항해99 8기
Node.JS

객체 리터럴, 그리고 구조분해할당

– 이민석, unchaptered(git)

항해99 8기 Node.JS

커피 좋아하는 개발자
낮잠 자는 걸 좋아하는 개발자

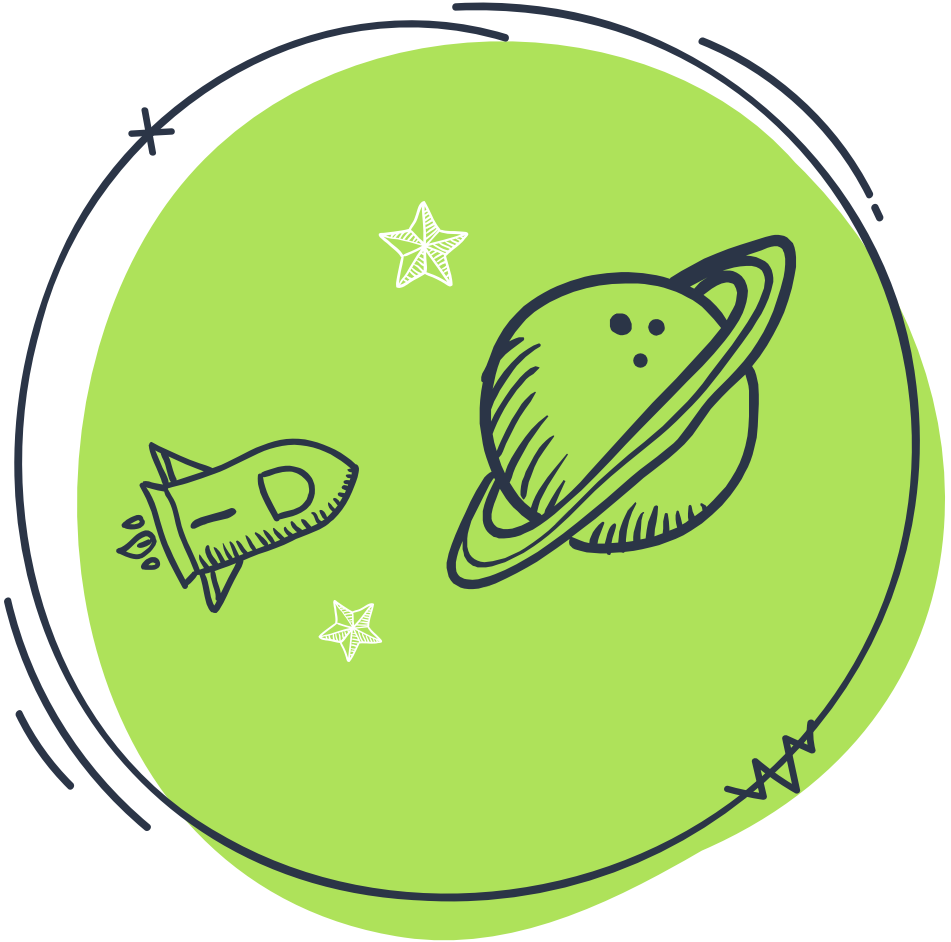
Loose JavaScript 문법이 재미 없는 이유

- Why?
- What?
- How?

WHY

객체 리터럴/구조분해할당

for, 안정적이고 효율적인 코딩



객체 리터럴

Object Literal

Coding?

Literal?

```
const username = 'unchaptered';
```

변수명, 식별자명 값, 그리고 문자열 리터럴

Many Literal

```
const name = 'unchaptered';
const age = 27;
const true_or_false = true;
```

리터럴
= 값

Object Literal

```
const user = {
    username: 'unchaptered',
    age: 27
};
```

객체 리터럴
= 객체를 의미하는 값

WHY ?

YOU → FRIENDS

WHY ?

FUNCTION ⟶ FUNCTION

```
const user = {
    username: 'unchaptered',
    age: 27
};
```

Object Literal

```
const user = {
    username: 'unchaptered',
    age: 27
};
```

작성 방법은 {} 괄호 안에,
왼쪽에는 변수명, 오른쪽에는 값
왼쪽을 Key, 오른쪽을 Value 라고 부른다.

PRAC, JOIN API (1)

```
function try_join() {

    const user = {
        username: 'unchaptered',
        age: 27
    }

    return act_join(user);

}
function act_join(user) {

    return user.age + '살의 ' + user.usrename + '님 반갑습니다.';

}
```

객체의 Key 를 호출할 수 있는 방법은 2가지가 있다.

사용 방법

1. user.username
2. user['username']

PRAC, JOIN API (2)

```
function try_join() {

    const user = {
        username: 'unchaptered',
        email: 'unchaptered@gmail.com',
        password: 'hello',
        meta: {
            age: 27,
            phone: '010-0000-000',
            hobbies: [ 'a', 'b', 'c' ]
        }
    }

    return act_join(user);

}
```

하지만 user 안에 정보가 많다면?

user.meta.age?
user.meta.phone?

이렇게 작성할 것인가?

PRAC, JOIN API (2)

```
const user = {
    username: 'unchaptered',
    email: `unchaptered@gmail.com`,
    password: '1234'
}

console.log(user.meta.age);
```

```
TypeError: Cannot read properties of undefined (reading 'task')
```

호출한 친구가
user 안에 없다면?

2

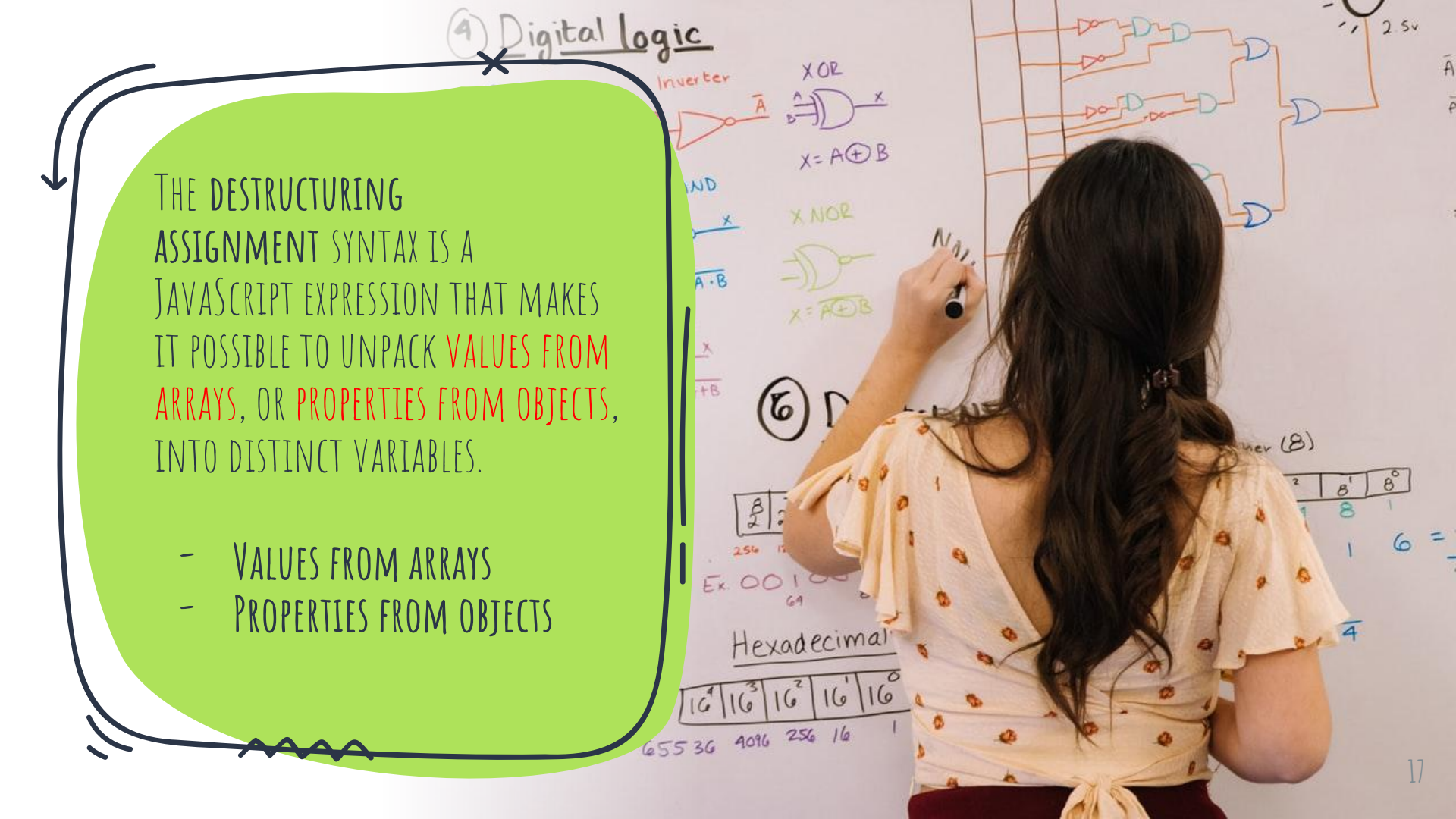
객체의 Key 를 더 짧게 부르기 위해서…

구조 분해 할당

Destructuring

The **destructuring assignment** syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

- Values from arrays
- Properties from objects



KEYWORD

구조 → 분해 → 할당

재포장?

KEYWORD

```
const user = {
    username: 'unchaptered',
    meta: {
        age: 28
    }
}


const { username } = user;
const { meta: { age } } = user;
```

등호 표시 '=' 를 기준으로,

우측에 있는 {} [] 는 리터럴로 객체 / 배열 생성

좌측에 있는 {} [] 는 구조분해 로 내부에 작성한 변수명 이 객체 안에 Key 로 존재한다면,

해당 Key 에 대응되는 Value 를 변수명에 할당

PRAC, JOIN API (3)

구조분해 방식

```
function try_join() {

    const user = {
        username: 'unchaptered',
        email: 'unchaptered@gmail.com',
        password: 'hello',
        meta: {
            age: 27,
            phone: '010-0000-000',
            hobbies: [ 'a', 'b', 'c' ]
        }
    }

    act_join(user);

}
```

객체 리터럴로 객체 생성

```
function act_join(user) {

    const {
        username,
        meta: { age: my_age }
    } = user;

    return my_age + '살의 ' + username + '님 반갑습니다.';

}
```

구조분해로 새 변수에 값 할당

PRAC, JOIN API (4)

매개변수 차원에서의 구조분해 방식

```
function act_join({ username, meta: { age }}) {

    return age + '살의' + username + '님 반갑습니다.'

}
```

```
function act_join({ username, meta: { age = 20 }}) {

    return age + '살의' + username + '님 반갑습니다.'

}
```

매개변수도 누군가에게 넘겨 받기 때문에,
구조분해로 새 변수에 값 할당 가능

더 나아가서

Object ++

Loop, with destructuring

```
const user_list = [
    { name: 'A', age: 10 },
    { name: 'B', age: 20 },
    { name: 'C', age: 30 }
]
for (let idx = 0; idx < user_list.length; idx++) {
    name = user_list[idx].name;
    age = user_list[idx].age;
    console.log(name, age);
}
```

Loop, with destructuring

```
const user_list = [
    { name: 'A', age: 10 },
    { name: 'B', age: 20 },
    { name: 'C', age: 30 }
]
for (const user of user_list) {
    name = user.name;
    age = user.age;
    console.log(name, age);
}
```

For of 문법은 배열 및 객체 를 대상으로

값, Value 자리에 있는 친구들을 꺼내준다.

배열은 Key 자리에 Index 가 있기 때문에..

Loop, with destructuring

```
const user_list = [
    { name: 'A', age: 10 },
    { name: 'B', age: 20 },
    { name: 'C', age: 30 }
]
for (const { name, age } of user_list) {
    console.log(name, age);
}
```

이 친구도 받는 부분이기 때문에 구조분해할당이 가능!!

Loop, with destructuring

```
const user_list = [
    { name: 'A', age: 10 },
    { name: 'B', age: 20 },
    { name: 'C', age: undefined }
]
for (const { name, age = 0 } of user_list) {
    console.log(name, age);
}
```

역시나, 기본값 할당도 가능!

조금만 더 나아가서

Array ++

Loop, with destructuring

```
const cards = [
    [100, 10],
    [200, 20],
    [300, 30]
];

for (const card of cards) {
    const width = card[0];
    const height = card[1];
    console.log(width, hieight);
}
```

Loop, with destructuring

```
const cards = [
    [100, 10],
    [200, 20],
    [300, 30]
];

for (const [ width, height ] of cards) {
    console.log(width, height);
}
```

Thanks!

Any questions?