

2018-2

웹 시스템 설계

실습

Week 4

JavaScript (Closure)

Composed by:

WISE Research Lab Ajou University



AJOU UNIVERSITY



Preliminary

■ 실습 목표

- JavaScript 의 Closures 개념을 이용하여 private 변수를 다루는 방법을 배우고, 이를 기반으로 웹 기반의 "Simple ATM"을 만들어본다.

■ Closures 란? (간단히)

- 함수와 그 함수가 선언된 *lexical* 환경과의 조합이다. - MND WEB DOC¹
- 어떤 함수 a() 내부에 정의된 지역 변수는, 함수 a()의 function scope 내에서만 참조가 가능하다.
- 그러나 함수 a() 내부에 또 다른 내부 함수 b()가 정의되어 있다면, 이 내부 함수 b()는 외부 함수 a()의 변수(혹은 scope)에 접근 할 수 있다. 이렇게 함수 b()가 외부 함수 a()의 function scope 를 참조 할 수 있는 것을 lexical scoping 이라 부른다.
- 위와 같은 lexical scoping 을 사용하여, 외부 함수의 변수를 이용한 연산 결과 혹은 변수 값 자체를 반환할 수 있는 내부 함수 혹은 객체를 Closure 라 부른다.
- 외부 함수에서 정의된 변수들은 외부 함수의 function scope 내부에서만 참조할 수 있기 때문에, 외부 함수의 바깥에서는 참조할 수 없다. 그러나 Closure 의 사용을 통해 외부에서 간접적인 참조가 가능해진다.

■ Closure 의 간단한 예제

- 아래는 Counter 를 만드는 Closure 예제코드와 실행 예시이다.
- makeCounter() 함수는 counter 라는 function scope 내의 변수를 갖고, 함수를 return 한다.
- makeCounter()에서 return 되는 함수는 자신이 참조 가능한 lexical scope 내에 있는 counter 라는 변수를 참조하여, 해당 변수를 1 증가시키고 그 값을 return 한다.
- 즉 makeCounter() 함수를 실행하여 return 받은 값을 저장한 counter1, counter2 는 makeCounter()가 실행될 때 생기는 lexical scope 에 존재하는 counter 값을 매번 1 증가시키는 함수를 저장하게 된다.
- counter1 과 counter2 는 서로 다른 lexical scope 를 갖기 때문에 서로 다르게 숫자가 증가하는 모습을 보인다.

¹ 클로저 - MND, "<https://developer.mozilla.org/ko/docs/Web/JavaScript/Guide/Closures>"

메모리 요약도	
Address-1	
Address-2	Lexical scope of counter2 var counter = 0;
Address-3	
Address-4	Lexical scope of counter2 var counter = 0;
...	
Address-N	

```

> function makerCounter() {
    var counter = 0;
    return function () { return counter += 1; }
}

> var counter1 = makerCounter();
             ↓

> counter1()
< 1

> counter1()
< 2

> counter1()
< 3

> var counter2 = makerCounter();
             ↓

> counter2()
< 1

> counter2()
< 2

```

Figure 1 makeCounter Closure 예제와 메모리 요약도

■ Closure 사용시 주의 사항

- 함수가 실행될 때마다 lexical scope 생성, Closure 가 이를 참조하기 위해 메모리에 따로 lexical scope 를 저장해 둔다.
- 이러한 이유로 무분별한 Closure 의 사용은 memory overflow 를 일으킬 가능성이 있다.

■ Private 범위: 객체지향 프로그래밍을 위한 Closure 활용

- lexical scope 를 사용한 변수 참조 방법을 통해, Closure 는 객체지향 프로그래밍의 private 범위를 제공할 수 있다. 예제는 다음과 같다.

```

> function user() { ← user()라는 객체 생성자를 만듦
  var name; ← 변수 name과 age를 Closure의 lexical scope에 선언함으로,
  var age; ← 외부에서 user 객체의 name, age를 직접 참조할 수 없게 함 (private 변수)

  return {
    getName: function () { return name; },
    getAge: function () { return age; }, ← Closure의 lexical scope에 있는 변수들을
    setAge : function (newAge) { age = newAge; } ← 참조하기 위한 함수들을 선언
  }
}
var u1 = user();
u1.setAge(12);

> u1.getAge() ← Closure에 선언된 getAge() 함수를 통해서 age 값을 참조할 수 있음
< 12
> u1.age; ← 변수 age를 직접 참조할 수 없어, undefined 출력
< undefined

```

Figure 2 Closure 기반 private 변수 활용 예시

- user()에서 선언한 name 과 age 는 user() function 밖에서 직접 참조할 수는 없지만, user() function 이 return 하는 객체 안의 함수들을 통해서 간접적으로 참조 수정할 수 있다.

■ Closure 참고

<https://developer.mozilla.org/ko/docs/Web/JavaScript/Guide/Closures>

Lab Assignment

■ 실습과제 목표

- JavaScript 의 Closure 를 학습한다
- Closure 를 통해 private 범위를 다루는 실습을 진행한다.
- DOM API 를 이용해서 Element 의 스타일을 변경하는 간단한 실습을 진행한다.

■ 구현 과제

- Closure 를 통해서 가상 계좌를 생성하고 입출금이 가능한 ATM 을 만들어본다.
- ATM 은 다음과 같은 기능이 있다.
 - ✓ 사용자의 이름과 비밀번호를 받아와 “MAKE NEW ACCOUNT” 버튼을 누르면, 새로운 계좌를 만든다.
 - ✓ 금액을 입력하고 “입금” 버튼을 누르면, 금액만큼 돈이 입금된다. 이후 잔고를 보여준다.
 - ✓ 비밀번호와 금액을 입력하고 “출금” 버튼을 누르면, 비밀번호가 맞는지 확인 후 금액에 맞게 출금한다. 이후 잔고를 보여준다.
 - ✓ 비밀번호를 입력하고 “잔고 확인” 버튼을 누르면, 잔고를 보여준다.

1) HTML 페이지 구현

- 만들어진 css 파일이 html 페이지에 load 되도록 <head> 태그 내에 <link> 태그를 작성한다. 예) <link rel="stylesheet" href="lab4.css">
- 다음은 HTML 의 전체적인 구조를 트리로 표현한 것이다.
트리와 같은 구조로 <body> 내에 태그 요소를 넣고, 지정된 id, class 이외는 자유롭게 id 와 class 속성값을 지정한다.

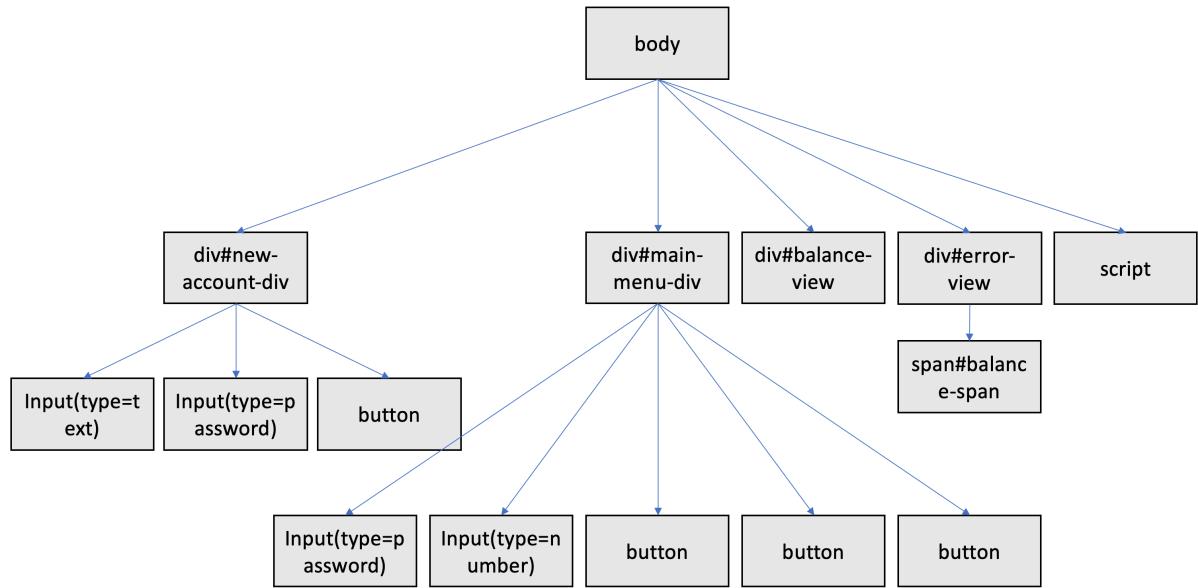


Figure 3 실습 과제 DOM tree 구조

- <body> 태그 내 가장 마지막에 <script> 태그로 js 파일을 불러온다.
 - ✓ <head>가 아니라 <body> 내 가장 마지막에 넣는 이유는 DOM load 가 완료된 후에 script 를 불러와서 element 들을 저장하는 변수들의 값이 null 이 되지 않도록 하기 위해서이다
- 브라우저에서 html 을 열면 다음과 같은 화면이 보이도록 한다.

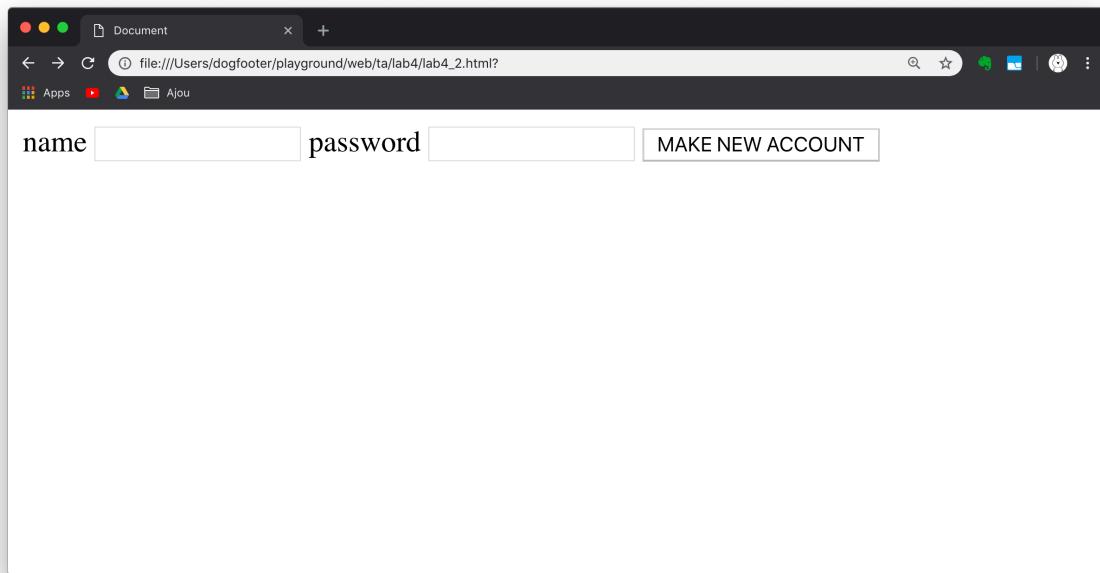


Figure 4 첫 화면

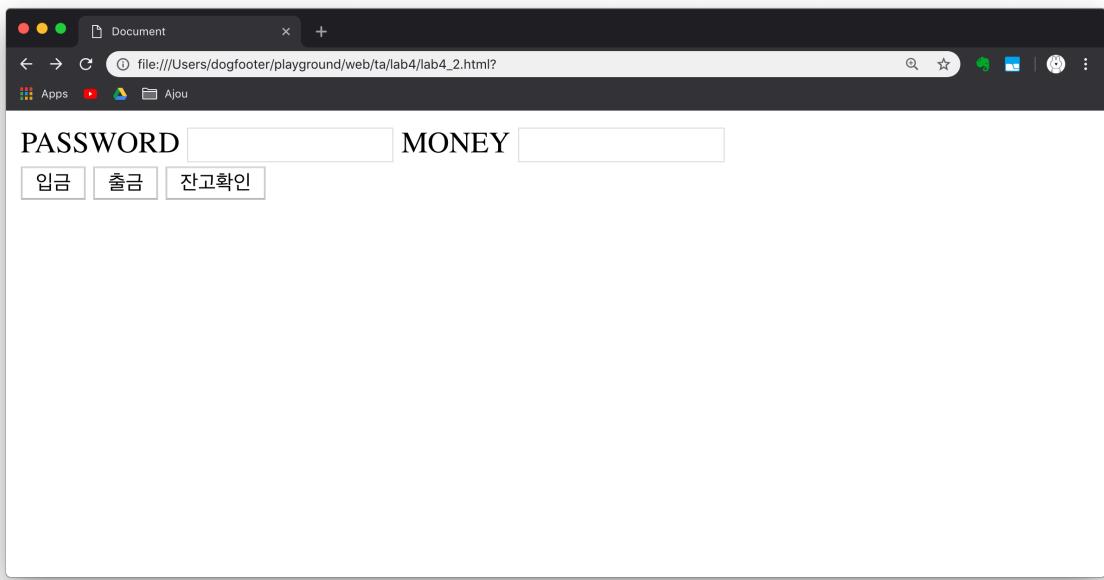


Figure 5 계좌 생성 후 화면

2) CSS 및 JS 구현

a) 이번 실습을 위해 필요한 계좌 객체를 만들기 위해서는 Closure 개념을 사용해야 한다.
다음은 계좌 객체에 관한 내용이다.

- Private 한 변수로 name 과 password, balance 를 갖는다.
 - ✓ name
 - 계좌 주인의 이름을 나타내며, 객체가 생성될 때 생성자의 파라미터로 입력 받아야 한다.
 - ✓ password
 - 계좌의 비밀번호를 나타내며, 객체가 생성될 때 생성자의 파라미터로 입력 받아야 한다.
 - ✓ Balance
 - 계좌의 잔고를 나타내며, 객체가 생성될 때 값을 0 으로 초기화 한다.
 - 이번 실습에서 금액은 정수 타입이다.
- 계좌 객체에는 입금 출금, 잔고 확인을 위한 함수들이 존재한다.
 - ✓ deposit
 - 금액을 입력받아, balance 에 금액을 추가한다. 이후 변경된 balance 를 반환한다.

- 금액은 양수의 정수만 입력받는다. 예외 상황 발생 시, 오류를 사용자에게 알린다.
- ✓ withdraw
 - 계좌 비밀번호와 금액을 입력받아, balance에서 금액만큼 차감한다. 이후 변경된 balance를 반환한다.
 - 입력받은 금액이 balance 보다 크거나 양의 정수가 아닐 시, 오류를 사용자에게 알린다.
- ✓ getBalance
 - 비밀번호를 입력받아 현재 balance를 반환한다.
 - ✓ 비밀번호를 입력받는 함수의 경우 비밀번호가 올바르지 못할 시 오류를 사용자에게 알린다.

b) lab4.js 파일 내에 다음 화면과 같이 각 버튼을 누르면 다음과 같이 동작하도록 함수를 구현한다.

- MAKE NEW ACCOUNT 버튼
 - ✓ 버튼을 누르면 내가 입력한 이름과 비밀번호 값을 받아와, 계좌를 생성하고 메인 메뉴(div#main-menu-div)를 보여준다.
 - 생성된 계좌 객체는 lab4.js의 글로벌 변수로 저장한다.
 - 메인 메뉴를 보여줄 때, div#new-account-div는 화면에서 사라지도록 한다.
- 입금 버튼
 - ✓ 버튼을 누르면 사용자가 입력한 금액을 받아와, 계좌 객체의 deposit 함수를 실행한다. 입금에 성공 시, 변경된 balance 값을 화면에 보여준다.
 - Balance 값을 보여 줄 때, div#balance-view를 사용한다.
 - 입금 실패 시 반환된 오류를 화면에 보여준다. 이 때, div#error-view를 사용한다.
- 출금 버튼
 - ✓ 버튼을 누르면 사용자가 입력한 금액과 비밀번호를 받아와, 계좌 객체의 withdraw 함수를 실행한다. 출금에 성공 시, 변경된 balance 값을 화면에 보여준다.
 - Balance 값을 보여 줄 때, div#balance-view를 사용한다.
 - 출금 실패 시 반환된 오류를 화면에 보여준다. 이 때, div#error-view를 사용한다.
- 잔고 확인 버튼
 - ✓ 버튼을 누르면 사용자가 입력한 비밀번호를 받아와, 현재 가진 balance를 화면에 보여준다.
 - Balance 값을 보여 줄 때, div#balance-view를 사용한다.

- 잔고 확인 실패 시, 오류를 화면에 보여준다. 이 때, div#error-view 를 사용한다.
 - 모든 버튼 클릭 이벤트 실행 시, 기존에 나와 있던 오류나 input 입력값들을 지운다.
 - ✓ 오류를 나타내는 element 를 화면에 보이지 않게 하라는 뜻
 - 위에 언급한 버튼 이벤트로 인한 사라지고 표시되는 element 는 모두 구현되어야 한다.
- c) lab4.css 는 다음의 항목을 따라서 구현한다.
- 제일 처음 html 파일을 브라우저에서 실행할 때, 화면에 보여주는 element 는 div#new-account-div 뿐이다.
 - Balance 를 표시하는 div 에서 금액을 나타내는 글자에 crimson 색을 넣는다.
 - ✓ 혹은 자유로운 색상 가능 (블랙 제외).
 - 기타 언급하지 않은 CSS 는 자유롭게 설정할 수 있다. (언급하지 않은 style 은 꼭 적용할 필요 없음).

3) 실습 추가 자료: DOM API 를 이용하여 Element Style 변경

- DOM API 를 통해서 노드들의 값이나 attribute 를 수정할 수 있는 것뿐만 아니라, 노드들의 스타일 값도 직접 수정이 가능하다. 이를 통해 동적인 이벤트에 따라서 변하는 스타일 값을 적용할 수 있다.
- DOM API 에서 노드의 스타일을 변경하는 코드는 다음과 같다.

```
1. document.getElementById("ID-of-DOM-you-want").style.color = "blue";
2. document.getElementById("ID-of-DOM-you-want").style.fontSize = "14px";
3. document.getElementById("ID-of-DOM-you-want").style.display = "none";
```

- Element 내의 style object property 를 참조하여, style object 내의 원하는 CSS style property 를 수정하면 된다.
- 기존의 hyphen ("")이 들어간 style property 의 경우 camelCase 로 바꾸어 사용하면 된다.
 - ✓ ex) font-size → fontSize
- (참고) CSS style property 중, display 는 browser 가 DOM tree 를 rendering 할 때 특정 element 를 rendering 에 포함 시킬지 제외할지, 포함한다면 어떤 레이아웃으로 구성할지 나타내는 property 이다.

■ 제출양식

- 프로젝트 폴더명은 Lab4-{학번}으로 하며
 - Ex) Lab4-201820000

- 프로젝트 내의 html, js, css 파일명은
 - lab4.html
 - lab4.js
 - lab4.css
- 제출하는 압축파일의 이름은 Lab4-{학번}.zip 으로 한다
 - Ex) Lab4-2018200000.zip
- 명시된 파일명, 폴더명을 반드시 지키기 바랍니다. (감점요인)

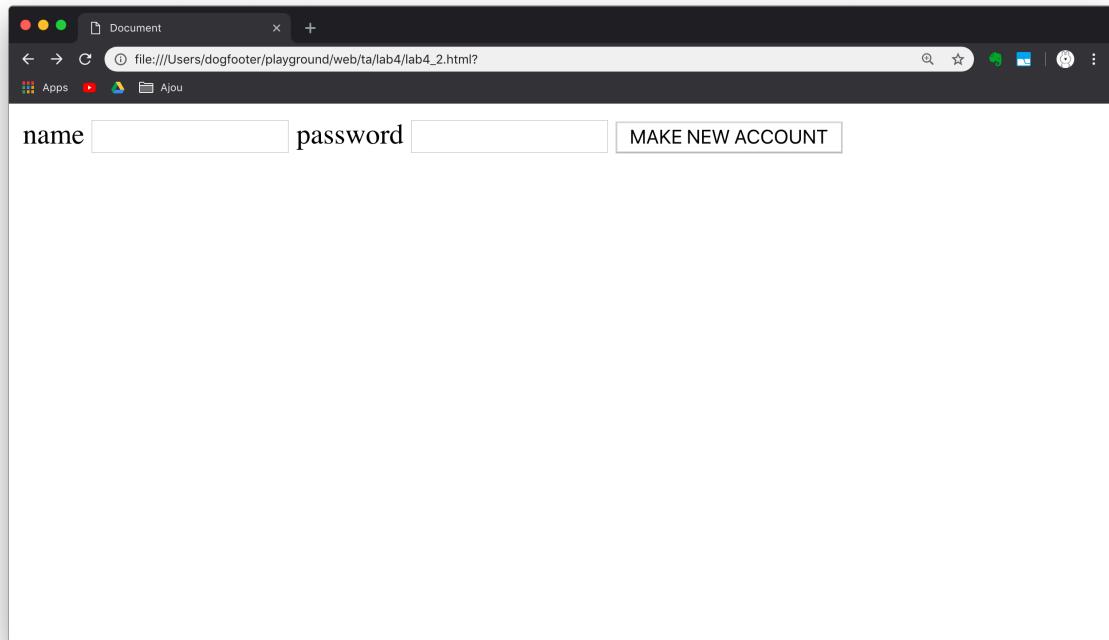
■ 제출기한

- 금요일 5:59 am 까지 제출: 100% 점수인정
- 금요일 5:59 pm 까지 제출: 70% 점수인정
- 그 이후 제출 시 0%. (안 봄)

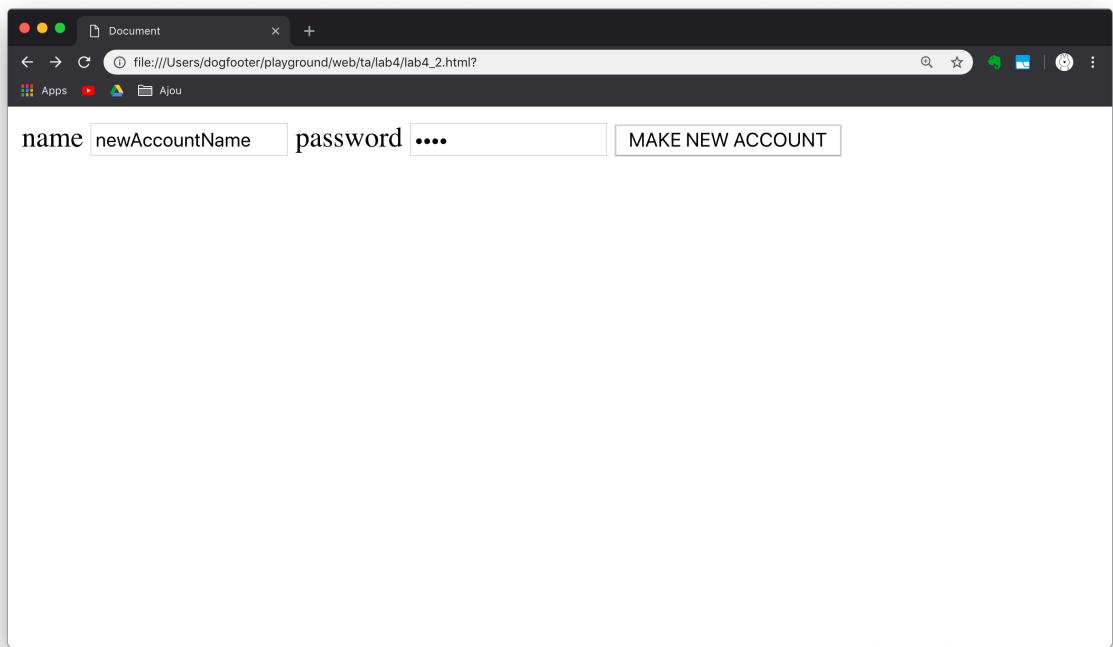
■ 주의사항

- 모든 실습과제는 본인이 수행한 결과물만 제출
- 처음 Copy 적발 시 해당 실습과제 0점 처리
- Copy 재적발 시 실습과제 전체점수 0점 처리

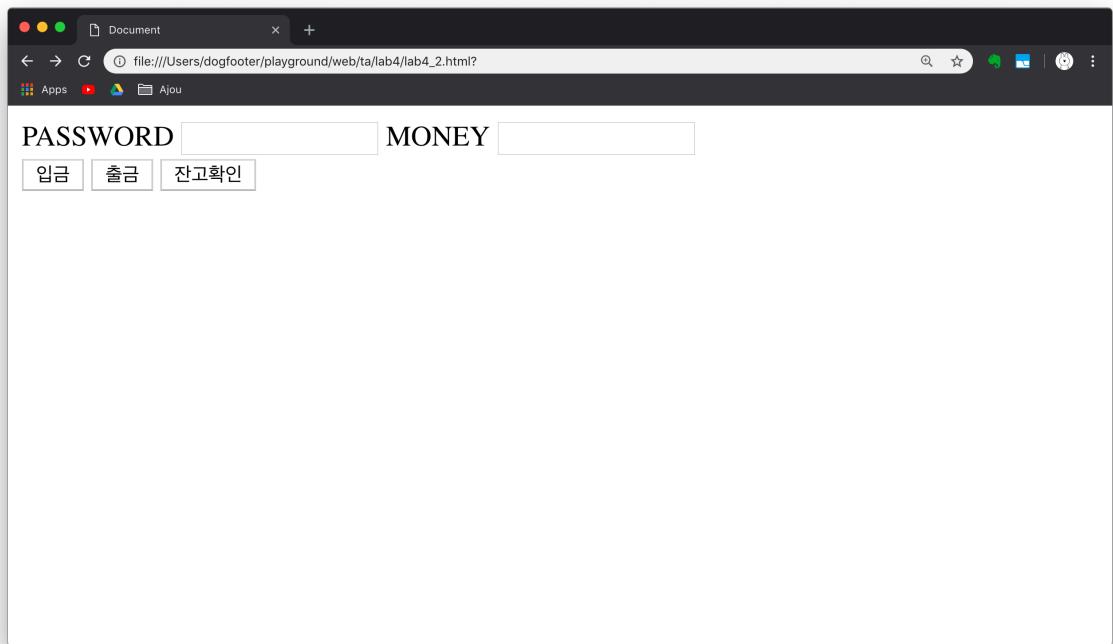
■ 실행 예시



- 새로운 계좌를 생성하는 화면



- 정보를 입력하여 MAKE NEW ACCOUNT 버튼을 누른다



- 계좌를 생성하고 나오는 main menu 의 모습

A screenshot of a web browser window titled "Document". The address bar shows the URL "file:///Users/dogfooter/playground/web/ta/lab4/lab4_2.html?". The page contains two input fields labeled "PASSWORD" and "MONEY". Below these fields are three buttons: "입금" (Deposit), "출금" (Withdrawal), and "잔고확인" (Check Balance). The "잔고확인" button is highlighted with a red box. Below the buttons, the text "잔고 : 0 원" (Balance : 0 Won) is displayed.

- 비밀번호를 입력하고 잔고확인 버튼을 누른 화면

A screenshot of a web browser window titled "Document". The address bar shows the URL "file:///Users/dogfooter/playground/web/ta/lab4/lab4_2.html?". The page contains two input fields labeled "PASSWORD" and "MONEY". Below these fields are three buttons: "입금" (Deposit), "출금" (Withdrawal), and "잔고확인" (Check Balance). The "입금" button is highlighted with a red box. Below the buttons, the text "잔고 : 500000 원" (Balance : 500000 Won) is displayed.

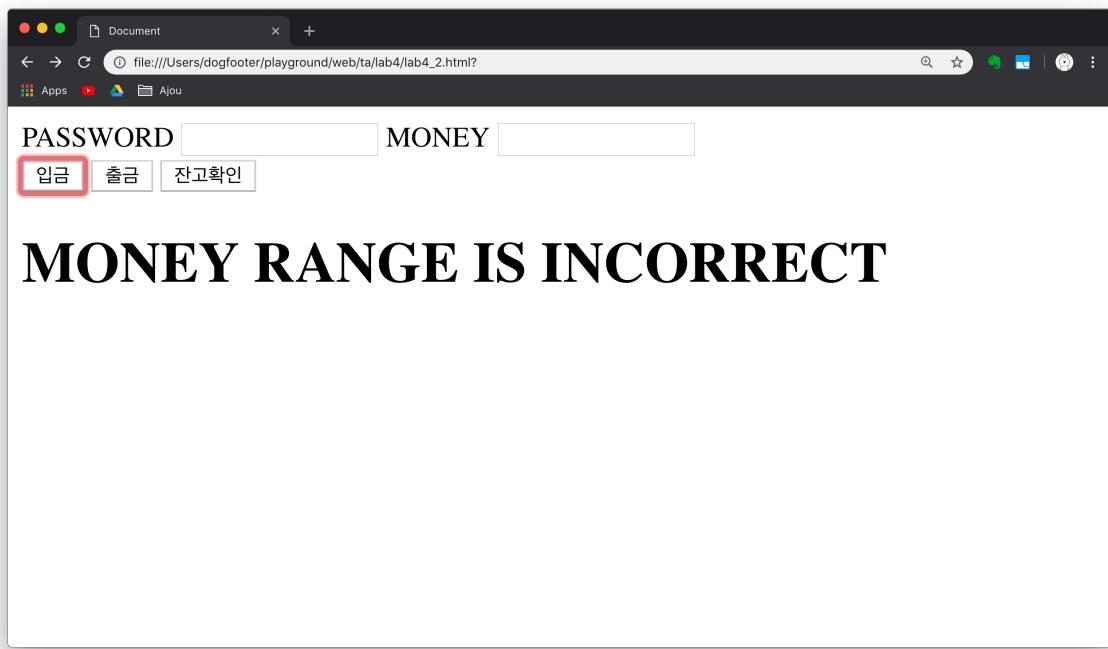
- 금액을 입력하고 입금 버튼을 누른 화면

A screenshot of a web browser window titled "Document". The address bar shows the URL "file:///Users/dogfooter/playground/web/ta/lab4/lab4_2.html?". The page contains two input fields: "PASSWORD" and "MONEY". Below the fields are three buttons: "입금" (Deposit), "출금" (Withdrawal), and "잔고확인" (Check Balance). The "출금" button is highlighted with a red border. The text "잔고 : 496000 원" (Balance : 496,000 won) is displayed in large red font below the buttons.

- 금액과 비밀번호를 입력하고 출금 버튼을 누른 화면

A screenshot of a web browser window titled "Document". The address bar shows the URL "file:///Users/dogfooter/playground/web/ta/lab4/lab4_2.html?". The page contains two input fields: "PASSWORD" and "MONEY". Below the fields are three buttons: "입금" (Deposit), "출금" (Withdrawal), and "잔고확인" (Check Balance). The "출금" button is highlighted with a red border. The text "PASSWORD IS INCORRECT" is displayed in large black font below the buttons.

- 비밀번호 오류가 발생한 화면



- 입력 금액 오류가 발생한 화면