서버파트

JavaScript 기초

1. 자바스크립트 소개

- 특징

2. 문법 기본

- 자료형
- 연산자
- JSON 객체
- 배열
- 함수

3. 문법 심화

- 실행 컨텍스트
- 스코프
- 호이스팅

01. 자바스크립트 소개

1.자바스크립트 소개

특징

- 웹브라우저에서 동작하는 스크립트언어로 시작.
- Node.js의 등장으로 서버개발에서도 사용가능!
- 다양한 분야에서 활용 가능, 풀스택 개발까지 가능함.
- 문법의 종류가 다양함.
- Github에서 가장 많이 사용되고 있는 언어.

1.자바스크립트 소개

특징

- 대부분의 개념은 객체(기본데이터타입, null, undefined)
- 함수조차 객체로 취급! -> 일급객체(First class object) 로 다뤄짐.
- 모든 객체는 프로토타입을 가짐.
- 실행 컨텍스트가 독특하다.

  -> 스코프(객체의 유효범위)가 일반적이지 않다!
- 클래스를 지원하지 않지만 객체지향프로그래밍 가능.

02. 문법 기본

2. 문법 기본

자료형

자료형의 종류

- Number, String, Boolean, undefined, null, Object
- 모든 자료형을 var, let, const로 표시.

-> var, let, const는 스코프의 차이, 뒤에 설명.

- Object 에는 배열, 함수, 정규표현식 등이 포함.
- 숫자의 자료형이 Number 하나다???

-> JavaScript에서 숫자 자료형은 모두 64bit 실수형.

C++, JAVA의 double과 같다.

-> 정수형을 제대로 표기할 수 있을까??

53bit 정확도로 정수 표기 가능. 즉, int형 완벽히 표기가능.

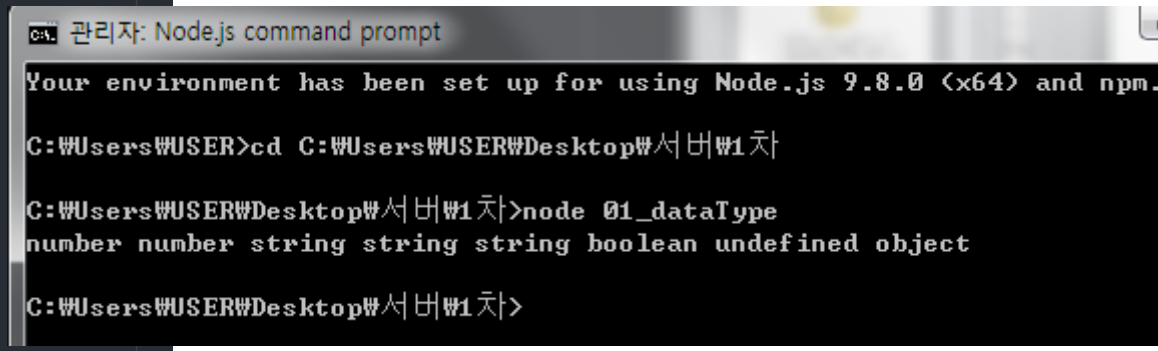
-> 근데 비트연산자는 제대로 작동함. But 느리다!!

- String은 '' , "" 둘다 가능하며 charactor도 String 타입.
- undefined는 타입인 동시에 값, 아무것도 없음을 의미. 선언만 되었다.
- null을 typeof로 확인하면 Object -> ===로만 null임을 확인가능!!

2. 문법 기본

자료형

```
var integerType = 5;
var floatType = 0.5;
//정수, 실수 둘다 Number 타입이고 이는 64bit 실수형입니다.

var singleQuoteStringType = '작은따옴표 스트링';
var doubleQuoteStringType = "큰따옴표 스트링";
//큰따옴표를 쓰던, 작은따옴표로 쓰던 둘다 스트링으로 처리합니다.
var charactorType = 'A';
//한글자의 charactor도 string으로 처리합니다.

var booleanType = true; // true false는 소문자로 적습니다.

var undefinedType;

var nullType = null; //null도 소문자로 적습니다.


//출력은 console.log(내용)을 사용합니다.
console.log(
  typeof integerType,
  typeof floatType,
  typeof singleQuoteStringType,
  typeof doubleQuoteStringType,
  typeof charactorType,
  typeof booleanType,
  typeof undefinedType,
  typeof nullType
); // console.log 끝에는 개행문자가 자동으로 삽입됩니다.

```



- Node.js Command prompt 창에서 실습해보기

2. 문법 기본

연산자

+연산자

- Number + Number 일 경우에는 더하기 연산 수행
- String + String , Number + String, String + Number 일 경우에는 문자열 연결 연산 수행
- 그럼 String + Number + Number 혹은 Number + Number + String 은????

  ㄴ 실습 02_operator

/연산자

- console.log(5/2)의 결과는????

연산자

typeof 연산자

- 피연산자의 타입을 String 형태로 반환하는 연산자.

숫자	“number”
문자열	“string”
boolean	“boolean”
null	“object”
Undefined	“undefined”
객체	“object”
배열	“object”
함수	“function”

└ 실습 02_operator

연산자

Equality(동등, 동치...)연산자

== , !=

두 값을 비교하는데 타입이 다를경우 묵시적 형변환 후 비교

Identity(일치)연산자

=== , !==

두 값을 비교하는데 타입이 다를경우 형변환 하지않고 비교

```
console.log( 1 == “1”);
console.log( 1 === “1”);
```

두 연산의 결과는 어떻게 출력될까??

ㄴ 실습 02_operator 에 추가해서 출력해보세요

JSON 객체

JSON (JavaScript Object Notation)

- 속성 , 값 의 쌍(property)으로 이루어진 데이터를 전달하기 위한 개방형 표준 포맷.
- 프로그래밍 언어에 독립적! -> 자바스크립트에서만 사용하는 것 아님!
- 인터넷에서 자료를 주고받을때 자료를 표현하는 방법
- 공식 미디어 인터넷 타입은 application/json
- 프로퍼티 사이의 구분은 , 으로 하고 마지막 프로퍼티 끝에는 붙이지 않음.
- "프로퍼티 이름" – "프로퍼티 값" 으로 구성되어 있을 때 이름의 "" 는 빠져도 됨.

```
{
"name" : "정보경",
"age" : 24,
"married" : false
}
```

둘다 가능!

```
{
name : "정보경",
age : 24,
married : false
}
```

└ 실습 03_object

JSON 객체

객체 생성

- Object()생성자 함수 이용

```
var server = new Object();
server.name = “김연태”;
server.age = 26;
server.married = false;
```

- 객체 리터럴(표현식) 방식 이용

```
var server = {
name : “김연태”,
age : 26,
married : false
}
```

— 실습 03_object

JSON 객체

프로퍼티 읽기

- 대괄호 표기법 이용

  server[“name”]

  server[“age”]

  server[“married”]

  -> 프로퍼티 이름을 문자열 형식으로 적어줘야 함!!

  만약 server[name]으로 접근시 undefined

- 마침표 표기법 이용

  server.name

  server.age

  server.married

└ 실습 03_object

JSON 객체

객체 프로퍼티 순회

for in문을 사용해 가능!

객체에 포함된 모든 프로퍼티에 대해 루프 수행

객체 프로퍼티 삭제

delete 연산자를 통해서 가능!

delete 연산자가 객체의 프로퍼티는 삭제할 수 있지만

객체 자체를 삭제할 수는 없다.

ㄴ 실습 03_object

배열

```
var NameofArray = [ 1 , 2.5 , “LikeLion” , true , { “part” : “server”, “name” : “정보경” } ];
```

- 배열의 원소에는 다른 데이터타입들이 들어갈 수 있음.
- 함수도 객체이므로 들어갈 수 있다.

배열

배열 요소 추가

- 동적으로 배열 원소를 추가할 수 있다.
- 특히나, 값을 순차적으로 넣을 필요 없이 아무데나 넣을 수 있다!!
- array.push(넣을값) -> 가장 끝에 있는 인덱스 뒤에 넣음

  ㄴ 실습 04_array

배열 요소 삭제

- delete를 통해 삭제 할 수 있다.

배열

배열 요소 접근

- array[음이 아닌 정수 or 변수] 로 접근가능
- 현재 배열 내 없는 인덱스로 접근하더라도 out of bound 나지 않음!

배열 요소 순회

- for(var i = 0 ; i < array.length ; i++){

console.log(array[i]);

} 배열 내 없는 원소까지 undefined로 모두 출력

- for(var j in array){

console.log(array[j]);

} 배열 내 없는 원소는 출력하지 않음.

배열

배열도 객체다!!

- 배열이름.프로퍼티이름 = 값 으로 프로퍼티 추가 가능
- length도 메소드가 아니라 객체의 프로퍼티다!
- 그래서 실제 배열의 길이와 관계없이 변경가능.
- 인덱스에만 관계있기 때문에 배열객체의 프로퍼티는 length에 포함되지 않음

함수

일급객체란?

- 자바스크립트에서 함수가 일급객체(first class object)로 취급된다.
- 변수 or 데이터구조에 담을 수 있다.
- 다른 함수의 파라미터로 전달할 수 있다.
- 반환 값으로 사용할 수 있다.
- 익명으로 생성할 수 있다.
- 런타임 시 생성될 수 있다.

함수

함수의 생성 방법

- 함수 선언문을 사용한 생성
- 함수 표현식을 사용한 생성
- 생성자 함수를 사용한 생성

함수가 생성되고 기능하는 것은 같지만 호이스팅에서 가장 큰 차이를 보임!

-> 문법 심화의 호이스팅에서 다시 다룰 것.

함수

함수 선언문을 사용한 생성

- 반드시 함수명이 정의되어 있어야 한다.
- 일반적인 함수 리터럴과 같음.
- function이라는 키워드를 사용.
- 매개변수의 타입을 표시하지 않음.

```
function add(x, y){
return x+y;
}
```

함수

함수 표현식을 사용한 생성

- 일급객체이므로 변수에 할당하는 것이 가능하다.
- 함수 리터럴로 하나의 함수를 만들고 변수에 할당하는 방식.
- 함수 이름은 선택사항이며 보통 익명함수로 생성 후 변수명으로 사용.
    -> 함수 표현식에서 사용된 함수 이름이 코드에서 접근불가하기 때문!

```
var add = function (x, y){
return x+y;
}
```

함수

생성자 함수를 통한 함수 생성

- 안씀.
- 그냥 이런게 있구나 알고만 있고 넘어갑시다.
- new Function(arg1, arg2, ... , argN, functionBody)

```
var add = new Function('x', 'y', 'return x + y')
```

03. 문법 심화

실행 컨텍스트

실행 컨텍스트란?

- 자바스크립트가 실행될 때 생성되는 하나의 실행 단위
- C, C++, JAVA 등의 콜스택에 들어가는 하나의 실행정보와 비슷한 개념.
- 실행 가능한 자바스크립트 코드 블록이 실행되는 환경.

현재 실행되는 컨텍스트에서 이 컨텍스트와 관련 없는 실행 코드가
실행되면 새로운 컨텍스트가 생성되어 스택에 들어가고 제어권이
그 컨텍스트로 이동한다.

실행 컨텍스트

실행 컨텍스트 생성 과정

1. 활성 객체 생성
2. arguments 객체 생성
3. 스코프 정보 생성
4. 변수 생성
    -> 여기서 선언이 이루어짐!
5. this 바인딩
6. 코드 실행
    -> 여기서 할당이 이루어짐!

호이스팅

호이스팅이란?

- 변수, 함수의 선언부가 스코프 가장 위로 끌어올려지는 것
- 블록 내부에 정의된 변수는 블록이 포함된 함수 전체에 선언되는 것과 같으므로 유효범위가 함수 전체로 확대된다.
- 반복문, 조건문 내부에 사용된 변수를 같은 함수 내라면 바깥에서 접근가능.
- 함수 표현식으로 정의되어 있으면 호이스팅이 발생하지 않음!

└ 실습 05_hoisting