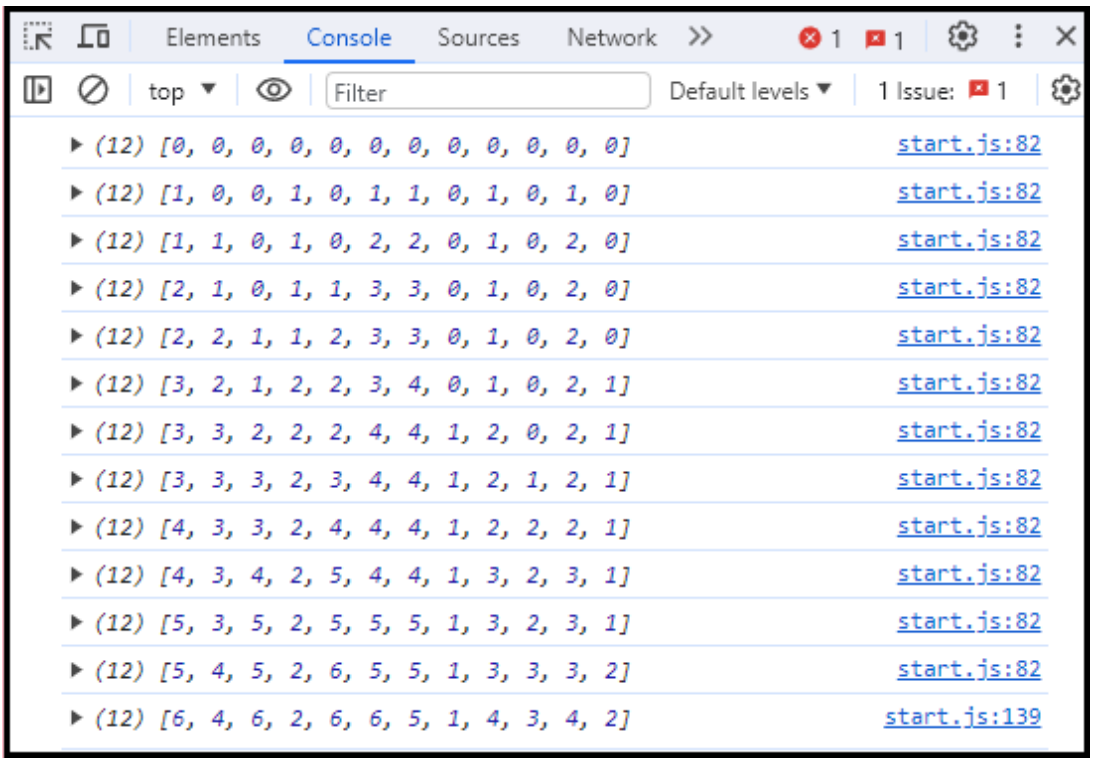
- 상황

```
let select = [0,0,0,0,0,0,0,0,0,0,0,0];
```

```
function calResult(){
    console.log(select);
    var result = select.indexOf(Math.max(...select));
    return result;}
```

  - …select : [0,0,0,0,0,0,0,0,0,0,0,0]
  - Math.max() : ( ) 안의 배열중 최대값
  - select.indexOf(n) : n 의 의미??
  - return result : 만약 result 가 6이면, calResult() 가 6이라는 의미??

- 개발자도구(f12) 를 통한 확인



  - 결과값으로 6이 여러개인데 어떤 기준으로 선택되는가?

- indexof : text 의 띄어쓰기 미포함하며 순서대로 인식

```
let s = "one two one two";
console.log(s.indexOf("one"));      0
console.log(s.indexOf("two"));      4
console.log(s.indexOf("two", 6));   12
console.log(s.indexOf("Two"));      -1
```

```
const str = "abab";
console.log(str.indexOf('ab'));     0
console.log(str.indexOf('ba'));     1
console.log(str.indexOf('abc'));    -1
console.log(str.indexOf('AB'));     -1
```

  - 출력값 0 : text의 띄어쓰기 미포함, 0번째 text에 있음을 의미
  - 출력값 4 : text의 띄어쓰기 미포함, 4번째 text에 있음을 의미
  - indexof(6) : 6번째부터 찾기
    - 출력값 12 : 띄어쓰기를 포함한 12번째 text에 있음을 의미
  - 출력값 -1 : 없음, 대문자 T 가 없음을 의미

- ... : 배열 앞쪽에 위치한 값 몇 개만 필요, 그 이후 이어지는 나머지 값(rest)들을 모아서 저장

```
let [name1, name2, ...rest] = ["Julius", "Caesar", "Consul", "of the Roman Republic"];
console.log(name1);
console.log(name2);
console.log(rest[0]);
console.log(rest[1]);
console.log(rest.length);
```

```
Julius
Caesar
Consul
of the Roman Republic
2
```