js 200519

- 함수
  - 선언적 함수(기본)
    - 매개변수, 리턴은 필수가 아님
    - ex.
      - function printTen()
        {
        console.log(10);
        }
      - function addNum(x,y)
        {
        return x + y;
        }
  - 익명함수
    - 이름이 없음
      - 변수에 할당
        - ex.
          - var plusTen2= function(x)
            {
            return x+10;
            }
          - plusTen2(); //익명함수 호출
            var plus3 = plusTen2 //함수의 복사
            console.log(plusTen2(10));
    - 굳이 함수의 재사용이 필요 없을 때 쓰이곤 함
  - 콜백함수
    - 매개변수가 함수
      - ex

- function myFunc(x)
  {
  // console.log(x+"의 타입은 " + typeof x); 여기서 맨 앞의 x에서도 함수가 실행됨.
  console.log(typeof x);
  x();
  }

  function myCall()
  {
  console.log('myCall이 호출');
  }
  - myFunc(myCall)
    - [console]
      function
      function.js:86 myCall이 호출
- 예제
  -
  var question = "동의하세요?";
  function ask(question, yes, no) //매개변수의 이름과
  {
  if (confirm(question)) yes()
  else no();
  }

  function agree() //함수의 이름은 달라도 됨. 상관 없음.
  {
  alert("동의했습니다.");
  }

  function cancle()
  {
  alert("취소했습니다.");
  }
- 타이머 함수
  - 내장함수
  - 특정 시간에 특정한 함수를 실행
  - !타이머 함수는 자바스크립트 코드 블록이 모두 실행된 후에서야 실행됨!
    - ex

- alert('1'); //1
  setTimeout(function(){ //3 마지막에 실행
  alert('2');
  }, 0);
  alert('3'); //2
- 매개변수
  - 함수
  - 시간
    - millisecond
    - 1000millisecond = 1초
  - 종류
    - setTimeout(함수, 시간)
      - 일정 시간 후 함수 실행
    - setIntervel(함수, 시간)
      - 일정 간격으로 함수 반복 실행
        - var intervalID= setInterval(seconds, 3000);
          - seconds 함수를 3000ms동안 반복하는 setInterval함수를 변수 intervalID에 할당
    - clearInterval(함수, 시간)
      - 일정 간격으로 함수를 반복해서 실행하는 것을 중지
        - ex
          - setTimeout(function(){
            clearInterval(intervalID);
            }, 5000);
            - 함수에 이름이 없어서 익명함수를 씀
            - 5000ms가 지나면 intervalID 종료

- 객체
  - 객체
    - 속성들을 포함
    - { }
    - 가장 상위 객체
      - window
        - 생략 가능
      - 중간 객체 생략 불가
        - ex

        - window.write-- 에러
        - document.write o
    - 속성
        - 속성이름, 속성값 쌍
        - 모든 형태의 자료형 가능
            - 배열, 함수, 객체,..
- DOM
    - Document Object Model 문서객체모델
        - 일종의 인터페이스
        - 모든 요소 정의
            - 문서 내
        - 요소에 접근하는 방법 제공
        - 자바스크립트가 객체모델을 이용해 할 수 있는 것
            - HTML요소 속성 추가, 제거, 변경
            - CSS 변경
            - HTML 이벤트 추가 및 반응
        - 계층구조
    - Document 객체
        - HTML 요소의 선택(DOM 셀렉터)
            - 단수
                - .getElementById
                    - 아이디 선택
                - .querySelector
                    - CSS선택자처럼 사용
                        - ex.
                            - querySelector('li')
                                - 제일 첫번째 li태그 리턴
                            - document.querySelector('#first')
            - 복수
                - 배열
                    - 인덱스를 활용할 수 있음
                - .getElementsByTagName
                - .getElementsByClassName

        - .querySelectorAll
    - 속성 get/set
        - .getAttribute
        - .setAtrribute
        - 속성
            - =
            - .textContent
                - 내용추가
                - ex.
                    - document.querySelector('h1').textContent= '안녕';
    - CSS 속성 바꾸기
        - .className
        - .classList
            - .add
                - 클래스 추가
            - .remove
                - 클래스 제거
            - .toggle
                - true/false
    - 기타
        - .parentElement
            - 부모 요소 불러옴
            - .children
                - 자식 요소들
            - ex.
                - li.parentElement.children[0]
                    - li의 부모요소의 자식들 중에 첫번째 요소 불러옴
- Q 콜백함수
    - 매개변수가 있는 함수는 매개변수로 못 넣나?
- 자바스크립트로 CSS에 있는 건 못 건드리나?

js 200520

- 1. 주사위 게임
  - html
    - ```
      <!DOCTYPE html>
      <html lang="ko">
        <head>
          <meta charset="utf-8">
          <title>Dicee</title>
          <link rel="stylesheet" href="styles.css">
          <link href="https://fonts.googleapis.com/css?family=Indie+Flower|Lobster"
      rel="stylesheet">

        </head>
        <body>

            <h1>Refresh Me</h1>

              Player 1
              <img class="img1" src="">

              Player 2
              <img class="img2" src="">

          <script src="app.js">

          </script>

        </body>

        <footer>
            © 2020 . busanIT. co. kr
        </footer>
      </html>
      ```
  - js
      ```

- ```
  var img1 = document.querySelector('.img1');
  var img2 = document.querySelector('.img2');
  var min= 1; // 최소
  var max= 6; // 최대
  var randomNumber1 = getRandomInt(min, max);
  var randomNumber2 = getRandomInt(min, max);
  var heading1 = document.querySelector('h1');

  // Math.random()은 0이상 1미만의 난수를 발생시킴
  function getRandomInt(min, max) //랜덤함수
  {
     return Math.floor(Math.random()* (max - min + 1) + min); //최댓값, 최솟값 포함.
  }


  //주사위 이미지 랜덤
  img1.setAttribute('src','images/dice'+ randomNumber1+'.png');
  img2.setAttribute('src','images/dice'+ randomNumber2+'.png');


  //누가 이겼는지 체크
  if(randomNumber1>randomNumber2)
  {
     heading1.textContent= '🚩player1 Wins!';
  }

  else if(randomNumber1===randomNumber2)
  {
     heading1.textContent= 'Draw!';
  }

  else
  {
     heading1.textContent= 'player2 Wins!🚩';
  }
  ```

- 자주 쓸 값을 변수가 가리키게 함
  - ex
    - var img1 =document.querySelector('.img1')
- 속성값 세팅
  - ex
    - img1.setAttribute('src','images/dice6.png')
  ```

- 1~6까지 난수 발생
  - Math.random()
    →참고링크 Math.random() - JavaScript | MDN
    - 0이상 1미만 난수 발생
    - 최솟값, 최댓값 포함하는 정수 난수
      - function getRandomInt(min, max)
        {
          return Math.floor(Math.random()* (max - min + 1) + min); //최댓값, 최솟값 포함.
          //Math.floor 버림.
        }
- 난수에 맞는 이미지 불러오기
  - var min= 1;
    var max= 6;
  - var randomNumber1 = getRandomInt(min, max);
    var randomNumber2 = getRandomInt(min, max);

    img1.setAttribute('src','images/dice'+ randomNumber1+'.png');
    img2.setAttribute('src','images/dice'+ randomNumber2+'.png');
- 승자 알림말 띄우기
  - 난수 비교
    - if else
      - var heading1 = document.querySelector('h1');
      - if(randomNumber1>randomNumber2)
        {
          heading1.textContent= '🚩player1 Wins!';
        }

        else if(randomNumber1===randomNumber2)
        {
          heading1.textContent= 'Draw!';
        }

        else
        {
          heading1.textContent= 'player2 Wins!🚩';
        }
  - h1 내용 변경하기
    - textContent

- 2. 이벤트

- <!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM 예제: 쇼핑리스트</title>
  </head>

  <body>
  <h1>쇼핑 리스트</h1>
  오늘 끝내자
  반드시!

  <input id="userInput" type="text" placeholder="리스트 추가">
  <button id="enter">입력!</button>

    <li random ="23" class="red blue">노트북
    가방
    책
    연습장
    볼펜
    지우개

  <script>
  var head1 = document.querySelector('h1');
  head1.className='cool'; // h1에 클래스이름 부여

  // style에 done클래스의
  var li = document.querySelector('li:last-child');
  li.classList.add('done');
  </script>

  <script src="shopping.js"></script>

  </body>
  </html>

```
var button = document.getElementById('enter');
var input = document.getElementById('userInput');
var ul = document.querySelector('ul');

button.addEventListener('click', function(){
    if(input.value!=="")
    {
        createListElement();

        // // console.log('클릭');
        // // 2. 새 태그 요소 만들기
        // var newList= document.createElement('li'); //아직 아무 곳에도 연결이 안 됨

        // // 3. 태그 요소에 textnode(요소 노드 안에 있는 글자) 내용 넣기  about node
        // newList.appendChild(document.createTextNode(input.value));
        //newList로 새 태그요소 li를 생성하면서, createTextNode로 텍스트요소 추가

        // ul.appendChild(newList); //새로 만든 태그 요소를 document.querySeeldt 연결
        // input.value=''; //입력후 value를 공백으로
    }

})

input.addEventListener('keypress', function(event){
//4. 어떤 키가 눌려졌는지 확인해야함. 키코드
// console.log(event.keyCode);

    if((input.value!=="") && event.keyCode===13) //13은 엔터
    {
            createListElement();

           // var newList= document.createElement('li');
            //newList.appendChild(document.createTextNode(input.value));
            //ul.appendChild(newList);
            //input.value='';
    }
})

function createListElement() {
    var newList = document.createElement('li');
    newList.appendChild(document.createTextNode(input.value));
    ul.appendChild(newList);
    input.value = '';
}
```

- .addEventListener
  →링크 [이벤트 참조 | MDN]
- 새 태그 요소 만들기
  - document.createElement('추가할요소')
    - ex
      - var newList= document.createElement('li'); //아직 아무 곳에도 연결이 안 됨
- 태그 요소에 textNode(요소 노드 안에 있는 글자)넣기
  - document.createTextNode('넣을값')
- 덧붙이기
  - .appendChild
    - (method) Node.appendChild<Text>(newChild: Text): Text
  - 만든 것과 기존의 것, 만든 것들 간에 잘 연결해줘야할 듯. 상하위 요소 잘 구분
- 코드 리팩토링
  - 중복되는 과정을 함수로 만들기
    - function createListElement() {
      var newList = document.createElement('li');
      newList.appendChild(document.createTextNode(input.value));
      ul.appendChild(newList);
      input.value = '';
      }
  - vscode
    - ctrl+shift+R

3. 배경생성기

- ```
  <!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>배경생성기</title>

    <link rel="stylesheet" href="style.css">
    <link rel="shortcut icon" href="#" type="image/x-icon">
  </head>


  <body id="gradient">
    <h1>배경 생성기</h1>

    <!-- 색을 입력하면 그 값들로 그라디언트 백그라운드를 줄 것 -->
    <input class= "color1" type="color" value="#00ff00">
    <input class= "color2" type="color" value="#ff0000">

    <h2>현재 CSS 배경</h2>
    <h3></h3>


    <script src="app.js"></script>

  </body>
  </html>
  ```
- ```
  var css = document.querySelector('h3');
  var color1= document.querySelector('.color1');
  var color2= document.querySelector('.color2');
  var body= document.getElementById('gradient');


  //세미콜론 위치 주의

  // input이벤트. 이 경우 색을 선택했을 때
  color1.addEventListener('input', setGradient);

  color2.addEventListener('input', setGradient);


  function setGradient() {
    body.style.background = "linear-gradient(to right," + color1.value + "," + color2.value
  + ")";
    css.textContent = body.style.background + ";";
  }
  ```

- .addEventListener의 두번째 매개변수는 함수가 들어가는 자리, 함수를 만들어서 그냥 그 자리에 넣어주면 됨

- 4. 드럼 킷
  - 오디오 재생
  - 타겟
    - this
      - 이벤트리스너가 동작한 태그 요소가 선택됨
        - console.log(this.textContent); //event.currentTarget
    - event.currentTarget
    - event.target

- 변수 써버릇하기
  - 반복되는 것
- function 오타 주의
- 대소문자 주의
- 세미콜론 위치 주의

js 200521

- 1. 드럼 킷
  - <!DOCTYPE html>
    <html lang="kr">

    <head>
      <meta charset="utf-8">
      <title>Drum Kit</title>
      <link rel="stylesheet" href="styles.css">
      <link href="https://fonts.googleapis.com/css?family=Arvo" rel="stylesheet">
      <link rel="shortcut icon" href="#" type="image/x-icon">

    </head>

    <body>

      <h1 id="title">Drum 🤖 Kit</h1>

        <button class="w drum">w</button>
        <button class="a drum">a</button>
        <button class="s drum">s</button>
        <button class="d drum">d</button>
        <button class="j drum">j</button>
        <button class="k drum">k</button>
        <button class="l drum">l</button>

      <footer>
        Made with ♡ in London.
      </footer>

    <script src="index.js"></script>
    </body>

    </html>

```
body {
  text-align: center;
  background-color: #283149;
}

h1 {
  font-size: 5rem;
  color: #DBEDF3;
  font-family: "Arvo", cursive;
  text-shadow: 3px 0 #DA0463;

}

footer {
  color: #DBEDF3;
  font-family: sans-serif;
}

.w {

  background: url('images/tom1.png') no-repeat center;
  background-size: cover;
}

.a {
  background: url('images/tom2.png') no-repeat center;
  background-size: cover;

}

.s {
  background: url('images/tom3.png') no-repeat center;
  background-size: cover;
}

.d {
  background: url('images/tom4.png') no-repeat center;
  background-size: cover;
}

.j {
  background: url('images/snare.png') no-repeat center;
  background-size: cover;
}

.k {
```

```
  background: url('images/crash.png') no-repeat center;
  background-size: cover;
}

.l {
  background: url('images/kick.png') no-repeat center;
  background-size: cover;
}

.set {
  margin: 10% auto;
}

.game-over {
  background-color: red;
  opacity: 0.8;
}

.pressed {
  box-shadow: 0 3px 4px 0 #DBEDF3;
  opacity: 0.5;
}

.red {
  color: red;
}

.drum {
  outline: none;
  border: 10px solid #404B69;
  font-size: 5rem;
  font-family: 'Arvo', cursive;
  line-height: 2;
  font-weight: 900;
  color: #DA0463;
  text-shadow: 3px 0 #DBEDF3;
  border-radius: 15px;
  display: inline-block;
  width: 150px;
  height: 150px;
  text-align: center;
  margin: 10px;
  background-color: white;
}
```

- //모든 버튼 선택

```
var drums = document.querySelectorAll('.drum');

var audioRoute = '';

for (i = 0; i < drums.length; i++) {
    drums[i].addEventListener('click', clickHandle);
    // drums[i].addEventListener('mouseleave', leaveBtn);
}

document.addEventListener('keypress', keyHandle);
// document.addEventListener('keyup', leaveKey)



//코드 리팩토링
function drumBit(element) {
    switch (element) {
        case 'w':
            // drums[0].style.color='white';
            audioRoute = "tom-1";
            break;

        case 'a':
            // drums[1].style.color='white';
            audioRoute = "tom-2";
            break;

        case 's':
            // drums[2].style.color='white';
            audioRoute = "tom-3";
            break;

        case 'd':
            // drums[3].style.color='white';
            audioRoute = "tom-4";
            break;

        case 'j':
            // drums[4].style.color='white';
            audioRoute = "snare";
            break;

        case 'k':
            // drums[5].style.color='white';
            audioRoute = "crash";
```

```
            break;

        case 'l':
            // drums[6].style.color='white';
            audioRoute = "kick-bass";
            break;

        default:
            return;
    }
    var audio = new Audio("sounds/" + audioRoute + ".mp3");
    audio.play();

    animationDrum(element);
}


function animationDrum(key) //key값이 넘어옴
{
    // 키값을 받아 해당 버튼 선택
    var el = document.querySelector('.'+key);
    el.classList.add('pressed'); //클래스 추가

    setTimeout(function(){
        el.classList.remove('pressed');
    }, 200);
}


function keyHandle(e) {
    // console.log('키눌러짐');
    // console.log(this); //#document
    // console.log(e.key);
    drumBit(e.key);
}


function clickHandle() {
    // console.log(this);
    // console.log(this.textContent); //event.currentTarget
    // console.log(event.target);
    drumBit(this.textContent);
}


// function leaveKey(e) 키를 뗐을 때 해당 drum클래스 요소의 색 바꾸기
```

```
// {
//     switch(e.key)
//     {
//         case 'w':
//             drums[0].style.color= '#DA0463';
//             break;

//         case 'a':
//             drums[1].style.color= '#DA0463';
//             break;

//         case 's':
//             drums[2].style.color= '#DA0463';
//             break;

//         case 'd':
//             drums[3].style.color= '#DA0463';
//             break;

//         case 'j':
//             drums[4].style.color= '#DA0463';
//             break;

//         case 'k':
//             drums[5].style.color= '#DA0463';
//             break;

//         case 'l':
//             drums[6].style.color= '#DA0463';
//             break;

//         default:
//             return;
//     }
// }

// function leaveBtn() {
//     this.style.color = '#DA0463';
// }
```

- 오디오 재생
- 타겟
  - this
- 키보드를 눌렀을 때 이벤트의 대상이 어디?

- 해당 드럼킷 화면에서 키보드를 누르면 소리가 나야함
- document
- function keyHandle(e)
  {
  // console.log('키눌러짐');
  // console.log(this); //#document
  console.log(e.key);
  }

- 코드 리팩토링
- 드럼 애니메이션
  - 눌렀을 때 색깔 바뀌도록
    - 클래스 적용
    - 일정 시간 이후 클래스 삭제
    - function animationDrum(key) //key값이 넘어옴
      {
      // 키값을 받아 해당 버튼 선택
      var el = document.querySelector('.'+key);
      el.classList.add('pressed'); //클래스 추가

      setTimeout(function(){
      el.classList.remove('pressed');
      }, 200);
      }

- this

  ➞링크 this | PoiemaWeb

  - 이벤트리스너가 동작한 태그 요소가 선택됨
    - console.log(this.textContent); //event.currentTarget
  - 자바스크립트의 함수는 호출될 때, 매개변수로 전달되는 인자값 이외에, arguments 객체와 this를 암묵적으로 전달 받는다.
  - 자바와 개념이 다소 다름
  - 동적으로 결정
    - 현재 실행 문맥
    - this에 바인딩할 객체
    - 함수가 어떻게 호출되었는지에 따라
      - 함수 호출
        - 기본적으로는 전역객체에 바인딩
          - 전역함수

- 내부함수
  - 외부함수에 바인딩x
  - 어디에서 선언되든
    - 전역객체 바인딩
    - 회피 방법이 있긴 함
  - 메소드의 내부함수
  - 콜백함수
- 메서드 호출
  - 메소드를 소유한 객체
- 생성자 함수 호출
  - 기존 함수에 new 연산자를 붙여서 호출하면 생성자 함수로 동작
    - 생성자 함수명 첫글자 대문자 약속
- apply/call/bind 호출

- 전역객체(Global Object)
  - 모든 객체의 최상위 객체
    - window
      - browser-side
    - global
      - server-side(Node.js)
  - 전역변수(global variable)를 프로퍼티로 소유
    - 전역 스코프
  - 글로벌 영역에 선언한 함수
- 2. 숫자 맞히기

- ```
  <!DOCTYPE html>
  <html lang="ko">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- 스켈레톤 CDN -->
    <link rel="stylesheet"
  href="https://cdnjs.cloudflare.com/ajax/libs/skeleton/2.0.4/skeleton.min.css">
    <title>숫자 맞추기</title>
  </head>

  <body>

      <h1>숫자 맞추기</h1>

        맞춰보세요  and 

        <input type="number" id="guess-input" placeholder="숫자 입력...">
        <input type="submit" value="확인하기" id="guess-btn">

    <script src="app.js"></script>
  </body>
  </html>
  ```
  ```

```
// 상수로 태그 요소들 사용
const game = document.querySelector('#game');
const minNum = document.querySelector('.min-num');
const maxNum = document.querySelector('.max-num');
const guessBtn = document.querySelector('#guess-btn');
const guessInput = document.querySelector('#guess-input');
const message = document.querySelector('.message');

const hintNum= document.querySelector('.hintNum');

let min = 1;
let max = 300;
let winningNum = getRandomNum(min, max);
let guessLeft = 10; //기회는 3번, 깎일 값

let chance = 10; //전체 횟수
let choiceNum= [];


minNum.textContent = min;
maxNum.textContent = max;

// 최솟값과 최댓값을 포함해 랜덤 값 생성
function getRandomNum(min, max) {
    return Math.floor(Math.random() * (max - min + 1) + min);
}


// 메시지와 색을 입력 받아서 화면에 출력
function setMessage(msg, color) {
    message.style.color = color;
    message.textContent = msg;
}


//확인하기 버튼을 눌렀을 때 이벤트
guessBtn.addEventListener('click', clickHandle);

function clickHandle() {
    // 값을 입력받아서 숫자로 변환
    let guess = parseInt(guessInput.value); //input창의 value값은 기본적으로 String임.
parseInt로 숫자로 바꿔줌
    // console.log(typeof guess);
    // console.log(isNaN(guess)); //true면 숫자가 아님

    if (isNaN(guess) || guess < min || guess > max) //숫자가 아니거나 min보다 작거나
```

max보다 큰 경우
```
    {
        setMessage(`숫자를 입력해주세요(${min}~ ${max})`, 'red');
        guessInput.value = '';
        guessInput.style.borderColor = 'red';
        return;
    }

    else if (guess === winningNum) {
        gameFinishWin(true,`${winningNum} 맞아요, YOU WIN!`);
    }

    // else
    // {
    //     guessLeft --;
    //     guessInput.style.borderColor = 'red';
    //
    //     if(guessLeft===0)
    //     {
    //         gameFinishWin(false,`Game Over... 정답은 ${winningNum} 입니다`);
    //         return;
    //     }
    //     setMessage(`${guess}는 틀렸어요, ${guessLeft}번 남았습니다.`, 'red');
    // }


    else {
        guessLeft--;
        guessInput.style.borderColor = 'red';

        if (guessLeft === 0) {
            gameFinishWin(false,`Game Over... 정답은 ${winningNum} 입니다`);
            return;
        }

        else if (guess > winningNum) {
            setMessage(`${guess}는 틀렸어요, 답은 그보다 작습니다. ${guessLeft}번 남았습
니다.`, 'red');
        }

        else {
            setMessage(`${guess}는 틀렸어요, 답은 그보다 큽니다. ${guessLeft}번 남았습니
다.`, 'red');
        }

        ///
```

```
        guessInput.value= '';
        choiceNum[chance-guessLeft-1] = guess;
        hintNum.textContent= choiceNum.toString();

        }
    }



// 리팩토링
function gameFinishWin(win, msg)
{
    let color;
    win === true ? color = 'green' : color = 'red' //true면 두번째, false면 세번째 실행

    guessInput.disabled = true; //입력 더이상 안 받음
    guessInput.style.borderColor = color;
    setMessage(msg, color);

    // 다시하기
    guessBtn.value = '다시 하기'; //버튼 글자 바꿈
    guessBtn.classList.add('play-again'); //클래스 추가

    game.addEventListener('mousedown', function(e){//위에 클릭 이벤트가 있어서 겹칠
수 있으니 mousedown을 씀
        console.log('클릭');
        if(e.target.className==='play-again')
        {
            window.location.reload(); //새로고침
        }
    })
}
```

- 배열 안의 값을 전부 출력하게 하기 위해
- toString()을 사용해봄

- 스켈레톤
  - CSS 라이브러리의 일종
  - html에 CDN링크 추가
- 상수- 태그 요소 접근
- 최솟값과 최댓값을 포함한 난수 생성->답
- 메시지와 색을 입력받아 메시지 출력
- 확인하기 버튼을 눌렀을 때 이벤트
  - 값을 입력받아 숫자로 변환

- input창의 value값은 기본적으로 String
- parseInt로 숫자로 바꿔줌

- 입력값이 숫자인지 확인
  - .isNaN()
    - 주어진 값이 Nan인지 판별
  - Number.isNaN()
    - 더 엄격
    - 강제로 매개변수를 숫자로 변환 하지x
    - 이는 이제 보통NaN으로 변환됐을 값이 안전하게 전달되지만, 실제로는 NaN과 같은 값이 아님을 의미합니다. 이는 또한 오직 숫자형이고 또한 NaN인 값만이 true를 반환함을 뜻합니다.
  - NaN
    - Not-A-Number
      - console.log(isNaN(guess)); //true면 숫자가 아님

- Template literals
  https://developer.mo... Template literals - JavaScript | MDN
  - 문자열 리터럴
    - 내장된 표현식을 허용
  - 괄호 대신 `(1 옆의)
  - ${ }
  - ex
    - setMessage(`숫자를 입력해주세요(${min} ~ ${max})`, 'red');

- 자바
  - 자바의 정석
  - 조건, 반복문
    - Q4_9
      - [4-9] 숫자로 이루어진 문자열 str이 있을 때
        각 자리의 합을 더한 결과를 출력하는 코드를 완성하라
        만일 문자열이 "12345" 면 , '1+2+3+4+5' 결과인 15를 출력

        [Hint] String charAt(int i) 클래스 사용

- package prac;

```
public class Q4_9 {
	public static void main(String[] args) {

		String str = "12345";

		int sum = 0;


		for(int i=0; i < str.length(); i++) {

			sum += str.charAt(i)- '0';
//			sum += Integer.parseInt(String.valueOf(str.charAt(i)));

		}

		System.out.println("sum="+sum);
	}
}
```

- Q4_10
  - package prac;

```
public class Q4_10 {
	public static void main(String[] args) {

		//
		//		int 타입 num
		//		각 자리의 합을 출력하는 코드
		//		문자열로 변환하지 말고 숫자로만 처리.


		int num = 12345;
		int sum = 0;

		while(num > 0) {
			sum += num%10;
			num /= 10;
			}

		System.out.println("sum="+sum);

	}

}
```

- 10으로 나눈 나머지를 구해서 sum에 더하고 num을 10으로 나누는 걸 반복
- 내 풀이
  - //각 자리를 나눠주기
    for(int i=10000; i>=1;)
    {

    sum+= num/i;
    num= num-(num/i*i);
    i=i/10;
    }
  - 내 경우엔
    - num을 i(초기값 10000)으로 나눠준 뒤에 sum에 더하고, num에서 i단위의 수를 빼주고, i의 자릿수를 한 단계 낮춤
    - 앞의 수 부터 얻어나감
    - 뒤에서 수를 얻어나간 책의 풀이가 더 단순해보임

- Q4_11
  - package prac;

    public class Q4_11 {
    public static void main(String[] args) {

    //피보나치수열 10번째 값 1, 1, 2, 3,,...

    int firstNum = 1;
    int secondNum = 1;
    int presentNum;
    int goalNum= 20;

    System.out.print(firstNum+ " " + secondNum+" ");

    for(int i=1; i<=goalNum-2; i++)
    {
    presentNum= firstNum + secondNum;
    System.out.print(presentNum+ " ");
    firstNum= secondNum;
    secondNum= presentNum;
    }
    }
    }
    - 풀이와 방식이 같았음

- 추가링크
  https://www.zerocho.com/category/Ja...
- 프로그래밍이란?
  https://poiemaweb.c... Progmamming | PoiemaWeb
  - 일종의 커뮤니케이션
    - 컴퓨터에게 실행을 요구
  - 문제해결능력 필요
    - 문제(요구사항)을 명확히 이해
      -
    - 문제해결 방안 정의
    - 복잡함을 단순하게 분해(Decomposition)
      - 작은 단위로
    - 자료를 정리하고 구분(Modeling)
  - 컴퓨터 관점에서의 문제 바라보기
    - Computational thinking
    - 과제 분해
    - 패턴화
    - 평가 가능
      - 명확한 수치화
  - 프로그래밍 언어
    - 기본 개념과 동작 원리 이해
      - 문맥에 맞는 정확한 용어의 사용
- 결국 프로그래밍은 요구사항의 집합을 분석하여 적절한 자료구조와 함수의 집합으로 변환한 후, 그 흐름을 제어하는 것이다