프론트엔드 개발자를 위한 Vue.js 시작하기

Vue.js 시작하기

ES2015 기초문법

한국기술교육대학교
온라인평생교육원

학습내용

- Constants & Template Literals
- Scoping
- Arrow Function

학습목표

- ES6의 Constants와 Template Literals에 대해 설명할 수 있다.
- ES6의 Variable scope에 대해 설명할 수 있다.
- ES6의 Arrow Function에 대해 설명할 수 있다.

Constants & Template Literals

Constants & Template Literals

1 Constants

① 상수변수

```
const PI = 3.141593
```

Constants & Template Literals

1 Constants

② 초기값 변경 불가

```
const a1 = "hello";
a1 = 'world'; //-> 에러 (Uncaught TypeError: Assignment to constant variable.)
```

Constants & Template Literals

Constants & Template Literals

1 Constants

③ ES5 스타일 코드

```
(function () {
  var name = 'hello';
  console.log(name); //hello
  name='world'
  console.log(name); //world

  function someFun(){
    console.log("someFun 1......")
  }
  someFun = function(){
    console.log("someFun 2....")
  }
  someFun()//someFun 2....
})();
```

Constants & Template Literals

1 Constants

④ ES6 스타일 코드

```
(function () {
  const name = 'hello';
  console.log(name);
  // name='world' //error
  console.log(name);

  const someFun = function(){
    console.log("someFun 1......")
  }
  // someFun = function(){//error
    // console.log("someFun 2....")
  // }
  someFun()
})();
```

Constants & Template Literals

Constants & Template Literals

2 Template Literals

① 문자열에 동적 데이터 추가

⋯› 큰 따옴표(“ ”)혹은 작은 따옴표(‘ ’)로 묶이는 문자열에
동적 데이터 추가는 + 를 이용하는 것이 기본

```
var name = "홍길동"
var message = "안녕하세요." + name + "님, 만나서 반갑습니다."
console.log(message)
```

Constants & Template Literals

2 Template Literals

② 문자열 내에 동적 데이터를 ${ } 내에 표현

⋯› 문자열이 backtick(` `) 으로 묶여 있어야 함

```
var name = "홍길동"
var message = `안녕하세요. ${name} 님, 만나서 반갑습니다.`
console.log(message)
```

Constants & Template Literals

Constants & Template Literals

2 Template Literals

③ 실행 결과

```
안녕하세요. 홍길동 님, 만나서 반갑습니다.
```

Constants & Template Literals

2 Template Literals

Tip!

backtick(`) 으로 묶이는 문자열 데이터 내의 특수 키 값 유지

```
var order = { amount: 7, product: "Book", price: 100 }
var order_message=
`
product : ${order.product}
amount : ${order.amount}
price : ${order.price}
`
console.log(order_message)
```

Constants & Template Literals

Constants & Template Literals

2 Template Literals

backtick(` `) 으로 묶이는 문자열 데이터 내의 특수 키 값 유지

실행 결과

```
product : Book
amount : 7
price : 100
```

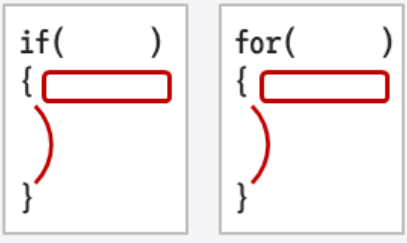
Scoping

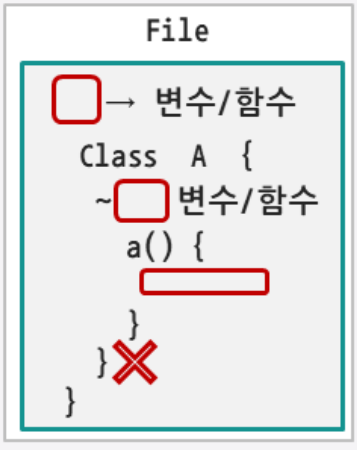
Scoping

1 Scoping의 개념

변수 및 함수의 적용 범위

- ES5에서는 함수단위 Scoping만 지원함
- 블록 단위 Scoping은 지원하지 않음





Scoping

1 Scoping의 개념

ES5 스타일 코드

```
(function () {
  var name = 'hello'
  function a(){
    var name="world"
    console.log(name)//world
  }
  a()
  console.log(name)//hello
  if(true){
    var name="홍길동"
  }
  console.log(name)//홍길동
  {
    var name=" 김길동"
  }
  console.log(name)//김길동
})();
```

Scoping

Scoping

2 let 키워드

ES6에서는 변수 선언을 위한
let 키워드 제공

let 키워드로 선언된 변수는
블록단위 Scoping

Scoping

2 let 키워드

ES6 스타일 코드

```
(function () {
  let name = 'hello'
  function a(){
    let name="world"
    console.log(name)//world
  }
  a()
  console.log(name)//hello
  if(true){
    let name="홍길동"
  }
  console.log(name)//hello
  {
    let name=" 김길동"
  }
  console.log(name)//hello
})();
```

Arrow Function

Arrow Function

1 Arrow Function의 개념

화살표(=>)로 선언하는 함수

ES5 스타일 코드

```
evens.map(function (v) { return v + 1; });
evens.map(function (v) { return { even: v, odd: v + 1 }; });
evens.map(function (v, i) { return v + i; });
```

Arrow Function

1 Arrow Function의 개념

화살표(=>)로 선언하는 함수

ES6 스타일 코드

```
evens.map(v => v + 1)
evens.map(v => ({ even: v, odd: v + 1 }))
evens.map((v, i) => v + i)
```

Arrow Function

Arrow Function

2 Arrow Function 작성규칙

① 매개변수가 없는 Arrow Function

```
var fun1 = () => { console.log('i am fun1') }
```

Arrow Function

2 Arrow Function 작성규칙

② 매개변수를 가지는 Arrow Function

```
var fun2 = x => { console.log(`i am fun2, argument : ${x}`) }
var fun3 = (x, y) => { console.log(`i am fun3, argument : ${x}, ${y}`) }
```

Arrow Function

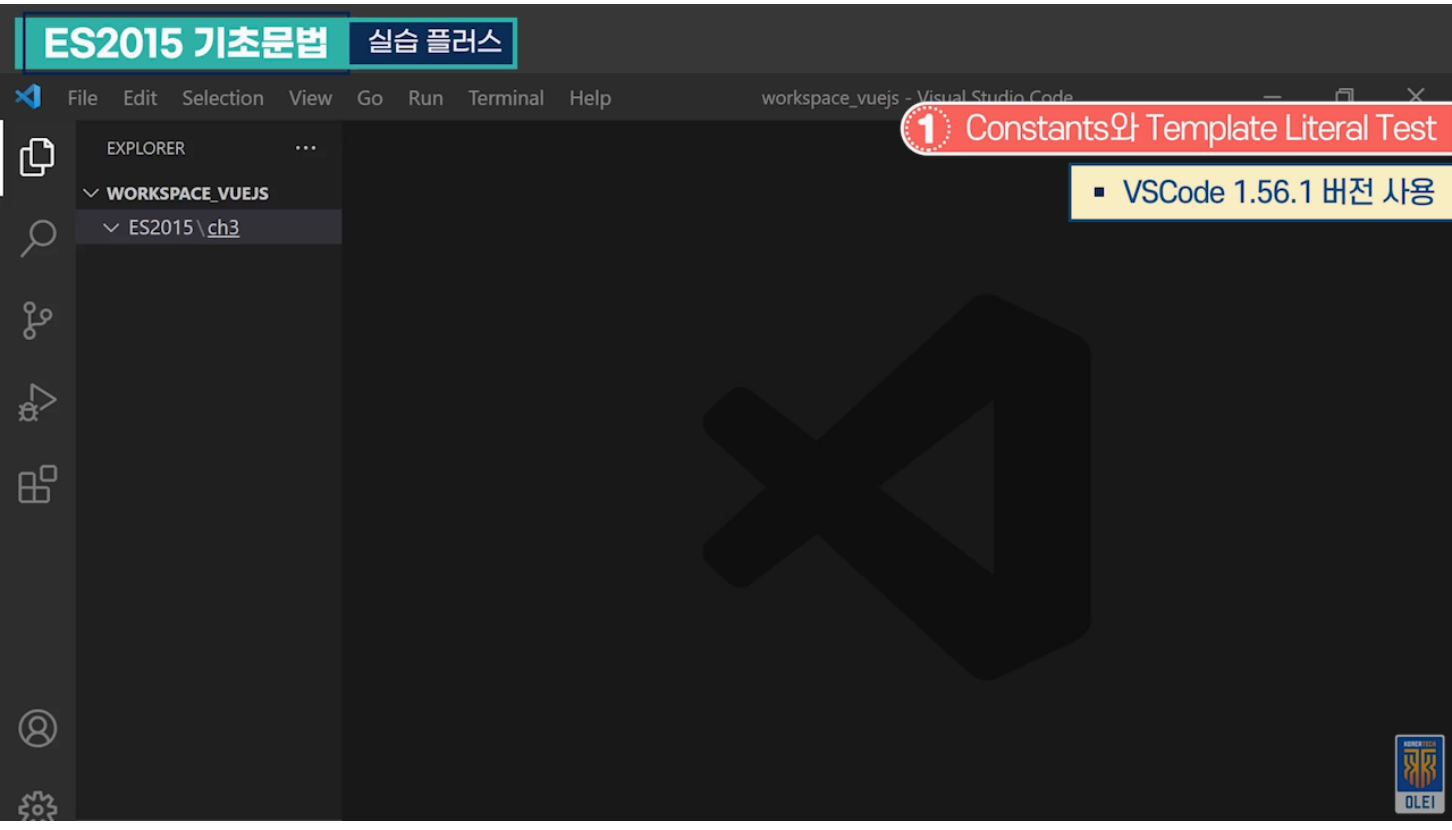
Arrow Function

2 Arrow Function 작성규칙

③ 리턴 값이 있는 Arrow Function

```
var fun4 = x => { return x*x }
var fun5 = x => x*x
var fun6 = x => {
    const y = x*x
    return y
}
```

실습 플러스



실습단계
폴더 만들기 : ES2015 - ch4
ch4 선택 후 New File - main.js 입력
function을 만들고 호출해서 테스트
변수 선언, 값이 입력되어 있다는 가정
변숫값을 다른 문자열에서 포함시켜 출력해야 함
Message 변수를 만들어서 name 변숫값을 뒤에 출력
문자열이 길고 동적데이터가 많을 경우 Template Literal 활용
문자열을 backtick(` `) 으로 표현
Backtick 내 문자열에 동적데이터 표현 부분을 ${} 로 표현
결과값 확인은 console.log(message) 출력, name 변숫값 포함 여부 확인

실습 플러스

main.js

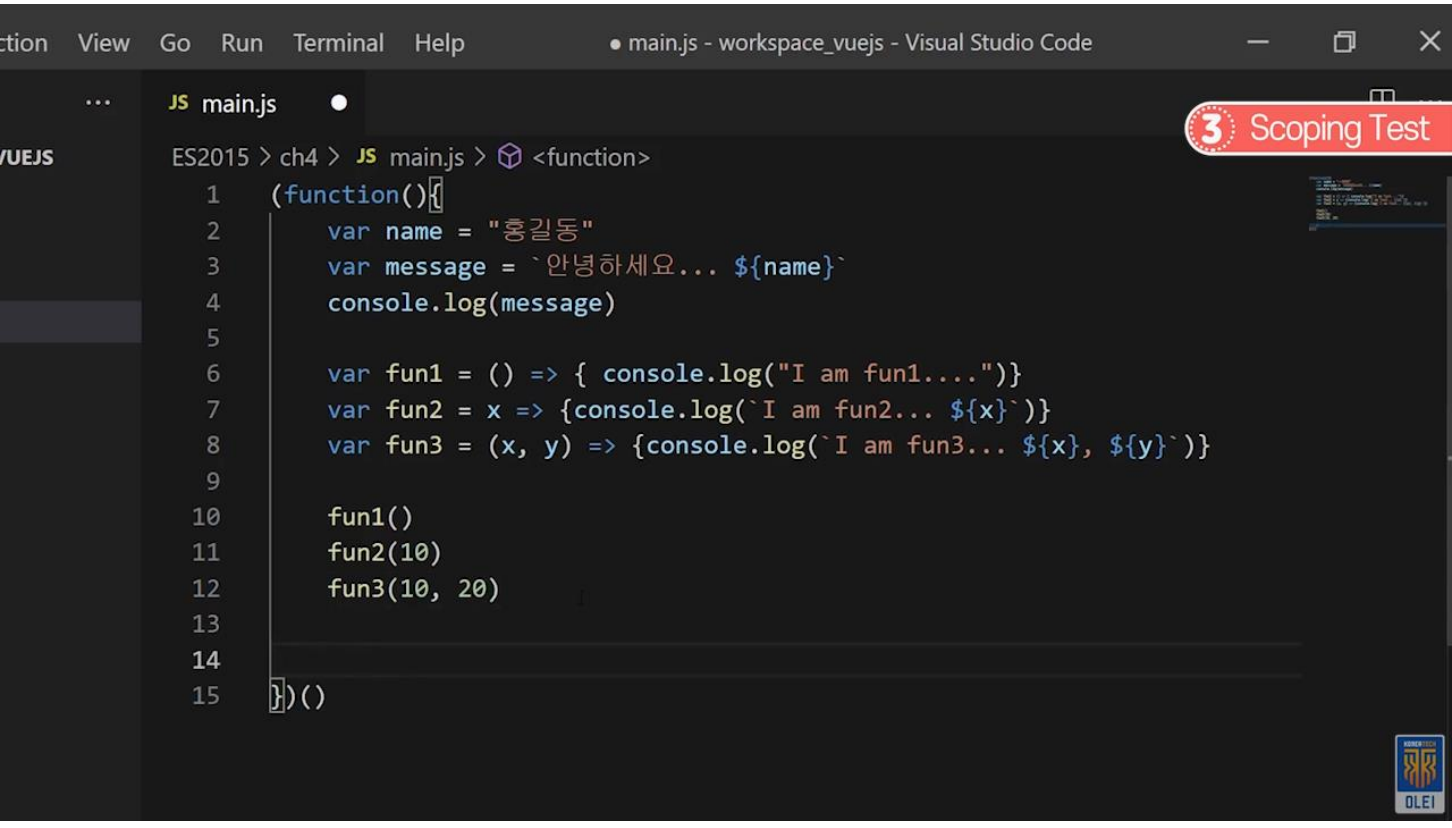
ES2015 > ch4 > main.js > <function>

2 Arrow Function Test

```
1  (function(){
2      var name = "홍길동"
3      var message = `안녕하세요... ${name}`
4      console.log(message)
5
6
7  })()
```

실습단계
간단한 함수는 화살표(=>) 함수로 표현
fun1 변수에 화살표 함수 대입, 함수도 데이터로 취급, 변수에 대입
화살표(=>)를 기준으로 왼쪽이 매개변수, '()'는 매개변수가 없는 함수
함수가 정상적으로 Call 되었는지 확인을 위해 console.log("I am fun1....") 입력
매개변수를 지정하여 다른 함수 선언
매개변수 x(매개변수를 x로 핸들링) console.log(`I am fun2... ${x}`)
매개변수가 여러 개일 경우 (x, y) => {console.log(`I am fun3... ${x}, ${y}`)}
함수 호출 및 매개변수 값 전달

실습 플러스



실습단계
변수 선언을 var로 하는 대신, let으로 선언
테스트를 위해 if문 작성
* Scoping 안 되었을 경우 Outer 변수명과 if문 안의 변수명이 동일하여 Outer 변숫값이 "world" 로 바뀜
* Scoping 될 경우 영역이 달라 if문 내에서만 "world" 값이 유지됨
console.log(data)로 무슨 값이 나오는지 확인
HTML을 만들어야 자바스크립트 테스트 가능
ch4 - New File - index.html
"main.js"를 단순하게 실행
Open with Live Server 클릭
빈 화면에서 오른쪽 마우스 클릭, 검사 - console 작성한 내용 확인