

# Learning Script - 1

# Quick Start

보안기술 본부 김윤경

2018.10.30

# The subject comes first, the medium second.

주제가 제일 중요하고 도구는 그 다음이다.

- Richard Prince

스크립트는 단지 하나의 도구입니다.

가장 중요한 것은 원래 하려는 목표를 정확히 이해하고, 관련 지식을 갖춰 적합한 과정으로 수행하는 것입니다.

스크립트는, 할 일의 프로세스가 결정되고 검증된 후, 그 일에 도움을 주려고 사용하는 도구중 하나입니다.

## Script는 로 실행한다.

**스크립트는 일종의 대본입니다.**

목표를 달성하기 위한 작업 프로세스를, 명령 해석기의 문법에 맞춰 정의한 것입니다. 이 해석기를 인터프리터(Interpreter)라고 합니다.  
인터프리터는 스크립트에 정의된 일련의 과정을 하나씩 해석하여 수행합니다.

- (1) Bash, Windows 파워셸 같이 특정 OS의 명령어를 해석하여 수행하는 인터프리터,
  - (2) Python, Ruby, JavaScript, TCL와 같이 프로그래밍 문법을 갖춘 전용 인터프리터
- 크게 두 분류가 있습니다.

인터프리터의 속성, 문법 또는 명령어세트를 아는 것이 스크립트 활용에 중요합니다.

스크립트	확장자	인터프리터	시스템
Windows Batch	.bat	COMMAND.COM cmd.Exe	DOS Windows
Linux Shell	.sh	bash, csh, ksh...	LINUX 계열 OS (Red Hat, CentOS, Ubuntu, BSD)
Python	.py	Python	-
Java Script	.js	브라우저	-

# Java Script — 1.생일축하

자바스크립트의 인터프리터는 브라우저입니다.

다음과 같이 자바 스크립트가 포함된 html을 브라우저로 열면 어떻게 될까요?

```
<!doctype html>
<html>
<head>
  <title>Test</title>
</head>
<body>
  <h1>greeting</h1>
</body>
<script>
  birthday_member = "오픈베이스"
  member = prompt("이름을 입력하세요: ");
  if (member == birthday_member) {
    alert(member + "님 생일축하합니다!");
  }
  else {
    alert(member + "님 반갑습니다!");
  }
</script>
</html>
```

주황색 부분은 자바스크립트입니다.  
직접 쳐서 실행해보시기 바랍니다.

# Java Script — 1.생일축하

브라우저가 html과 자바스크립트를 해석/실행 합니다.

이름 입력 프롬프트에 "오픈베이스"를 넣으면 "생일축하합니다", 다른 이름을 넣으면 "반갑습니다"로 인사합니다.

```
<!doctype html>
<html>
<head>
  <title>Test</title>
</head>
<body>
  <h1>greeting</h1>
</body>
<script>
  birthday_member = "오픈베이스"
  member = prompt("이름을 입력하세요: ");
  if (member == birthday_member) {
    alert(member + "님 생일축하합니다!");
  }
  else {
    alert(member + "님 반갑습니다!");
  }
</script>
</html>
```

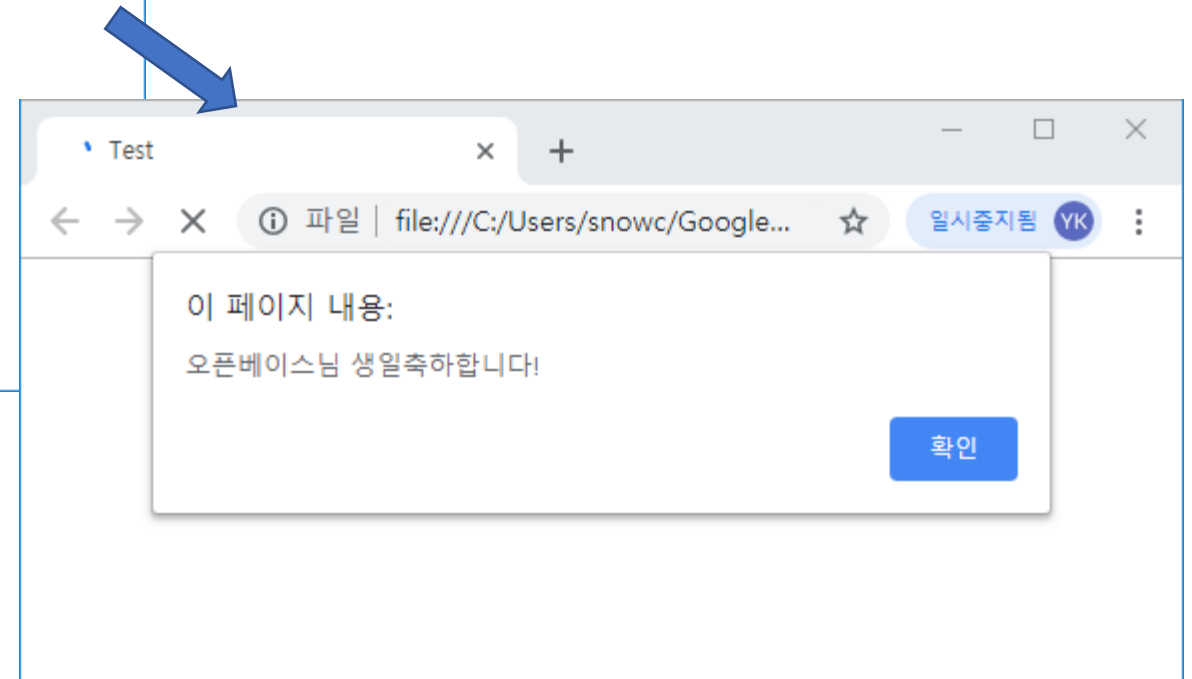
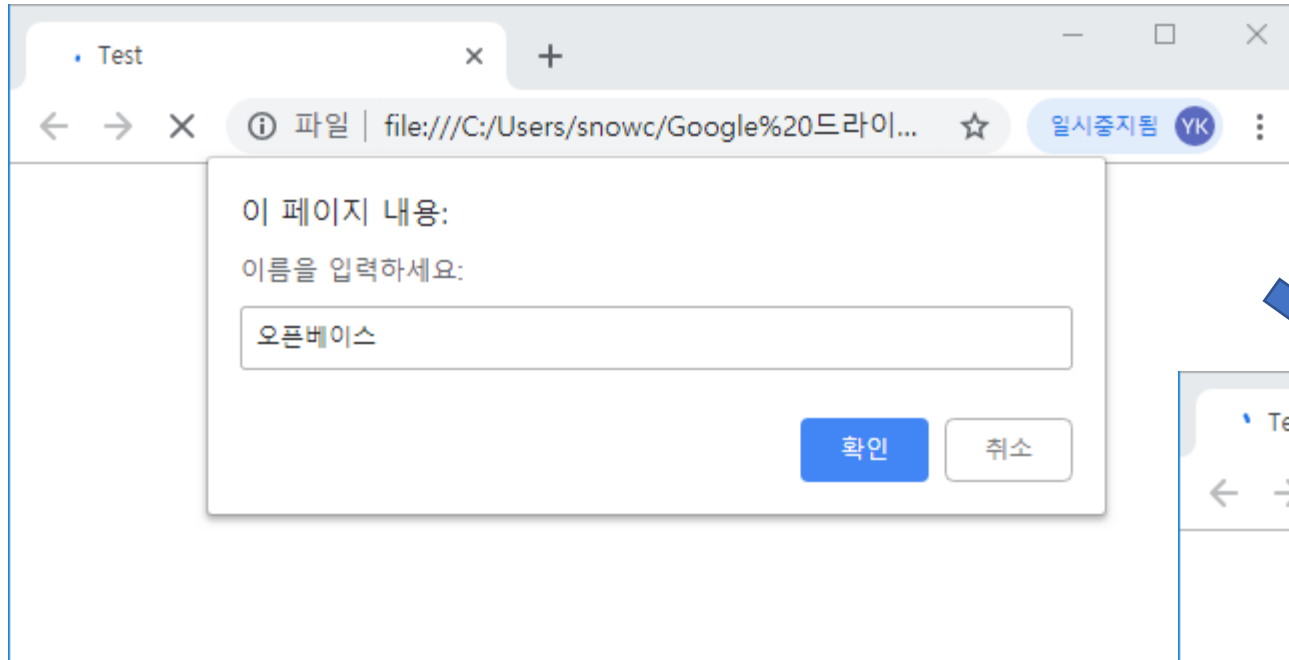
생일인 멤버의 이름은 " 오픈베이스"

입력 받은 이름을 생일인 멤버이름과 비교하여 일치하는 경우에 생일축하 인사를 한다.

# Java Script — 1.생일축하

브라우저가 html과 자바스크립트를 해석/실행 합니다.


이름 입력 프롬프트에 "오픈베이스"를 넣으면 "생일축하합니다", 다른 이름을 넣으면 "반갑습니다"로 인사합니다.



# Java Script — 1.생일축하

html에서 자바스크립트 부분을 greeting.js 파일로 분리 저장해도 똑같은 기능을 수행합니다.

```
<!doctype html>
<html>
<head>
  <title>Test</title>
</head>
<body>
  <h1>greeting</h1>
</body>
<script src="greeting.js"></script>
<!--
<script>
  birthday_member = "오픈베이스"
  member = prompt("이름을 입력하세요: ");
  if (member == birthday_member) {
    alert(member + "님 생일축하합니다!");
  }
  else {
    alert(member + "님 반갑습니다!");
  }
</script>
-- >
</html>
```



```
birthday_member = "오픈베이스"
member = prompt("이름을 입력하세요: ");
if (member == birthday_member) {
  alert(member + "님 생일축하합니다!");
}
else {
  alert(member + "님 반갑습니다!");
}
```

자바스크립트만 별도 파일에 분리하고 src 속성에 파일 이름을 지정하여 정의할 수 있습니다.  
<!-- --> 은 html의 주석입니다.



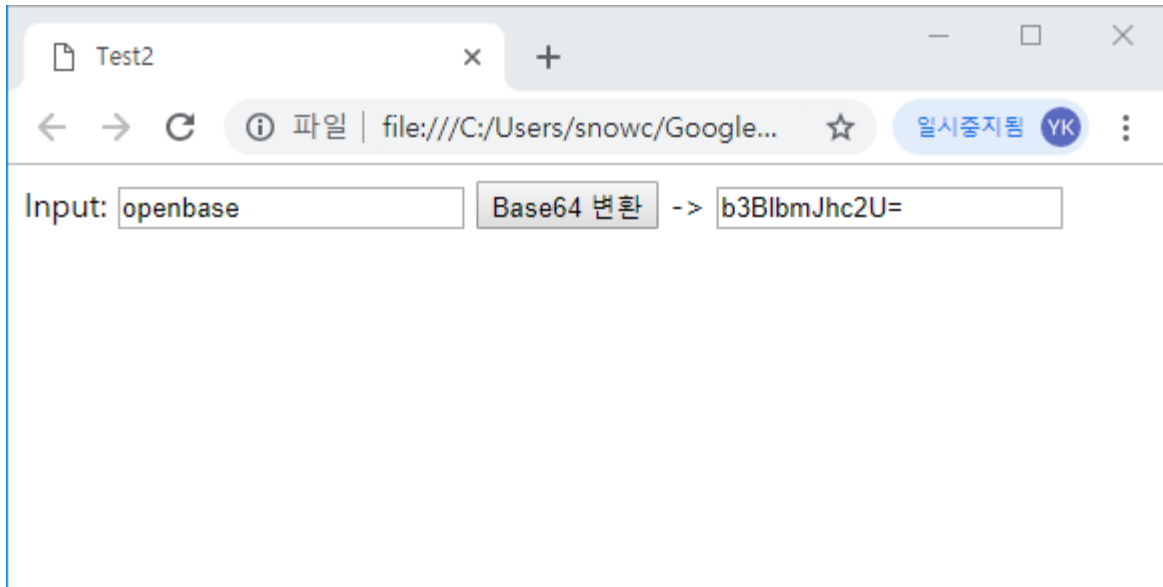
자바스크립트를 이용한 '평문 -> Base64' 변환기

```
<!doctype html>
<html>
  <head>
    <title>Test2</title>
  </head>
  <body>
    <form id="testform">
      Input: <input type="text" name="user_input">
      <input type="button" value="Base64 변환" onclick="convert2Base64()"></input>
      ->
      <input type="text" name="base64">
    </form>
  </body>
  <script type="text/javascript">
    function convert2Base64()
    {
      result = window.btoa(document.forms["testform"]["user_input"].value)
      document.forms["testform"]["base64"].value = result;
    }
  </script>
</html>
```

브라우저에는 btoa()라는 Base64 변환 함수가 내장  
돼 있습니다.  
html form을 이용해서 문자열을 입력받아 자바스크  
립트로 Base64 변환을 해 봅니다.

# Java Script — 2. Base64 변환기

자바스크립트를 이용한 '평문 -> Base64' 변환기



# Linux Shell — 기본 셸 Bash



리눅스는 GUI로도 액세스하지만 CLI(Command Line Interpreter)도 많이 쓰기 때문에 셸 스크립트를 빈번하게 사용합니다. 기본적인 사항을 알아봅니다.

```
# 리눅스에서 사용 가능한 셸
$ cat /etc/shells

# 현재 셸 확인
$ echo $SHELL
```

여러 리눅스 셸(인터프리터) 중 기본 셸은 bash(Born Shell) 입니다. 그러므로 **보통 리눅스에 로그인하면 default로 bash 셸을 하나 얻게 되는 것**입니다. bash만 알고 있어도 충분합니다.

아래와 같이 CentOS에서 셸 실행파일 4개는 모두 같은 bash 입니다.

`/bin/sh = /bin/bash = /usr/bin/sh = /usr/bin/bash`

```
$ ls -l /bin/sh
lrwxrwxrwx. 1 root root 4 Apr  6  2018 /bin/sh -> bash

$ ls -l /usr/bin/sh
lrwxrwxrwx. 1 root root 4 Apr  6  2018 /usr/bin/sh -> bash

$ ls -l /bin
lrwxrwxrwx. 1 root root 7 2018-04-06 14:24:29 /bin -> usr/bin
```

리눅스 셸은 명령어에 즉시 반응하는 인터프리터입니다.

스크립트 파일을 만들지 않고 프롬프트에서 덧셈을 해 보겠습니다.

아래와 같이 각각의 명령어를 치면서 "???"부분에 나오는 결과를 확인해 보세요.

```
$ a=1
$ b=100

$ ret1=$a+$b
$ echo $ret1
???
```

```
$ ret2=`expr $a + $b`
$ echo $ret2
???
```

```
$ let ret3=$a+$b
$ echo $ret3
???
```

변수에 값을 할당할 때는 '=' 앞뒤에 space를 두면 안 됩니다.

- a = 1 (X)

정의된 변수를 사용할 때는 '\$'를 앞에 붙입니다.

ret1의 결과는 나열한 문자열을 붙인 것입니다. '+'가 덧셈기호가 아닌 하나의 문자로 처리됩니다.  
수식을 연산하게 하기 위해서는 ret2, ret3의 경우 처럼 합니다.

```
$ a=1
$ b=100

$ ret1=$a+$b
$ echo $ret1
1+100

$ ret2=`expr $a + $b`
$ echo $ret2
101

$ let ret3=$a+$b
$ echo $ret3
101
```

가상의 상황으로, 서비스(포트 5601) 연결이 불규칙하게 자주 끊어지는데 메모리 사용량과 관련이 있는 것 같아서 연결상태가 끊어질 때 메모리 상태를 같이 볼 필요가 있다고 합니다.

다음과 같은 리눅스 명령어를 이용하겠습니다.

- `date`: 시스템 시간 확인. 체크한 시간을 확인하는데 이용
- `netstat -an | grep 5601`: 포트 5601 상태 보기
- `vmstat`: 시스템 메모리 사용 확인

위 명령어는 매번 체크할 때 세트로 사용할 것이므로 다음과 같이 bash 스크립트 파일로 만듭니다.

```
#!/bin/bash

date
netstat -an | grep 5601
vmstat
echo -----
```

파일을 저장한 후 다음과 같이 실행 퍼미션을 주고 실행하면 정보를 볼 수 있습니다.

```
$ chmod 755 my_script.sh
$ ./my_script.sh
```

계속 모니터링을 하려면 어떻게 해야 할까요?

언제 문제가 생길지 모르니 포트 연결 상태와 메모리 상태를 3초마다 모니터링하게 합니다. 다음과 같이 while 문을 적용합니다.

- 구조: while [ 유지 조건 ] do 실행할 일들 done
- 한 회차의 체크 후 다음 체크 사이에 3초 간격이 생기도록 sleep을 넣습니다.

```
#!/bin/bash

while [ 1 ]
do
    date
    netstat -an | grep 5601
    vmstat
    echo -----
    sleep 3
done
```

## #! (Shebang line)

- = Hash(#) + Bang(!)
- 스크립트를 실행할 해석기를 지정
- 스크립트 파일 첫 줄에 정의

while loop 유지조건의 1은 'true'입니다. 그러므로 이 스크립트는 외부에서 중단시키기 전까지 실행됩니다.

- [1] (X)
- [ 1 ] (O)

실행하면 3초마다 체크를 반복하며 결과를 화면에 찍습니다. 그런데 한밤에는 모니터링하는 사람이 없습니다. 데이터를 파일에 기록할 필요가 있습니다.

모니터링한 정보를 파일에 기록하여 유지합니다.

```
#!/bin/bash

while [ 1 ]
do
    date >> report.dat
    netstat -an | grep 5601 >> report.dat
    vmstat >> report.dat
    echo ----- >> report.dat
    sleep 3
done
```

> : 파일을 새로 만들어서 기록. 기존에 파일이 있으면 삭제하고 새로 만들.  
>> : 파일의 끝에 내용을 추가 기록. 기존에 파일이 없으면 새 파일을 만들어서 기록.

접속 터미널을 열어서 다음과 같이 실행중인 스크립트 프로세스를 확인합니다.

```
$ ps -ef | grep bash
```

데이터가 쌓이는 현황을 보려면 다음과 같이 합니다.

```
$ tail -f ./report.dat
```



# Windows Batch — 특정 종류 파일 작업

Windows에서 수행할 대량의 반복적인 작업, 자동으로 실행할 일련의 작업, 또는 자주 수행하는 명령어를 batch 스크립트로 구성해 실행합니다.

다음은 특정 폴더 하위의 \*.png파일(또는 모든 파일)을 지정 위치로 복사하는 스크립트입니다.

복사 대신, zip 파일 압축을 풀거나, curl로 파일을 API 서버에 POSTing하는데 이용 할 수도 있습니다.

```
@echo off

set RootDir=C:\data\source
set DstDir=C:\data\destination

mkdir %DstDir%

rem root 하위 디렉토리를 순환하면서 모든 png 파일을 특정 위치로 복사
rem /r: recursive folders, %%f : each file
for /r "%RootDir%" %%f in (*.png) do copy "%%f" "%DstDir%"

echo copy finished
```

자료 :

[https://secuwave.github.io/secure3/learn\\_script/main](https://secuwave.github.io/secure3/learn_script/main)