프론트엔드 개발자를 위한 Vue.js 시작하기

Vue.js 시작하기

ES2015 - 다양한 기법

한국기술교육대학교
온라인평생교육원

학습내용

- Argument, Parameter, Operation
- Destructed, Object literal, Modules

학습목표

- Default argument, Rest parameter, Spread operation에 대해 설명할 수 있다.
- Destructed assignment, Object literal, Modules 이용에 대해 설명할 수 있다.

Argument, Parameter, Operation

Argument, Parameter, Operation

1 Default Argument

- 함수의 매개변수 선언할 때 기본 값 지정

```
function some(x, y = 10, z = 20) {
      console.log(`x=${x}, y=${y}, z=${z}`)
}
some(1)//x=1, y=10, z=20
some(1, 2)//x=1, y=2, z=20
some(1, 2, 3)//x=1, y=2, z=3
```

Argument, Parameter, Operation

2 Spread operator

- 배열, 객체를 각 요소로 분리시키는 연산자
- 배열 복사, 객체 결합 등에 유용하게 사용됨

Argument, Parameter, Operation

Argument, Parameter, Operation

2 Spread operator

배열 복사

```
var array1=[10,20,30]
var array2 = [1, 2, ...array1]
console.log(array2)//[1, 2, 10, 20, 30]

array2.push(...array1)
console.log(array2)//[1, 2, 10, 20, 30, 10, 20, 30]
```

Argument, Parameter, Operation

2 Spread operator

Set 객체 복사

```
var datas = new Set([1,1,3,4])
var array3 = [10, 20, ...datas]
console.log(array3)//[10, 20, 1, 3, 4]
```

Argument, Parameter, Operation

Argument, Parameter, Operation

2 Spread operator

배열 복사

```
var obj = {
    data1: 'hello',
    data2: 10
}
var obj2 = {
    ...obj,
    data3: true
}

console.log(obj2)//{data1: "hello", data2: 10, data3: true}
```

Argument, Parameter, Operation

3 Rest Parameter

- 함수의 Argument를 Spread operator(…)을 이용하여 선언하는 기법
- Rest parameter 값은 함수 내에서 배열로 이용

Argument, Parameter, Operation

Argument, Parameter, Operation

3 Rest Parameter

- 함수의 마지막 Argument만 Rest parameter로 선언이 가능

```
function some (x, y, ...z) {
        console.log(z.length)//3
        z.forEach( arg => {
                console.log(arg)//hello - true - 2
        })
}
some(1, 2, "hello", true, 2)
```

Destructed, Object literal, Modules

Destructed, Object literal, Modules

1 Destructed assignment

- 객체 혹은 배열을 분해 하는 작업
- 변수 명 변경 혹은 default 값 명시 가능

Destructed, Object literal, Modules

1 Destructed assignment

- Destructed assignment를 사용하지 않은 경우의 예

```
var obj = {
    data1: 'hello',
    data2: 10,
    data3: true
}

let data1 = obj.data1
let data2 = obj.data2
let data3 = obj.data3
```

Destructed, Object literal, Modules

Destructed, Object literal, Modules

1 Destructed assignment

- Destructed assignment 사용 예

```
var obj = {
    data1: 'hello',
    data2: 10,
    data3: true
}

let {data1, data2, data3} =
obj
```

Destructed, Object literal, Modules

1 Destructed assignment

- Default 값 명시

```
var obj = {
    data1: 'hello',
    data2: 10,
    data3: true
}

let {data1, data2 = 20, data3, data4 = 'world'} = obj
```

Destructed, Object literal, Modules

Destructed, Object literal, Modules

1 Destructed assignment

- 이름 변경 획득

```
var obj = {
    data1: 'hello',
    data2: 10,
    data3: true
}

let {data1: a1, data2: a2 = 20, data3: a3, data4: a4 = 'world'} = obj
```

Destructed, Object literal, Modules

1 Destructed assignment

- 배열 분해도 제공
- 배열 분해는 중괄호({ })를 사용하지 않고 대괄호 ([ ])를 사용함

```
var array = ["hello","world"]
let [b1, b2, b3 = true] = array
```

Destructed, Object literal, Modules

Destructed, Object literal, Modules

2 JSON

- Key - value 쌍으로 표현하는 데이터 표현 방식

```
var json = {
    "data1": "hello",
    "data2": 10,
    "data3": true
}
```

Destructed, Object literal, Modules

3 Object Literal

JSON	특징
• JavaScript Object Notation	• Key: Value 쌍으로 표현하는 객체 표현 방식

```
var user = {
    name: 'hong',
    email: 'a@a.com',
    age: 30
}

var name = user.name
```

Destructed, Object literal, Modules

Destructed, Object literal, Modules

3 Object Literal

- 객체 표현임으로 함수 등록 가능

```
var user = {
    name: 'hong',
    email: 'a@a.com',
    age: 30,
    sayHello: function(arg){
        console.log(`Hello.. ${arg}`)
    }
}
```

Destructed, Object literal, Modules

3 Object Literal

- Object 구성할 때 축약형으로 구성 가능

```
var name = 'hong'
var email = 'a@a.com'

var user = {
    name: name,
    email,
    age: 30,
}
```

Destructed, Object literal, Modules

Destructed, Object literal, Modules

3 Object Literal

- 객체 내의 멤버는 this 예약어로 이용

```
var user = {
    name: name,
    email,
    age: 30,
    sayHello: function(){
        console.log(`Hello.. ${this.name}, ${this.email}, ${this.age}`)
    }
}
```

Destructed, Object literal, Modules

3 Object Literal

- JSON 문자열 파싱
- JSON 문자열을 Object로 파싱해서 사용

Destructed, Object literal, Modules

Destructed, Object literal, Modules

3 Object Literal

- JSON.parse() 이용

```
const json = '{"result":true, "count":42}'
console.log(json.result)//undefined
```

```
const json = '{"result":true, "count":42}'
const obj = JSON.parse(json)
console.log(obj.result)//true
```

Destructed, Object literal, Modules

3 Object Literal

- Object Literal을 JSON 문자열로 변환
- JSON.stringify() 이용

Destructed, Object literal, Modules

Destructed, Object literal, Modules

3 Object Literal

```
var user = {
    name: name,
    email,
    age: 30,
    sayHello: function(){
        console.log(`Hello.. ${this.name}, ${this.email}, ${this.age}`)
    }
}

const str = JSON.stringify(user)
console.log(str)
console.log(user)
```

Destructed, Object literal, Modules

3 Object Literal

```
{"name":"hong","email":"a@a.com","age":30}
▼{name: "hong", email: "a@a.com", age: 30, sayHello: ƒ}
    age: 30
    email: "a@a.com"
    name: "hong"
  ▶ sayHello: ƒ ()
  ▶ [[Prototype]]: Object
```

Destructed, Object literal, Modules

Destructed, Object literal, Modules

4 Modules

- 자바스크립트 파일의 구성요소(변수, 함수, 클래스)를 외부 파일에 공개 및 이용

```
//  a.js
export function sum (x, y) { return x + y }
export var pi = 3.141593

//  b.js
import * as math from “a”
console.log(“2π = “ + math.sum(math.pi, math.pi))

//  c.js
import { sum, pi } from “a"
console.log("2π = " + sum(pi, pi))
```

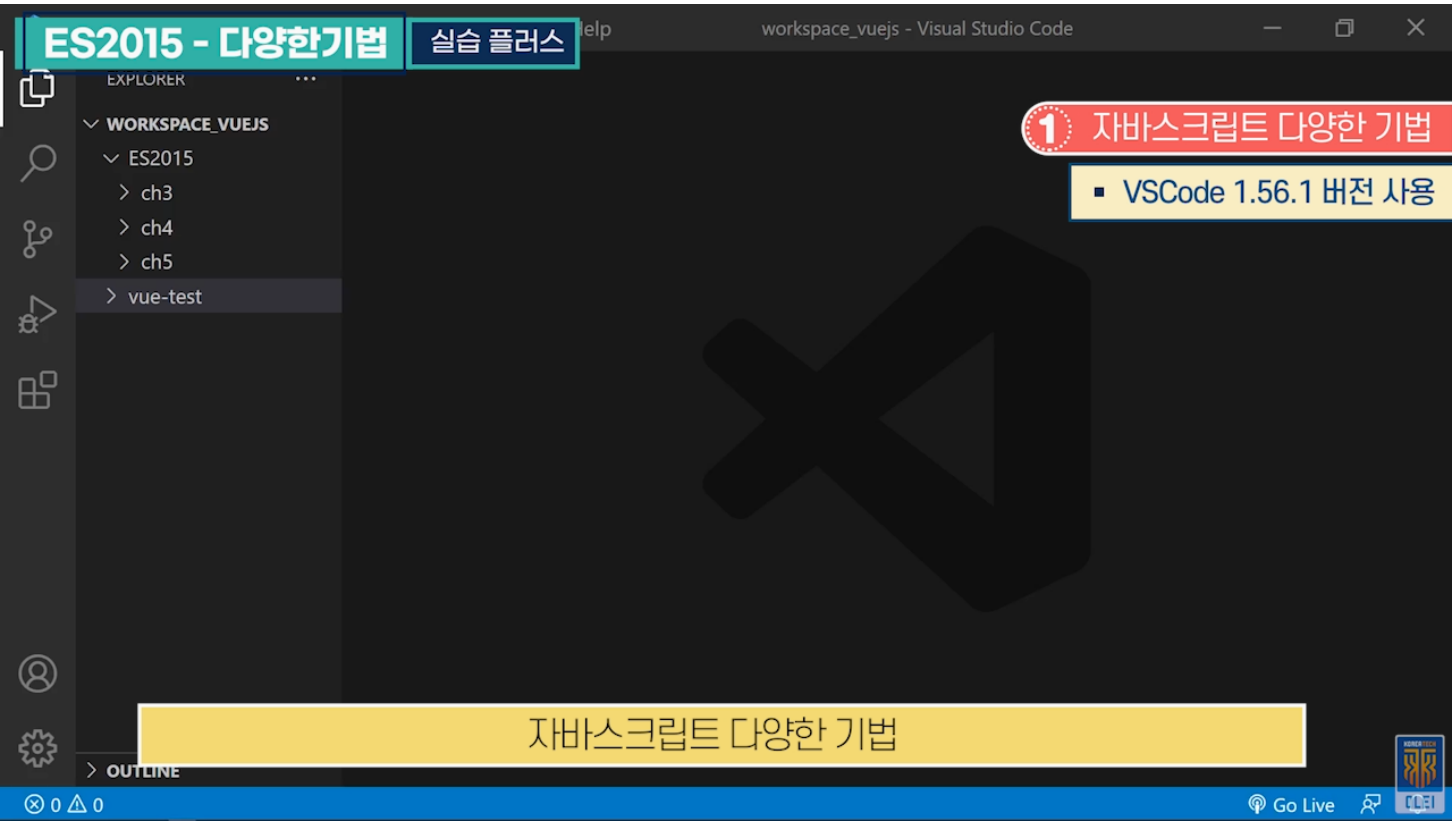
Destructed, Object literal, Modules

4 Modules

- import 방법

1. `import myModule from 'my-module';`
2. `import 'my-module' as myModule;`
3. `var myModule = require('my-module');`

실습 플러스



실습단계
실습을 위해 임의의 폴더를 하나 만들어서 테스트, 폴더명 ch6
New File → 자바스크립트 파일 제작, 파일명 main.js
테스트를 구분하기 위해 console 사용
제일 먼저 스프레드 연산자(Spread operator) 테스트
임의의 배열이 선언됨, 배열을 복제하여 다른 배열을 만들겠다는 가정
array2라는 배열 선언, array1의 모든 배열 데이터를 복제하겠다는 의미
원래 배열 데이터를 하나하나 추출해서 담으면 됨, 배열데이터가 많으면 스프레드 연산자로 한번에 복제 가능
스프레드 연산자로 Object literal의 구성 요소 복제 가능
스프레드 연산자로 한번에 복제 가능
Rest Parameter 테스트, 함수의 매개변수 부분
함수의 매개변수를 정확하게 지정하는 것이 좋음
함수 선언하면서 매개변수 개수가 많거나 예측하기 힘들 때 Rest parameter를 사용하면 유용함
함수를 호출했을 때, 세 번째 매개변수에 여러 개의 값이 들어옴

실습 플러스

실습단계
각각의 값이 대입되었는지 console로 확인
some 함수를 호출하면 x, y에는 두 개의 데이터를 대입하고 나머지 매개변수는 Rest Parameter에 대입
호출하면, 세 개의 값은 Rest Parameter로 되어 있는 매개변수에 대입하게 됨
Destructed assignment 테스트, Object literal이 필요함
여러 개를 추출할 때 유용하게 사용할 수 있는 기법
obj3를 대입하면, obj3의 데이터 중에서 data1 값을 추출하여 a1이라는 변수에 담으라는 것
데이터를 추출하여 a2에 담고, 데이터가 없을 경우 디폴트 값은 20이라는 선언
제대로 데이터가 추출돼서 변수에 들어갔는지 log로 확인
테스트하기 위해 HTML 파일 하나가 필요함, ch6 → 파일명 index.html 생성
HTML을 Live Server로 테스트 진행
검사를 눌러서 console로 확인
Array, Object literal 데이터, Rest Parameter 값, Destructed assignment 데이터 추출 결과 값 확인