

Ionic Study

Day 3

오늘 할 것들

- 자바스크립트란?
- 자바스크립트 기본 문법 알아보기
 - 자료형, 연산자, 제어구조

자바스크립트

- 웹브라우저 위에서 동작하는 언어.
 - 객체 기반의 스크립트 프로그래밍 언어.
- 웹프로그래밍을 하려고 한다면 자바스크립트는 사실상 필수라고 할 수 있음.
- 최근에는 node.js라는 서버측 자바스크립트가 각광받고 있으며, 자바스크립트는 다양한 플랫폼을 프로그래밍적으로 제어하기 위한 도구로 폭넓게 채택되고 있음.

자바스크립트로 할 수 있는 것

- 웹 클라이언트 기반
 - html 요소들을 동적으로 제어 (DOM 조작)
 - 페이지 이동 없이 http 요청 주고받기 (Ajax)
- 웹 서버 기반
 - PHP, JSP등을 배우지 않아도 자바스크립트만으로 웹 서버의 구축이 가능 (Node.js)

ECMAScript

- 자바스크립트의 표준
 - 그 이전까지는 Javascript, Jscript등으로 비표준 기능들이 난립하였음.
 - 자바스크립트가 자바스크립트가 아니게 되어버리는 현상을 우려하여 ECMA라는 단체에서 표준 규격을 제창.
- ES3, ES5, ES6(ES2015)등의 규격버전이 존재. 숫자가 높을수록 최신 규격임.

자바스크립트 따라해보기

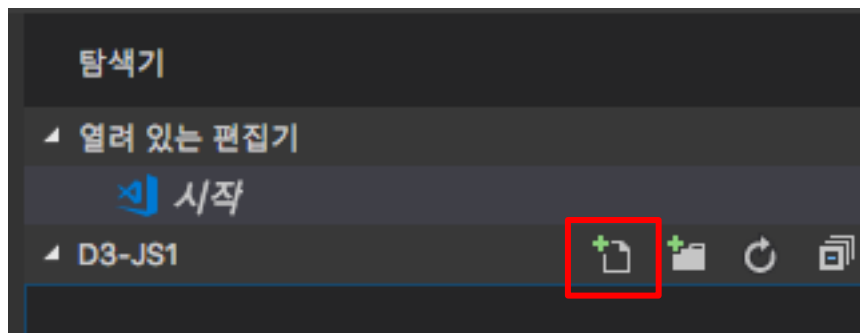
- html문서 내에서 실행
 - `<script>` 태그 내에서 자바스크립트 구문 삽입
 - `<script src...>`로 스크립트 파일 로드
- NodeJS로 실행
 - nodejs 내부에서 즉시 (Immediately) 실행
 - nodejs로 스크립트 파일 로드

자바스크립트 따라해보기

- 프로젝트 폴더 만들기
- Visual Studio Code 실행
 - [파일] - [열기] - 프로젝트 폴더 선택 후 열기
 - '탐색기' 영역에서 '새 파일' 아이콘 클릭하여
새 파일 추가

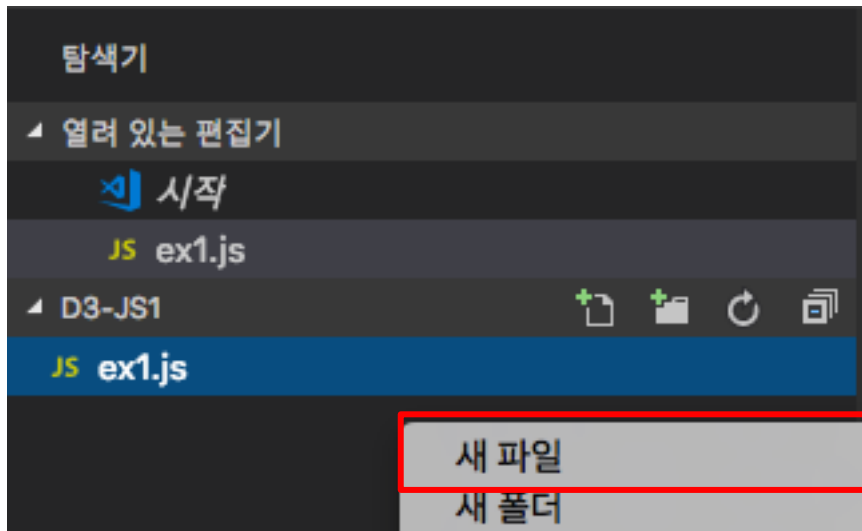
자바스크립트 따라해보기

- Visual Studio Code 실행
 - '탐색기' 영역에서 '새 파일' 아이콘 클릭하여
새 파일 추가



자바스크립트 따라해보기

- Visual Studio Code 실행
 - '탐색기' 영역에서 마우스 오른쪽 버튼 클릭하여 [새 파일] 클릭.

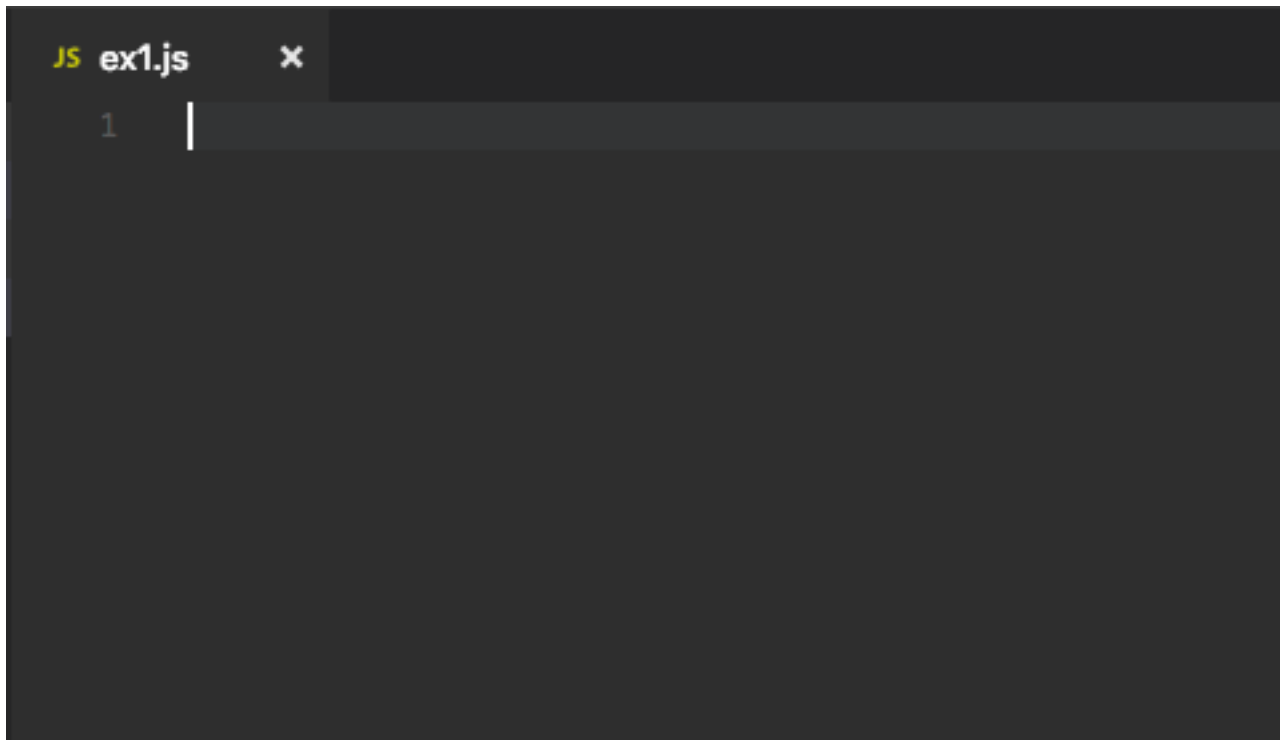


자바스크립트 따라해보기

- 새 파일 만들어보기
 - 파일명은 자유롭게.
 - 확장자는 끝에 js를 붙임.
- 편집을 해보자
 - 우측 편집기 영역에서 파일 에디트 가능.

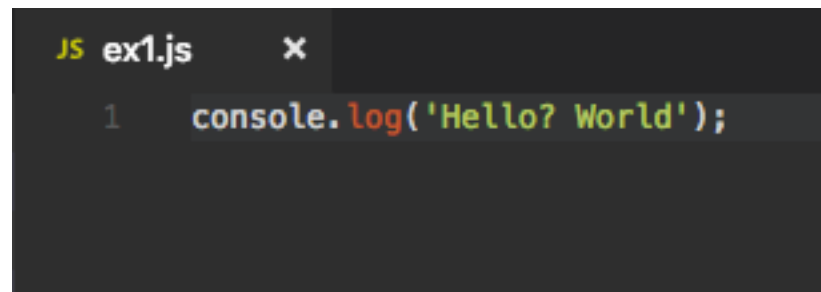
자바스크립트 따라해보기

- Visual Studio Code의 편집기 영역



자바스크립트 따라해보기

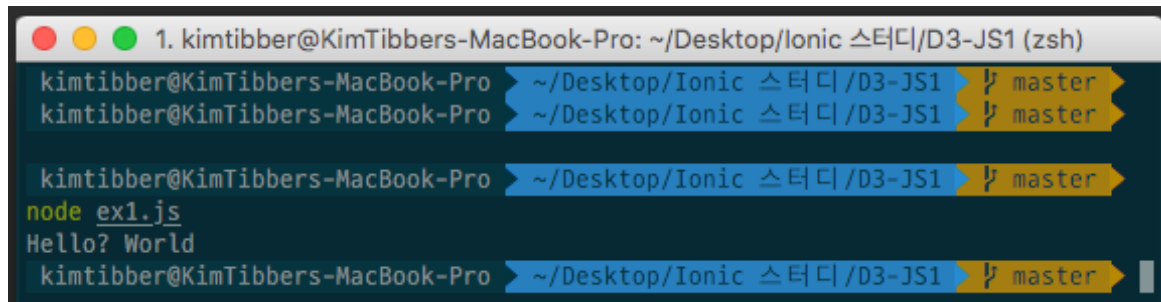
- Hello? World를 콘솔로 출력해보기.
- `console.log('Hello? World');`
- 입력 후 저장 (Ctrl+S)



```
JS ex1.js x
1 console.log('Hello? World');
```

자바스크립트 따라해보기

- 만든 파일 실행해보기
 - 터미널 열고 프로젝트 폴더 이동 후, 아래 명령어 실행
 - `node ex1.js` (본인이 만든 파일명)



```
1. kimtibber@KimTibbers-MacBook-Pro: ~/Desktop/Ionic 스테디/D3-JS1 (zsh)
kimtibber@KimTibbers-MacBook-Pro > ~/Desktop/Ionic 스테디/D3-JS1 > master
kimtibber@KimTibbers-MacBook-Pro > ~/Desktop/Ionic 스테디/D3-JS1 > master

kimtibber@KimTibbers-MacBook-Pro > ~/Desktop/Ionic 스테디/D3-JS1 > master
node ex1.js
Hello? World
kimtibber@KimTibbers-MacBook-Pro > ~/Desktop/Ionic 스테디/D3-JS1 > master
```

자바스크립트 따라해보기

- console 객체
 - 콘솔 제어를 위한 객체
 - log 메소드를 이용하면 콘솔 화면에 문자열의 출력이 가능.

자바스크립트 시작해보기

- 기본 약속

- 한 줄이 끝나면 줄 끝에 세미콜론 (;) 붙이기.
- 줄이 끝났는데도 세미콜론을 쓰지 않는다면
그 줄이 끝나지 않은것으로 보고 문법 오류를
일으키게 될 것임...

자바스크립트 시작해보기

- 변수
 - 값을 저장하기 위한 기억공간.
 - 어떤 값이 저장에 따라 **변**하는 **수**
- 변수는 var, let등으로 할당을 통해 사용이 가능.

자바스크립트 시작해보기

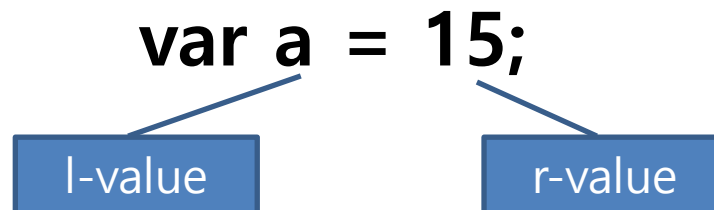
- 변수의 명명 규칙
 - 변수명은 영문과 숫자를 사용.
 - ex: greeting, sum_2017
 - 특히 영문은 대소문자를 가림 (Case-sensitive)
 - Alex와 alex는 서로 다른 변수
 - 변수의 첫문자는 영문 대소문자, _를 사용하며, 숫자는 사용이 불가능함.
 - ex: 1stGrade등
 - 변수명은 사용자가 지정할 수 있지만, 자바스크립트에서 이미 사용중인 예약어를 사용하는것은 불가능.

자바스크립트 시작해보기

- 선언, 할당, 대입
 - 선언 : 변수를 지정하는 과정
 - 할당 (대입) : 변수에 값을 지정하는 과정
 - var, let, const 키워드
 - 변수 재선언 여부의 차이.
 - var는 변수의 재선언이 가능함.
 - `var a = 2;`
`var a = 3;` 과 같은 형태가 가능
 - let는 변수의 재선언이 불가능.
 - `var a = 2;`
`var a = 3;` 은 불가능하지만, 값을 재할당하는것은 가능.
 - const는 변수의 재선언, 재할당이 불가능. => 상수

자바스크립트 시작해보기

- l-value, r-value
 - 등호 (=)를 기준으로 왼쪽에 있는 식을 l-value, 오른쪽에 있는 식을 r-value라고 함.
 - l-value에는 할당될 변수명이 위치함.
 - r-value에는 할당할 값 또는 변수명이 위치함.



자바스크립트 시작해보기

- ex2.js

```
var a = 10;
```

```
var a = 15;
```

```
let b = 20;
```

```
let b = 25;
```

```
const c = 30;
```

```
c = 35;
```

- 실행 결과는?

자바스크립트 시작해보기

- 변수와 자료형
 - 변수 : 값을 담을 수 있는 그릇.
 - 자료형 : 그릇안에 들어갈 내용물의 종류를 결정
 - 정수, 실수, 문자, 함수, 객체등이 있다.

자바스크립트 시작해보기

- 자료형

- 변수가 다룰 수 있는 값의 종류에 대한 정의.
- 단, 자바스크립트에서는 C나 자바처럼 자료형을 미리 정의할 필요는 없다 (실행단계에서 자동으로 파악됨)
 - 같은 변수에 여러 타입의 값을 넣는 것이 가능.
- 위와 같은 특성으로 인해, 변수값이 잘못된 자료형으로 오염될 수 있음.

자바스크립트 시작해보기

- 자료형의 종류

- 기본 자료형

- Boolean : true와 false 두 가지 값을 가질 수 있는 논리적인 요소
 - Null : 빈 값을 논리적으로 나타내는 표현.
 - Undefined : 값을 할당하지 않은 변수가 가지게 되는 자료형
 - Number : 정수 또는 실수로 표현.
 - String : 문자열

- 객체

- Object : 키와 값으로 이루어진 속성이 하나 이상 존재하는 집합.
 - Array : 키를 정수로 가지는 객체. 배열이라고 함.

자바스크립트 시작해보기

- ex3.js

```
JS ex1.js    JS ex2.js    JS ex3.js    x
1  let var_num, var_str, var_bool, var_null, var_undefined;
2  var_num = 10;
3  var_str = 'Hello';
4  var_bool = true;
5  var_null = null;
6  console.log('var_num : ' + var_num);
7  console.log('var_str : ' + var_str);
8  console.log('var_bool : ' + var_bool);
9  console.log('var_null : ' + var_null);
10 console.log('var_undefined : ' + var_undefined);
```


자바스크립트 시작해보기

- 연산자
 - 프로그래밍에서 쓰이는 기호.
 - 많은 연산자들이 있음.
 - 피연산자의 갯수로 구분 (단항, 이항, 삼항)
 - 연산자의 역할로 구분 (산술, 문자열, 증감...)

자바스크립트 시작해보기

- 산술 연산자
 - 하나의 숫자 값을 반환하는 연산자.
 - $+$, $-$, $*$, $/$ 의 사칙연산과 $\%$ 로 구성.
 - $+$: 두 값을 더하는 연산자
 - $-$: 두 값을 빼는 연산자
 - $*$: 두 값을 곱하는 연산자
 - $/$: 두 값을 나누는 연산자
 - $\%$: 두 값을 나눴을 때, 발생하는 나머지를 구하는 연산자

자바스크립트 시작해보기

- 문자열 연산자
 - + 연산자는 숫자간의 계산뿐만이 아니라, 문자열과 다른 데이터를 연결하는데에도 사용이 된다.
 - 문자열이 아닌 데이터는 문자열로 바뀌어서 연결됨.
 - 숫자와 문자열을 동시에 취급하고자 할 때 주의해야 함. 산술보다 문자열 연산이 우선임.

자바스크립트 시작해보기

- ex4.js

```
1.js    JS ex2.js    JS ex3.js    JS ex4.js    x
let plus, minus, mul, div, mod;
plus = 5 + 3;
minus = 10 - 7;
mul = 3 * 4;
div = 10 / 2;
mod = 8 % 3;
console.log('5 + 3 = '+ plus);
console.log('10 - 7 = '+ minus);
console.log('3 * 4 = '+ mul);
console.log('10 / 2 = '+ div);
console.log('8 % 3 = '+ mod);

let st1, st2, st3;
st1 = 'I am' + 'Boy';
st2 = 'Since' + 2009;
st3 = st1 + st2;
console.log('st1 :'+ st1);
console.log('st2 :'+ st2);
console.log('st3 :'+ st3);
```

자바스크립트 시작해보기

- 증감 연산자
 - ++, -- 가 있음.
 - 단항 연산자 (피연산자가 1개)
 - 변수에 1을 더하거나 1을 빼주는 연산자
 - ++나 --를 붙이는 위치에 따라 결과가 달라짐.
 - 앞에 붙은것을 전위증감연산자, 뒤에 붙은것을 후위증감연산자라고 부름.

자바스크립트 시작해보기

- 증감 연산자

- 전위증감연산자

- 먼저 1을 더하거나 빼고
다음 동작을 취함.

- 후위증감연산자

- 동작을 먼저 한 다음, 1을 더하거나 뺌.

- 혼란스러울 수도 있다...



자바스크립트 시작해보기

- ex5.js

```
js      JS ex2.js      JS ex3.js      JS
let i = 0;
i++;
console.log('i++ : ' + (i++));
console.log('i : ' + i);
console.log('++i : ' + (++i));
console.log('i : ' + i);
i = i + 1;
console.log('i : ' + i);
```

자바스크립트 시작해보기

- 대입 연산자
 - 변수에 값을 대입하는 연산자.
 - 산술 연산자와 결합하여 '복합대입연산자' 를 만들어 낼 수도 있다.
 - $+=$, $-=$, $*=$, $\%=$
 - ex) $i += 2$ // $i = i + 2$ 와 같다.

자바스크립트 시작해보기

- ex6.js

```
let val = 10;  
val += 2;  
console.log('val += 2 : '+ val);  
val -= 3;  
console.log('val -= 2 : '+ val);  
val *= 4;  
console.log('val *= 2 : '+ val);  
val %= 5;  
console.log('val %= 2 : '+ val);  
val /= 6;  
console.log('val /= 2 : '+ val);
```

자바스크립트 시작해보기

- 비교 연산자

- 값을 비교하여 논리적으로 참, 거짓을 가려내는 연산자.
- 크다 ($>$), 작다 ($<$)와 등호를 결합하여 이상 ($>=$), 이하 ($<=$), 동치 ($==$), 동치 부정 ($!=$) 의 사용이 가능.

자바스크립트 시작해보기

- ex7.js

```
let a = 10, b = 15;  
console.log('a = ' + a + ', b = ' + b);  
console.log('a > b : ' + (a > b));  
console.log('a < b : ' + (a < b));  
console.log('a >= b : ' + (a >= b));  
console.log('a <= b : ' + (a <= b));  
console.log('a == b : ' + (a == b));  
console.log('a != b : ' + (a != b));
```

자바스크립트 시작해보기

- 논리 연산자
 - and, or, not를 표현 가능한 연산자.
 - true와 false를 가지는 값에 대하여 논리 연산자로 논리 연산이 가능.

자바스크립트 시작해보기

- 논리 연산자
 - &&
 - AND 연산자.
 - op1과 op2가 모두 true일 때만 true를 가짐.
 - 그 외의 경우엔 false

자바스크립트 시작해보기

- 논리 연산자

- ||

- OR 연산자.
 - op1 또는 op2 중에 하나라도 true일 경우 true를 가짐.
 - op1과 op2가 모두 false라면 false를 가짐.

자바스크립트 시작해보기

- 논리 연산자

- !

- NOT 연산자.

- op1의 반대 값을 돌려줌.

- ex: true => false, false => true.

자바스크립트 시작해보기

- ex8.js

```
let true_value = true,  
    false_value = false;  
  
console.log('true AND true = ' + (true_value && true_value));  
console.log('true AND false = ' + (true_value && false_value));  
console.log('true OR true = ' + (true_value || true_value));  
console.log('true OR false = ' + (true_value || false_value));  
console.log('NOT true = ' + (!true_value));  
console.log('NOT false = ' + (!false_value));  
  
let op1 = 10,  
    op2 = 20;  
console.log(op1 > op2 && op1 <= op2 || !(op1 == op2));|
```


자바스크립트 시작해보기

- 제어 구조
 - 흐름을 제어하는 구문들
 - 분기 발생, 반복으로 이루어짐.

자바스크립트 시작해보기

- if...elseif...else
 - 조건식에 따라 분기를 발생시키는 구문.
 - if (조건식1) { ... } else if (조건식2) { ... } else { ... }

자바스크립트 시작해보기

```
if (expr1) {
```

```
...   <= expr1이 참이라면 실행
```

```
} else if (expr2) {
```

```
...   <= expr1은 거짓이지만, expr2가 참이라면 실행
```

```
} else {
```

```
...   <= expr1과 expr2가 모두 거짓이라면 실행
```

```
}
```

자바스크립트 시작해보기

- ex9.js

```
const a = 10, b = 15;
let c = a + b;

if (c < 10) {
    console.log('c는 10보다 작아여!');
} else if (c > 10 && c <= 40) {
    console.log('c는 10보다 큰데 40 밑이에여!');
} else {
    console.log('c는 40보다 큰듯...');
}
```

자바스크립트 시작해보기

- switch...case
 - 변수의 값에 따라 분기를 발생시키는 구문
 - 값으로 비교가 이루어짐.
 - 실행이 끝나면 break를 이용하여 분기를 끊어줘야 함
(break를 안쓰면 계속 아래 코드를 실행함)
 - 비교할 값이 없는 경우엔 default를 사용 (선택사항)

자바스크립트 시작해보기

```
switch (vari) {  
  case 'A':  
    ...           <= vari = 'A'라면 실행  
    break;  
  default:  
    ...           <= vari = 'A' 이외의 값일때 실행  
}
```

자바스크립트 시작해보기

- ex10.js

```
let kor = 60,  
    eng = 80,  
    avg = (kor + eng) / 2,  
    grade;  
  
if (avg > 70) {  
    grade = 'A';  
} else if (avg <= 70 && avg > 60) {  
    grade = 'B';  
} else if (avg <= 60 && avg > 50) {  
    grade = 'C';  
} else {  
    grade = 'F';  
}  
  
switch (grade) {  
    case 'A':  
        console.log('A등급입니다. ');  
        break;  
    case 'B':  
        console.log('B등급입니다. ');  
        break;  
    case 'C':  
        console.log('C등급입니다. ');  
        break;  
    case 'F':  
        console.log('F등급입니다. ');  
        break;  
}
```

오늘은 여기까지~

See you next day!