UI 프론트엔드 프로그래밍 과정

JavaScript

JavaScript

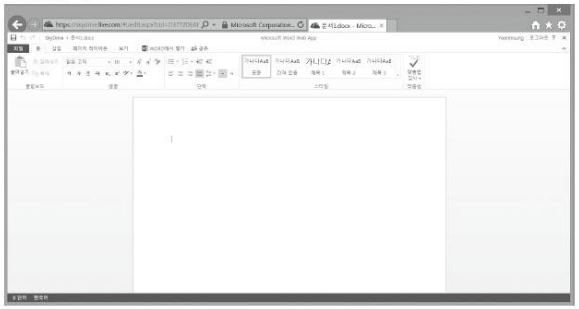
1.1 자바스크립트의 역사

- Javascript란?
  - 웹 브라우저에서 많이 사용하는 프로그래밍 언어
  - 넷스케이프 사의 브랜든 아이히(Brendan Eich)에 의해 모카라는 이름으로 시작 (모카라이브 스크립트로 이름 변경)
  - 넷스케이프 사가 썬 마이크로시스템과 함께 자바스크립트라는 이름을 붙이고 본격적 발전

1.2 자바스크립트의 활용

- 웹에서 웹 애플리케이션으로
  - 초기의 웹
    - 변화 없는 정적 글자들의 나열
    - 웹은 하이퍼링크라는 매개체를 사용해 웹 문서가 연결된 거대한 책에 불과
  - 자바스크립트의 등장
    - 웹 문서의 내용을 동적으로 바꾸거나 마우스 클릭 같은 이벤트 처리
  - 웹은 애플리케이션의 모습에 점점 가까워짐
    - 대표적인 예는 아래 그림과 같은 웹 문서 작성 도구
    - 구글 , 마이크로소프트에서는 웹 브라우저만으로 워드, 엑셀, 파워포인트 같은 애플리케이션 사용 가능
    - 웹 애플리케이션은 웹 브라우저만 있으면 언제 어디서나 사용 가능

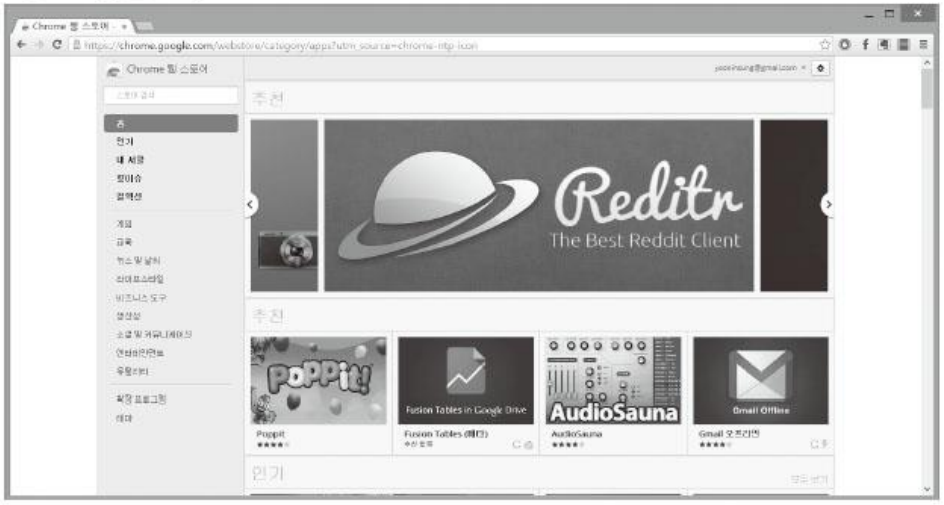
그림 1-1 웹 오피스



1.2 자바스크립트의 활용

- 구글에서 제공하는 크롬 웹 스토어
  - 웹 스토어에서 웹 브라우저상에서 실행되는 웹 애플리케이션 거래

그림 1-2 크롬 웹 스토어



1.2 자바스크립트의 활용

- 인터넷 연결 없이 웹 브라우저에서 실행 가능한 웹 애플리케이션
  - 스마트 폰이나 스마트패드 내의 애플리케이션을 만들 때에도 자바스크립트 사용

1.2 자바스크립트의 활용

- 파이어폭스 OS
  - 자바스크립트로 애플리케이션을 만듬
  - 그 밖에 웹 애플리케이션 사용이 가능한 장치

  → 데스크톱, 노트북, 스마트폰 등

그림 1-3 파이어폭스 OS

1.2 자바스크립트의 활용

- 3D 테트리스
  - HTML5의 도입 → 웹에서 플래시나 실버라이트 없이
    게임과 같은 복잡한 프로그램 작성 가능

그림 1-4 http://www.benjoffe.com/code/games/torus/

1.3 자바스크립트의 종류

◆ 자바스크립트의 종류

- 유럽 컴퓨터 제조 협회(European Computer Manufacturer's Association)는 자바스크립트를 ECMAScript라는 이름으로 표준화
- 웹 브라우저나 애플리케이션에 내장된 자바스크립트의 종류
- ECMAScript와 Jscript는 모두 자바스크립트를 의미

표 1-1 애플리케이션에 따른 자바스크립트의 종류

애플리케이션	자바스크립트 종류
Mozilla Firefox	JavaScript
Google Chrome	JavaScript
Internet Explorer	JScript
Opera	ECMAScript
Apple Safari	JavaScript
Microsoft .NET Framework	JScript.NET
Adobe Flash & Adobe Flex	ActionScript
Adobe Acrobat	JavaScript

1.4 HTML 파일 만들기

◆ 모든 예제는 HTML5 표준 형식 사용

코드 1-2 HTML5 기본 페이지(2)

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
</head>
<body>

</body>
</html>
```

◆ 자바 스크립트 코드 위치

- 기본 페이지의 head 태그 사이에 script 태크 삽입
  - script 태그 사이에 자바스크립트 코드 입력
- HTML5에서는 script 태그에 type 속성을 적지 않는 게 원칙

코드 1-3 기본 구성

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <script>

    </script>
</head>
<body>

</body>
</html>
```

JavaScript 기본문법

2.1 기본 용어

◆ 표현식과 문장

- 표현식이란?
  - 값을 만들어내는 간단한 코드
  - Ex)

```
273
10 + 20 + 30 * 2
'RintIanTta'
```

- 문장이란?
  - 하나 이상의 표현식이 모인 것
  - 문장이 모여 프로그램 구성
  - **문장의 끝에는 세미콜론을 찍어 문장의 종결을 알려줌**
  - **하나의 표현식에도 세미콜론만 찍히면 문장**
  - Ex)

```
10 + 20 + 30 * 2;
var rintiantta = 'Rint' + 'Ian' + 'Tta';
alert('Hello JavaScript..!');
273;
```

2.1 기본 용어

◆ 키워드

- 키워드란?
  - 자바스크립트가 처음 만들어질 때 정해진 특별한 의미가 있는 단어
  - 모든 브라우저에서 28개의 키워드를 지원

break	else	instanceof	true
case	false	new	try
catch	finally	null	typeof
continue	for	return	var
default	function	switch	void
delete	if	this	while
do	in	throw	with

2.1 기본 용어

◆ 키워드

- 미래에 사용될 가능성이 있는 자바스크립트 키워드
  - W3C에서는 자바스크립트 프로그램 작성 시 아래 키워드를 사용하지 않기를 권고
  - const나 debugger 같은 키워드는 이미 일부 브라우저에서 사용

abstract	double	implements	private	throws
boolean	enum	import	protected	transient
byte	export	int	public	volatile
char	extends	interface	short	
class	final	long	static	
const	float	native	super	
debugger	goto	package	synchronized	

2.1 기본 용어

◆ 식별자

- 자바스크립트에서 이름을 붙일 때 사용
- 식별자의 예
  - **변수 명과 함수 명**
- 식별자 생성 시 규칙
  - 키워드를 사용 불가
  - 숫자로 시작하면 불가
  - 특수 문자는 _과 $만 허용
  - 공백 문자 포함 불가
  - Ex)

```
alpha
alpha10
_alpha
$alpha
AlPha
ALPHA
```

- 식별자로 사용 불가능한 단어

```
break
273alpha
has space
```

2.1 기본 용어

◆ 식별자 생성 규칙

- 모든 언어가 사용 가능하나 알파벳 사용이 개발자들 사이 관례
- Input, output 같은 의미 있는 단어 사용
- 자바 스크립트 개발자가 식별자를 만들 때 지키는 관례
  - 생성자 함수의 이름은 대문자로 시작
  - 변수와 인스턴스, 함수, 메서드의 이름은 항상 소문자로 시작
  - 식별자가 여러 단어로 이루어지면 각 단어의 첫 글자는 대문자
  - Ex)

```
will out          ⇨ willOut
will return       ⇨ willReturn
i am a boy        ⇨ iAmABoy
```

2.1 기본 용어

◆ 자바스크립트의 식별자 종류

- 크게 네 종류
  - 더 많은 종류로 나누기도 하나 이 책에선 네 가지로 구분

구분	단독으로 사용	다른 식별자와 사용
식별자 뒤에 괄호 없음	변수	속성
식별자 뒤에 괄호 있음	함수	메서드

- 식별자 구분해보기

- **alert**('Hello World') ⇨ 함수
- Array.**length** ⇨ 속성
- **input** ⇨ 변수
- **propmt**('Message', 'Defstr') ⇨ 함수
- Math.**PI** ⇨ 속성
- Math.**abs**(-273) ⇨ 메서드

2.1 기본 용어

◆ 주석

- 프로그램 진행에 영향을 끼치지 않음
- 코드의 특정 부분을 설명
- 1) HTML 태그 주석
  - <!-- -->로 문자열을 감싸 생성

```
<!DOCTYPE html>
<html>
<head>
    <!-- 주석입니다. -->
    <script>
    </script>
</head>
<body>
    <!-- <h1>주석입니다.</h1> -->
</body>
</html>
```

2.1 기본 용어

◆ 주석

▪ 2) 자바스크립트 주석

- //를 사용해 한 줄 주석 표현
  - // 뒤의 문장은 실행되지 않음
  - Ex) // 주석문
- /*와 */을 사용해 여러 줄 주석 표현
  - /*와 */ 사이의 문장은 실행되지 않음
  - Ex) /*
    주석문
    주석문
    */

```
<script>
    // 주석은 코드의 실행에 아무 영향을 미치지 않습니다.
    /*
    alert('Hello JavaScript');
    alert('Hello JavaScript');
    alert('Hello JavaScript');
    */
</script>
```

2.2 출력

◆ 자바스크립트 출력

- 기본 출력 방법 : alert ( ) 함수를 사용할 구성 예제 코드

```
<!DOCTYPE html>
<html>
<head>
    <script>

    </script>
</head>
<body>

</body>
</html>
```

2.2 출력

◆ 자바스크립트 출력

▪ Alert ( ) 함수

- 가장 기본적인 출력 방법
- 브라우저에 경고창을 띄울 수 있음
- alert ( ) 함수의 사용 예 :
  - 함수의 괄호 안에는 문자열 입력

```
<script>
    alert('Hello JavaScript..!');
</script>
```

2.2 출력

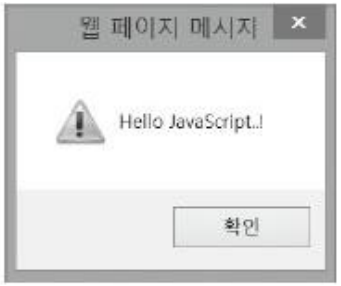
◆ 자바스크립트 출력

■ 매개 변수

- 함수의 괄호 안에 들어가는 것

```
alert()
alert([String message])
```

- 실행 결과



2.3 문자열

문자열이란?

- 문자를 표현할 때 사용하는 자료의 형태
- alert() 함수의 매개 변수로 쓰인 'Hello JavaScript..!' 와 같은 자료
- 문자열을 만드는 방법
  - 큰따옴표의 사용
    - "동해물과 백두산이"
  - 작은따옴표의 사용
    - '동해물과 백두산이'
  - 두 가지 방법 중 어떤 문자열로도 사용 가능하지만 일관되게 사용해야 함

2.3 문자열

◆ 예외적인 문자열 사용법

- 문자열 안의 따옴표 사용
  - 내부에 작은 따옴표 사용 시 외부에 큰 따옴표
  - 내부에 큰 따옴표를 사용 시 외부에 작은 따옴표

```
<script>
    alert('This is "string"');
    alert("This is 'string'");
</script>
```

2.3 문자열

◆ 예외적인 문자열 사용법

- 이스케이프 문자의 사용
  - 한 가지 따옴표로만 사용하고 싶을 때

- 이스케이프 문자란?
  - 특수한 기능을 수행하는 문자
  - 따옴표를 사용하고 싶을 때 이스케이프 문자를 사용
    - 따옴표 앞에 ₩를 사용해 따옴표를 문자 그대로 사용하겠다고 표시 (₩와 ₩ 는 같은 표시)

```
<script>
    alert("This is \"string\"");
    alert('This is \'string\'');
</script>
```

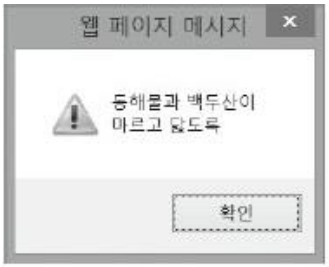
2.3 문자열

◆ 예외적인 문자열 사용법

- 이스케이프 문자의 특수 기능
  - **\n :** 문자열을 줄 바꿈 할 때 사용
  - 사용 예 :

```
<script>
    alert('동해물과 백두산이\n마르고 닳도록');
</script>
```

  - 출력 결과



2.4 숫자

◆ 숫자 자료 형

- **정수와 유리수의 구분 없이 숫자는 모두 숫자**
- Ex)

```
273
52.273
```

- 문자열과 마찬가지로 alert() 함수의 괄호 안에 사용해 출력

```
<script>
    alert(273);
    alert(52.273);
</script>
```

2.4 숫자

◆ 숫자를 사용한 기본적인 사칙 연산

- 연산자를 사용해 기본적인 사칙연산 가능

연산자	설명	연산자	설명
+	더하기 연산자	*	곱하기 연산자
–	빼기 연산자	/	나누기 연산자

- 자바스크립트에서는 연산자 우선순위 고려함
  - 다음 그림의 결과 값은?

```
5 + 3 * 2
```

  - 덧셈을 먼저 실행하고 싶다면 괄호를 사용

```
(5 + 3) * 2
```

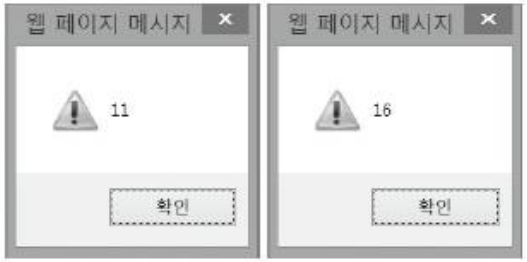
2.4 숫자

◆ 숫자를 사용한 기본적인 사칙 연산

- 자바스크립트 코드를 이용한 연산

```
<script>
    alert(5 + 3 * 2);
    alert((5 + 3) * 2);
</script>
```

- 출력 결과
  - 왼쪽이 5 + 3 * 2의 결과/ 오른쪽이 (5 + 3) * 2의 결과



2.4 숫자

◆ 숫자를 사용한 기본적인 사칙 연산

▪ 나머지 연산자 : %

- 좌변을 우변으로 나눈 나머지를 표시하는 연산자

연산자	설명
%	나머지 연산자

- 사용 예 :

```
<script>
    alert(10 % 7);
</script>
```

- 출력 결과

2.5 불

◆ 불 자료형

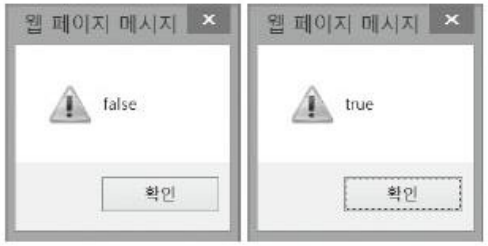
- 자바스크립트에서는 참과 거짓이라는 값을 표현할 때 사용
- 예제

```
52 > 273
52 < 273
```

- 사용 예 :

```
<script>
    alert(52 > 273);
    alert(52 < 273);
</script>
```

- 출력 결과



2.5 불

◆ 비교 연산자

- 비교 연산자란?
  - 두 대상을 비교할 수 있는 연산자

- 비교 연산자의 종류

연산자	설명
>=	좌변이 우변보다 크거나 같습니다.
<=	우변이 좌변보다 크거나 같습니다.
>	좌변이 큽니다.
<	우변이 큽니다.
==	좌변과 우변이 같습니다.
!=	좌변과 우변이 다릅니다.

- 문자열 비교
  - **국어사전의 앞 쪽에 있을 수록 값이 작음**

```
'가방' > '하마'  ⇨  false
```

2.5 불

◆ 비교 연산자

- 유니코드 문자로 비교
  - 모든 언어 비교 가능

```
'尹' == '尹'  ⇨  true
```

- 불끼리의 크기 비교
  - **자바스크립트는 true를 1 / false를 0으로 변환 후 비교 연산**

```
<script>
    alert(true > false);
</script>
```

  - alert (1 > 0)로 변환 → true 출력

2.5 불

◆ 불 사용 예

- 조건문에서 불의 사용
  - 조건문 괄호 안의 불 표현식이 참이면 중괄호 속 문장 실행
  - 거짓이면 중괄호 속 문장 무시

```
if(불 표현식) {
    불 표현식이 참일 때 실행할 문장
}
```

- 사용 예
  - 출력 결과 : 불 표현식이 참인 두 번째 경고창 출력

```
<script>
    if(273 < 52) {
        alert('273은 52보다 작습니다.');
    }
    if (273 > 52) {
        alert('273은 52보다 큽니다.');
    }
</script>
```

2.5 불

◆ 자바 스크립트 논리연산자의 종류

연산자	설명
!	논리 부정 연산자
&&	논리곱 연산자
||	논리합 연산자

- 논리부정 연산자
  - 참을 거짓으로, 거짓을 참으로 바꿈

```
<script>
    alert(!true);
    alert(!false);
</script>
```

2.5 불

◆ 자바 스크립트 논리연산자의 종류

- 논리곱 연산자
  - 좌변과 우변이 모두 참일 때만 참

좌변	우변	결과
true	true	true
true	false	false
false	true	false
false	false	false

- 논리합 연산자
  - 좌변과 우변이 모두 거짓일 때만 거짓

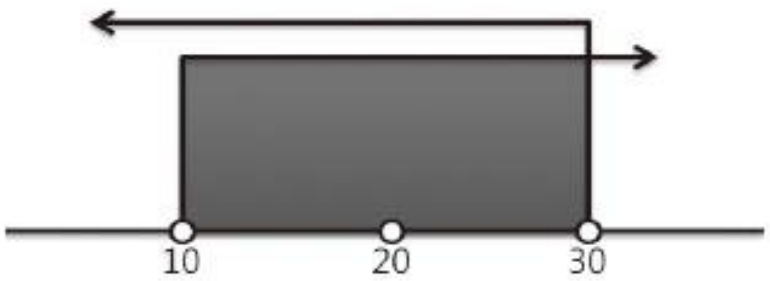
좌변	우변	결과
true	true	true
true	false	true
false	true	true
false	false	false

2.5 불

◆ 비교 연산자와 논리 연산자의 적절한 사용 필요

```
<script>
    alert(30 > 20 && 20 > 10);
</script>
```

alert(30>20 && 20>10);
↓
alert(true && true);
↓
alert(true);



2.6 변수

◆ 변수

- 값을 저장할 때 사용하는 식별자
  - 숫자뿐 아니라 모든 자료형 저장 가능
  - **변수를 사용하려면?**
    - **1. 변수 선언 : 변수를 만듦**
    - **2. 변수에 값 할당**
  - **변수 선언 방법**
    - **Var 식별자 ;**

```
<script>
    // 변수를 선언합니다.
    var pi;
</script>
```

2.6 변수

◆ 변수 할당

- 변수 초기화 : 변수 선언 후 처음 값을 할당 하는 것

```
<script>
    // 변수를 선언합니다.
    var pi;

    // 변수에 값을 할당합니다.
    pi = 3.14159265;
</script>
```

```
<script>
    // 변수를 선언 및 초기화합니다.
    var pi = 3.14159265;
</script>
```

2.6 변수

◆ 변수 할당

- 변수 사용(1)

```
<script>
    // 변수를 선언하고 초기화합니다.
    var pi = 3.14159265;

    // 출력합니다.
    alert(pi);
</script>
```

2.6 변수

◆ 변수 할당

▪ 변수 사용(2)

- 숫자가 들어간 변수는 숫자와 관련된 연산자 사용
- 문자열이 들어간 변수는 문자열과 관련된 연산자 사용

```
<script>
    // 변수를 선언 및 초기화합니다.
    var radius = 10;
    var pi = 3.14159265;

    // 출력합니다.
    alert(2 * pi * radius);
</script>
```

2.6 변수

◆ 변수 할당

■ 변수 사용(3)

- var 키워드를 사용해 여러 변수를 한꺼번에 선언
  - var 키워드 뒤에 쉼표를 사용해 식별자를 연속으로 입력

```
<script>
    // 변수를 선언합니다.
    var radius, pi;

    // 변수에 값을 할당합니다.
    radius = 10;
    pi = 3.14159265;

    // 출력합니다.
       alert(2 * pi * radius);
  </script>
```

2.6 변수

◆ 변수에 값 할당하기

- 자료형
  - 문자열, 숫자, 불리언과 같은 것, 함수, 객체
  - 자바스크립트에는 총 여섯 가지 자료형이 있음
    - **Cf. undefined 자료형**
      - » **선언되지 않거나 할당되지 않은 변수**
      - » **변수에 저장해도 의미가 없음**

```
<script>
    // 변수를 선언합니다.
    var stringVar = 'String';
    var numberVar = 273;
    var booleanVar = true;
    var functionVar = function () { };
    var objectVar = {};
</script>
```

2.6 변수

◆ 복합 대입 연산자

- 대입 연산자와 다른 연산자를 함께 사용하는 연산자

연산자	설명
+=	기존 변수의 값에 값을 더합니다.
-=	기존 변수의 값에 값을 뺍니다.
*=	기존 변수의 값에 값을 곱합니다.
/=	기존 변수의 값에 값을 나눕니다.
%=	기존 변수의 값에 나머지를 구합니다.

2.6 변수

◆ 복합 대입 연산자의 사용

- 변수 value를 10으로 초기화
- 이후 + = 복합 대입 연산자를 사용해 value의 기존 값에 10을 더함
- 결과는 20 출력

```
<script>
    // 변수를 선언합니다.
    var value = 10;

    // 연산자를 사용합니다.
    value += 10;

    // 출력합니다.
    alert(value);
</script>
```

2.6 변수

◆ 복합 대입 연산자의 사용

- 변수 list를 빈 문자열('')로 초기화
- + = 복합 대입 연산자를 사용해 문자열 만들고
- HTML 문서의 body 태그에 넣음

```
<script>
    window.onload = function () {
        // 변수를 선언합니다.
        var list = '';

        // 연산자를 사용합니다.
        list += '<ul>';
        list += '    <li>Hello</li>';
        list += '    <li>JavaScript..!</li>';
        list += '</ul>';

        // 문서에 출력합니다.
        document.body.innerHTML = list;
    };
</script>
```

- Hello
- JavaScript..!

2.6 변수

◆ 증감 연산자

- 복합 대입 연산자를 간략하게 사용한 형태

연산자	설명
변수++	기존의 변수 값에 1을 더합니다(후위).
++변수	기존의 변수 값에 1을 더합니다(전위).
변수--	기존의 변수 값에 1을 뺍니다(후위).
--변수	기존의 변수 값에 1을 뺍니다(전위).

2.6 변수

◆ 증감 연산자의 활용 (1)

- 변수 number를 초기화하고 + + 증감 연산자 사용
  - 코드를 실행하면 10에 1을 더한 11이 출력
  - + + number로도 출력해봐도 차이가 없음
  - 한 줄에 독립적 증감 연산자를 사용할 때는 전위와 후위의 차이가 없음
  - 다른 연산자나 함수와 함께 사용할 때 차이가 있음

```
<script>
    // 변수를 선언합니다.
    var number = 10;

    // 연산자를 사용합니다.
    number++;

    // 출력합니다.
    alert(number);
</script>
```

2.6 변수

◆ 증감 연산자의 활용 (2)

- 실행하면 순서대로 10, 11, 12 출력
- 후위는 해당 문장을 실행한 후에 값을 더하라는 의미
- alert(number+ +)는 alert(number)를 실행한 후 숫자 1을 더함

```
<script>
    // 변수를 선언합니다.
    var number = 10;

    // 출력합니다.
    alert(number); number += 1;
    alert(number); number += 1;
    alert(number); number += 1;
</script>
```

```
<script>
    // 변수를 선언합니다.
    var number = 10;

    // 출력합니다.
    alert(++number);
    alert(++number);
    alert(++number);
</script>
```

2.6 변수

◆ 증감 연산자의 활용 (4)

```
<script>
    // 변수를 선언합니다.
    var number = 10;

    // 출력합니다.
    alert(number++);
    alert(++number);
    alert(number--);
    alert(--number);
</script>
```

2.6 변수

◆ 변수의 특성

- 변수는 하나만 담을 수 있음

```
<script>
    // 변수를 선언합니다.
    var cup = 'Coffee';
    cup = 'green Tea';
    cup = 'Water';

    // 출력합니다.
    alert('Drink ' + cup + '..!');
</script>
```

2.6 변수

◆ 변수의 재선언 (1)

- 변수는 하나만 담을 수 있음

```
<script>
    // 변수를 선언합니다.
    var favoriteFood = '김치 찌개';
    var favoriteFood = '라면';
    var favoriteFood = '냉면';

    // 출력합니다.
    alert(favoriteFood);
</script>
```

2.6 변수

◆ 변수의 재선언 (3)

- 예제 코드의 출력 값은?

```
<script>
    // 1번 문제
    var value = 10;
    value += 20;
    alert(value);

    // 2번 문제
    var value = 'Hello' + '..!';
    alert(value + ' JavaScript');
</script>
```

2.7 자료형 검사

◆ 자료형

- 숫자, 문자열, 불리언 같은 자료의 형태
- typeof 연산자
  - 자료형을 확인할 때 사용

```
<script>
    alert(typeof ('String'));
    alert(typeof (273));
</script>
```

2.7 자료형 검사

◆ 자료형을 출력하는 예제 코드

- Undefined
  - **정의하지 않은 자료형 의미**
  - 선언하지 않은 식별자 alpha 사용

```
<script>
    // 문자열
    alert(typeof ('String'));
    // 숫자
    alert(typeof (273));
    // 불
    alert(typeof (true));
    // 함수
    alert(typeof (function () { }));
    // 객체
    alert(typeof ({}));
    // undefined
    alert(typeof (alpha));
</script>
```

2.8 undefined 자료형

- undefined 자료형
  - **'존재하지 않는 것'은 undefined 자료형으로 표현**
  - 변수로 선언하지 않은 식별자가 갖는 자료형

```
<script>
    // 출력합니다.
    alert(typeof (variable));
</script>
```

2.8 undefined 자료형

- undefined 자료형
  - 변수를 선언했지만 초기화하지 않았을 때 undefined 자료형을 갖음

```
<script>
    // 변수를 선언합니다.
    var variable;

    // 출력합니다.
    alert(typeof (variable));
</script>
```

2.9 입력

◆ 문자열을 입력하는 방법

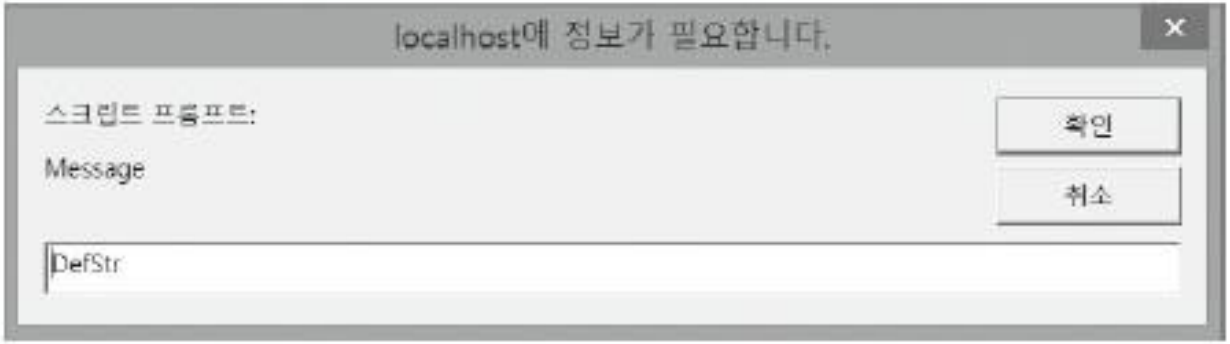
- 숫자를 입력 받는 방법
  - 문자열을 입력 받은 후 숫자로 변환
  - **문자열을 입력을 할 때 사용하는 함수는 prompt() – 매개변수 두 개 필요**

```
prompt()
String prompt([String message], [String defaultValue])
```

```
<script>
    // 변수를 선언합니다.
    var input = prompt('Message', 'DefStr');

    // 출력합니다.
    alert(input);
</script>
```

2.9 입력

- Prompt() 함수 실행
  - 입력 칸에 변수를 입력하면 그대로 코드 변경

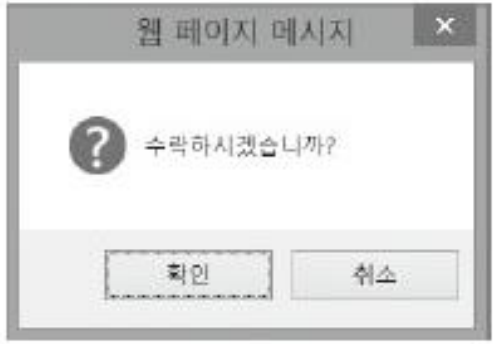


2.9 입력

Confirm( ) 함수

- **불을 입력 받을 때 사용**
- prompt() 함수와 비슷한 방식으로 사용
  - **사용자가 확인을 누르면 true 리턴**
  - **취소를 누르면 false 리턴**
  - 변수 input에 불이 들어가고 곧바로 변수 input을 출력

```
<script>
    // 변수를 선언합니다.
    var input = confirm('수락하시겠습니까?');

    // 출력합니다.
    alert(input);
</script>
```



2.10 숫자와 문자열 자료형 변환

- 문자열과 숫자를 더하는 자료형
  - 숫자와 문자열을 덧셈 연산하면 문자열 우선

```
<script>
    // 1번
    alert('52 + 273');
    // 2번
    alert(52 + 273);
    // 3번
    alert('52' + 273);
    // 4번
    alert(52 + '273');
    // 5번
    alert('52' + '273');
</script>
```

2.10 숫자와 문자열 자료형 변환

◆ 문자열과 숫자를 곱하는 자료형

- 더하기 연산자를 제외한 사칙 연산자는 숫자가 우선
  - 첫 번째를 제외하면 14196을 출력

```
<script>
    alert('52 * 273');
    alert(52 * 273);
    alert('52' * 273);
    alert(52 * '273');
    alert('52' * '273');
</script>
```

2.10 숫자와 문자열 자료형 변환

- 강제로 자료형 변환시키기
  - 다른 자료형을 숫자로 - Number() 함수
  - 다른 자료형은 문자열로 - String() 함수

2.10 숫자와 문자열 자료형 변환

- prompt() 함수
  - 문자열만 입력 가능
  - 아무리 숫자를 입력해도 문자열의 자료형 출력

```
<script>
    // 변수를 선언합니다.
    var input = prompt('숫자를 입력해주세요.', '숫자');

    // 출력합니다.
    alert(typeof (input));
</script>
```

2.10 숫자와 문자열 자료형 변환

Number() 함수

- **숫자로 바꾸려면 Number() 함수 사용**
- 문자열로 입력받고 숫자로 변환


```
<script>
    // 변수를 선언합니다.
    var input = prompt('숫자를 입력해주세요.', '숫자');
    var numberInput = Number(input);

    // 출력합니다.
    alert(typeof (numberInput) + ': ' + numberInput);
</script>
```

2.10 숫자와 문자열 자료형 변환

NaN

- **NaN(Not a Number) 값 출력 : 숫자가 아닌 값 입력 시**
- Nan은 자료형 숫자이나 자바스크립트로 나타낼 수 없는 숫자를 의미

```
<script>
    // 변수를 선언합니다.
    var number = Math.sqrt(-3);

    // 변수를 출력합니다.
    alert(number);
</script>
```

2.11 불 자료형 변환

- Boolean() 함수
  - **불 자료형으로 변환할 때 사용**
  - 다섯 가지 경우가 false로 변환
    - 이 다섯 가지를 제외한 모든 경우 true로 변환
      - 문자열 '0'과 문자열 'false'는 문자열이므로 true

```
<script>
    alert(Boolean(0));
    alert(Boolean(NaN));
    alert(Boolean(''));
    alert(Boolean(null));
    alert(Boolean(undefined));
</script>
```

2.11 불 자료형 변환

- 조건문 사용/ 논리 부정 연산자 사용
  - 자동으로 불리언 자료형으로 변환
  - **undefined 자료형은 false**

```
<script>
    alert(!!0);
    alert(!!NaN);
    alert(!!'');
    alert(!!null);
    alert(!!undefined);
</script>
```

2.12 일치 연산자

◆ 자동 자료형 변환

- 비교 연산자를 사용할 때 뜻하지 않는 경우가 발생
  - 네 가지 모두 true 출력

```
<script>
    alert('' == false);
    alert('' == 0);
    alert(0 == false);
    alert('273' == 273);
</script>
```

2.12 일치 연산자

◆ 일치 연산자의 용도

- 자료형이 다른 것을 확실하게 구분 짓고 싶을 때 사용
  - 예제 코드의 결과는 모두 false

연산자	설명
===	양 변의 자료형과 값이 일치합니다.
!==	양 변의 자료형과 값이 다릅니다.

```
<script>
    alert('' === false);
    alert('' === 0);
    alert(0 === false);
    alert('273' === 273);
</script>
```

JavaScript 조건문

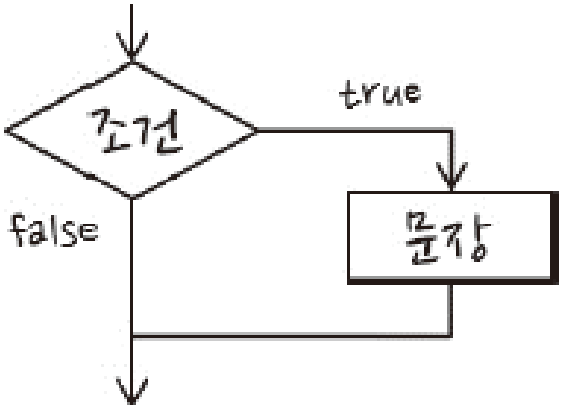
3.1 if 조건문

if 조건문

- 자바스크립트에서 가장 일반적인 조건문
  - 형태

```
if (불 표현식) {
    문장
}
```

  - 불리언 표현식이 true면 문장 실행
  - false면 문장 무시
    - 조건문에 의해 여러 문장을 실행할 때는 중괄호로 감싸야 함



3.1 if 조건문

◆ 예제 (1)

- 실행 결과 : '종료'가 입력된 경고창만 출력

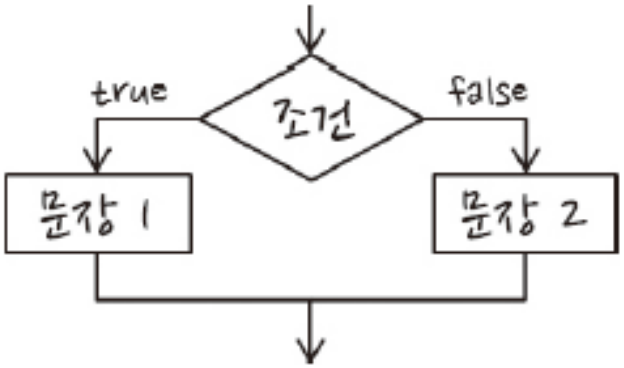
```
<script>
    // 조건문
    if (273 < 100) {
        // 표현식 "273 < 100"이 참일 때 실행합니다.
        alert('273 < 100 => true');
    }

    // 프로그램 종료
    alert('종료');
</script>
```

3.2 if else 조건문

if else 조건문

- 서로 반대되는 조건에 사용하는 조건문
- else 키워드는 if 조건문과 함께 사용하는 키워드
  - if 조건문의 바로 뒤에 붙여 사용
- 조건문의 형태

```
if (불 표현식) {
    문장 A
} else {
    문장 B
}
```



3.3 중첩 조건문

◆ 중첩 조건문

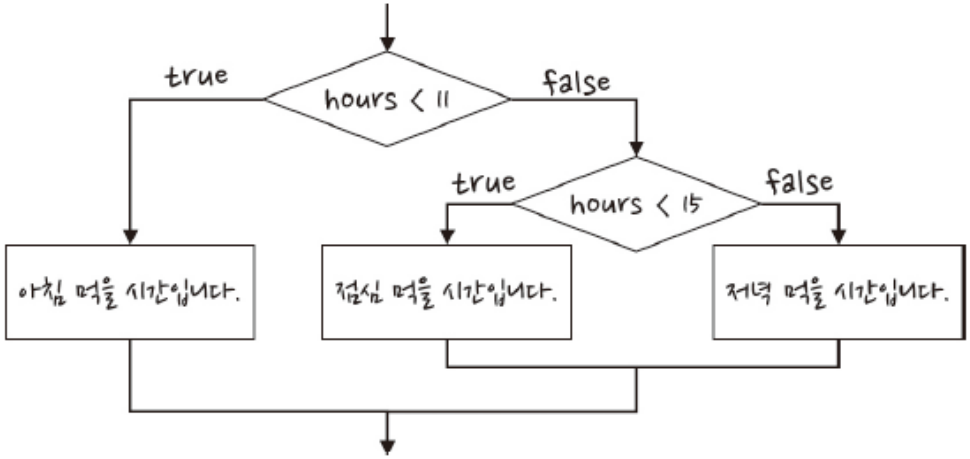
- 조건문 안에 조건문을 중첩해 사용하는 형식
- 중첩 조건문의 형태
  - 여러 번 중첩해도 상관 없음

```
if (불 표현식) {
    if (불 표현식) {
        문장
    } else {
        문장
    }
} else {
    if (불 표현식) {
        문장
    } else {
        문장
    }
}
```

3.3 중첩 조건문

◆ 중첩 조건문

- 진행과정




```
<script>
    // 변수를 선언합니다.
    var date = new Date();
    var hour = date.getHours();

    // 조건문
    if (hour < 11) {
        // "hour < 11"이 참일 때 실행합니다.
        alert('아침 먹을 시간입니다.');
    } else {
        // "hour < 11"이 거짓일 때 실행합니다.
        if (hour < 15) {
            // "hour < 15"가 참일 때 실행합니다.
            alert('점심 먹을 시간입니다.');
        } else {
            // "hour < 15"가 거짓일 때 실행합니다.
            alert('저녁 먹을 시간입니다.');
        }
    }
</script>
```

3.4 if else if 조건문

If else if 조건문

- If else if 조건문의 형태
- 충첩 조건문을 if else if 조건문은로 형태로 변경
  → 한쌍의 중괄호를 삭제

```
if (불 표현식) {
    문장
} else if (불 표현식) {
    문장
} else if (불 표현식) {
    문장
} else {
    문장
}
```

3.4 if else if 조건문

- if else if 조건문 실습
  - 사용자에게 숫자를 입력 받아 양수, 0, 음수를 구분하는 프로그램
  - 사용자에게 숫자를 입력 받아 홀수와 짝수를 구분하는 프로그램

3.5 switch 조건문

- switch 조건문의 기본 형태
  - default 부분은 생략 가능

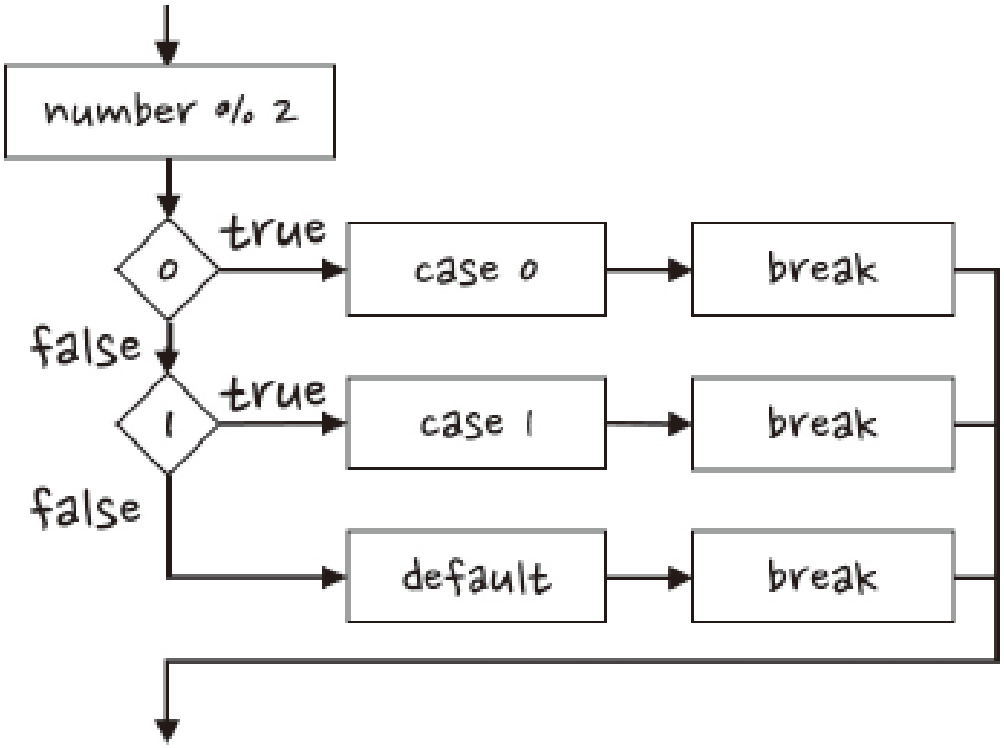
```
switch (비교할 값) {
    case 값:
        문장
        break;
    case 값:
        문장
        break;
    default:
        문장
        break;
}
```

3.5 switch 조건문

Break 키워드의 의미

- switch 조건문이나 반복문을 빠져 나가려고 사용하는 키워드
- switch 조건문의 괄호 안에는 비교할 값을 입력
  - 입력한 값을 기준으로 특정 코드 실행
  - 입력한 표현식과 case 키워드 옆의 표현식이 같음

» case 키워드 바로 다음에 오는 문장 실행



3.6 삼항 연산자

◆ 삼항 연산자

- 연산자지만 프로그램의 진행이 조건에 따라 변화 가능
- 삼항 연산자의 기본적인 형태
  - (불리언 표현식)?(참일 때 실행하는 문장):(거짓일 때 실행하는 문장)
- 삼항 연산자는 한 줄로 표시할 수 있을 때만 사용
- 코드 길이를 줄일 수 있다는 장점숫자의 부호를 비교하고 자연수인지 아닌지 판별하는 예제

```
(불 표현식)?(참일 때 실행하는 문장):(거짓일 때 실행하는 문장)
```

```
<script>
    // 변수를 선언합니다.
    var input = prompt('숫자를 입력해주세요.', '');
    var number = Number(input);

    // 조건문
    (number > 0) ? alert('자연수입니다.') : alert('자연수가 아닙니다.');
</script>
```

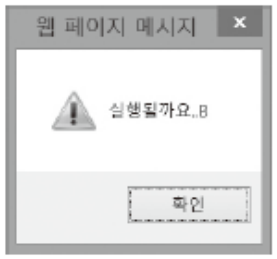
3.7 짧은 조건문

- 논리 연산자가 가지고 있는 특성을 조건문으로 사용
  - 논리합 연산자를 사용한 표현식은 뒤에 어떠한 값이 들어가도 항상 참
    - true || ○○○
    - 첫 번째 문장 : 좌변이 참이므로 우변 무시
    - 두 번째 문장 : 좌변이 거짓이므로 우변이 참인지 거짓인지 검사

코드 3-10 짧은 조건문(1)

```
<script>
    true || alert('실행될까요..A');
    false || alert('실행될까요..B');
</script>
```

그림 3-7 짧은 조건문



3.7 짧은 조건문

◆ 논리 연산자를 이용한 짧은 조건문

- 논리합 연산자

```
(불 표현식) || (불 표현식이 거짓일 때 실행할 문장)
```

- 논리곱 연산자

```
(불 표현식) && (불 표현식이 참일 때 실행할 문장)
```

JavaScript 반복문

4. 반복문

◆ 반복문

- 여러 번 반복해야 하는 일을 간편하게 처리하는 반복문

```
<script>
    alert('출력');
    alert('출력');
    alert('출력');
    alert('출력');
    alert('출력');
</script>
```

```
<script>
    for (var i = 0; i < 100; i++) {
        alert('출력');

    }
</script>
```

4.1 배열

◆ 배열이란?

- 여러 개의 변수를 한꺼번에 다룰 수 있는 자료형
- 객체 중 하나
- 대괄호([ ])로 생성, 대괄호 사용 후 안에 쉼표로 구분해 자료를 입력

코드 4-1 배열

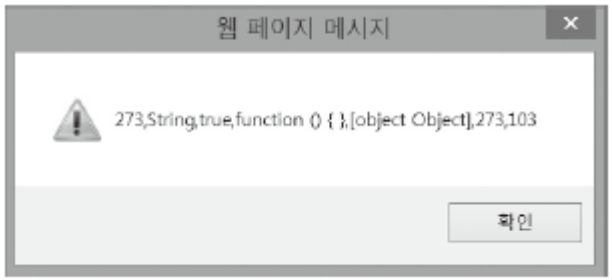
```
<script>
    // 변수를 선언합니다.
    var array = [273, 32, 103, 57, 52];
</script>
```

4.1 배열

◆ 배열 요소

- 어떠한 종류의 자료형도 배열 요소가 될 수 있음
- 열 안에 자바스크립트의 모든 자료형이 넣어져 있음

```
<script>
    // 변수를 선언합니다.
    var array = [273, 'String', true, function () { }, {}, [273, 103]];

    // 출력합니다.
    alert(array);
</script>
```



4.1 배열

◆ 배열 요소에 접근하는 방법

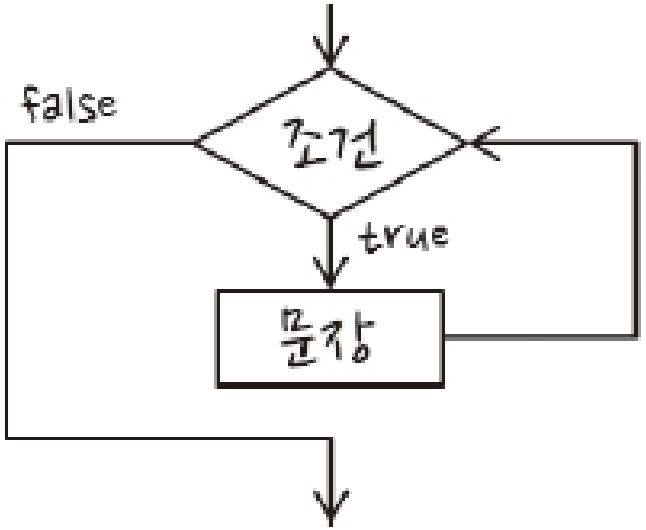
- 가장 앞에 있는 요소를 0번째라 표현

```
<script>
    // 변수를 선언합니다.
    var array = [273, 32, 103, 57, 52];

    // 출력합니다.
    alert(array[0]);
    alert(array[2]);
    alert(array[4]);
</script>
```

4.2 while 반복문

- While 반복문
  - 가장 간단한 반복문
  - if 조건문 형태가 비슷

```
while (불 표현식) {
    문장
}
```



4.2 while 반복문

- While 반복문
  - 무한루프 : 무한히 반복문이 반복
  - 익스플로러에서 무한루프의 비극적인 결과로 실행 하지 않도록 주의

```
while (true) {
    alert('반복문');
}
```

4.2 while 반복문

While 반복문

- while 반복문 종료
  - 숫자 증가
  - 내부적으로 변화

```
<script>
    // 변수를 선언합니다.
    var value = 0;

    // 반복문
    while (value < 5) {
        alert(value + '번째 반복');
        value++;
    }
</script>
```

4.2 while 반복문

While 반복문

- while 반복문 종료
  - 시간을 조건으로 변화

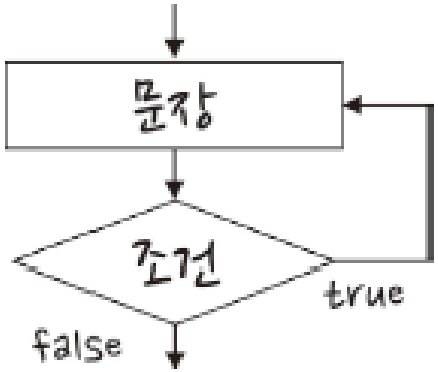
```
<script>
    // 변수를 선언합니다.
    var value = 0;
    var startTime = new Date().getTime();

    // 반복문
    while (new Date().getTime() < startTime + 1000) { value++; }

    // 출력합니다.
    alert(value);
</script>
```

4.3 do while 반복문

do While 반복문

- 조건의 참 거짓 여부와 상관없이 내부의 문장을 최소한 한 번은 실행해야 하는 경우 사용
- while 반복문과 형태가 비슷
- do while 반복문의 기본 형태

```
do {
    문장
} while (불 표현식);
```

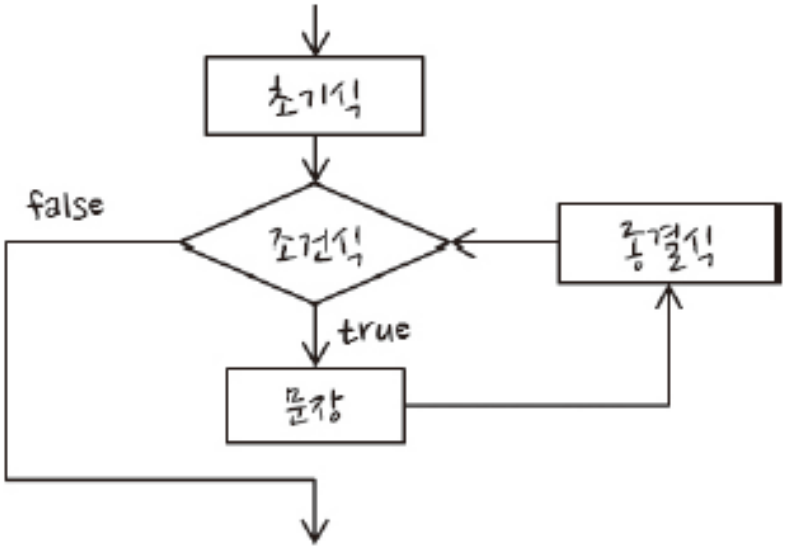


4.4 for 반복문

for 반복문

- 조건(while 반복문)보다 횟수에 비중을 둘 때 사용 하는 반복문
- 초기식과 종결식이 있음

```
for (초기식; 조건식; 종결식) {
    문장
}
```



4.4 for 반복문

- for 반복문 실행 순서
  - 초기식 실행
  - 조건식 비교/ 조건이 거짓이면 반복문 종료
  - 문장을 실행
  - 종결식 실행
  - 다시 '조건식 비교' 이후를 반복

4.4 for 반복문

for 반복문 사용

- 초기문에 선언하는 변수는 간단한 한 글자로 만듦
- 단순 for 반복문 : for (var i = 0; i < length; i++)와 같은 형태
- 역 for 반복문 : 배열 요소를 역으로 출력

```
for (var i = 0; i < length; i++) {
    문장
}
```

4.5 for in 반복문

for in 반복문의 형태

- for in 반복문은 단순 for 반복문과 같은 기능

```
for (var i in array) {

}
```

4.6 중첩 반복문

중첩 반복문

- 반복문을 여러 겹 중첩해서 사용
- 중첩 반복문의 문제, 스크립트 코드를 작성해보자.

4.7 break 키워드

- break 키워드
  - switch 조건문이나 반복문을 벗어날 때 사용
  - 다음 반복문은 조건이 항상 참으로 무한 반복
  - 무한루프는 break 키워드 사용으로 탈출

```
while (true) {

}
```

4.8 continue 키워드

- continue 키워드
  - 현재 반복을 멈추고 다음 반복을 진행
  - continue 키워드

```
<script>
    // 반복문

    for (var i = 0; i < 5; i++) {
        // 현재 반복을 중지하고 다음 반복을 수행합니다.
        continue;
        alert(i); // (ㅠㅜ)
    }
</script>
```

4.8 continue 키워드

continue 키워드

- 0부터 10까지의 짝수 합을 구하는 예제
  - 조건문을 사용해서 홀수일 때는 continue 키워드를 만나 바로 다음 반복으로
    - 짝수의 합만 구해짐

```
<script>
    // 변수를 선언합니다.
    var output = 0;

    // 반복문
    for (var i = 1; i <= 10; i++) {
        // 조건문
        if (i % 2 == 1) {
            // 홀수이면 현재 반복을 중지하고 다음 반복을 수행합니다.
            continue;
        }
        output += i;
    }
    // 출력합니다.
    alert(output);
</script>
```