F.N.S 6th Week

JavaScript Notion - 추상화, 제어문

목차

다룰 주제

JavaScript 추상화

JavaScript 제어문

간단한 실습

Q&A

추상화

추상화란

'추상'이란 '사물이나 표상을 어떤 성질, 공통성, 본질에 착안하여 그것을 추출하여 파악하는 것' 을 말한다.
즉, 추상화란 핵심적인 개념 또는 공통적인 기능을 추출하는 것을 말한다.

추상화하는 이유

1. 가독성 증가 - 복잡한 로직은 숨기고 간결하게 표현하면 읽기도 쉽고 이해하기도 쉬움
2. 생산성 및 재사용성 증가 - 공통적인 부분을 추출해 잘 만든 클래스 또는 함수는 재사용 가능하며 생산성을 높여줌

- 자료 추상화는 불필요한 정보는 숨기고 중요한 정보만을 표현함으로써 프로그램을 간단히 만드는 것.
- 자료 추상화를 통해 정의된 자료형을 추상 자료형이라고 함.
- 객체 지향 프로그래밍에서 일반적으로 추상 자료형을 클래스, 추상 자료형의 인스턴스를 객체, 추상 자료형에서 정의된 연산을 메소드(함수), 메소드의 호출을 생성자라고 한다.

추상화 - 할당 연산자

할당 연산자 (=)

할당 연산자는 오른쪽 피연산자의 값을 왼쪽 피연산자에 할당한다.
= 등호를 사용 (수학적으로 왼쪽 오른쪽의 값이 같다 가 아닌 값을 할당한다라는 의미를 가진다.)
ex) let x = 5;
x = x - 2;
console.log(x); -> 3

복합 할당 연산자

할당 연산자와 결합해서 쓰이는 표현 . 보다 간략하게 코드를 작성하도록 해준다.
변수의 값을 1씩 증가 시키거나 감소 시킬때는 증가, 감소 연산자를 사용한다.
i++; >증가 연산자 / i--; >감소 연산자

복합 할당 연산자

이름	단축 연산자	의미
할당	x = y	x = y
덧셈 할당	x += y	x = x + y
뺄셈 할당	x -= y	x = x - y
곱셈 할당	x *= y	x = x * y
나눗셈 할당	x /= y	x = x / y
나머지 연산 할당	x %= y	x = x % y
지수 연산 할당	x **= y	x = x ** y
왼쪽 시프트 할당	x <<= y	x = x << y
오른쪽 시프트 할당	x >>= y	x = x >> y
부호없는 오른쪽 시프트 할당	x >>>= y	x = x >>> y
비트 AND 할당	x &= y	x = x & y
비트 XOR 할당	x ^= y	x = x ^ y
비트 OR 할당	x |= y	x = x | y

추상화 - return 문

return문

- 어떤 값을 되돌려주는 아웃풋의 역할.(함수에서 결괏값을 반환할 때 사용.)
- 함수 자체의 실행을 중단한다.
- return문이 실행되고 나서는 함수 실행이 중단되기에 return문 밑에 코드는 실행 될 일이 없다.

```
function 함수명() {
  let code = 1;
  return code;

  //return이 이미 실행되서 이 코드는 의미 없다.
  let code_2 = 2;
}
```

return 과 console.log 의 차이점

- console.log 는 값을 결과로 정의하지 않고, 단순 결과값을 콘솔창에 표시
- return 은 다음의 결과로 함수의 값을 정의하고 함수를 종료하겠다는 의미. (콘솔에 값을 반환하는 것이 아니다.)

추상화 - 옵셔널 파라미터

옵셔널 파라미터

- 파라미터가 있는 함수에 아무런 값을 전달하지 않고 함수를 호출하는 경우 undefined값이 출력됨.
- 필요에 따라서 undefined값이 아닌 다른 값이 자동으로 전달되게끔 파라미터에 기본 값을 설정하는 방법.
- 함수를 호출 할 때 파라미터 값을 전달해도 되고 생략해도 되기에 선택적으로 전달을 받는다고 해서 이런 파라미터를 옵셔널 파라미터라고 함.
- 옵셔널 파라미터는 파라미터 중 가장 뒤 쪽으로 설정.

```
function 함수명(name, nationality = '한국')

/*출력할 때 nationality 값을 생략하면 한국이라고 출력된다.
출력할 때 nationality 값을 전달하면 지정한 값으로 출력된다*/
```

추상화 - 상수

상수 (const)

- 절대 변하지 않고 항상 일정한 값을 상수라고 한다.
- const 로 지정. ex) const myNumber = 01012345678;
- 수정할 일이 없고 고정된 값을 유지하는 것을 상수로 선언하는것이 좋음.
- 값의 변경이 불가능함.
- 상수의 이름은 대문자로 쓰고 두 단어 이상일 때 언더바_를 사용하는 암묵적인 규칙이 있음
- ex) const MY_NUMBER = 01012345678;

```
const myNumber = 01012345678;
console.log(myNumber);

// 대문자, 언더 바 사용
const MY_NUMBER = 01012345678;
console.log(MY_NUMBER);
```

제어문 - if문

if 문

```
if (조건부분) {
 동작부분1 ;
} else {
  동작부분2 ;
}
```

- 조건이 맞으면 if문의 동작부분 1 실행
- 조건이 틀리면 else문의 동작부분2 실행

else if 문

```
if (조건부분) {
 동작부분1 ;
} else if (조건부분2) {
 동작부분2 ;
} else {
 동작부분3;
}
```

- 여러가지 조건이 있을 때 if문 안에 else if 문을 활용함.

제어문 - switch 문

switch 문

- break 문을 만나야 동작을 종료함.
- switch문을 if문으로 대체할 수 있음.
  - 범위를 만족하는 조건식을 만들 때는 if문을 활용하는 것이 효과적이고.
  - 특정값을 만족하는 조건식을 만들 때는 switch문이 효과적이다.
- switch 문은 값들을 비교할 때 자료형을 엄격히 구분한다.
- if문으로 대체할 때는 조건식에서
  반드시 등호 3개를 사용하여 일치 비교를 해야한다.

```
switch (비교할_값) {
  case 조건값_1:
    동작부분;
    break;
  case 조건값_2:
    동작부분;
    break;
  default:
    동작부분;
}
```

제어문 - for반복문

for 반복문

```
var brands = ['Samsung', 'Apple', 'Starbucks',
 'Mcdonald','Amazon','Google'];

for (var i = 0; i < 6; i = i+1) {
  console.log(brands[i]);
}
```

- var i =0 → 반복문에 사용할 변수 선언
- i < 6 → 반복 조건
- i = i + 1 → 반복문이 끝나고 실행될 코드
- console.log(brands[i]); → 반복시키고 싶은 코드
- 반복문 초기값이 0으로 설정되고 조건을 통과하면 반복코드 실행 후 1이 추가됨. 조건을 통과할 때마다 1이 추가되고 조건을 통과 하지 못할 때 반복문 종료.

제어문 - while반복문

while 반복문

```
//while문을 이용해 i보다 큰 수 중에서 가장 작은 7의 배수를 찾는 코드.

let i = 30; /*while 문에는 초기화부분이 없어 반복에 필요한 횟수를 카운트하려면 반복문 밖에서 글로벌변수를 지정.*/

while (i % 7 !== 0) {
  i++; //반복횟수를 증가시키는 추가동작부분이 없기때문 동작부분안에 작성.
}

console.log(i);

//출력결과
35
```

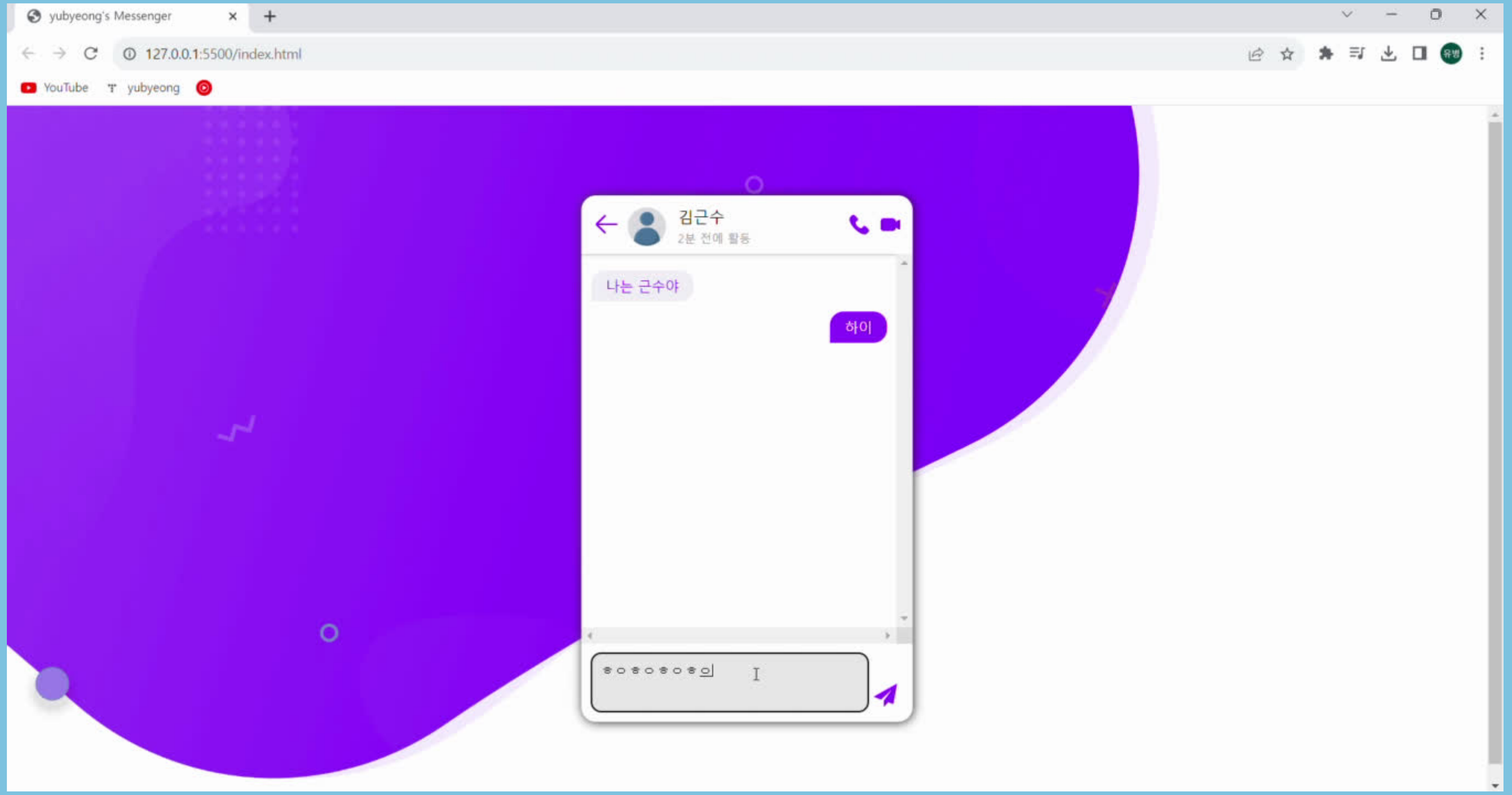
```
while (조건부분) {
  동작부분
}
```

- 반복문 밖에 글로벌 변수를 가지고 조건을 평가하고 반복문 안에서도 변수를 다루고 반복문이 다 실행된 후에 반복문 밖에서도 변수를 활용.
- 변수를 반복문 실행 전 실행 후에도 사용할 때 while반복문 사용.

JS 활용 실습

HTML, CSS, JS를 활용한 간단한 채팅 웹

```
const chatBox = document.querySelector('#chat-box');
const input = document.querySelector('#input');
const send = document.querySelector('#send');

function sendMyText() {
  const newMessage = input.value;
  if (newMessage) {
    const div = document.createElement('div');
    div.classList.add('bubble', 'my-bubble');
    div.innerText = newMessage;
    chatBox.append(div);
  } else {
    alert('메시지를 입력하세요...');
  }

  input.value = '';
}

send.addEventListener('click', sendMyText);

/*ketpress 타입을 사용하여 shift + enter 키로 줄바꿈이 일어나고
 메세지는 보내지지 않게 하는 코드. */
function sendMyTextByEnter (e) {
  if (e.key === 'Enter' && !e.shiftKey) {
    sendMyText();
    e.preventDefault();
  }
}

input.addEventListener('keypress', sendMyTextByEnter);
```


yubyeong's Messenger
127.0.0.1:5500/index.html
YouTube
yubyeong
김근수
2분 전에 활동
나는 근수야
하이

Q&A

감사합니다