**JavaScript 재입문하기**

**■ 개요**

JavaScript는 유형 및 연산자, 표준 내장 객체 및 메소드가 있는 다중 패러다임, 동적 언어

객체 프로토 타입을 사용하여 객체 지향 프로그래밍을 지원

함수형 프로그래밍 지원

ECMAScript edition 5 (2009년 12월에 출간), ECMAScript edition 6 (2015년에 출간) 규격의 근간을 형성

**■ 데이터 타입**

1. 수 (Number)
2. 문자열 (String)
3. 부울 (Boolean)
4. 기호 (Symbol)
5. 객체 (Object)
   * 함수 (Function)
   * 배열 (Array)
   * 날짜 (Date)
   * 정규식 (RegExp)
6. 널 (Null)
7. 정의되지 않음 (Undefined)

**■ 변수**

1. let - 변경 가능한 변수 선언
2. const - 변경되지 않는 변수 선언
3. var - 명확하지 않은 변수 선언, 블록을 무시하여 사용됨

**■ 연산자**

|  |  |  |
| --- | --- | --- |
| **종 류** | **기 호** | **비 고** |
| 산술 | +, -, \*, /, % |  |
| 비교 | <, >, <=, >=, ===, !== |  |
| 증가와 감소 | ++, -- |  |
| 지정 | +=, -=, \*=, /= |  |

**■ 제어 구조**

1. 조건문
   * If 문

|  |
| --- |
| //예시  var name = 'kittens';  if (name == 'puppies') {  name += ' woof';  } else if (name == 'kittens') {  name += ' meow';  } else {  name += '!';  }  name == 'kittens meow'; |

* + Switch 문

|  |
| --- |
| //예시  switch(action) {  case 'draw':  drawIt();  break;  case 'eat':  eatIt();  break;  default:  doNothing();  } |

1. 반복문
   * while 문

|  |
| --- |
| //예시  while (true) {  // 무한루프!} |

* + do-while 문

|  |
| --- |
| //예시  var input;  do {  input = get\_input();  } while (inputIsNotValid(input)); |

* + for 문

|  |
| --- |
| //예시1  for (var i = 0; i < 5; i++) {  // 내부 동작을 5번 반복합니다  }  //예시2  for (let value of array) { // value로 작업을 실행합니다}  //예시3  for (let property in object) {  // object의 항목(property)으로 작업을 실행합니다  } |

**■ 객체**

JavaScript 객체는 name-value pairs의 모임

1. 배열

|  |
| --- |
| //예시  var a = ["dog", "cat", "hen"];  for (var i = 0; i < a.length; i++) {  // a[] 로 뭔가를 수행  }  for (const currentValue of a) { // currentValue 로 뭔가를 수행}  ['dog', 'cat', 'hen'].forEach(function(currentValue, index, array) {  // currentValue나 array[index]로 뭔가를 수행} |

1. 함수

|  |
| --- |
| //예시  function add(x, y) {  var total = x + y;  returntotal;  } |

1. 사용자 정의 객체

function을 class로 사용

|  |
| --- |
| //예시1  function Person(first, last) {  this.first = first;  thislast = last;  this.fullName = function() {  return this.first + ' ' + this.last;  };  this.fullNameReversed = function() {  return this.last + ', ' + this.first;  };  }  var s = new Person('Simon', 'Willison');  //예시2  function personFullName() {  return this.first + ' ' + this.last;}function personFullNameReversed() { return this.last + ', ' + this.first;}function Person(first, last) {  this.first = first;  this.last = last;  this.fullName = personFullName;  this.fullNameReversed = personFullNameReversed;} |

1. Class

|  |
| --- |
| //예시  class Person {  constructor(name, age) {  this.name = name;  this.age = age;  } speak() {  console.log(`${this.name} : hello!`);  }  }  const person = new Person(‘Bob’, 20); |

**■ 클로져 (Closures)**

함수와 함수에 의해 생성되는 범위 객체를 함께 지칭하는 용어

상태를 저장할 수 있도록 허용

|  |
| --- |
| //예시  function makeAdder(a) {  return function(b) {  returna + b;  };  }  var add5 = makeAdder(5);  var add20 = makeAdder(20);  add5(6); // 11  add20(7); // 27 |

클로져 문제점은 Internet Explorer에서 메모리 누출

Internet Explorer는 JavaScript와 다른 가비지 컬렉션 방식 사용

JavaScript 객체와 고유 객체간에 참조하는 중 자기 자신을 참조하는 경우 발생

|  |
| --- |
| 예시//  function leakMemory() {  Var el = document.getElementById('el');  varo = { 'el': el };  el.o = o;  } |