삼육대학교 SW융합교육원 김진호

자바스크립트 기초교육

<class>
</body>
<head>
<title>

02. variable (변수)

2-1. data type

2-2. dynamically typed language

2-3. implicit-coercion

2-4. explicit coercion

2. variable

2. 변수(variable)

■ **변수란?**

- 변하는 값을 저장하기 위한 메모리상의 공간이다.

■ **변수를 사용하는 방법**

- 변수를 선언한다.
- 변수에 리터럴(값)을 대입한다. (값을 최초 대입하는 행위를 초기화 라고 한다.)

```
var num;   // 변수 선언
num = 20; // 값 대입
```

■ **리터럴이란?**

- 값 그 자체를 의미한다.
- 값의 타입 별로 약속된 리터럴의 형태가 다르다.

2. variable

2-1. data type (데이터 타입)

data type은 값의 종류를 말하며 자바스크립트(ES6)는 7개의 데이터 타입을 제공한다.

2-1-1. number (숫자 타입)

```
var integer = 10;
var double = 5.5;
var negative = -10;
```

- 자바의 경우 정수와 실수를 구분해 int, long, float, double 등과 같은 다양한 숫자 타입을 제공한다.
- 하지만 자바스크립트의 경우 하나의 숫자 타입만 존재하고 모든 수를 실수로 처리한다.
- 숫자 타입은 추가적으로 세 가지 특별한 값도 표현할 수 있다.
  - Infinity : 양의 무한대
  - -Infinity : 음의 무한대
  - NaN : 산술 연산 불가(not-a-number)

2. variable

2-1. data type (데이터 타입)

2-1-2. string (문자열 타입)

```
var string;
string = 'JavaScript'; // 작은 따옴표
string = "JavaScript"; // 큰 따옴표
string = `JavaScript`; // 백틱(ES6)
```

- 문자열 타입은 텍스트 데이터를 나타내는데 사용한다.
- 문자열은 작은 따옴표(''), 큰 따옴표(""), 또는 백틱(``)으로 텍스트를 감싼다.
- 자바는 문자열을 객체로 표현하지만 자바스크립트의 문자열은 원시 타입이며, 변경 불가능한 값이다.
- 템플릿 리터럴(``)
  - ES6부터 도입된 멀티라인 문자열, 표현식 삽입 등의 편리한 문자열 처리 기능을 제공하는 문자열 표기법이다. 이 때는 작은 따옴표, 큰 따옴표 대신 백틱을 사용해 표현한다.

2. variable

2-1. data type (데이터 타입)

2-1-3. boolean (논리 타입)

```
var flag = true;
console.log(flag);
flag = false;
console.log(flag);
```

- 불리언 타입의 값은 논리적 참, 거짓을 나타내는 true와 false뿐이다.

2. variable

2-1. data type (데이터 타입)

2-1-4. undefined and null

```
var undef;
console.log(undef);
```

- **undefined 타입**
  - undefined 타입의 값은 undefined가 유일하다.
  - var 키워드로 선언한 변수는 암묵적으로 undefined로 초기화 되므로 변수를 선언한 이후 값을 할당하지 않은 변수를 참조하면 undefined가 반환된다.
  - undefined는 자바스크립트 엔진이 변수를 초기화 할 때 사용하는 값이므로 개발자가 의도적으로 변수에 할당하는 것은 본래 취지와 어긋나고 혼란을 줄 수 있으므로 지양한다.

2. variable

2-1. data type (데이터 타입)

2-1-4. undefined and null

```
var nullVal = 'something';
nullVal = null; // 이전 참조를 제거하여 더 이상 'something'을 참조하지 않는다.
console.log(nullVal);
```

- **null 타입**
  - null 타입의 값은 null이 유일하다.
  - 변수에 값이 없다는 것을 의도적으로 명시할 때 사용한다.
  - 이전에 할당 되어 있던 값에 대한 참조를 명시적으로 제거하지 않는 것을 의미하고 자바스크립트 엔진은 누구도 참조하지 않는 메모리 공간에 대해 가비지 콜렉션을 수행하게 된다.

2. variable

2-1. data type (데이터 타입)

2-1-5. symbol and object

```
var key = Symbol('key');
console.log(typeof key);
```

- **Symbol 타입**
  - 심벌은 ES6에서 추가 된 7번째 타입으로, 변경 불가능한 원시 타입의 값이다.
  - 다른 값과 중복 되지 않는 유일무이한 값으로 이름 충돌할 위험이 없는 객체의 유일한 프로퍼티 키를 만들기 위해 사용한다.
  - 심벌 이외의 원시 값은 리터럴을 통해 생성하지만 심벌은 Symbol 함수를 통해 호출해 생성한다.
  - 지금은 데이터 타입의 한 종류 라고만 이해하고 넘어가면 된다.

2. variable

2-1. data type (데이터 타입)

2-1-5. symbol and object

```
var obj = {};                // 객체 생성
var func = function () {};   // 함수 생성
var arr = [];                // 배열 생성
```

- **객체 타입**
  - 자바스크립트의 데이터 타입은 크게 원시 타입과 객체 타입으로 분류한다.
  - 자바스크립트는 객체 기반의 언어이며, 자바스크립트를 이루고 있는 거의 모든 것이 객체이다.
  - number, string, boolean, undefined, null, symbol 6가지 데이터 타입 이외의 값은 모두 객체 타입이다.
  - 객체, 함수, 배열 등에 대해서 추후 하나씩 자세히 다뤄볼 것이다.

2. variable

2-2. dynamically typed language (동적 타입 언어)

- 정적 타입(static/strong type) 언어(C, C++, Java, Kotlin 등)
  - 변수를 선언할 때 데이터 타입을 사전에 선언(명시적 타입 선언)해야 한다.
  - 변수의 타입을 변경할 수 없으며, 변수에 선언한 타입에 맞는 값만 할당할 수 있다.
  - 컴파일 시점에 타입 체크를 수행하는데 타입의 일관성을 강제하여 런타임 에러를 줄인다.

2. variable

2-2. dynamically typed language (동적 타입 언어)

- **동적 타입(dynamic/weak type) 언어(JavaScript, Python 등)**
  - 자바스크립트는 var, let, const 키워드를 사용해 변수를 선언할 뿐 데이터 타입을 사전에 선언하지 않는다.
  - 즉, 동적 타입 언어는 변수 선언이 아닌 할당에 의해 타입이 결정(타입 추론)되며 재할당에 의해 변수의 타입은 언제든지 동적으로 변할 수 있다.
  - 변수의 값이 언제든지 변경 될 수 있기 때문에 값을 확인하기 전에는 타입을 확신할 수 없다.
  - 개발자의 의도와 상관 없이 자바스크립트 엔진에 의해 암묵적으로 타입이 자동 변환되기도 한다.
  - 즉, 유연성은 높지만 신뢰성은 떨어진다.
  - 이로 인해 변수를 사용하기 전 데이터 타입 체크를 하기도 하는데 이는 번거롭기도 하고 코드의 양도 증가한다.

2. variable

2-2. dynamically typed language (동적 타입 언어)

- **동적 타입 언어의 단점을 보완하기 위해서 변수 사용 시 유의할 점**

1. 변수는 꼭 필요한 경우에 한해 제한적으로 사용
2. 변수의 유효 범위(스코프)를 최대한 좁게 만듦
3. 전역 변수 지양
4. 변수보다 상수를 사용해 값의 변경 억제
5. 변수명을 통해 변수의 목적이나 의미를 파악할 수 있도록 함

2. variable

2-3. implicit-coercion (암묵적 타입 변환)

개발자가 의도적으로 값의 타입을 변환하는 것을 명시적 타입 변환(explicit coercion)이라고 하며 자바스크립트 엔진에 의해 암묵적으로 타입이 자동 변환 되는 것을 암묵적 타입 변환(implicit coercion)이라고 한다. 명시적 타입 변환은 코드에서 드러나지만 암묵적 타입 변환은 드러나지 않으므로 타입 변환 된 결과를 예측할 수 있어야 오류를 방지할 수 있다.

2-3-1. convert to string (문자열 타입으로 변환)

```
console.log(10 + '20');
```

- 문자열 연결 연산자(+)는 문자열 타입이 아닌 피연산자를 문자열 타입으로 암묵적으로 변환한다.

2. variable

2-3. implicit-coercion (암묵적 타입 변환)

2-3-2. convert to number (숫자 타입으로 변환)

```
console.log(10 - '5');      // 산술 연산자
console.log(10 > '5');      // 비교 연산자
console.log(+'');           // + 단항 연산자
```

- 산술 연산자의 피연산자는 모두 숫자 여야 하므로 숫자가 아닌 피연산자를 숫자 타입으로 암묵적 타입 변환한다.
- 비교 연산자로 크기를 비교하기 위해 모두 숫자 타입이여야 하므로 숫자가 아닌 피 연산자를 숫자 타입으로 암묵적 타입 변환한다.
- + 단항 연산자는 피연산자가 숫자 타입의 값이 아니면 숫자 타입의 값으로 암묵적 타입 변환한다.

2. variable

2-3. implicit-coercion (암묵적 타입 변환)

2-3-3. convert to boolean (논리 타입으로 변환)

```
if(undefined) console.log("if(undefined)");          // false
if(null) console.log("if(null)");                    // false
if(0) console.log ("if(0)");                         // false
if(NaN) console.log("if(NaN)");                      // false
if('') console.log("if('')");                        // false
if('JavaScript') console.log("if('JavaScript')");    // true
```

- 자바스크립트 엔진은 불리언 타입이 아닌 값을 Truthy 값(참으로 평가 되는 값) 또는 Falsy 값(거짓으로 평가되는 값)으로 구분한다.
- Truthy -> true, Falsy -> false로 암묵적 타입 변환 된다.

2. variable

2-4. explicit coercion(명시적 타입 변환)

개발자의 의도에 따라 값의 타입을 변환하는 것이다. 자바스크립트에서 기본 제공하는 표준 빌트인 생성자 함수(String, Number, Boolean)를 new 연산자 없이 호출하는 방법, 빌트인 메서드를 사용하는 방법, 암묵적 타입 변환을 이용하는 방법이 있다.

2-4-1. convert to string (문자열 타입으로 변환)

```
console.log(String(10));        // String 생성자 함수
console.log ((10).toString()); // Object.prototype.toString 메서드
console.log (10 + '20');        // 문자열 연결 연산자
```

- String 생성자 함수를 new 연산자 없이 호출한다.
- Object.prototype.toString 메서드를 사용한다.
- 문자열 연결 연산자를 이용한다.

2. variable

2-4. explicit coercion(명시적 타입 변환)

2-4-2. convert to number (숫자 타입으로 변환)

```
console.log(Number('10'));          // Number 생성자 함수
console.log (parseInt('10'));       // parseInt 함수
console.log(parseFloat('10.01'));   // parseFloat 함수
console.log(+'');                   // + 단항 연산자
console.log('10' * 1);              // * 산술 연산자
```

- Number 생성자 함수를 new 연산자 없이 호출한다.
- parseInt, parseFloat 함수를 이용한다. (문자열 -> 숫자만 가능)
- + 단항 산술 연산자를 이용한다.
- 산술 연산자를 이용한다.

2. variable

2-4. explicit coercion(명시적 타입 변환)

2-4-3. convert to boolean (논리 타입으로 변환)

```
console.log(Boolean('JavaScript'));   // Boolean 생성자 함수
console.log(!!'JavaScript');          // ! 부정 논리 연산자 두 번 사용
```

- Boolean 생성자 함수를 new 연산자 없이 호출한다.
- ! 부정 논리 연산자를 두 번 사용한다.