

The background features a series of flowing, wavy green lines that create a sense of movement and depth. These lines are layered, with some appearing more prominent than others, and they span the width of the image. The colors range from a vibrant lime green to a deeper, more muted green. The overall effect is modern and dynamic.

Javascript Language Specification

자바스크립트의 정의

■ 등장 배경

- 클라이언트쪽에서 독립적으로 실행되는 프로그램을 작성하기 위한 스크립트 언어로 넷스케이프사에서 LiveScript 발표
- 썬마이크로시스템사와 공동으로 라이브스크립트를 확장한 JavaScript 탄생
- ECMA 인터내셔널에서 넷스케이프사의 자바스크립트와 마이크로소프트사의 Jscript 기반 ECMAScript(ECMA-262)를 Technical Committee 39(TC-39)에서 정의

- 자바 스크립트는 스크립트 언어의 특성 상 플랫폼에 독립적이며 넷스케이프, 익스플로러 등 모든 웹 브라우저에서 동일한 실행 결과를 얻을 수 있음(무료, 쉽고 자유로움)

자바스크립트의 특징

- 웹 문서(XHTML)에 끼워서 사용하는 스크립트 언어다.
- 웹 브라우저에서 웹 문서를 실행할 때 프로그램 코드가 해석되기 때문에 웹 브라우저에 자바스크립트 해석기가 있어야 한다.
- 인터프리터 언어의 형태를 띄고 있다.
- 컴파일 과정을 거치지 않기 때문에 비교적 자료형 조사를 철저하게 하지 않는다
- 몇 가지 객체 지향 요소를 포기하고 있다. 가령 클래스를 정의할 수 없다든지, 클래스를 상속할 수 없는 것이 이러한 부분이다. 대신 객체를 정의하여 사용할 수는 있다.

구분	자바스크립트	자바
작성 및 사용 방법	XHTML 내에 작성하며, 가공하지 않고 그대로 사용한다.	자바 에디터에서 작성 후 컴파일하며, 자바 가상 머신을 사용하여 읽는다.
방식	인터프리터 방식	컴파일러 방식(인터프리터 겸)
OOP (객체 지향)	OOP 기반으로 되어 있지만 클래스가 없고 상속할 수도 있다.	완벽한 OOP로 클래스를 가지고 있으며 상속할 수 있다.
보안성	소스가 노출되어 보안성이 없다.	컴파일된 파일로 만들어져 보안성이 있다. 하지만 디컴파일러를 통해서 역으로 변환할 수 있다.

자바스크립트의 기능

- 일반적으로 XHTML로만 작성된 웹 문서는 단순히 정보만을 제공할 뿐, 웹 문서에 동적으로 움직일 수 있는 프로그램적인 요소를 추가할 수 없다. 이 프로그램적인 요소를 자바스크립트를 이용하면 해결할 수 있다.
- 자바스크립트는 어떤 이벤트에 대한 반응을 할 수 있도록 웹 문서를 구성할 수 있다. 예를들어, 사용자가 웹 문서 내 특정 부분을 클릭했거나 웹 문서가 호출된 후에 웹 문서에서 어떤 작업이 진행되도록 구성할 수 있다.
- 자바스크립트는 웹 문서 내의 XHTML 요소의 내용을 읽거나 바꿀 수 있다.
- 자바스크립트는 클라이언트의 정보를 서버에 보내기 전에 먼저 검증할 수 있으며, 이로 인해 서버의 작업 부담도 줄여줄 수 있다.
- 자바스크립트는 클라이언트가 어떤 웹 브라우저를 사용하는지 알아내어 사용자의 웹 브라우저에 맞는 웹 문서를 호출하는 맞춤형 서비스를 제공할 수 있다.
- 자바스크립트는 클라이언트 컴퓨터에 쿠키를 새로 만들거나 생성된 쿠키를 삭제 및 회수할 수도 있다.

자바스크립트의 장점과 단점

■ 자바스크립트의 장점

- 작업 속도가 빠르다
 - 자바스크립트는 HTML 파일 내에서 작성할 수 있으므로 개발 속도가 빠르다.
 - 자바, C/C++ 등 프로그래밍 언어의 기본적인 틀을 그대로 갖추고 있어 사용자로부터 익숙한 작업 가능.
- 운영체제의 제한을 받지 않는다.
- 배우기 쉽다.
 - 객체의 단순화로 인한 프로그램 제작 용이

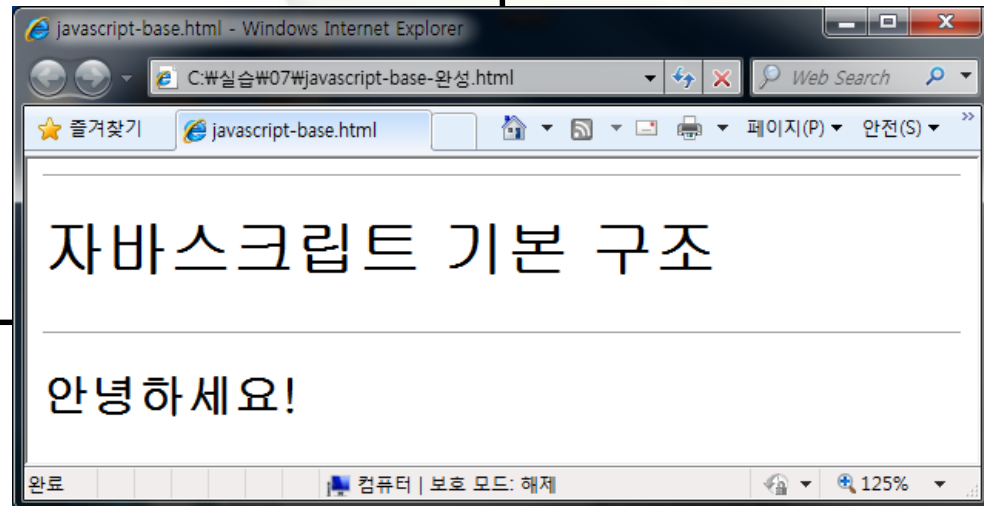
■ 자바스크립트의 단점

- 소스 코드가 노출된다. 컴파일하지 않는 언어이므로 복사하여 사용할 수 있다.
- 한정된 객체와 객체 함수다. 제한된 객체와 객체 함수를 사용하므로 복잡한 게임, 메신저, 채팅방 등의 프로그램은 개발할 수 없다.

자바스크립트의 기본 구조

- <script> 태그를 사용
- type 속성값으로 “text/javascript” 를 입력

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=euc-kr />
    <title> javascript-base.html </title>
  </head>
  <body>
    <hr /><h1> 자바스크립트 기본 구조 </h1><hr />
    <script type="text/javascript">
      <![CDATA[
        document.write("<h2>안녕하세요!</h2>");
      //]]>
    </script>
  </body>
</html>
```



자바스크립트 실행 위치 (1)

- 자바스크립트가 <head> 태그 영역에 배치되는 경우

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
03 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">  
04   <head>  
05     <meta http-equiv="content-type" content="text/html; charset=euc-kr" />  
06     <title> javascript-head.html </title>  
07     <script type="text/javascript">  
08       function message( ){  
09         alert("이 메시지 상자는 onload 이벤트가 발생할 때 보인다.");  
10       }  
11       document.write("<h2>자바스크립트 세계에 오신 걸 환영합니다.</h2>");  
12     </script>  
13   </head>  
14   <body onload="message( );">  
15     <hr /><h1> 자바스크립트가 head 요소 영역에 위치했을 때 </h1><hr />  
16   </body>  
17 </html>
```

2. 자바스크립트 실행 위치 (2)

- 자바스크립트가 <body> 태그 영역에 배치되는 경우

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
02 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
03 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ko" lang="ko">  
04   <head>  
05     <meta http-equiv="content-type" content="text/html; charset=euc-kr" />  
06     <title> javascript-body.html </title>  
07   </head>  
08   <body>  
09     <hr /><h2> 자바스크립트가 body 요소 영역 내에 위치했을 때 </h2><hr />  
10     <script type="text/javascript">  
11       document.write("자바스크립트의 문장");  
12     </script>  
13     <p> xhtml의 문장 </p>  
14   </body>  
15 </html>
```


2. 자바스크립트 실행 위치

- 자바스크립트를 외부 파일로 사용되는 경우
 - <head> 태그 부분이든 <body> 태그 부분이든 기능을 구현할 곳에 삽입
 - 외부 파일 자바스크립트의 확장자는 ".js"
 - script element의 src 속성값이 호출할 자바스크립트의 파일명
 - 사용 방법

```
<script type="text/javascript" src="호출할 자바스크립트 파일명.js"></script>
```

기본 구문 규칙 - 표현식과 문장

■ 표현식이란?

- 값을 만들어내는 간단한 코드

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

■ 문장이란?

- 하나 이상의 표현식이 모인 것
- 문장이 모여 프로그램 구성
- 문장의 끝에는 세미콜론을 찍어 문장의 종결을 알려줌

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

기본 구문 규칙 - 키워드

- 자바 스크립트에 이미 정의된 특별한 의미가 있는 단어
 - 초기 키워드

표 1-2 자바스크립트의 키워드 (1)

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

- W3C에서 자바스크립트 프로그램 작성시 추가 키워드

표 1-3 자바스크립트의 키워드 (2)

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

기본 구문 규칙 - 식별자

- 식별자는 변수, 함수 등에 붙여진 이름

- 필수 식별자 규칙

- 키워드를 사용할 수 없음
- 숫자로 시작할 수 없음
- 특수 문자는 _와 \$만 허용
- 공백 문자 포함할 수 없음

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

- 권장 식별자 규칙

- 모든 언어의 문자 사용 가능하나 영문자 사용이 개발자들 사이 관례
- 대상의 의미를 정확하게 표현할 수 있는 단어 사용
- 생성자 함수의 이름은 항상 대문자로 시작
- 변수, 인스턴스, 함수, 메서드의 이름은 항상 소문자로 시작
- 여러 단어로 이뤄진 식별자는 각 단어의 첫 글자를 대문자로

기본 구문 규칙 - 주석

- 엔진에 의해 처리되지 않는 영역으로 코드에 대한 설명을 위해 작성
- HTML 태그 주석
 - <!-- 주석 내용 -->로 문자열을 감싸 생성
- 자바스크립트 주석
 - 한 줄 주석 : //뒤의 문장은 실행되지 않음
 - 영역 주석 : /* 주석 내용 */을 사용해 주석 영역 표현

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

변수

- 프로그래밍에서 데이터를 담을 수 있는 메모리 할당 영역
- 변수 선언

```
var A=8;  
var people_name="kim";
```

- 잘못 된 변수 이름 사용 예

변수명	올바르지 않은 이유	올바른 변수명의 예시
Str*	특수문자 포함	Str
1234Num	숫자로 시작	Num1234
if	예약어 사용	ifn
마우스	한글 사용	mouse

자료형

- 크기, 저장 형식 등 데이터의 정보를 의미
- 명시적인 자료형 표현을 사용하지 않습니다.
- 데이터에 따라 자료형 구분
 - 문자열
 - 숫자 (정수, 부동소수점)
 - Boolean
 - null

숫자 자료형

■ 정수

```
var num1 = 64; // 10진수
var num2 = 0100; // 8진수, 10진수로 64
var num3 = 0x40; // 16진수, 10진수로 64
document.write("num1의 값은 " + num1 + " 입니다 <br/>");
document.write("num2의 값은 " + num2 + " 입니다 <br/>");
document.write("num3의 값은 " + num3 + " 입니다 <br/>");
```

■ 실수

```
var num1 = 3.2541, num2 = 2e3; // 소수점 4자리수와 2,000의 지수표현
document.write("num1은 "+num1+"이고 num2는 "+num2+"입니다 <br/>");
```


문자형

- 큰따옴표(“ ”) 또는 작은따옴표(‘ ’) 사이에 들어가는 0개 이상의 문자들

```
var str = "야구"; // 나중에 바뀔 수 있는 문자열을 변수에 입력
document.write("사람들이 좋아하는 \"스포츠\"는 "+str+"이고 "+str+"는 \'재미\' 있다.");
```

- 이스케이프 문자 : 자바스크립트에서 사용되는 특수한 문자형

특수문자	내용
\n	커서를 다음 줄로 이동
\t	커서를 탭(Tab) 이동
\b	커서를 앞 문자를 지우며 이동
\f	커서를 다음 페이지 처음으로 이동
\r	커서를 그 줄의 처음으로 이동
\"	큰따옴표
\'	작은따옴표
\\	역슬래시

논리형

- 크기가 1비트. 참(true)과 거짓(false)의 두 값만 가짐.
- 비교(<, >, =) 연산, 논리 연산, 함수 반환 값으로 사용.

```
var t = (3>2);  
var f = (2>3);  
document.write("참일 때에는 "+t+"로표현되 고거짓일때에는 "+f+"로 표현된다.");
```

연산자

- 데이터의 처리 명령을 표현하는 기호
- 종류
 - 산술 연산자
 - 대입 연산자
 - 관계 연산자
 - 논리 연산자
 - 증감 연산자
 - ...

산술 연산자

- 정수형 변수, 실수형 변수, 정수값, 실수값을 이용하여 산술 연산

연산자	사용 예	설 명
-	$C = -A$	A 값이 양수이면 음수로, 음수이면 양수로 변환
-	$C = A - B$	A에서 B를 뺀 차를 C에 저장, 뺄셈 연산
+	$C = A + B$	A와 B의 합을 C에 저장, 덧셈 연산
*	$C = A * B$	A와 B의 곱을 C에 저장, 곱셈 연산
/	$C = A / B$	A를 B로 나눈 몫을 C에 저장, 나눗셈 연산
%	$C = A \% B$	A를 B로 나누었을 때 나머지를 C에 저장, 나머지 연산
++	$C = ++A$	A 값에 1을 더한 값을 C에 저장, 증가 연산
--	$C = --A$	A 값에서 1을 뺀 값을 C에 저장, 감소 연산

대입 연산자

- 연산에 사용되는 변수와 연산 결과를 저장하는 변수가 동일할 경우에 사용
- 수식을 축약하여 표현할때 사용

연산자	사용 예	동일 연산	설명
<code>+=</code>	<code>C += A</code>	<code>C = C + A</code>	C와 A의 합을 C에 저장
<code>-=</code>	<code>C -= A</code>	<code>C = C - A</code>	C에서 A를 뺀 차를 C에 저장
<code>*=</code>	<code>C *= A</code>	<code>C = C * A</code>	C와 A의 곱을 C에 저장
<code>/=</code>	<code>C /= A</code>	<code>C = C / A</code>	C를 A로 나눈 몫을 C에 저장
<code>%=</code>	<code>C %= A</code>	<code>C = C % A</code>	C를 A로 나누었을 때 나머지를 C에 저장

관계 연산자

- 변수 사이의 일치성 판단하는 조건문에 사용
- 두 개의 값을 비교하여 참(true)과 거짓(false) 값으로 반환.

연산자	사용 예	설명
==	A == B	A와 B의 값이 같은지 비교
===	A === B	A와 B의 값뿐만 아니라 자료형도 같은지 비교
!=	A != B	A와 B의 값이 다른지 비교
>	A > B	A가 B보다 큰지 비교
>=	A >= B	A가 B보다 크거나 같은지 비교
<	A < B	A가 B보다 작은지 비교
<=	A <= B	A가 B보다 작거나 같은지 비교

논리 연산자

- 변수 또는 값의 조합이 논리적으로 참(true)인지 거짓(false)인지를 판별
- True : “1” , false : “0” 반환

연산자	사용 예	설명
&&	A && B	and 연산, A와 B 둘 다 참일 때만 참이다.
	A B	or 연산, A와 B 둘 중에 하나만 참이면 참이다.
!	!A	not 연산, A가 참이면 거짓이 되고, 거짓이면 참이 된다.

A	B	A && B	A B	!A
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

삼항 연산자

- 조건 결과값이 참(true)이면 “처리내용 1” , 거짓(false)이면 “처리내용 2” 반환

값 = 조건식 ? 처리내용 1 : 처리내용 2 ;

```
val = ("a" == "a") ? "같습니다." : "같지 않습니다."  
document.write("a와 a가 같나요? " + val + "<br />");  
val = ("a" == "b") ? "같습니다." : "같지 않습니다."  
document.write("a와 b가 같나요? " + val + "<br />");
```


증감 연산자

- “++” : 증가 연산자. 1씩 증가
- “--” : 감소 연산자, 1씩 감소
- 전위 증감 연산사 : 변수 앞쪽에 위치
- 후위 증감 연산사 : 변수 뒤쪽에 위치

연산자	설 명	사용 예	연산자 구분
++	1씩 증가	++A : 먼저 1 증가한 값 출력 A++ : 처음은 증가하지 않고 다음 값부터 1씩 증가	전위 증감 후위 증감
--	1씩 감소	--A : 먼저 1 감소한 값 출력 A-- : 처음은 감소하지 않고 다음 값부터 1씩 감소	전위 증감 후위 증감

제어문

- 실행 코드의 실행 흐름을 논리적으로 제어하는 구문
- 선택문과 반복문으로 구분
- 기본 구조
 - 선택문 : 조건 검사 ➔ 선택
 - 반복문 : 조건 검사 ➔ 반복
- 종류
 - 선택문
 - if - else
 - switch - case
 - 반복문
 - while
 - for
 - do - while

if - else 구문

▪ if 문

- if 문의 조건식에 만족(true)하면 { } 안의 문장을 수행

```
if(조건식) {  
    문장... 문장...  
}
```

▪ if~else 문

- 조건식에 만족하면 문장 1 수행, 만족하지 않으면 else 다음문장 수행

```
if(조건식) { 문장 1... }  
else { 문장 2... }
```

▪ 다중 if~else 문

- 여러 문장 중에서 조건에 만족하는 문장을 실행

```
if(조건식) { 문장1... }  
else if(조건식) { 문장2... }  
else if(조건식) { 문장3... }  
else if(조건식) { 문장4... }  
else { 문장5... }
```

switch – case 구문

- 다중 if 문을 대신하여 switch 문을 사용하면 간단하게 구현이 가능하다.
- switch 문에 제공된 식의 결과 값과 일치하는 case 문 실행

```
switch(변수) {  
    case 상수1 : 문장1; break;  
    case 상수2 : 문장2; break;  
    case 상수3 : 문장3; break;  
    default : 문장4;  
}
```

반복문 (1)

▪ while 문

- 주어진 조건을 만족하는 동안 반복해서 지정된 기능을 수행
- 조건이 참인 동안 계속해서 문장이 실행되고, 거짓이 되는 순간 루프를 벗어남
- 초기값이 while 문 밖에 존재

```
while(조건식) {  
    문장...  
}
```

▪ do~while 문

- do~while 문은 최소한 한 번은 루프를 실행 - while 문과의 차이점
- 조건이 아래에 있기 때문에 루프를 실행한 후에 조건을 체크

```
do {  
    문장...  
} while(조건식)
```

반복문 (2)

▪ for 문

- 초기값과 조건식, 증가분을 한꺼번에 지정하는 반복문
- 일반적으로 반복 횟수가 정해져 있거나 확인할 수 있을 때 사용
- 변수에 초기값을 주고 이 값이 조건을 만족시키는 동안에만 작동

```
for (초기값; 조건식; 증감식) {  
    실행문...  
}
```

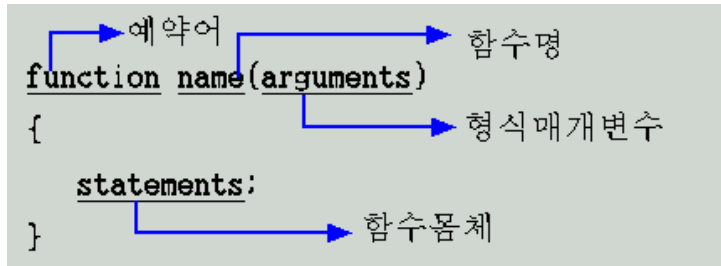
▪ 다중 for 문

- 단일 for 문을 동시에 여러 번 사용하는 반복문
- 조건이 아래에 있기 때문에 루프를 실행한 후에 조건을 체크

```
for (초기식; 조건식; 증감식) {  
    실행문...  
    for (초기식; 조건식; 증감식) {  
        실행문...  
    }  
}
```

함수

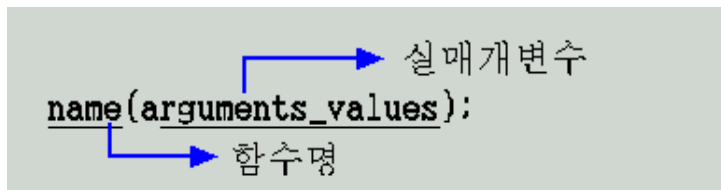
- 프로그램 내에서 특정 작업 수행 위해 독립적으로 만들어진 하나의 단위
- 예약어 'function' 으로 시작
- 함수의 시작과 끝을 구분하는 기호: 중괄호({ })
- 호출 영역에서 함수 영역으로 데이터를 전달하기 위해 전달인자사용
- 함수 정의 구문 구조



```
function name(arguments)
{
    statements;
}
```

The diagram illustrates the components of a function definition. Blue arrows point from labels to parts of the code: '예약어' (keyword) points to 'function', '함수명' (function name) points to 'name', '형식매개변수' (formal parameter) points to 'arguments', and '함수몸체' (function body) points to the block between the curly braces. The word 'statements;' is also present within the body.

- 함수 호출 구문 구조



```
name(arguments_values);
```

The diagram illustrates the components of a function call. Blue arrows point from labels to parts of the code: '실매개변수' (actual parameter) points to 'arguments_values', and '함수명' (function name) points to 'name'.