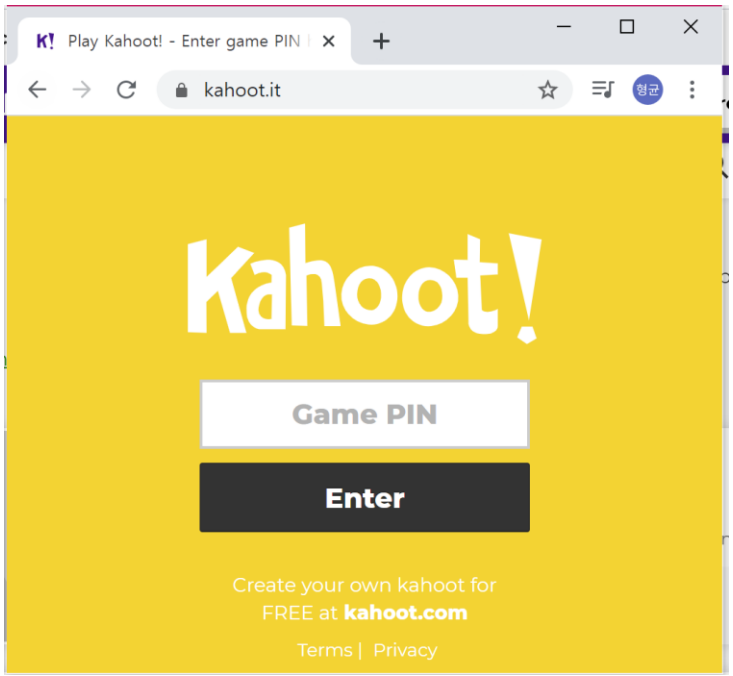
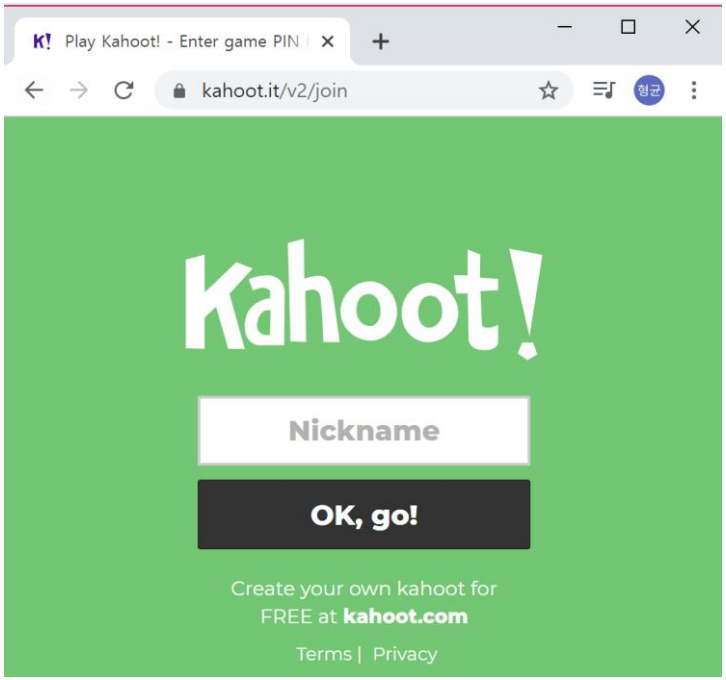
6장 복습문제 풀이

1

- 본인 컴퓨터에서 Kahoot.it 사이트 접속후 라이브화면에서 제시되는 pin번호를 입력
- 닉네임은 본인의 학번으로 반드시 입력
  - 퀴즈 점수에 반영 예정(상위 30% : 2점, 하위 70% 1점, 미참여 : 0점)
- 라이브화면에서 제시되는 문제를 보고 본인 컴퓨터에서 정답 선택
- 최대한 빠르게 선택해야 높은 점수



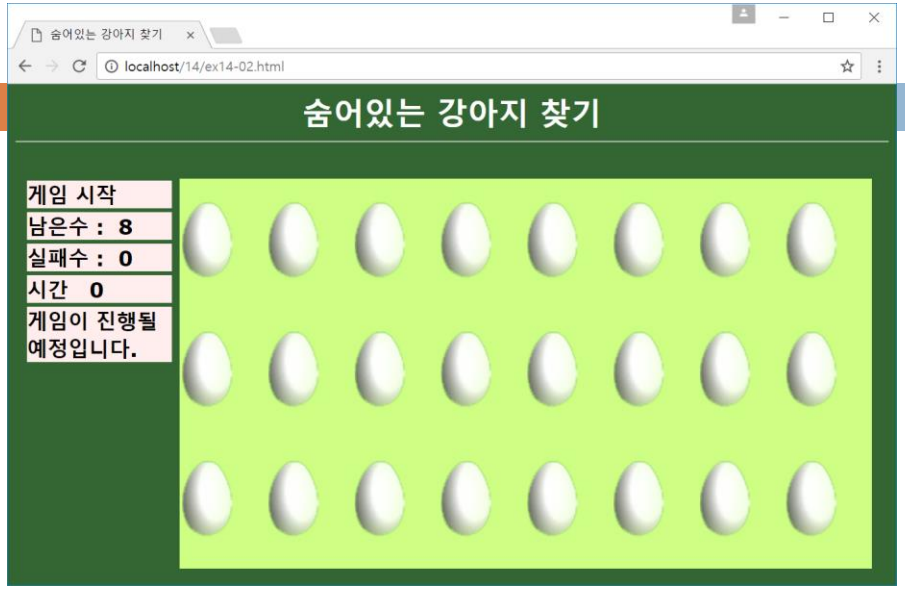


2020학년도 1학기 학사운영 관련 변경 사항 안내

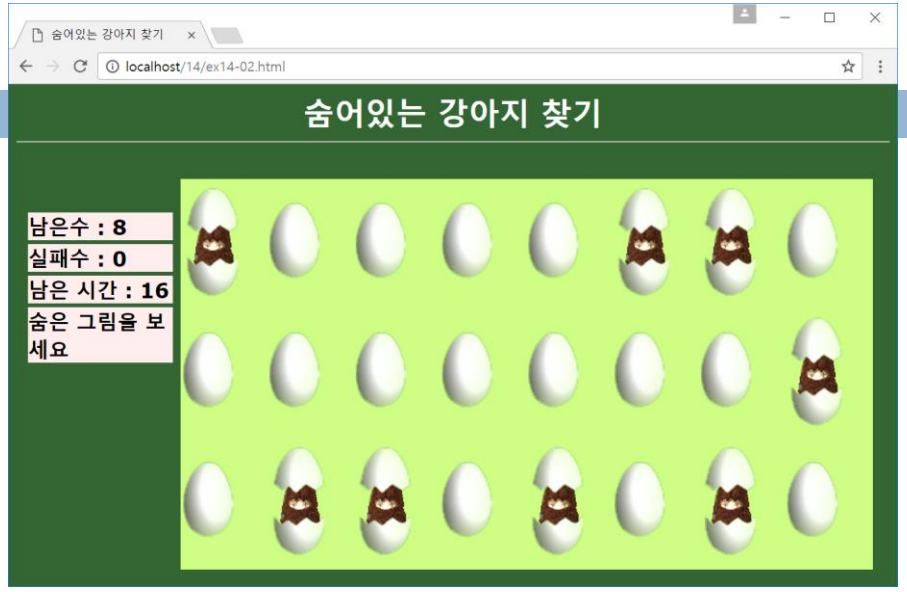
- 2020년 1학기 학사일정
  - 학기말까지 온라인 수업 기조 유지
- 성적평가: 2020학년도 1학기 모든 수업 절대평가 허용
- 중간고사 실시하지 않음
- 기말고사 : 대면시험 시행 (6월 25일 목요일 예정)

시험			수행과제			참여	
중간고사	기말고사	퀴즈	프로젝트	과제물	발표	출석	수업 참여도
0%	50%	20%	10%	10%	%	10%	0%

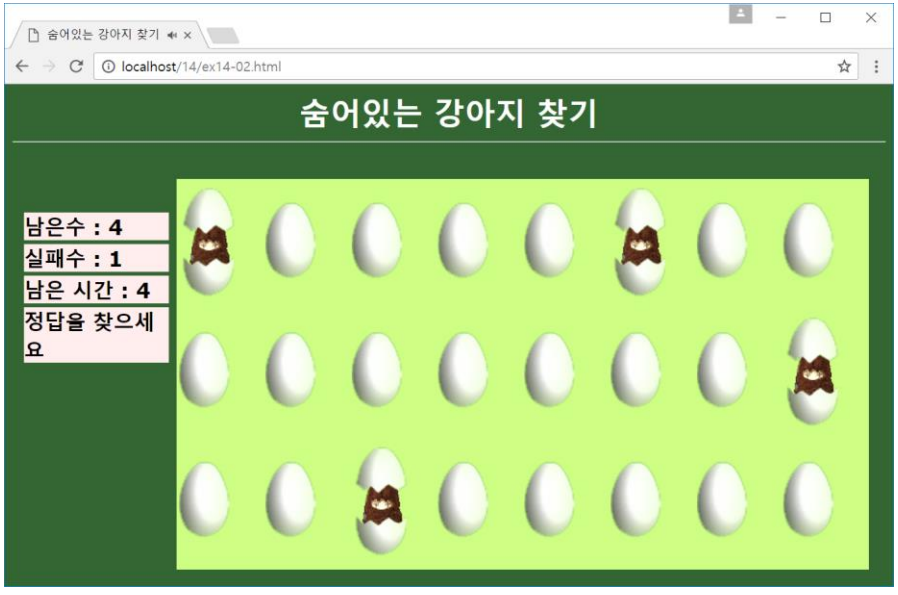
프로젝트 "숨어 있는 강아지 찾기"



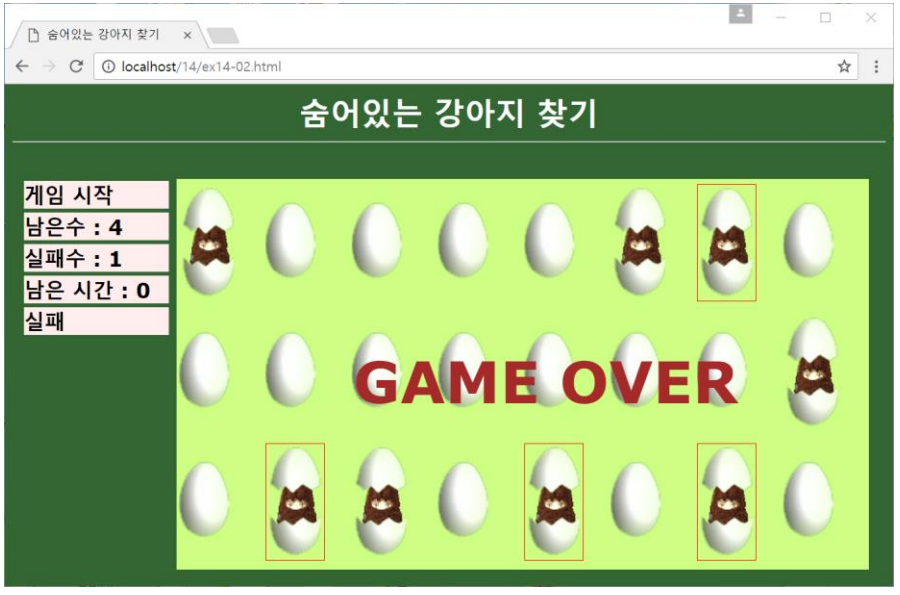
초기 화면



게임 시작을 눌러 숨은 그림을 보여주는 화면



마우스를 클릭하여 숨은 강아지를 찾고 있는 모습



제한 시간 동안 찾지 못해 실패한 경우. 숨은 강아지 보여줌

프로젝트 "숨어 있는 강아지 찾기"

- 작성요령
  - 앞서 설명한 기본 상황을 포함하며 업그레이드 한 게임을 제작
  - 기본 이미지 제공예정
  - 보고서에 게임 기획의도와 업그레이드 상황 설명
  - 보고서에는 소스 코드의 설명과 상황별 실행화면을 포함한다
  - 소스코드는 : html, css, js 포함
- 과제 제출물
  - 프로젝트 보고서 : ppt파일 1개
  - 프로그램 소스코드 및 이미지 압축파일 1개
- 제출일
  - 제출기한 : 6월 21일(일) 자정까지 E-CAMPUS 업로드

07

자바스크립트 코어 객체와 배열

강의 목표

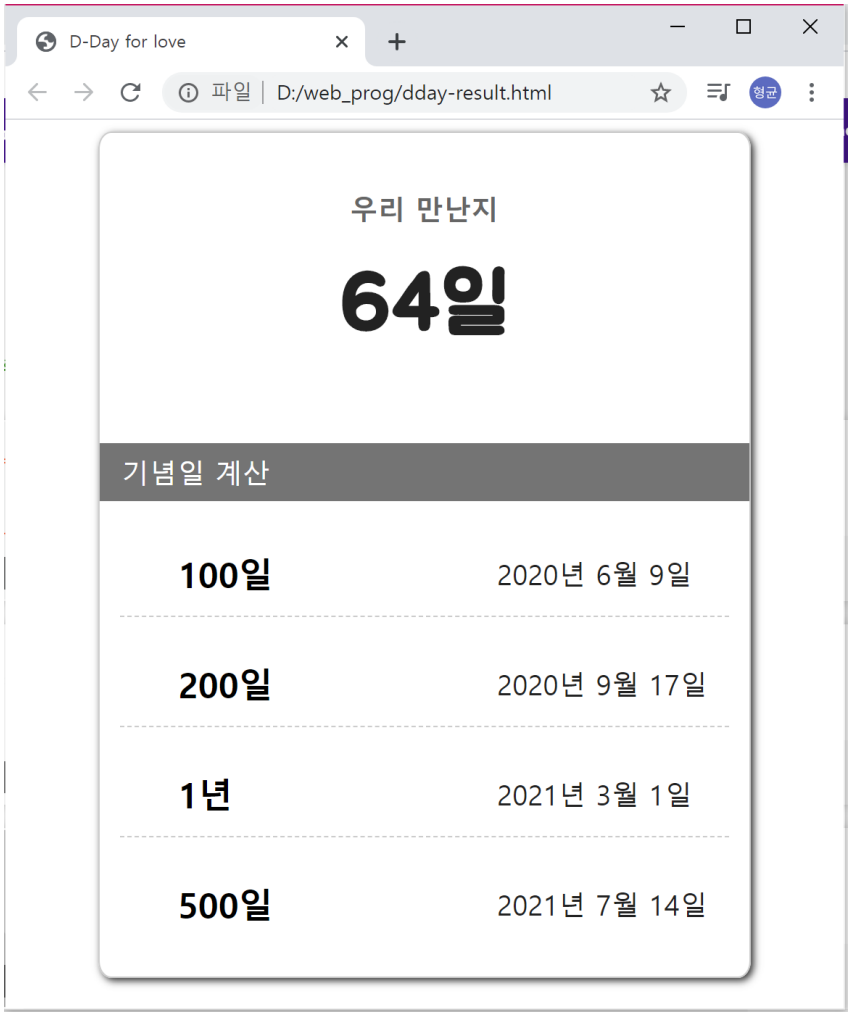
1. 객체의 기본 개념을 간단히 이해한다.
2. 브라우저가 제공하는 기본 객체(코어 객체)들의 종류를 알고 사용할 수 있다.
3. Date 객체를 활용할 수 있다.
4. String 객체를 활용할 수 있다.
5. 자바스크립트 배열을 만들 수 있다.
6. Array 객체를 이용하여 배열을 만들고 활용할 수 있다.
7. Math 객체를 활용할 수 있다.

기념일 계산 프로그램 만들기

- E-CAMPUS 자료실 다운로드
  - dday-result.html, d-day.css
  - 자바스크립파일 작성

CSS 선택자를 이용하여 DOM을 조작하기

```
document.querySelector('#accent').innerText
  = passedDay + "일";
```



```html
<body>
    <div class="container">
        <div class="day1">
            <h3>우리 만난지</h3>
            <p id="accent" class="accent">
                <span style="font-size:0.6em; font-style:italic">며칠?
                </span></p>
        </div>
        <div class="bar">기념일 계산</div>
        <div class="day2">
            <ul>
                <li class="item-title">100일</li>
                <li class="item-date" id="date100"></li>
            </ul>
            <ul>
                <li class="item-title">200일</li>
                <li class="item-date" id="date200"></li>
            </ul>
            <ul>
                <li class="item-title">1년</li>
                <li class="item-date" id="date365"></li>
            </ul>
            <ul>
                <li class="item-title">500일</li>
                <li class="item-date" id="date500"></li>
            </ul>
        </div>
    </div>
    <script src="dday-result.js"></script>
</body>
```

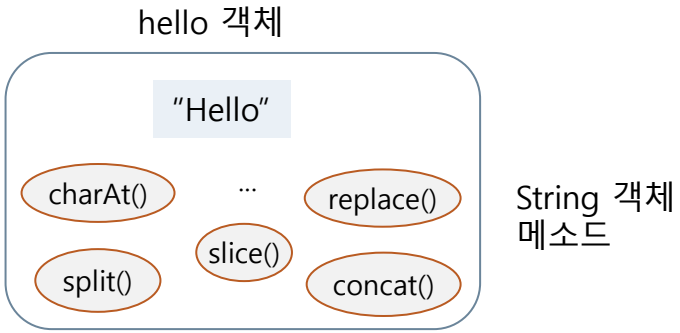
```
var now = new Date();
// 오늘 날짜 정보를 Date 객체의 인스턴스 now 객체로 만듭니다.
var firstDay = new Date("2020-03-01");
// 처음 만난 날의 날짜 정보를 firstDay 객체로 만듭니다.
var toNow = now.getTime();
// 오늘 날짜를 밀리초로 바꿉니다.
var toFirst = firstDay.getTime();
// 처음 만난 날을 밀리초로 바꿉니다.
var passedTime = toNow - toFirst;
// 처음 만난 날과 오늘 사이의 차이 (밀리초)
var passedDay = Math.round(passedTime/(24*60*60*1000));
// 밀리초를 일로 변환 후 반올림합니다.
document.querySelector('#accent').innerText = passedDay + "일";
// #accent 영역에 표시합니다.


function calcDate(days) { ···
}
```

solution

String 객체

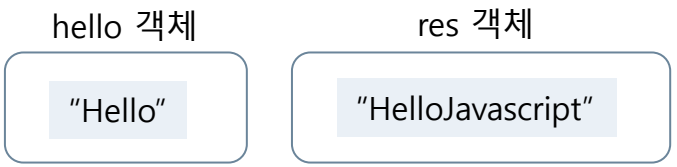
- String
  - 문자열을 담기 위한 객체

```
// 2 경우 모두 오른쪽 String 객체 생성

var hello = new String("Hello");
var hello = "Hello";
```



  - String 객체는 일단 생성되면 수정 불가능

```
var hello = new String("Hello");
var res = hello.concat("Javascript");

// concat() 후 hello의 문자열 변화 없음
```



String 객체의 특징

- 문자열 길이
  - String 객체의 length 프로퍼티 : 읽기 전용

```
var hello = new String("Hello");
var every = "Boy and Girl";
var m = hello.length;              // m은 5
var n = every.length;              // n은 12
```

```
var n = "Thank you".length;    // n은 9
```

- 문자열을 배열처럼 사용
  - [] 연산자를 사용하여 각 문자 접근

```
var hello = new String("Hello");
var c = hello[0];      // c = "H". 문자 H가 아니라 문자열 "H"
```

String 객체의 메소드 활용

메소드와 사용법	하는 일
charAt(index)	지정된 위치에서 문자 찾기
indexOf(string)	지정된 문자의 위치를 왼쪽부터 찾기
lastIndexOf(string)	지정된 문자의 위치를 오른쪽부터 찾기
substring(index1, index2)	지정된 위치에 있는 문자열 리턴
toLowerCase()	소문자로 변환하기
toUpperCase()	대문자로 변환하기
concat(string)	두 문자열을 합치기
slice(start_index, end_index)	문자열의 일부를 추출하기
split([분리자])	문자열을 분리하기
substr(start_index, length)	문자열을 length만큼 잘라내기
charCodeAt([index])	문자열의 ISO Latin-1 값 알아내기
fromCharCode("n1", ..., "nn")	ISO Latin-1 값의 문자열 알아내기

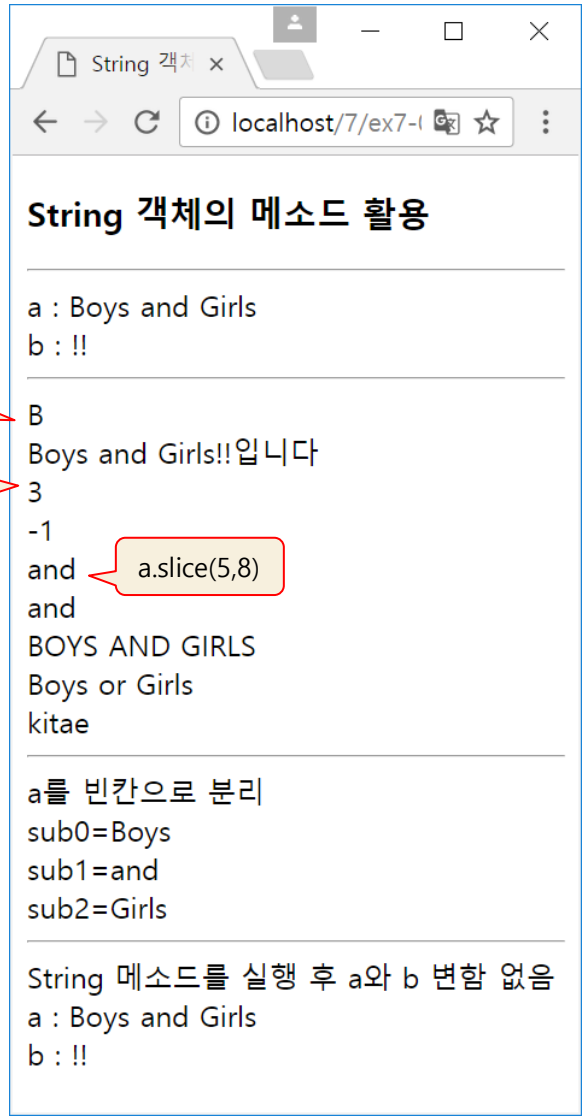
예제 7–7 String 객체의 메소드 활용


```
<!DOCTYPE html>
<html><head><title>String 객체의 메소드 활용</title></head>
<body>
<h3>String 객체의 메소드 활용</h3>
<hr>
<script>
var a = new String("Boys and Girls");
var b = "!!";
document.write("a : " + a + "<br>");
document.write("b : " + b + "<br><hr>");

document.write(a.charAt(0) + "<br>");
document.write(a.concat(b, "입니다") + "<br>");
document.write(a.indexOf("s") + "<br>");
document.write(a.indexOf("And") + "<br>");
document.write(a.slice(5, 8) + "<br>");
document.write(a.substr(5, 3) + "<br>");
document.write(a.toUpperCase() + "<br>");
document.write(a.replace("and", "or") + "<br>");
document.write("    kitae    ".trim() + "<br><hr>");

var sub = a.split(" ");
document.write("a를 빈칸으로 분리<br>");
for(var i=0; i<sub.length; i++)
    document.write("sub" + i + "=" + sub[i] + "<br>");

document.write("<hr>String 메소드를 실행 후 a와 b 변함 없음<br>");
document.write("a : " + a + "<br>");
document.write("b : " + b + "<br>");
</script>
</body>
</html>
```

a.charAt(0)

a.indexOf("s")

a.slice(5,8)



Math 객체

- Math
  - 수학 계산을 위한 프로퍼티와 메소드 제공
  - new Math()로 객체 생성하지 않고 사용

```
var sq = Math.sqrt(4);          // 4의 제곱근을 구하면 2
var area = Math.PI*2*2;         // 반지름이 2인 원의 면적
```

  - 난수 발생
    - Math.random() : 0~1보다 작은 랜덤한 실수 리턴
    - Math.floor(m)은 m의 소수점 이하를 제거한 정수 리턴

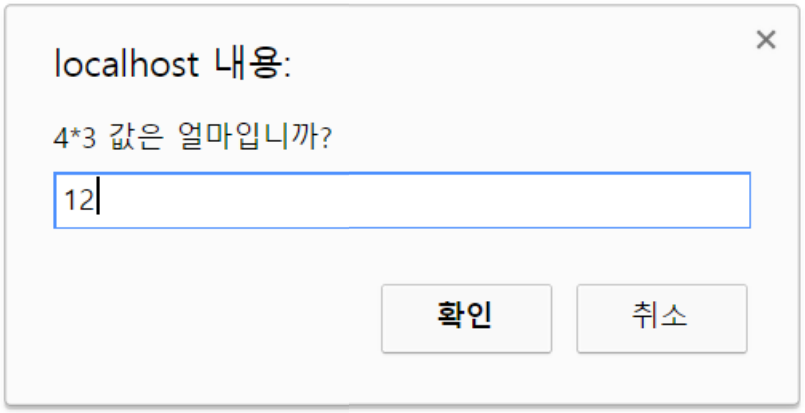
```
// 0~99까지 랜덤한 정수를 10개 만드는 코드
for(i=0; i<10; i++) {
   var m = Math.random()*100;  // m은 0~99.999... 보다 작은 실수 난수
   var n = Math.floor(m);          // m에서 소수점 이하를 제거한 정수(0~99사이)
   document.write(n + " ");
}
```

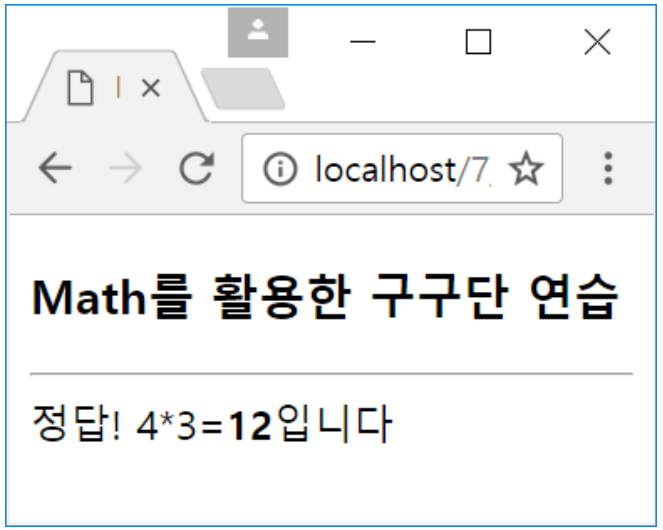
예제 7–8 Math를 이용한 구구단 연습

메소드	설명
Math.min(x, y, ...)	인수로 전달받은 값 중에서 가장 작은 수를 반환함.
Math.max(x, y, ...)	인수로 전달받은 값 중에서 가장 큰 수를 반환함.
Math.random()	0보다 크거나 같고 1보다 작은 랜덤 숫자를 반환함.
Math.round(x)	x를 소수점 첫 번째 자리에서 반올림하여 그 결과를 반환함.
Math.floor(x)	x와 같거나 작은 수 중에서 가장 큰 정수를 반환함.
Math.ceil(x)	x와 같거나 큰 수 중에서 가장 작은 정수를 반환함.
Math.abs(x)	x의 절댓값을 반환함.
Math.cbrt(x)	x의 세제곱근을 반환함.
Math.sqrt(x)	x의 제곱근을 반환함.
Math.pow(x, y)	x의 y승을 반환함.
Math.trunc(x)	x의 모든 소수 부분을 삭제하고 정수 부분만을 반환함.

실습 Math를 이용한 구구단 연습

- 문제는 randomInt() 함수를 정의해 2개의 난수를 발생해 출제한다.
  - 1~9사이의 난수발생 2개 : 4 * 3
- 취소 버튼이 클릭된 경우 "구구단 연습을 종료합니다"
- 정답일 경우 "정답! 2*4=8입니다"
- 정답이 아닐경우 "아니오! 2*9=18입니다"

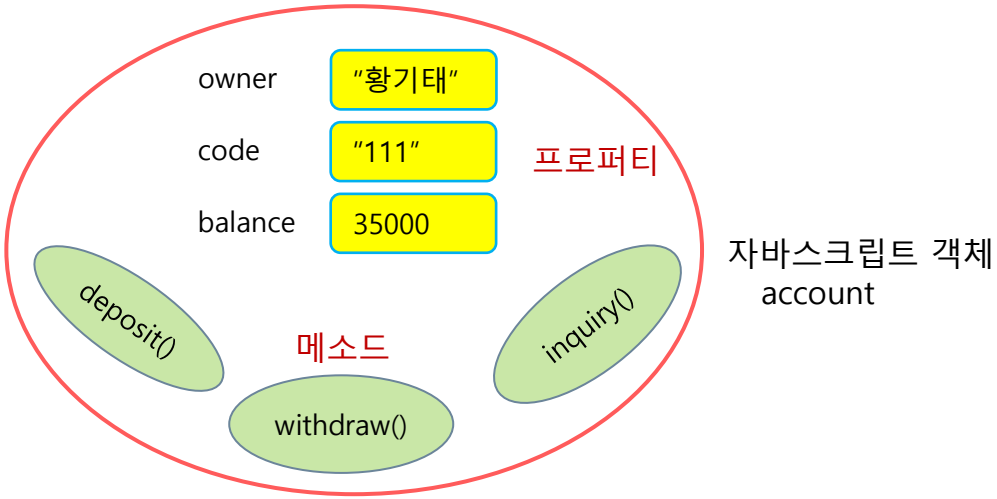




solution

사용자 객체 만들기

- 사용자가 새로운 타입의 객체 작성 가능 : 3 가지 방법
  - 1. 직접 객체 만들기
    - new Object() 이용
    - 리터럴 표기법 이용
  - 2. 객체의 틀(프로토타입)을 만들고 객체 생성하기
- 샘플
  - 은행 계좌를 표현하는 account 객체



new Object()로 객체 만들기

- 과정
  - 1. new Object()로 빈 객체 생성
  - 2. 빈 객체에 프로퍼티 추가
    - 새로운 프로퍼티 추가(프로퍼티 이름과 초기값 지정)
  - 3. 빈 객체에 메소드 추가
    - 메소드로 사용할 함수 미리 작성
    - 새 메소드 추가(메소드 이름에 함수 지정)

new Object()로 객체 만들기

- 과정
  - 1. new Object()로 빈 객체 생성
  - 2. 빈 객체에 프로퍼티 추가
    - 새로운 프로퍼티 추가(프로퍼티 이름과 초기값 지정)

```
var account  = new Object();
account.owner = "황기태";       // 계좌 주인 프로퍼티 생성 및 초기화
account.code = "111";           // 코드 프로퍼티 생성 및 초기화
account.balance = 35000;        // 잔액 프로퍼티 생성 및 초기화
```

new Object()로 객체 만들기

- 과정
  - 3. 빈 객체에 메소드 추가
    - 메소드로 사용할 함수 미리 작성
    - 새 메소드 추가(메소드 이름에 함수 지정)

```
var account   = new Object();
account.owner = "황기태";       // 계좌 주인 프로퍼티 생성 및 초기화
account.code = "111";           // 코드 프로퍼티 생성 및 초기화
account.balance = 35000;        // 잔액 프로퍼티 생성 및 초기화
```

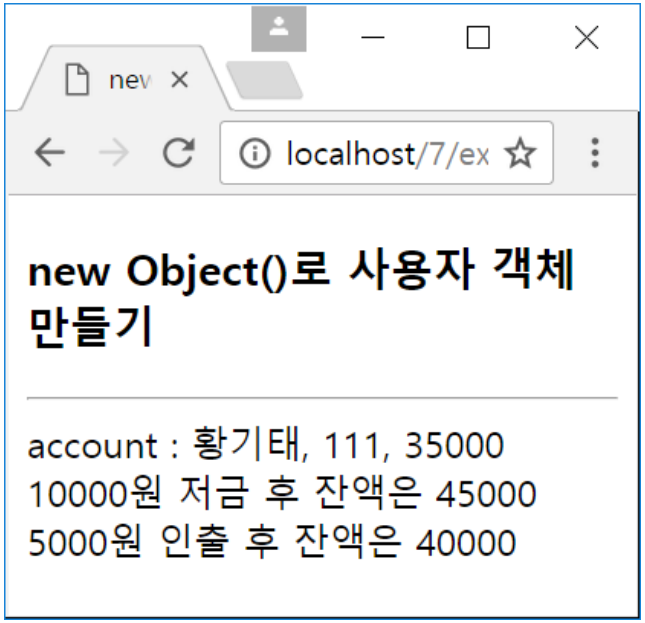
```
function deposit(money){         // 메소드로 사용할 함수 작성
   this.balance += money;
}
account.deposit = deposit;       // 메소드 만들기 완성
```

```
account.deposit(1000);    // 메소드 실행
```

예제 7-9 new Object()로 계좌를 표현하는 account 객체 만들기


```
<!DOCTYPE html>
<html><head><title>new Object()로 사용자 객체 만들기</title>
<script>
    //메소드로 사용할 3 개의 함수 작성
    function inquiry() { return this.balance; } // 잔금 조회
    function deposit(money) { this.balance += money; } // money 만큼 저금
    function withdraw(money) { // 예금 인출, money는 인출하고자 하는 액수
                                    // money가 balance보다 작다고 가정
        this.balance -= money;
        return money;
    }

    // 사용자 객체 만들기
    var account  = new Object();
    account.owner = "황기태"; // 계좌 주인 프로퍼티 생성 및 초기화
    account.code = "111"; // 코드 프로퍼티 생성 및 초기화
    account.balance = 35000; // 잔액 프로퍼티 생성 및 초기화
    account.inquiry = inquiry; // 메소드 작성
    account.deposit = deposit; // 메소드 작성
    account.withdraw = withdraw; // 메소드 작성
</script></head>
<body>
<h3>new Object()로 사용자 객체 만들기</h3>
<hr>
<script>
    // 객체 활용
    document.write("account : ");
    document.write(account.owner + ", ");
    document.write(account.code + ", ");
    document.write(account.balance + "<br>");

    account.deposit(10000); // 10000원 저금
    document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
    account.withdraw(5000); // 5000원 인출
    document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```

this.balance는 객체의 balance 프로퍼티



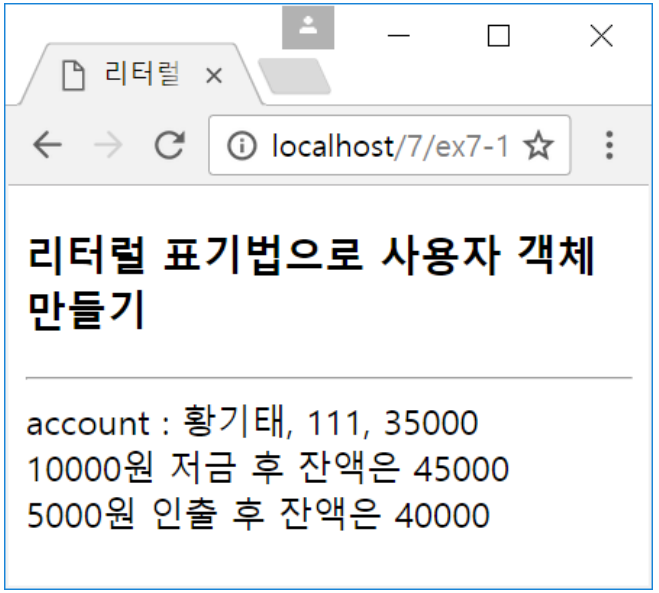
리터럴 표기법으로 만들기

- 과정
  - 중괄호를 이용하여 객체의 프로퍼티와 메소드 지정

```
var account = {
   // 프로퍼티 생성 및 초기화
   owner : "황기태",    // 계좌 주인 프로퍼티 추가
   code : "111",        // 계좌 코드 프로퍼티 추가
   balance : 35000,     // 잔액 프로퍼티 추가

   // 메소드 작성
   inquiry : function () { return this.balance; }, // 잔금 조회
   deposit : function(money) { this.balance += money; }, // 저금. money 만큼 저금
   withdraw : function (money) { // 예금 인출, money는 인출하고자 하는 액수
      // money가 balance보다 작다고 가정
      this.balance -= money;
      return money;
   }
};
```

예제 7-10 리터럴 표기법으로 계좌를 표현하는 account 객체 만들기


```
<!DOCTYPE html>
<html>
<head><title>리터럴 표기법으로 사용자 객체 만들기</title>
<script>
//사용자 객체 만들기
var account = {
    // 프로퍼티 생성 및 초기화
    owner : "황기태", // 계좌 주인
    code : "111", // 계좌 코드
    balance : 35000, // 잔액 프로퍼티

    // 메소드 작성
    inquiry : function () { return this.balance; }, // 잔금 조회
    deposit : function(money) { this.balance += money; }, // 저금. money 만큼 저금
    withdraw : function (money) { // 예금 인출, money는 인출하고자 하는 액수
                                  // money가 balance보다 작다고 가정
        this.balance -= money;
        return money;
    }
};
</script></head>
<body>
<h3>리터럴 표기법으로 사용자 객체 만들기</h3>
<hr>
<script>
    document.write("account : ");
    document.write(account.owner + ", ");
    document.write(account.code + ", ");
    document.write(account.balance + "<br>");

    account.deposit(10000); // 10000원 저금
    document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
    account.withdraw(5000); // 5000원 인출
    document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```



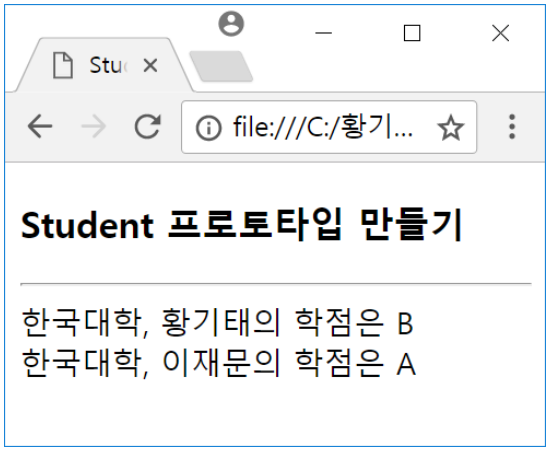
프로토타입

- 프로토타입(prototype)이란?
  - 객체의 모양을 가진 틀
  - 붕어빵은 객체이고, 붕어빵을 찍어내는 틀은 프로토타입
  - C++, Java에서는 프로토타입을 클래스라고 부름
  - Array, Date, String : 자바스크립트에서 제공하는 프로토타입
  - 객체 생성시 'new 프로토타입' 이용
    - var week = **new Array**(7); // Array는 프로토타입임
    - var hello = **new String**("hello"); // String은 프로토타입임

프로토타입 만드는 사례 : Student 프로토타입

- 프로토타입은 함수로 만든다
  - 프로토타입 함수를 생성자 함수라고도 함

```
// 프로토타입 Student 작성
function Student(name, score) {
    this.univ = "한국대학"; // this.univ을 이용하여 univ 프로퍼티 작성
    this.name = name; // this.name을 이용하여 name 프로퍼티 작성
    this.score = score; // this.score를 이용하여 score 프로퍼티 작성
    this.getGrade = function () { // getGrade() 메소드 작성
            if(this.score > 80) return "A";
            else if(this.score > 60) return "B";
            else return "F";
    }
}
```



- new 연산자로 객체를 생성한다

```
var kitae = new Student("황기태", 75);          // Student 객체 생성
var jaemoon = new Student("이재문", 93); // Student 객체 생성
document.write(kitae.univ + ", " + kitae.name + "의 학점은 " + kitae.getGrade() + "<br>");
document.write(jaemoon.univ + ", " + jaemoon.name + "의 학점은 " + jaemoon.getGrade() + "<br>")
```

예제 7-11 프로토타입으로 객체 만들기


```
<!DOCTYPE html>
<html><head><title>프로토타입으로 객체 만들기</title>
<script>
    // 프로토타입 만들기 : 생성자 함수 작성
    function Account(owner, code, balance) {
        // 프로퍼티 만들기
        this.owner = owner;     // 계좌 주인 프로퍼티 만들기
        this.code = code;       // 계좌 코드 프로퍼티 만들기
        this.balance = balance; // 잔액 프로퍼티 만들기

        // 메소드 만들기
        this.inquiry = function () { return this.balance; }
        this.deposit = function (money) { this.balance += money; }
        this.withdraw = function (money) { // 예금 인출, money는 인출하는 액수
            // money가 balance보다 작다고 가정
            this.balance -= money;
            return money;
        }
    }
</script></head>
<body>
<h3>Account 프로토타입 만들기</h3>
<hr>
<script>
    // new 연산자 이용하여 계좌 객체 생성
    var account = new Account("황기태", "111", 35000);

    // 객체 활용
    document.write("account : ");
    document.write(account.owner + ", ");
    document.write(account.code + ", ");
    document.write(account.balance + "<br>");

    account.deposit(10000); // 10000원 저금
    document.write("10000원 저금 후 잔액은 " + account.inquiry() + "<br>");
    account.withdraw(5000); // 5000원 인출
    document.write("5000원 인출 후 잔액은 " + account.inquiry() + "<br>");
</script>
</body></html>
```