2020. 07. 01 – 2020. 07. 02
학습내용 발표

알고리즘

2020.07.01 - 문자열 내 p와 y의 개수

문제

대문자와 소문자가 섞여있는 문자열 s가 주어집니다. S에 'p'의 개수와 'y'의 개수를 비교해 같으면 True, 다르면 False를 return 하는 solution를 완성하세요. 'p', 'y' 모두 하나도 없는 경우는 항상 True를 리턴합니다. 단, 개수를 비교할 때 대문자와 소문자는 구별하지 않습니다.

제한 조건

문자열 s의 길이 : 50 이하의 자연수

문자열 s는 알파벳으로만 이루어져 있습니다.

입출력 예시

s	answer
pPoooyY	true
Pyy	false

```
var temp = [];
var p = [];
var y = [];
temp = s.split("");
for(var i=0; i<temp.length; i++){
    if(temp[i] =="p" || temp[i] == "P"){
        p.push(temp[i]);
    }
    if(temp[i] =="y" || temp[i] == "Y"){
        y.push(temp[i]);
    }
}
p.length == y.length ? answer=true : answer=false;
return answer;
```

JavaScript for Web Browser (생활코딩)

2020.07.01 (1강 ~ 8강)

- HTML에서 JavaScript를 로드하는 방법 4가지 (inline / script태그 / 외부파일로드 / onload)

inline방식 : 정보와 제어가 같이 있기 때문에 유지보수 문제, 검색엔진이 해석하기 어려워지는 문제등이 발생할 수 있다. 이를 해결하기 위한 방법이 script태그.

scrpit태그 방식 : inline태그보다는 정보와 제어를 구분하지만, 아직은 html안에 포함되어있는 방식.

외부파일 로드 방식 : HTML파일이 정보'만'을 갖게됨. JS가 여러개의 HTML파일을 공통으로 사용할 수 있게 작업할 수 있음. 중복제거, 유지보수의 편의성 등의 장점 그리고 브라우저가 캐시를 통해 같은파일을 요청받았을때는 빠르게 불러오기 때문에 네트워크의 대역폭 지연현상 감소, 서버측 부담 감소, 전송과정에서 발생하는 비용감소 등 여러 장점이 있다.

onload방식 : window.onload 가 하는 역할 => 웹페이지가 모두 읽히고, 모든 소스들이 다운로드가 끝난 후에, onload라는 메소드를 호출한다. 조금 더 빠르게 로딩되기 위해서는, body태그 끝나는 곳에 script를 호출하는것이 바람직하다.

Do it! 리액트 프로그래밍 정석

2020.07.01 (~48p)

리액트 ES6문법 액기스 01. 템플릿문자열

기존 자바스크립트에서는 문자열과 문자열 또는 문자열과 변수를 연결하려면 병합연산자(+)를 사용해야 했습니다. 병합 연산자를 사용했을때의 코드의 복잡성을 해결하기 위해 ES6에서는 템플릿 문자열을 도입했습니다. 템플릿 문자열을 작은따옴표 대신 백틱 (`) 으로 문자열을 표현합니다. 또한 템플릿 문자열에 특수기호 $를 사용하여 변수 또는 식을 포함할 수도 있습니다.

ex)

(병합연산자방식) '장바구니에' + cart.name + '가 있습니다. 총 금액은' + getTotal(cart) + '입니다.';
(템플릿문자열방식) `장바구니에 ${cart.name}가 있습니다. 총 금액은 ${getTotal(cart)} 입니다.`

Do it! 리액트 프로그래밍 정석

2020.07.01 (~48p)

리액트 ES6문법 액기스 02. 전개연산자

전개연산자는 함수를 호출하는 인자로 배열을 사용하고 싶을 때나 배열을 정의하는 리터럴 내에서 사용할 수 있습니다. 사용 방법은 배열이나 객체, 변수명 앞에 마침표 세개 (…) 를 입력합니다. 다만, 배열, 객체, 함수 인자 표현식 ([ ].{ },( )) 안에서만 사용해야 합니다.

S6이전 문법에서는 배열의 일부 요소만 잘라내거나 연결하려면 배열 인덱스와 함께 배열 내장함수들을 사용해야 했고, 객체또한 객체의 키나 값을 추출할때 객체 내장 함수를 사용했습니다. 이제는 전개연산자를 이용하여 코드를 간결하게 작성 할 수 있습니다.

ex)

1) 전개연산자의 함수호출용의 용도

```
const add = (a, b, c) => {
    return a+b+c;
}
var arr = [2, 4, 5];
add(...arr); // 11이 출력
```

2) 배열 리터럴에서 전개연산자를 활용하는 방법

```
var arr1 = [3, 4, 5];
var arr2 = [1, 2, ...arr1, 6, 7];

console.log(arr2); // [1, 2, 3, 4, 5, 6, 7]
```

Do it! 리액트 프로그래밍 정석

2020.07.01 (~48p)

리액트 ES6문법 액기스 02. 전개연산자 추가개념 : Rest Parameter (나머지 매개변수)

함수를 선언 할 때, 해당 함수가 호출될 때 값으로 들어올 인자를 정하는 것을 매개변수라고 합니다. 이 매개변수를 지정할 때 매개변수의 이름 앞에 ...을 붙이면 이는 나머지 매개변수를 의미합니다. 그리고 이는 함수내에서 배열로 인식됩니다. 나머지 매개변수는 arguments와 다르게 유사 배열이 아닌 자바스크립트 표준 배열로 대체되고, **마지막 파라미터만 Rest 파라미터가 될 수 있습니다**. 그리고 **나머지 매개변수는 반드시 하나여야 합니다.** 또한, arguments 객체는 호출될 때 들어온 모든 인자를 의미하지만 나머지 매개변수는 사용자가 원하는 인자만 배열로 정의할 수 있습니다.

```
function add2(a, b, ...rest) {
    console.log(arguments);
    console.log(rest);
}
add2(2,3,4,5,6);
```

Do it! 리액트 프로그래밍 정석

2020.07.01 (~48p)

리액트 ES6문법 액기스 03. 가변 변수와 불변 변수

기존 자바스크립트 문법은 변수선언에 var 키워드를 사용했지만, ES6에서는 값을 수정할 수 있는 가변 변수를 위한 let 키워드와, 값을 수정할 수 없는 불변 변수를 위한 const키워드를 사용합니다.

가변 변수 사용법 : **let**키워드로 선언합니다. let으로 선언한 변수는 읽거나 수정할 수 있습니다.
불변 변수 사용법 : **const**키워드로 선언합니다. const로 선언한 변수는 읽기만 가능합니다.

불변 변수는 값을 다시 할당할 수 없는 것이지, 값을 변경할 수는 있습니다. 하지만 불변 변수로 정의된 배열이나 객체를 수정하는 것은 '**무결성 제약조건에 위반되었다**' 라고 합니다. 무결성을 유지하면서 불변 변수의 값을 수정해야 하는 경우가 있을때에는, 수정할 불변 변수를 새로 만들어 새 값을 할당하는 방법으로 수정해야 합니다. 불변 변수를 사용하면 무결성 제약 규칙에 의해 변수가 변하는 시점을 쉽게 파악할 수 있고, 수행 전과 후의 변수값을 비교할 수 있어 가변 변수보다 더 유용합니다.

알고리즘

2020.07.02 - 문자열 내 마음대로 정렬하기

문제

문자열로 구성된 리스트 strings와, 정수 n이 주어졌을 때, 각 **문자열의 인덱스 n번째 글자를 기준으로 오름차순 정렬하려** 합니다.

제한 조건

strings는 길이 1 이상, 50이하인 배열입니다.

strings의 원소는 소문자 알파벳으로 이루어져 있습니다.

trings의 원소는 길이 1 이상, 100이하인 문자열입니다.

모든 strings의 원소의 길이는 n보다 큽니다.

인덱스 1의 문자가 같은 문자열이 여럿 일 경우, 사전순으로 앞선 문자열이 앞쪽에 위치합니다.

입출력 예시

strings	n	return
[sun, bed, car]	1	[car, bed, sun]
[abce, abcd, cdx]	2	[abcd, abce, cdx]

```
function compare(a, b) {
    if (a.charAt(n) > b.charAt(n)) return 1;
    else if (b.charAt(n) > a.charAt(n)) return -1;
    else if (a.charAt(n) == b.charAt(n)) {
       return a.localeCompare(b);
    }
}
answer = strings.sort(compare);
```

JavaScript for Web Browser (생활코딩)

2020.07.02 (9강 ~ 20강)

- BOM (Browser Object Model)

웹브라우저를 제어하기 위해서 브라우저가 제공하는 객체들. (자바스크립트를 통해 브라우저에 새창을 연다거나, 현재창의 URL을 알아낸다거나, 현재 동작하고 있는 제품명, 버전명 등을 알아낸다거나 를 할수있도록)

- 사용자와의 커뮤니케이션 내장함수 (alert , confirm , prompt)

- location 객체를 통한 문서의 주소정보 확인 및 변경 (console.log를 통해 확인)

location.href : 주소확인

location.protocol : 프로토콜 확인

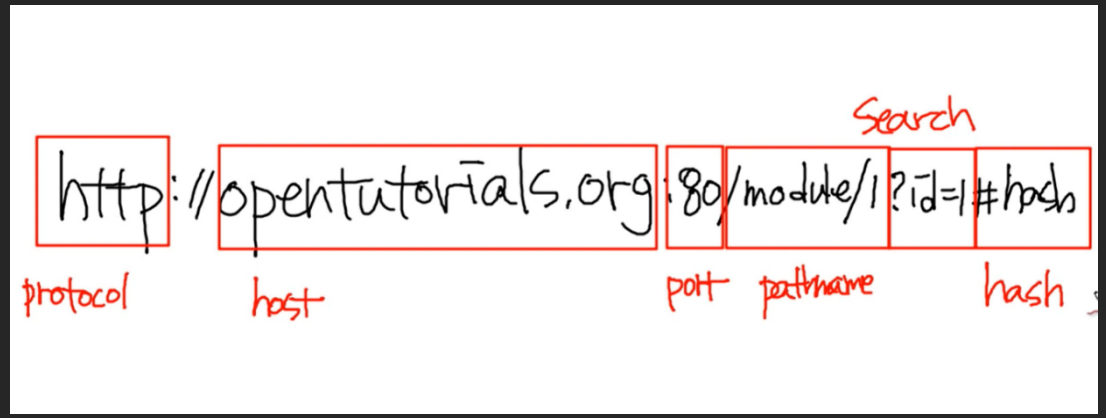
location.host : domain 확인

location.port : port번호 확인

location.pathname : 주소에서 domain을 제외한 구체적인 정보값

location.search : 주소에서 ? 뒤에 오는 값

location.hash : 주소에서 #뒤에 오는값. Bookmark값



JavaScript for Web Browser (생활코딩)

2020.07.02 (9강 ~ 20강)

- Navigator 객체

크로스 브라우징을 위해 필요한 정보를 알아오기 위한 객체.

※ console.dir을 통해 해당 객체가 가지고 있는 프로퍼티를 열람할 수 있다.

Navigator.appName / appVersion / userAgent / platform 등

Navigator 객체는 브라우저 호환성을 위해 사용되지만 모든 브라우저에 대응하는것은 쉬운일이 아니므로, 기능테스트를 사용하는것이 더 선호되는 방법이다.

```
// From https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/(
if (!Object.keys) {
  Object.keys = (function () {
    'use strict';
    var hasOwnProperty = Object.prototype.hasOwnProperty,
        hasDontEnumBug = !({toString: null}).propertyIsEnumerable('toString'),
        dontEnums = [
          'toString',
```

JavaScript for Web Browser (생활코딩)

2020.07.02 (9강 ~ 20강)

- 창 제어

Window.open('주소', 인자) ※ a태그의 target 에 들어가는 인자와 동일

인자가 없는 경우 : 이름이 붙지 않은 새 창 오픈

인자가 _self : 스크립트가 실행되는 창에서 오픈

인자가 _blank : 새 창

인자에 이름을 넣어준다면 : 새로 오픈되는 창이 해당 이름을 갖게 되고, 다시 열릴 경우, 그 이름을 갖고있는 창에서 오픈이 일어난다.

인자를 추가로 넣어줄수 있는데, 새로 열리는 창의 사이즈 등을 지정해 줄 수 있다.

window.open 을 특정 변수에 담아, 변수.document~ 등으로 해당 창의 내용을 제어할 수 있고, 변수.close()등으로 해당 창을 종료시킬 수도 있다.

Do it! 리액트 프로그래밍 정석

2020.07.02 (~52p)

리액트 ES6문법 액기스 04. 클래스

ECMAScript 5 이전 버전의 JavaScript에는 클래스가 없었습니다. 클래스에 가장 가까운 방법은 생성자를 생성한 다음 생성자의 프로토 타입에 메서드를 할당하는 것으로, 일반적으로 사용자 정의 타입 생성이라고 하는 접근 방식입니다.
클래스는 조금 더 연관있는 데이터(속성값(field) 과 행동 (method))를 하나로 묶어놓는 컨테이너 같은 개념. 클래스 안에서도 내부적으로 보여지는 변수와 외부로 보이는 변수를 나누어서 캡슐화를 해줍니다. 클래스를 통해 상속과 다양화가 일어날 수 있습니다.
Class는 template, 즉 틀과 같은 역할을 해서, 이러이러한 데이터가 들어갈 수 있다고 정의해서 선언해주고, 실제로 데이터를 넣어 만들어 낸 것은 object 라고 합니다.

알아야 하는 개념이 많아서, 기본적인 class의 개념만 찾아서 정리했습니다.

Do it! 리액트 프로그래밍 정석

2020.07.02 (~52p)

리액트 ES6문법 액기스 05. 화살표 함수

화살표함수는 ES6에 추가된 표현식을 사용하는 함수로, 화살표기호 (=>) 로 함수를 선언합니다.

함수를 선언할때, function 키워드를 생략하고 인자블록 ( ) 과 본문블록 { } 사이에 =>를 표기합니다.

ex) 기존방식 : var add = function(first,second){ } 화살표함수 : var add = (first,second) => { }

화살표 함수를 통해, 계단형 함수 구조가 만들어지지 않게 할 수 있고, 콜백함수의 this 범위로 생기는 오류를 피하기 위해 bind()

함수를 사용하여 this 객체를 전달하는 과정을 포함하고 있어서, 해당 과정을 생략해서 사용할 수 있습니다.