



자바스크립트 기본

- 2.1 자바스크립트 선언 및 실행하기
- 2.2 데이터 입/출력
- 2.3 주석문, 변수와 연산자, 자료형
- 2.4 배열 Array 객체
- 2.5 함수의 정의와 호출
- 2.6 자바스크립트 객체의 멤버 접근
- 2.7 이벤트 핸들러와 이벤트 처리

2.1 자바스크립트 선언 및 실행하기

2.1.1 자바스크립트 실행하기

그럼, 이제부터 자바스크립트를 직접 사용해보도록 하자. 자바스크립트는 다양한 브라우저에서 실행되어야 하는 특징과 강한 유연성으로 인해 처음에 쉽게 생각했다간 나중에 예상 외의 결과로 마음에 상처를 받을 수도 있다. 그러므로 독자들도 자바스크립트와 같은 인터프리터 언어를 공부하기 위해서는 어느 정도 유연한 사고방식을 갖는 것이 좋을 것이다. 특히, 자바나 C 언어와 같은 문법이 엄격한 프로그래밍 언어를 먼저 사용해본 사람들이라면 더욱 명심해줄 길 바란다.

자바스크립트를 실행하기 위해선 `<script>`와 `</script>` 태그 사이에 자바스크립트 문장을 넣어주면 된다. 혹시나 있을 자바스크립트 기능을 꺼놓은 브라우저나 자바스크립트가 아예 지원되지 않는 브라우저에서 안내 문구가 출력하도록 하기 위해서는 `<noscript>` 태그를 이용할 수 있다. 소스 2-1은 자바스크립트를 구현하기 위해 `<script>` 태그를 선언한 것이다. 그리고 아래엔 스크립트가 지원되지 않는 브라우저에서 안내 문구를 출력할 목적으로 `<noscript>` 태그를 함께 사용한 예다.

소스 2-1 HTML 페이지 내에서 자바스크립트 코드 작성하기

CODE js2.1.1.html

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2 <html>
3 <head>
4 <!-- 자바스크립트가 지원되는 브라우저에서 실행 -->
5 <script type="text/javascript">
6     console.log("head 태그 안에서 선언되었다.");
7 </script>
8 <!-- 자바스크립트가 지원되지 않는 브라우저를 위한 조치 -->
9 <noscript>
10     이 브라우저는 자바스크립트가 지원되지 않습니다.
11 </noscript>
12 </head>
13 <body>
14
15 </body>
16 </html>
```

자바스크립트 문장은 언뜻 보기에 C 언어나 자바 언어와 유사한 부분이 많다. 그러나 내부적으로 볼 때 자바 언어는 컴파일러를 통해 번역된 바이트 코드를 컴퓨터가 통째로 해석하는 방식인데 비해서 자바스크립트는 인터프리터 방식으로 작동한다. 즉, 웹 브라우저에 내장되어 있는 자바스크립트 파서가 소스 코드를 한 줄씩 읽고 해석한다는 점에서 작동 원리나 문법의 상당 부분이 컴파일 언어와는 차이가 있다. 인터프리터 방식이란, 이와 같이 한 줄을 읽어들이고 그 결과를 화면에 뿌리는 방식이라고 이해하면 된다.

HTML 문서 안에 자바스크립트 코드를 문법에 맞게 작성하였다면 일단 실행해보도록 한다. 자바스크립트를 실행하는 방법은 매우 간단하다. 자바스크립트 코드가 포함된 HTML 파일을 웹 브라우저로 실행하면 끝이다. 이처럼 작성과 실행이 간단하기에 프로그래밍을 처음 공부하는 사람에겐 쉽게 접근할 수 있어서 좋다. 자바와 같은 언어는 설치가 반이라고 할 정도로 JDK를 설치한 후 코드를 작성하기까지가 초보자에겐 좀 까다로운 편이다.

소스 2-1의 코드는 아무런 문제가 없지만 실행하면 화면에 아무것도 보이지 않는다. 왜냐하면, `console.log()`의 기능은 브라우저 화면에 무엇을 출력하는 것이 아니라 브라우저에 내장되어 있는 Developer Tools의 Console(콘솔) 패널에 출력되기 때문이다. Console 기능은 인터넷 익스플로러 8 이하의 버전에서는 사용이 불가능한 기능이다. 이 예제는 구글 크롬(Google Chrome) 브라우저에서 실행하였다. (Developer Tools는 Inspector라고도 한다.)

크롬 웹 브라우저에서 Developer Tools를 실행하는 방법은 다음과 같다. 인터넷 익스플로러 9 이상에서는 키보드의 단축 버튼 F12를 눌러서 비슷한 기능을 실행할 수 있다.

■ 결과 확인 실행 단계 설명(크롬 브라우저에서 실행했을 경우)

- **1단계:** 웹 브라우저의 화면에서 마우스 오른쪽 버튼을 클릭한다 (우클릭).
- **2단계:** 브라우저 화면 위에 '바로가기' 메뉴 목록이 열린다.
- **3단계:** 다음 '바로가기' 메뉴 목록 하단에 있는 '요소 검사'를 선택하면 그림 2-2와 같이 Developer Tools 창이 뜬다.

Developer Tools가 실행되면 최초 Elements 탭이 자동으로 보여진다. 창의 상단 우측 맨 끝에 위치한 Console 탭을 열어보면 출력 내용 및 에러에 대한 설명을 볼 수 있다.

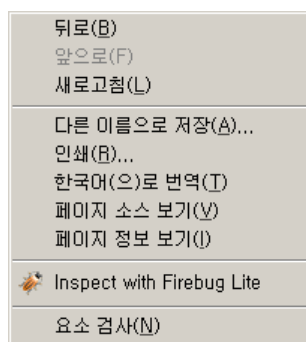



그림 2-1 실행 결과 확인. 크롬 화면에서 오른쪽 클릭_1단계

혹은 Elements 탭이나 Source 탭을 실행해서 창 하단의  부분을 선택해서 Console 기능을 함께 사용할 수 있다. Elements 탭에서는 Styles 항목의 스타일 속성을 변경하면서 화면에 보여지는 레이아웃 등의 결과를 즉시 확인할 수 있다. Elements 탭의 여러 기능을 잘 사용하면 매우 편리하다.

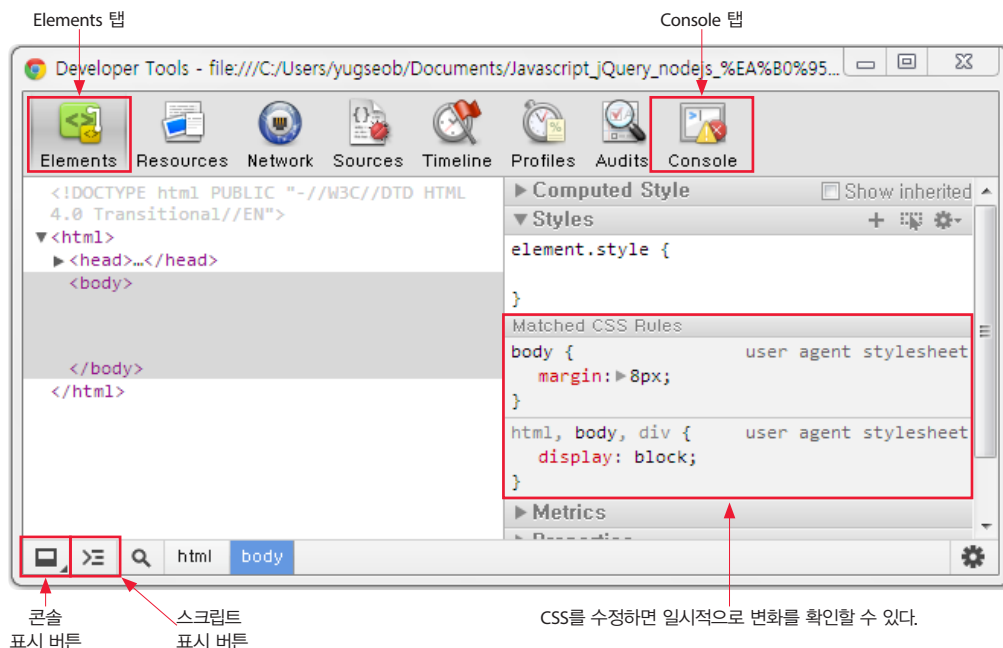
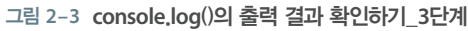


그림 2-2 console.log()의 출력 결과 확인하기_2단계

그림 2-3은 Elements 탭과 Console 탭을 함께 사용한 것을 참고하기 위한 것으로서 소스 2-1의 실행 결과와는 관련이 없다. 이 화면은 구글의 메인 화면에서 요소 검사를 실행한 것이다.

Developer Tools의 Console 탭을 열면 `<script>` 태그의 `console.log()`에 작성했던 내용이 콘솔 창에 출력된 것이 보인다. 자바스크립트가 실행되기 위해 특별히 사용해야 할 브라우저는 없다. 오히려 크로스 브라우징이 지원되는 코드를 만들기 위해선 여러 브라우저에서 자신의 코드를 테스트할 것을 권장한다. 이런 식의 테스트를 게을리하면 다 완성된 프로젝트를 열어야 하는 경우도 생긴다. 시쳇말로, ‘다 된 밥에 재 뿌리는 식’이 된다. 고스톱에서도 남의 점수를 꼭 보면서 쳐야 낭패를 안 당하는 이치와 같다. 늘 느끼는 것이지만, 필자가 얘기하는 비유가 더 어렵게 느껴진다. 앞으로 쓸 내용도 많은데 큰일이다.



2.1 _ 자바스크립트 선언 및 실행하기 15

2.1.2 자바스크립트 선언 위치

자바스크립트 코드가 작성되는 `<script>` 태그는 HTML 문서의 어느 곳에 오든지 상관없다. `<head>` 태그나 `<body>` 태그 안에 위치할 수 있는 것은 물론이고, 필요하다면 파일 내의 `<html>` 이 선언되는 부분 외부나 위에 있어도 아무런 문제가 없다. 그뿐만이 아니라 HTML 파일 내에 HTML 코드가 전혀 없어도 `<script>` 코드를 작성하고 실행하는 데는 큰 문제가 없다. 하지만 HTML 파일을 작성할 때 꼭 표준을 지키는 것이 좋다. 특히, 요즘은 법적으로 웹 접근성을 강제하기 때문에 그것을 고려하지 않은 개발을 할 경우 엄청난 손해를 감수할 수도 있다. 부디 웹 표준과 접근성에 대해서 잘 알아보고 작업하길 바란다.

소스 2-2 자바스크립트 코드가 작성되는 위치

CODE js2.1.2.html

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2 <script type="text/javascript">
3     console.log("body 태그 위에서 선언되었다.");
4 </script>
5 <html>
6 <head>
7 <!-- HTML페이지의 어느 위치에든
8     자바스크립트 태그를 선언해서 사용 가능하다.
9     자바스크립트의 실행은 위에서 아래로 순서대로 실행된다. -->
10 <script type="text/javascript">
11     console.log("head 태그 안에서 선언되었다.");
12 </script>
13 </head>
14
15 <body>
16 <script type="text/javascript">
17     console.log("body 태그 안에서 선언되었다.");
18 </script>
19 </body>
20 </html>
```

자바스크립트는 인터프리터 언어이기 때문에 위에서부터 아래로 순차적으로 실행된다. 이렇게 순차적으로 실행되는 방식을 절차지향 방식이라고 한다. 하지만 자바스크립트 언어가 절차적인 방식으로만 실행되는 것은 절대로 아니다. 자바스크립트 역시 상속, 다형성, 생성자 같은 객체지향 언어의 특징을 가지고 있다. 다시 말해, 자바스크립트도 자바 언어처럼 객체지향 방식의 프로그래밍 작성 방식을 지원한다는 것이다. 또한, 많은 자바스크립트 라이브러리가 그러한 객체지향 방식을 이용해서 만들어지고 있다. 이 책의 자바스크립트 파트 후반부에 객체지향 문법에 대해 다룰 것이다.

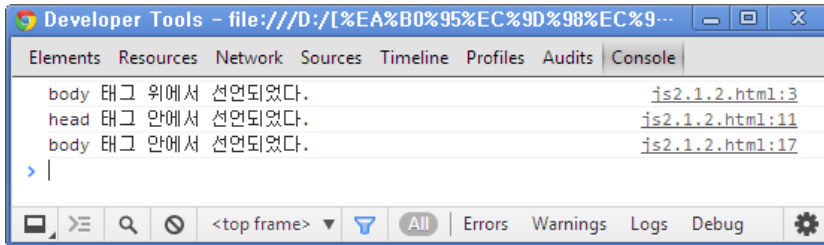


그림 2-5 자바스크립트 선언 위치에 따른 출력 결과

2.1.3 Inline, Internal, External 방식의 사용

지금까지 위에서 보여준 작성 방식은 일반적인 방식으로, HTML 파일 내에 선언하는 Internal 작성 방식이다. 이러한 방식은 자바스크립트의 간단한 기능을 테스트할 때 편리한 방식이다. 그러나 자바스크립트 코드의 행이 많아지거나 또는 반대로 자바스크립트의 양이 한두 줄 정도로 간단한 코드는 Internal 방식 외의 다른 작성 방식을 사용한다. 자바스크립트의 양이 많을 때 자바스크립트 코드 부분을 외부 파일로 분리해서 사용하는 External 방식으로 작성한다. 실제 프로젝트에서도 디자인과 스크립트를 분리하기 위해 External 방식을 선호한다.

1 External 방식의 선언

외부의 .js 파일에 자바스크립트 소스를 작성해두고 HTML 파일에서 불러들여서 사용하는 방법을 말한다. 이런 방법을 사용하는 이유는 여러 개의 HTML 파일에서 공통된 자바스크립트 기능을 사용할 수 있고, 자바스크립트의 유지 관리도 편리하기 때문이다. 소스 2-3의 예제에서는 외부의 파일에 단 한 줄의 코드를 작성한다. 그리고 소스 2-4에서와 같이 소스 2-3을 참조하는 External 선언 방식의 아주 간단한 예제를 실습해보도록 한다.

소스 2-3 외부 파일에 자바스크립트 선언

CODE js2.1.3(1)external.js

```
1 | alert("외부에 선언한 자바스크립트 파일");
```

외부의 자바스크립트 소스 파일에는 순수하게 자바스크립트 코드만 적어야 한다. 확장자가 .js 인 파일에는 HTML 코드가 있을 수 없다. HTML 주석도 사용하면 안 된다. 물론, 자바스크립트

트 관련 문법이나 자바스크립트 주석은 모두 사용 가능하다. 자바스크립트 주석과 HTML 주석은 다른 영역에 적용되는 다른 언어란 것을 명심하길 바란다.

소스 2-4 외부에 선언된 자바스크립트 소스 참조하기

CODE js2.1.3(1)external.html

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml">
4 <head>
5 <meta name="generator" content="editplus" />
6 <script src="js2.1.3(1)external.js"></script>
7 </head>
8 <body>
9
10 </body>
11 </html>
```

<script> 태그의 src 속성을 이용해서 외부의 자바스크립트 소스 파일을 참조하도록 한다. 나중에 공부할 jQuery 라이브러리도 외부에 선언된 자바스크립트 소스 파일이라고 생각할 수 있다. 즉, jQuery 같은 라이브러리도 이런 식으로 외부 소스 파일을 참조해서 사용한다.

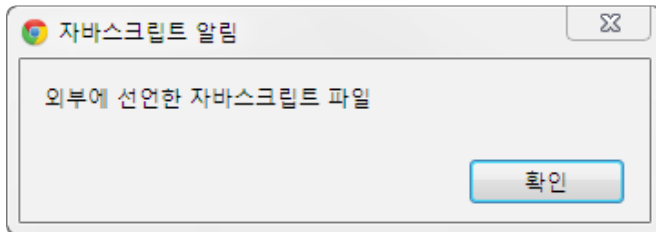


그림 2-6 External 방식의 실행 결과

브라우저로 실행해보면 그림 2-6처럼 외부 파일에서 작성한 alert() 기능이 적용되는 것을 확인할 수 있다.

2 Inline 방식의 선언

자바스크립트의 양이 한두 줄 정도로 소량일 때는 Inline 방식으로 사용한다. 이 방식은 고전적인 방식의 이벤트 핸들러 작성 방식을 사용한다. 자바스크립트를 적용하고자 하는 해당 태그의 이벤트 핸들러 속성을 이용해서 자바스크립트 내장 메소드를 호출하거나 개발자가 미리

선언해둔 사용자정의 함수를 호출하는 형태로 작성된다.

소스 2-5 Inline 방식으로 자바스크립트 사용

CODE js2.1.3(2)inline.html

```
9   <body>
10     <a href="#"
11       onclick="alert('Internal 방식 경고창 실행');return false;" >
12       Inline 선언 방식 </a>
13   </body>
```

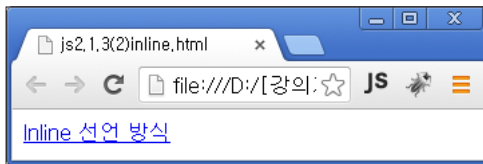


그림 2-7 Inline 방식으로 사용된 자바스크립트
_1단계 결과



그림 2-8 Inline 방식으로 사용된 자바스크립트
_2단계 결과

지금까지 자바스크립트를 선언하고 그 결과를 확인해보았다. 예제를 작성하는 데 사용된 `alert()`, `console.log()`, `document.write()` 같은 기능들을 함수 또는 메소드라고 부른다. 자바스크립트에서 사용하는 좀 더 많은 기본적인 데이터 입/출력 방식은 다음 절에서 자세히 설명하도록 하겠다. 또, 메소드의 사용법이나 사용자정의 함수를 선언하는 요령은 2.5절에서 좀 더 자세히 알아볼 것이다.

만일 소스 코드가 작성된 HTML 문서의 파일 인코딩 타입이 “UTF-8”로 되어 있다면, 브라우저에서 설정한 인코딩 타입과 맞지 않아서 한글이 깨져 보이는 현상이 발생할 수 있다. 이럴 경우에는 브라우저의 인코딩 타입 설정을 “UTF-8”로 바꾸든가, 아니면 문서 편집툴인 에디트플러스(EditPlus)에서 파일 저장 시 인코딩 타입을 변경할 수 있다. 에디트 플러스에서 파일 인코딩을 변경하는 방식은 에디트 플러스 상단 메뉴바에 있는 문서(Document) 메뉴의 File Encoding 메뉴를 이용해서 변경할 수 있다.

즉, 에디트플러스 편집툴의 Document 메뉴 ➡ File Encoding ➡ Change File Encoding...을 통해 변경할 수 있다. 사용하는 에디트 플러스가 한글 버전일 경우에는 문서(D) 메뉴 ➡ 파일 인코딩(D) ➡ 파일 인코딩 변경(C)...에서 변경할 수 있다.

2.2 데이터 입/출력

2.2.1 자바스크립트에서 데이터 출력해보기

자바스크립트는 HTML 문서 안에 있는 다양한 요소들을 접근하고 제어한다. 브라우저에서 실행되는 페이지와 페이지 사이에 전달되는 파라미터 정보를 받아들여서 활용하기도 하고, `prompt()`와 같은 내장 기능을 이용해서 사용자로 부터 데이터를 입력받을 수도 있다. 자바스크립트를 공부할 때 여러 가지 다양한 예제를 만들어보고 실행해보기 위해선 일단 입/출력 방법부터 익혀야 한다. 이번 절에서 다루는 내용은 자바스크립트를 이용해서 다양한 요소와 데이터에 접근해서 사용하는 기본적인 방법들이다. 앞으로 우리가 실습하는 데 필요한 기본적인 입/출력 방식이므로 잘 숙달시키도록 한다. DOM이나 BOM에 대해서는 뒤에 더 자세히 다룰 것이니 일단은 많이 사용할 기능부터 배워보도록 하자.

표 2-1 데이터를 출력하는 기본적인 방법들

| 기능 | 설명 |
|---------------------------------|---|
| <code>document.write(내용)</code> | <ul style="list-style-type: none">• 화면에 값을 추가한다. 콤마(,)를 이용해서 여러 개의 값을 추가할 수 있다.• 비슷한 기능으로는 <code>document.writeln()</code>이 있는데, 이것을 사용하면 출력 후 자동으로 줄 바꿈이 된다.• 내용을 엮어서 출력하고자 할 땐 내용을 콤마(,)로 이어준다.• 내부적으로 <code>document.open()</code>으로 시작되고 끝난 후엔 <code>document.write()</code>를 해주는 것이 원칙이지만, 일반적으로 그냥 생략하기도 한다. |
| <code>window.alert(내용)</code> | <ul style="list-style-type: none">• 내용을 경고 창으로 표시해준다.• HTML 문서가 실행되어 보여지는 브라우저 창은 자바스크립트에서 기본적으로 window 객체라고 보기 때문에 window. 부분은 생략이 가능하다. |
| <code>innerHTML = 내용</code> | <ul style="list-style-type: none">• DOM을 조작할 때 엘리먼트 요소의 내용을 가져오거나 새로운 내용으로 대체할 때 사용되는 아주 유용한 속성이다.• 처리 속도가 빠르다.• 내용을 가져올 때 브라우저마다 조금씩 다를 수 있다. 예를 들면, 인터넷 익스플로러 8에서는 태그를 대문자로 읽는다.• 비슷한 속성으로 텍스트만 읽어오는 <code>innerText</code>가 있다. |
| <code>console.log(내용)</code> | <ul style="list-style-type: none">• 내용을 Developer Tools의 콘솔에 표시한다. |

```

8   <script type="text/javascript">
9       document.write("hello1");           //화면을 덮어쓴다.
10      alert("hello2");                     //경고 창이 실행된다.
11      document.body.innerHTML += "hello3"; //화면의 내용에 추가
12      console.log("hello4");               //브라우저의 console에 나온다.
13  </script>

```

9행 document.write() 메소드 내부에는 document.open()이 사용되었다. write() 메소드의 기본적인 기능은 인자로 들어오는 내용으로 화면을 덮어쓴다.

10행 alert()는 window 객체의 멤버다. 팝업 경고 창을 띄우고 인자로 들어오는 값을 보여준다.

11행 document.write()를 먼저 사용하게 되면 document.body를 이용할 수 있다. innerHTML 속성을 이용해서 body 태그의 내용을 변경한다. 이때 대입연산자인 = 연산자를 이용하면 내용을 덮어쓴다. 복합배정대입 연산자인 += 연산자를 사용하면 기존 내용에 새로운 내용을 덧붙인다.

12행 크롬 등의 브라우저에서 제공하는 콘솔 기능을 이용해서 내용을 출력하게 한다. 인터넷 익스플로러는 9 이상에서 확인 가능하다. 인터넷 익스플로러 8 이하에선 지원하지 않는 기능이므로 에러로 표시된다.

화면에 내용을 출력하고 자동으로 경고 창이 팝업된다.

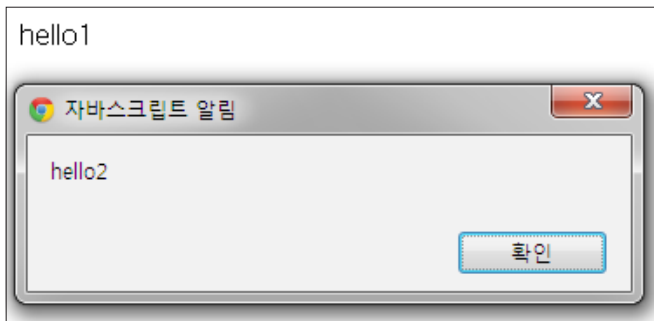


그림 2-9 다양한 방법으로 데이터 출력해보기_1단계 결과

경고 창의 확인 버튼을 누르면 다음 내용인 hello3이 innerHTML 기능에 의해 기존 내용에 붙여진다. 복합배정대입 연산자인 +=을 이용해서 기존 내용에 추가되는 것이다.

```
hello1hello3
```

그림 2-10 다양한 방법으로 데이터 출력해보기_2단계 결과

여기까진 자동으로 보여지고 다음은 브라우저에 내장된 콘솔에서 확인해야 한다. 참고로, 브라우저의 콘솔 기능은 인터넷 익스플로러 8 이하에선 지원하지 않는다.

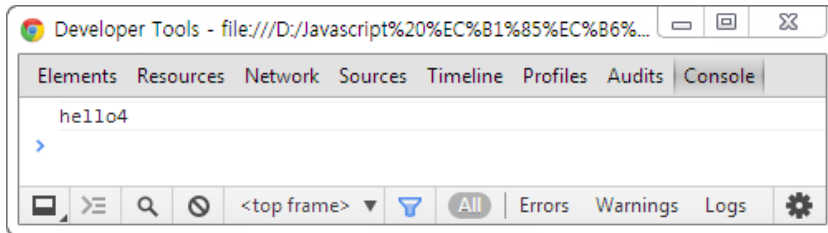


그림 2-11 다양한 방법으로 데이터 출력해보기_3단계 결과

Developer Tools의 콘솔 패널에서 마지막 내용의 출력 결과를 확인할 수 있다. 우측에 보여지는 파일 명을 클릭해보면 코드의 어떤 부분에 의해 내용이 출력되었는지 알 수 있다.

2.2.2 데이터 입력받기

자바스크립트에서 문서 내의 요소나 데이터에 접근해서 사용하는 방법은 아주 다양하다. 데이터를 입력받거나 문서 내부의 특정 요소를 다루는 여러 방법을 정리해보았다. 한마디로 말하면, 자바스크립트가 하는 일이 HTML 문서의 요소들을 제어하는 것이다. HTML 문서나 윈도우 화면에서의 데이터를 CSS로 변경하는 것이 자바스크립트가 일반적으로 하는 일의 대부분이라고 해도 과언은 아니다. 앞으로 이 책에 소개되는 자바스크립트의 대부분 핵심적인 기능이 이번 장에서 예제로 다루어질 것이다. 어찌 보면 후반부는 이번 장의 기능들이 융합되어 만들어지는 예제이기에 이번 장 역시 꼼꼼히 학습해야 한다.

1 confirm() 함수로 입력받기

어떤 질문 사항에 대한 답을 “예” 또는 “아니오”의 결과로 얻어야 할 경우, 대화 창을 통해 결정 지을 수 있다. 실제로 대화 창 버튼에는 “확인”과 “취소”로 표시된다.

표 2-2 confirm() 함수에 대한 설명

| 기능 | 설명 |
|----------------------|--|
| window.confirm(질문내용) | <ul style="list-style-type: none">• 팝업 창의 “확인”을 누르면 true 값을 돌려준다.• “취소”를 누르면 false 값을 돌려준다. |

소스 2-7 confirm() 팝업 다이얼로그로 실행

CODE js2.2.2(1)confirm0.html

```
8 <script type="text/javascript">
9   var isMan = confirm("당신은 남자입니까?");
10  document.write("결과: " + isMan + "<br />");
11 </script>
```

confirm()이 실행된 후 어떤 데이터가 isMan 변수에 담기는지 알아보기 위한 단순한 소스다. 일단, 자바스크립트 구현 원리를 알아보기 위해 단순한 형태로 실행하고 결과를 출력해보도록 한다.

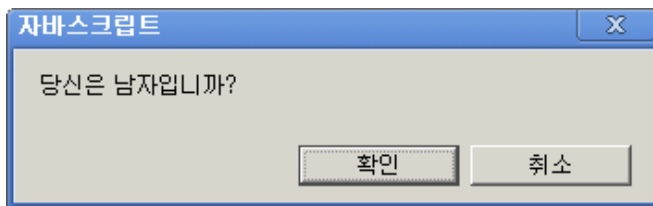


그림 2-12 confirm() 대화상자 팝업으로 열기_결과 1

결과: true

그림 2-13 confirm()으로 대화상자 팝업으로 열기_결과 2

팝업된 대화창의 “확인” 버튼을 누르면 isMan 변수에 true 값이 담긴다. “취소” 버튼을 누르면 isMan 변수에 false 값이 담기게 된다.

기본적이고 단순한 형태로 `confirm()` 함수가 어떤 기능인지를 확인했다면, 다음 예제를 통해 그 결과 값을 제어문으로 활용하는 테스트를 해보도록 한다.

소스 2-8의 예제에선 이 책의 다음 장에 나오는 제어문과 연산자가 먼저 등장한다. 어렵지 않은 제어문이니 아직 배워보지 않은 것을 먼저 실습해보고 공부하는 것도 자바스크립트를 이해하는 데 있어서 좋은 방법이 될 수 있다. 즉, 실행 결과를 먼저 보고 그것에 대해 개념을 잡아가는 것이 프로그래밍 학습 효과는 더 좋다(역공학). 가령, 음식 만드는 걸 배운다고 가정한다면, 맛있는 음식의 맛을 먼저 맛보고 음식을 만드는 것이 음식을 더 맛있게 만들 수 있다는 이치와 같다. 자격증 공부를 할 때도 본론으로 들어가기 전에 일단 기출문제부터 풀어보는 것이 학습 효과가 더 좋다. 만일 `if` 문과 같은 제어문이 어렵게 느껴진다면 다음 장을 먼저 보고 이번 장을 공부해도 나쁘진 않다. 자바스크립트와 같은 프로그래밍 언어는 단순히 문법만 안다고 해서 어디에 써먹을 수 있는 것이 아니다. 필요한 기능에 따른 여러 기술들이 어우러져서 하나의 덩어리(모듈)들이 만들어진다. 문법적인 요소를 공부하는 순서가 따로 있는 것도 아니기 때문에 프로그래밍 공부를 체계적으로 공부한답시고 목차 순서대로 접근하는 것도 나쁘지는 않겠지만, 자칫 ‘장님 코끼리 다리 만지듯’ 프로그래밍을 단편적으로 이해하는 우를 범할 수도 있게 된다. 무슨 일이든 마찬가지로, 프로그래밍 공부를 하기에 앞서서 나무를 보기 전에 숲을 먼저 봐야 한다는 진리를 기억하자.

소스 2-8 `confirm()`의 결과 값을 제어문으로 제어하기

CODE js2.2.2(1)confirm1.html

```
8   <script type="text/javascript">
9   var isMan = confirm("당신은 남자입니까?");
10  document.write("결과: " + isMan + "<br />");
11  if(isMan === true) {
12      gender = "남";
13  } else {
14      gender = "여";
15  }
16  document.body.innerHTML += gender + "탕으로 가시오.";
17  </script>
```

위 소스 코드는 `confirm()` 함수로부터 돌려받은 결과 값을 `isMan` 변수에 담은 후에 출력할 내용을 `if` 문으로 걸러주는 소스다. `if` 문은 이것 아니면 저것을 선택할 때 쓴다. 조건 분기문이라고도 한다.

| | |
|--------|---|
| 9행 | confirm() 함수를 이용해서 팝업을 띄운다. |
| 10행 | 만약 사용자가 팝업 대화상자에 있는 “확인”을 선택한다면 isMan 변수에 true 값이 담기게 된다. “취소”를 선택했다면 false 값이 담긴다. |
| 11~15행 | if 문을 이용해 isMan의 값이 true라면 gender 변수에는 “남”이 입력된다. isMan의 값이 false라면 gender 변수에 “여”가 입력된다. |
| 16행 | 이렇게 gender 변수에 걸려진 결과 값은 위 소스의 16행에 있는 innerHTML 속성에 의해 브라우저의 화면에 “남탕으로 가시오.”라고 출력하게 된다. |

소스 2-8은 어떤 변수에 결과를 담아서 그 값을 다시 if 문의 조건식으로 사용하고 있는데, 이 부분을 소스 2-9처럼 입력받고 제어하는 두 가지 기능을 하나로 합쳐서 사용하기도 한다.

소스 2-9 confirm()의 결과 값을 제어문의 조건으로 바로 사용하기

CODE js2.2.2(1)confirm2.html

```

8  <script type="text/javascript">
9  //var isMan = confirm("당신은 남자입니까?");
10 //document.write("결과: " + isMan + "<br />");
11 document.write("결과 : ");
12 if(confirm("당신은 남자입니까?") === true ) {
13     gender = "남";
14 } else {
15     gender = "여";
16 }
17 document.body.innerHTML += gender + "탕으로 가시오.";
18 //document.body가 로드되지 않은 상태에서
19 //document.body의 값을 사용하면 에러가 발생한다.
20 </script>

```

소스 2-9의 11번째 행에 있는 `document.write("결과 : ");`라고 적힌 부분을 생략하게 되면, 17번째 행의 `document.body.innerHTML += gender + "탕으로 가시오.";` 부분에 에러가 발생한다. 이 부분에 에러가 발생하는 이유는 <body> 태그가 자바스크립트 소스보다 아랫부분에 있기 때문이다. 위에서부터 아래로 읽혀지는 인터프리터 언어의 성격상 자바스크립트 소스보다 아래에 위치한 <body> 태그를 이보다 윗부분에 선언된 자바스크립트에서는 제어를 할 수 없게 된다. 이런 문제를 해결하기 위해선 자바스크립트 소스를 <body> 태그보다 아래에 위치하도록 하든가, 아니면 onload 이벤트 핸들러를 이용해야 한다. onload 이벤트 핸들러를 이용해서 문서의 내용이 모두 브라우저에 로드된 후 자바스크립트가 다시 호출되어 실행되도록 할 수 있다.

```

8  <script type="text/javascript">
9  //var isMan = confirm("당신은 남자입니까?");
10 //document.write("결과: " + isMan + "<br />");
11 window.onload = function() {
12     if(confirm("당신은 남자입니까?") === true ) {
13         gender = "남";
14     } else {
15         gender = "여";
16     }
17     document.body.innerHTML += gender + "탕으로 가시오.";
18     //window.onload 이벤트가 발생 후에 처리되기 때문에
19     //document.body를 사용하는 데 문제가 없다.
20 }
21 //window가 로드되기 전에 document.body 객체를 사용하면
22 //에러가 발생한다.
23 //document.write(gender + "탕으로 가시오." );
24 </script>

```

소스 2-10의 11번째 행은 window.onload 이벤트 속성에 자바스크립트 콜백 함수 기능을 이용한 것이다. 이렇게 하면 자바스크립트 코드가 <body> 태그보다 뒷부분에 위치해 있다 하더라도 17번 행에 문제가 발생하지 않게 된다.

소스 2-9에서는 document.write("결과 : "); 부분을 10번 행처럼 주석 처리하면 에러가 발생했다. document.write() 함수 사용 전에 document.body.innerHTML 속성을 사용하게 되면 에러가 발생하게 되고, document.write() 함수를 먼저 써주면 정상 작동된다. 이렇게 되는 이유는 document.write() 함수 내부에는 document.open() 함수가 포함되어 있기 때문이다. 말 그대로 document.write() 함수를 사용하게 되면 이미 문서를 열고 썼다는 것이 된다. 그래서 위에서처럼 document.write()나 document.writeln() 함수로 내용을 먼저 출력하고 나면 17번 행에 문제가 발생하지 않았던 것이다. 이 책에서는 자바스크립트 코드를 onload 이벤트 핸들러로 호출한 함수에 구현하거나 window.onload = function() { } 형태로 작성할 것이므로 크게 중요하지 않다.

다음 소스 코드는 앞에서 보여준 “if else” 제어문을 “조건연산자”를 이용해서 한 행으로 처리하는 것을 보여준다. 필자는 이런 식의 조건연산자 처리를 매우 좋아한다. 조건연산자에 익숙해지면 그 편리함에 중독되고 말 것이다.


```

8   <script type="text/javascript">
9   window.onload = function() {
10      /*if(confirm("당신은 남자입니까?") === true ) {
11          gender = "남";
12      } else {
13          gender = "여";
14      }*/
15      document.body.innerHTML +=
16      (confirm("당신은 남자입니까?") ? "남" : "여")+"탕으로 가시오.";
17      //삼항연산자
18      //(조건식) ? (참일 때 결과) : (거짓일 때 결과);
19      //결과는 필히 변수에 담는다.
20  }
21  </script>

```

if 문과 같은 제어문이나 삼항연산자에 대해서는 뒷부분에서 더 자세히 다룰 것이다. 이런 제어문이나 삼항연산자를 처음 접하는 독자들은 당장 이 소스 코드가 어렵게 느껴질 것이다. 그럼에도 굳이 지금 이런 소스 코드를 보여주는 이유는 자바스크립트 코딩에 대한 독자의 이해를 돕기 위한 것일 뿐 지금 당장 제어문을 사용하란 것은 아니다. 이 부분에선 그냥 제어문에 대한 감만 잡고 넘기길 바란다. 그보다는 우선 자바스크립트가 어떤 식으로 실행되는지 보는 게 좋다. 같은 결과를 만들어내더라도 내부적으로 코딩하는 스타일은 모두 다를 수 있다는 것을 이해하길 바란다. 이런 여러 가지 스타일 중에서 어떤 것이 정답이라고 하는 것은 매우 위험한 발상이다.

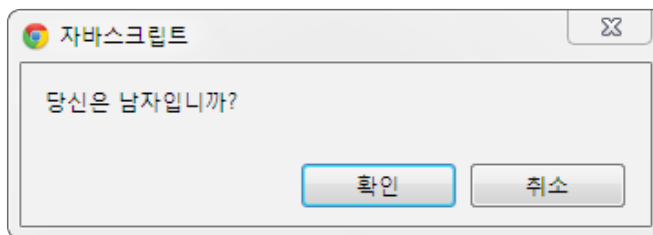


그림 2-14 confirm() 함수로 결정하고 결과 출력하기 1

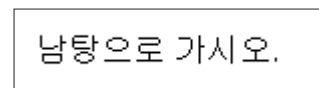


그림 2-15 그림 2-15 confirm() 함수로 결정하고 결과 출력하기 2

소스 2-11을 브라우저로 실행하면 자동으로 confirm() 대화상자가 열리고, “확인”을 누르면 브라우저에 “남탕으로 가시오.”라는 문장이 출력된다.

2 prompt() 함수로 외부 데이터 입력받기

prompt() 함수는 외부에서 데이터를 입력받고자 할 때 사용한다. 사용자가 어떤 데이터 정보를 입력하고자 할 때 경고 창처럼 팝업을 실행시켜서 입력받는 방법을 사용할 수 있다. 경고 창의 UI가 좀 투박하기 때문에 실제 프로젝트에서는 많이 사용하지 않지만, 자바스크립트 학습을 할 때나 간단한 입력이 필요한 테스트를 할 때 유용하게 사용할 수 있다.

소스 2-12 prompt() 함수로 외부 데이터 입력받기

CODE js2.2.2[2]prompt.html

```
6 <script type="text/javascript">
7   var hometown = prompt("고향이 어딘가요?");
8   alert("오늘 밤 "+hometown+"엔 꽃이 필게요.");
9 </script>
```

위 소스 코드를 실행하면 다음 그림과 같이 경고 창이 팝업된다. 팝업 창의 내용을 보면 “고향이 어딘가요?”란 물음과 그에 대한 답을 입력받을 수 있도록 입력 텍스트 필드가 준비되어 있다.

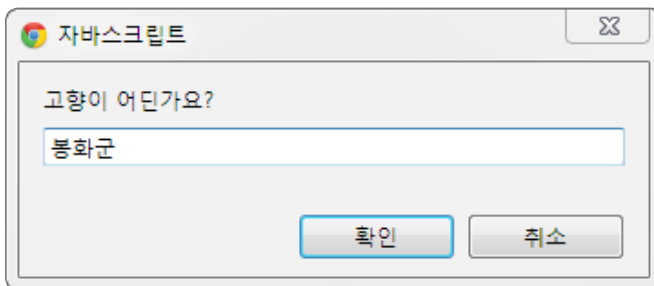


그림 2-16 prompt() 함수로 외부 데이터 입력받기_결과1

위 그림과 같이 입력 텍스트 필드에 “봉화군”이라고 입력하고 ‘확인’ 버튼을 누르면 alert() 함수 기능에 의해 다음 그림과 같이 팝업이 된다. 팝업의 내용을 보면 위 과정에서 입력한 “봉화군”이 소스에서 선언한 변수 hometown 대신 쓰여진 것을 확인할 수 있다.

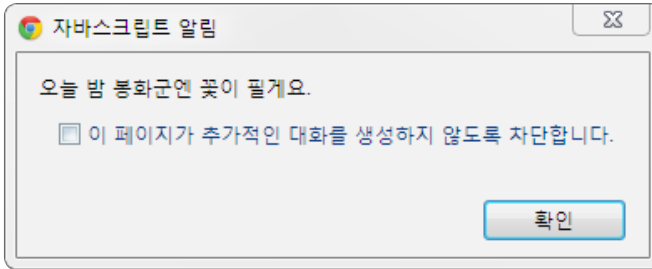


그림 2-17 prompt() 함수로 외부 데이터 입력받기 결과 2

이상 자바스크립트의 window 객체에서 기본 제공되는 alert(), confirm(), prompt()와 같은 기본 기능으로 데이터를 입력받는 법을 알아보았다. 다음은 자바스크립트 DOM의 기본적인 기능을 이용해서 문서 내의 요소에 접근하는 방법을 알아보도록 하자. 위에서도 언급했듯이, DOM에 대한 좀 더 자세한 설명은 뒷장으로 일단 미루도록 하겠다. 이번 장에서는 우선 자바스크립트라는 큰 산을 보기 위해 개념을 잡는 데 집중하도록 하자.

3 getElementById() 사용하기

자바스크립트 작성 시 많이 쓰는 요소는 크게 봤을 때 브라우저 객체 모델(이하 BOM으로 지칭한다)과 문서 객체 모델(이하 DOM으로 지칭한다)이 있다. 자바스크립트에서 DOM이란 document가 최상위로 되는 문서 내의 태그 요소들을 XML과 같이 계층적인 형태로 읽어들이는 것을 말한다. DOM을 자바스크립트에서 제어하기 위해서는 일단 문서에 있는 태그 엘리먼트를 자바스크립트로 받아와야 한다. 이럴 때 사용되는 document 객체의 메소드로는 getElementById()나 getElementsByName() 또는 getElementsByTagName() 같은 메소드를 이용한다.

표 2-3 태그 엘리먼트의 객체 정보를 가져오는 document 메소드들

| 메소드 | 설명 |
|---------------------------|---|
| getElementById(id 값) | • 태그의 id 속성 값을 이용해서 태그 엘리먼트 객체 정보를 가져온다. |
| getElementsByName(name 값) | • 태그의 name 속성 값을 이용해서 태그 엘리먼트의 객체 정보를 배열에 담아서 가져온다. |
| getElementsByTagName(태그명) | • 태그명을 이용해서 해당 태그들의 객체 정보를 배열에 담아서 가져온다. |

getElementById() 메소드는 해당 엘리먼트 객체를 직접 가져오지만, getElementsByName() 이나 getElementsByTagName() 메소드는 배열에 담아서 가져온다. 그 이유는 하나의 문서 내에 같은 id 이름을 여러 개 쓰는 것을 권장하지 않기 때문이다. 반면, name 속성이나 클래스 속성은 여러 개 중복 사용이 가능하다. 태그는 당연히 같은 태그를 중복해서 사용해야 한다.

소스 2-13 getElementById() 메소드를 이용한 DOM 객체 사용 준비

CODE js2.2.2(3)getElementById0.html

```

1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2  <html>
3  <head>
4  <meta http-equiv="content-type" content="text/html; charset=utf-8";    />
5  <style type="text/css">
6  #box {
7      width: 300px;
8      height: 200px;
9      border: 3px solid Red;
10     background-color: Pink;
11 }
12 </style>
13 <script type="text/javascript">
14 window.onload = function() {
15
16 }
17 </script>
18 </head>
19 <body>
20     <div id="box">사랑합니다.</div>
21 </body>
22 </html>

```

소스 2-13은 <body> 태그 내에 <div> 태그를 두고 id 속성의 값을 “box”로 해두었다. 6~11 행은 <div> 태그의 크기와 색상을 설정한 부분이다. 아직 자바스크립트 부분의 코드는 비어 있다. 이것을 브라우저에서 실행하면 그림 2-18과 같은 결과 화면이 나온다.

다음과 같이 자바스크립트에서 div 태그의 DOM 객체를 받아와서 제어해보도록 하겠다.

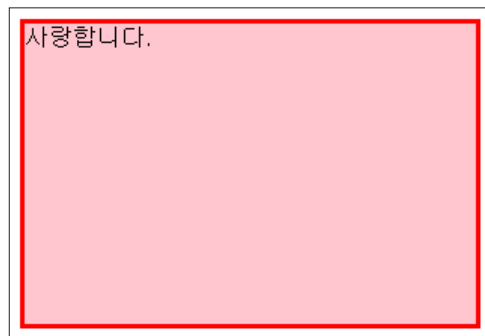


그림 2-18 getElementById() 메소드를 이용한 DOM 객체 사용 준비 결과 화면

```

13 <script type="text/javascript">
14   window.onload = function() {
15       //DOM을 이용해서 요소에 접근하기
16       var box = document.getElementById('box');
17       box.style.backgroundColor = "Green";
18       alert("미안합니다. " + box.innerHTML);
19       box.innerHTML = "고맙습니다.";
20   }
21 </script>

```

16행 문서 내에서 id 속성이 "box"인 태그의 DOM 객체를 받아온다.

17행 받아온 box 엘리먼트의 배경색을 녹색으로 변경한다.

18행 경고 창을 열고 "미안합니다."를 출력하고 box 엘리먼트의 내용 엔티티인 "사랑합니다."를 읽어와서 뒤에 붙여준다.

19행 box 엘리먼트의 내용을 "고맙습니다."로 변경한다.

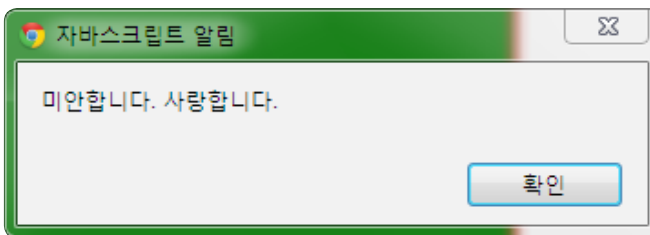


그림 2-19 getElementById() 메소드를 이용한 DOM 객체 사용_결과 1

소스 2-14를 실행하면 코드가 작성된 순서대로 실행된다. 먼저, <div> 태그 엘리먼트 객체를 받아온 후 배경색을 녹색으로 변경하고, 팝업을 띄워서 <div> 태그 엘리먼트의 엔티티 내용인 "사랑합니다."를 이어서 팝업에 보여준다. 팝업의 확인 버튼을 누르면 <div> 태그의 엔티티 내용이 "고맙습니다."로 바뀐다.

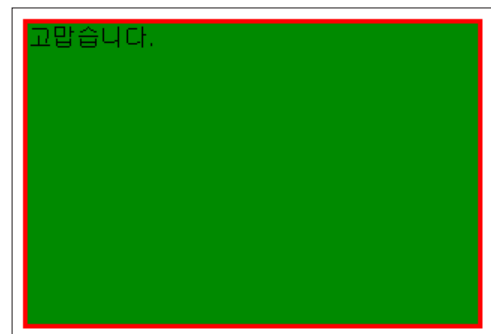


그림 2-20 getElementById() 메소드를 이용한 DOM 객체 사용_결과 2

4 getElementByTagName()

id 값을 이용해서 해당 태그를 직접 제어하는 방법 외에도 태그 이름을 이용해서 태그들을 한꺼번에 몽땅 가져와서 순서대로 제어하는 방법도 있다. 하나의 문서 안에는 같은 태그가 여러 개 존재할 수 있기 때문에 태그 이름으로 DOM을 얻어올 때는 배열에 담아서 넘겨준다. 결과가 단수가 아니라 복수이기 때문에 getElementByTagName()에는 getElement 다음에 's'가 붙는다는 것을 명심하길 바란다. 간혹 getElementByTagName()처럼 's'를 빼먹는 실수를 하기가 쉽다. 많이 사용하는 getElementById()에 익숙해서 생기는 실수인 것 같다.

소스 2-15 getElementByTagName() 메소드의 사용

CODE js2.2.2(4)getElementByTagName.html

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2  <html>
3  <head>
4  <meta http-equiv="content-type" content="text/html; charset=utf-8"; />
5  <script type="text/javascript">
6  start = function() {
7      var li = document.getElementsByTagName("li");
8      li[2].style.backgroundColor = "Yellow";
9      //li 요소 중 세 번째 요소의 스타일을 변경한다.
10 }
11 </script>
12 </head>
13
14 <body onload = "start()">
15 <!-- body가 로드되면 start() 함수가 호출된다. -->
16 <ul>
17     <li>Korea</li>
18     <li>Japan</li>
19     <li>China</li>
20     <li>Betnam</li>
21 </ul>
22 </body>
23 </html>
```

16~21행 자바스크립트에서 불러올 태그들의 묶음이다.

14행 <body> 태그가 모두 로드되면 자바스크립트로 선언된 사용자정의 함수 start()를 호출한다.

6~10행 사용자정의 함수인 start() 함수가 선언된 부분이다.

7행 getElementByTagName() 메소드를 이용해서 태그들의 DOM을 배열 형태로 변수 li에 담는다.

8행 li 변수에 담긴 태그 배열 세 번째 요소의 배경색을 노란색으로 변경한다.

- Korea
- Japan
- China
- Betnam

그림 2-21 getElementByTagName() 메소드의 사용 결과

위 소스 코드를 실행하면 세 번째 태그 항목인 China의 배경색이 바뀌는 것을 확인할 수 있다.

5 <input> 태그와 value 속성 사용

웹 페이지에서 사용자로부터 데이터를 입력받기 위해 많이 사용되는 <input> 태그가 있다. <input> 태그는 <form> 태그의 하위 태그로 주로 사용되지만, 필요에 따라서는 <input> 태그만 사용하기도 한다. <input> 태그의 type 속성에는 다양한 값들을 지정할 수 있는데, 주로 사용되는 값으로는 “text”, “button”, “radio”, “checkbox”, “submit” 등 아주 많이 있다.

다음은 <input> 태그의 value 속성 값을 읽어온 후 변경하는 예제다.

소스 2-16 <input> 태그의 value 속성 다루기 준비

CODE js2.2.2(5)input_value0.html

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml">
4  <head>
5  <meta http-equiv="content-type" content="text/html; charset=utf-8"; />
6  <script type="text/javascript">
7  checkValue = function(element) {
8      var inputText = document.getElementById("userName");
9      var userNameValue = inputText.value;
10     inputText.size = userNameValue.length;
11     element.value = userNameValue + "님은 이미 확인 하셨습니다.";
12     element.onclick = "";
13 }
14 </script>
15 </head>
16 <body>
17     <input type="text" id="userName" value="성명입력" onclick="this.value="" />
18     <input type="button" id="btn" value="확인하기" onclick="checkValue(this)" />
19 </body>
20 </html>

```

- 17행** 이벤트 핸들러인 onclick="this.value=" "의 뜻은 해당 <input> 태그를 클릭하면 기존의 내용을 지우는 것이다. this의 의미는 해당 태그 엘리먼트를 뜻한다.
- 18행** 해당 버튼을 클릭하면 7~13행에 선언되어 있는 사용자정의 함수 checkValue()를 호출한다. 인자로 this가 들어간 것은 함수에서 해당 버튼을 직접 제어하기 위한 것이다.
- 7~13행** 18행의 버튼을 클릭했을 때 호출될 사용자정의 함수를 정의한 것이다. 파라미터로 선언된 element는 18행에서 함수를 호출할 때 인자로 전달된 this가 저장된다. this는 버튼 태그를 가리키는 참조 값이다.
- 8행** 제어할 input 텍스트 필드의 DOM 정보를 가져와서 inputText 변수에 담는다.
- 9행** 17행의 텍스트 필드에 있는 데이터를 userNameValue 변수에 저장한다.
- 10행** 17행의 텍스트 필드의 넓이를 입력된 문자열의 길이만큼 변경한다.
- 11행** 18행의 버튼의 value를 변경한다. 버튼에 표시된 문구가 변경된다.
- 12행** 18행의 버튼 태그에 있던 이벤트 핸들러의 연결을 제거한다.

소스 2-16을 실행하면 다음과 같이 입력 텍스트 필드와 버튼이 화면에 보이게 된다.

그림 2-22 input 태그의 value 속성 다루기 준비_결과 1

위 소스에는 아직 특별한 제어문이 없기 때문에 버튼을 누르면 다음과 같이 버튼의 문구가 바로 바뀌게 된다.

그림 2-23 input 태그의 value 속성 다루기 준비_결과 2

소스 2-16 코드의 7행과 8행 사이에 소스 2-17과 같이 제어문을 추가해보겠다.

소스 2-17 input 태그의 value 속성 다루기

CODE js2.2.2(5)input_value1.html

```

5 | <script type="text/javascript">
6 |   checkValue = function(element) {
7 |     var inputText = document.getElementById("userName");
8 |     var userNameValue = inputText.value;
9 |     if(userNameValue == "성명입력" || userNameValue == "") {

```



```

10         alert("먼저 성명을 입력 하세요.");
11         inputText.value = "";
12         inputText.focus();
13         return;
14     }
15     inputText.size = userNameValue.length;
16     element.value = userNameValue + "님은 이미 확인 하셨습니다.";
17     element.onclick = "";
18 }
19 </script>

```

9~14행 텍스트 필드의 내용이 변경되지 않거나 없을 때 사용자가 내용을 입력하도록 유도하는 제어문이다. 참고로, 9행에서 사용된 ==는 뉴메릭과 넘버를 같은 것으로 취급하는 반면에 ===는 타입까지 모두 일치해야 true가 된다.

12행 focus() 메소드는 입력 커서가 해당 텍스트 필드에서 깜빡이게 한다.

13행 return;은 더 이상 함수를 아래로 진행하지 않고 해당 지점에서 끝내도록 한다.

위와 같이 텍스트 필드에 내용이 변경되었는지를 검사하는 제어문이 추가되면, 다음과 같이 “먼저 성명을 입력 하세요.”라고 안내문이 팝업으로 보여지게 된다.

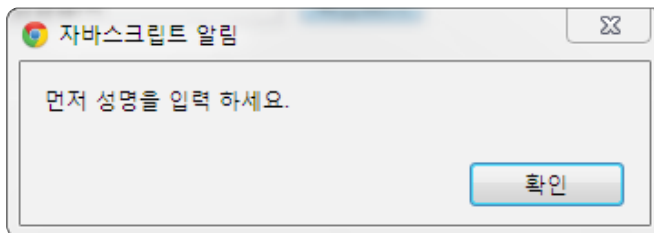


그림 2-24 input 태그의 value 속성 다루기_결과 1

팝업의 “확인” 버튼을 누르게 되면 다음과 같이 “성명입력”이라고 적혀 있던 부분이 지워지고 내용을 입력받을 수 있는 준비가 된다.



그림 2-25 input 태그의 value 속성 다루기_결과 2

텍스트 필드에 “홍길동”을 입력하고 “확인하기” 버튼을 누르면, 텍스트 필드의 크기가 줄어들고 버튼의 내용이 바뀌게 된다.

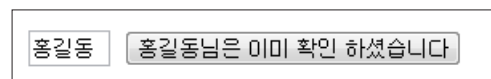


그림 2-26 input 태그의 value 속성 다루기_결과 3

2.3 주석문, 변수와 연산자, 자료형

2.3.1 주석 처리 방식

코딩의 완성은 주석 처리를 하는 것이라 할 정도로 주석은 중요하다. 주석은 인터프리터가 해석하지 않는 부분이므로 프로그래밍에 직접적인 영향을 주지는 않지만 프로그래밍 요소 중 아주 중요하다. 프로그래밍 언어 자체가 언어인데 주석이 왜 필요하냐고 하는 오만한 개발자들도 간혹 있다. 그러나 유능한 개발자가 되고자 한다면 처음부터 자신의 코드에 주석을 다는 습관을 가지길 바란다. 이것은 남을 위한 것이 아니라 가까운 미래의 자신을 위한 것이다. 이 사실은 개발을 시작해보면 곧 알게 될 것이다. 또한, 소스 코드는 인간이 일반적으로 사용하는 언어와 형태가 다르다. 코딩에 숙달된 개발자라 하더라도 남이 작성한 코드를 한눈에 이해하기란 절대로 쉬운 일이 아니다. 그러므로 나와 남을 위해 주석을 깔끔하게 달아주는 습관을 갖도록 해야 한다.

표 2-4 HTML 문서 내에서 사용하는 자바스크립트 주석과 HTML 주석

| | | |
|-----------|---------|--|
| 자바스크립트 주석 | 한 줄 주석 | <ul style="list-style-type: none">• //로 시작한다.• 행이 바뀌면 자동으로 주석이 해제된다.• C++에서 시초가 되었다. |
| | 여러 줄 주석 | <ul style="list-style-type: none">• /*로 시작하고 */로 끝낸다.• 여러 행의 코드나 설명을 주석으로 묶어줄 때 사용한다. C 언어에서 시초가 되었다.• HTML 문서에서는 자바스크립트에서만 아니라 스타일시트에서도 이 방식으로 주석 처리를 한다. |
| HTML 주석 | | <ul style="list-style-type: none">• <!-- 로 시작해서 -->로 끝낸다.• 자바스크립트나 스타일시트 태그 이외의 HTML 태그에서는 이 주석을 사용해서 주석 처리해주어야 한다. |

소스 2-18 HTML 문서 내의 자바스크립트 주석과 HTML 주석

CODE js2.3.1.comment.html

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2 <html>
3 <head>
```

```

4   <meta http-equiv="content-type" content="text/html; charset=utf-8"; />
5   <script type="text/javascript">
6       /*
7           여러 행을 주석으로 묶을 때 사용.
8       */
9       // 한 행만 주석으로 사용할 때 사용.
10      // 행이 바뀌면 주석은 자동 해제된다.
11  </script>
12  </head>
13  <!--
14      HTML 주석은 자바스크립트 내에서 적용되지 않고
15      자바스크립트 주석도 HTML 문서에서는 적용되지 않는다.
16  -->
17  <body>
18  </body>
19  </html>

```

6~8 행 자바스크립트 태그 내에서 자바스크립트 여러 줄 주석 처리를 하였다.

9~10 행 자바스크립트 태그 내에서 자바스크립트 한 줄 주석을 처리하였다. 한 줄 주석은 행이 바뀔 때마다 매번 주석 처리해주어야 한다.

13~16 행 HTML 문서 내의 HTML 주석 처리를 하였다. 자바스크립트 주석과 잘 구분해서 사용해야 한다.

2.3.2 변수와 연산자

1 변수 선언하기

프로그래밍에서 변수는 데이터를 담는 그릇과도 같다. 밥상에 음식을 차릴 때 그릇에 담아 두어야 먹기에도 좋다. 이렇듯 변수를 이용하면 프로그래밍에서 다룰 데이터들을 임시 저장하고 관리하기가 좋아진다. 프로그래밍 언어에서 변수가 없다면 매번 데이터를 메모리에 담고 해당 메모리의 위치 주소를 일일이 기억해야 하기 때문에 코딩하기가 너무 불편해질 것이다. 프로그래밍에서 변수를 꼭 사용해야 한다는 법은 없지만, 변수를 잘 쓰는 것이 코딩의 효율을 좋게 하고 소스 코드를 읽기 쉽게 하므로 유지보수할 때도 좋다.

변수는 크게 멤버 변수와 로컬 변수로 나눌 수 있다. 멤버 변수는 전역 변수라고도 하지만, 문서 내에선 특별히 정해주지 않으면 `window`의 멤버 변수가 된다. 그 외 사용자정의 함수에서 선언하는 것은 해당 멤버의 로컬 변수라고 생각할 수 있다. 즉, 변수는 어떤 영역에서 어떤 값을 사용하기 위해 저장하기 위한 용도로 사용된다.

```

5  <script type="text/javascript">
6      var name; //변수 선언 후 값을 초기화하지 않으면 null이다.
7      console.log(name.aaa);
8  </script>

```

6행 자바스크립트 변수 `name`을 선언한다. 변수 선언 시 `var` 지시어를 붙여주면 로컬 변수를 선언했다고 할 수 있다. `name` 변수 같은 경우, `var` 지시어를 붙여서 사용했어도 `window`의 멤버 변수가 된다. 변수를 선언만 하고 아무것도 초기화하지 않으면 변수는 `null`이라고 볼 수 있다. `null`이란 것은 아무것도 참조하지 않는다는 의미다.

7행 브라우저의 콘솔에 변수의 내용을 출력하는 문장이다. `name`이 `null`이므로 당연히 `name.aaa`는 존재하지 않는다. 존재하지 않는 대상을 접근하려고 하기 때문에 콘솔에는 `undefined`가 출력된다.

이런 변수를 선언할 땐 변수를 작명하는 간단한 규칙만 지켜주면 코딩하는 사람이 맘대로 지어 쓸 수 있다.

자바스크립트 식별자 명명 규칙

- 영문 대/소문자, 숫자, `_`, `$`만을 사용할 수 있다.
- 첫 글자로 숫자가 올 수 없다.
- 공백을 포함한 특수문자를 포함할 수 없다.
- 의미 있는 단어를 사용하는 것이 좋다.
- 예약어는 변수명으로 사용할 수 없으며, `for`, `if`, `while`, `var`, `function` 등과 같이 이미 자바스크립트에서 사용하고 있는 의미 있는 단어들을 뜻한다. 식별자로 사용 불가능한 예약어에는 어떤 단어들이 있는지 한번 찾아보자.
- 두 단어 이상을 결합해서 사용할 땐 낙타봉 표기법을 권장한다. (낙타봉 표기법이란 변수명이나 함수명과 같은 식별자를 만들 때 두 단어 이상이 포함될 경우 띄어쓰기를 할 수 없다. 따라서 단어의 의미를 알아보기 힘든 것을 개선하기 위해서 이어지는 단어를 대문자로 시작하거나 언더스코어(`_`)를 사용해서 단어를 구분하는 것을 말한다. 다른 말로 헝가리언 표기법이라고도 한다.)
- 자바스크립트에서는 변수와 같은 식별자를 한글로 만들 수 있다.

※ 식별자에 대한 예제 파일이 이 책과 함께 제공되는 소스에 있으니 참고하길 바란다.

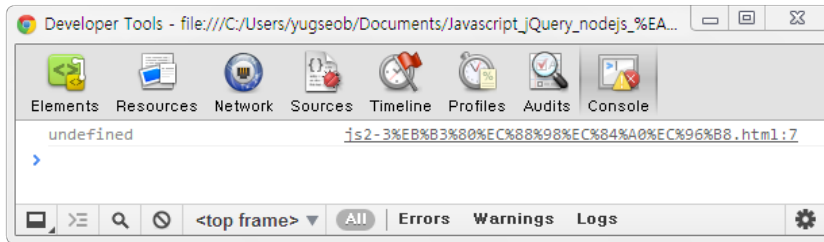


그림 2-27 자바스크립트에서 변수 선언과 사용 결과

변수에 저장된 값을 이용하여 원하는 정보를 얻기 위해선 변수의 값을 처리하고 가공하는 과정이 필요하다. 이렇게 변수의 데이터를 처리할 때 필요한 것이 연산자다. 프로그래밍에서 많이 사용되는 연산 대부분은 관계연산과 논리연산일 것이다. 그리고 또 사칙연산과 같은 산술 연산과 단항연산 등도 프로그래밍할 때 아주 많이 사용되는 연산자라고 볼 수 있다. 이 책에서는 모든 연산자에 대해 꼼꼼히 다루지는 않고 많이 사용되는 연산자 위주로 몇 가지 예제를 다룰 생각이다. 자바나 C와 같은 대부분의 컴퓨터 프로그래밍 언어에서 사용되는 연산자는 비슷한 특징을 가지고 있다. 그러므로 이 책에서 다루지 않는 일반적인 연산자의 특징은 다른 자료를 통해 참고할 수 있도록 한다.

2 멤버 변수와 로컬 변수

멤버 변수는 전역 변수라고도 하고, 로컬 변수는 지역 변수라고도 한다. 사실, 용어는 중요하지 않다. 용어는 달을 보라고 가리키는 손가락에 불과하기 때문이다. 중요한 것은 실체다. 자바스크립트를 작성할 때는 처음에 변수의 소속이 어디고, `this`가 가리키는 것이 무엇인지가 아주 헷갈린다. 어떨 땐 `this`가 `window`를 가리키다가 어떨 땐 `this`가 사용자가 정의한 객체를 가리키고 있기도 한다. 자바스크립트 문서 내에서 최상위 객체는 `window` 객체가 된다. 그러므로 `document`도 역시 `window`의 멤버가 된다. 단지 `window`나 `this`가 암묵적으로 생략됐다고 보면 된다. 따라서 `document.write('hello')`나 `this.document.write('hello')`나 `window.document.write('hello')`는 모두 같은 의미라고 할 수 있다.

표 2-5 멤버 변수와 로컬 변수

| 형식 | 설명 |
|----------------|--|
| 변수명 = 입력값; | <ul style="list-style-type: none"> • window.변수명 또는 this.변수명과 같은 의미 • 변수 선언 시 변수 앞에 var를 붙이지 않으면 전역 변수로 간주한다. |
| var 변수명 = 입력값; | <ul style="list-style-type: none"> • 변수 선언 시 앞에 var를 붙이면 지역 변수다. |

소스 2-20 멤버 변수와 로컬 변수의 사용 예제

CODE js2.3.2[2]member_local.html

```

8   <script type="text/javascript">
9   /*   스크립트를 선언한 후 바로 변수를 작성하면 전역 변수.
10      전역 변수는 멤버 변수라고도 하고, 지역 변수는 로컬 변수라고도 한다.
11      객체의 멤버 변수는 "속성" 또는 "Attribute"라고 한다.
12      객체의 멤버 함수는 "메소드" 또는 "Behavior"라고 한다.
13      전역 변수(window) : name = window.name = this.name      */
14   var name = "Hong micle";
15   grade = 9;
16   console.log(name + ", " + window.name + ", " + this.name);
17
18   function testFn() {
19       var school = "My school";
20       school2 = "Your school";
21       console.log(school.length);
22       console.log(window.school2);
23   }
24   testFn();
25   console.log("밖에서 ... " + school2);
26   console.log("밖에서 ... " + school);
27   </script>

```

14~16 행 전역에서 선언한 변수는 var를 쓰든 쓰지 않든 window 객체의 멤버다.

18~23 행 사용자 함수 testFn()을 선언한다.

19 행 testFn() 함수의 지역 변수다.

20 행 전역 변수로 간주한다.

24 행 선언된 함수를 호출해야 실행이 된다.

25 행 school2는 전역 변수로 본다.

26 행 school은 지역 변수이므로 함수 외부에선 not defined다.

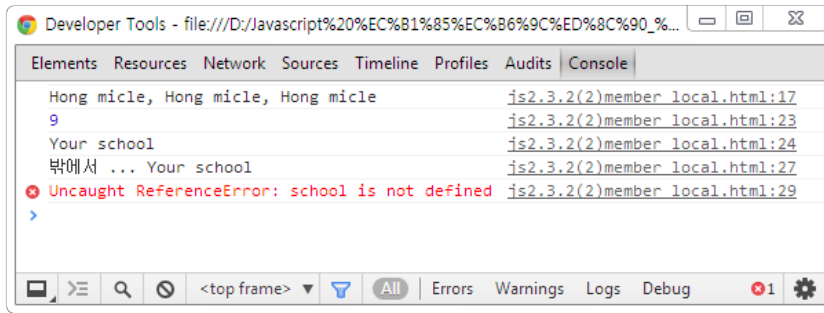


그림 2-28 멤버 변수와 로컬 변수의 사용 예제 결과

위 결과 화면에서 붉은색으로 출력된 마지막 라인은 `testFn()` 함수의 로컬 변수인 `school`을 전역에서 접근하려고 했기 때문에 발생한 것이다.

3 연산자 정리 – Operator 조작하는 자

■ 문장이 끝남을 세미콜론(;)으로 표시한다

자바스크립트는 명령문의 끝에 세미콜론(;)을 붙이지 않아도 큰 문제가 생기지 않는다. 그러나 자바스크립트도 자바 언어처럼 명령문의 끝을 세미콜론으로 마무리하면 가독성이 좋아진다. 단, 제어문이 끝날 때는 세미콜론을 쓰지 않는다. (do while 문은 예외)

■ 복문의 연산은 블록({ })으로 묶어준다

자바스크립트에서는 블록(와 } 사이) 안에 명령문의 수에 따라 단문과 복문으로 구분한다. 단 문은 하나의 명령문을 의미하고 복문은 복수의 명령어 문장을 의미한다.

■ 연산자의 종류

자바스크립트에서 필요한 정보를 얻고 기능을 수행하기 위해서 데이터를 연산해야 한다. 이런 데이터의 종류와 결과 값의 종류에 따라 다양한 연산자를 적절하게 사용하여야 한다. 이 소절에서 보는 연산자는 다른 컴퓨터 프로그래밍 언어에서 공통적으로 쓰여지기도 한다.

표 2-6 연산자의 종류와 연산자들

| 연산자 종류 | 연산자들 |
|--------------|-----------------|
| 괄호 | () |
| 단항연산자 | . [] ++ -- + - |
| 산술연산자 | * / % + - |
| 관계연산자 | == != > < >= <= |
| 논리연산자 | && |
| 배정대입 연산자 | = |
| 복합배정대입 연산자 | += -= *= /= %= |
| 조건연산자(삼항연산자) | ? : ; |

■ 연산자 우선순위

자바스크립트에서 연산의 우선순위가 가장 빠른 것은 괄호와 단항연산이다. 그 다음에 산술 연산을 수행한다. 반대로, 연산을 가장 나중에 하는 것은 배정, 대입연산이다. 그 외엔 크기와 관계 등을 비교하는 일들이다. 논리연산은 관계연산이 중복될 때 관계연산을 엮어주는 기능을 한다고 볼 수 있다.

단항연산 > 산술연산 > 논리연산 > 삼항연산 > 배정연산

■ 결합 방식

대부분의 연산은 오른쪽에서 왼쪽으로 결합하면서 연산한다. 단항연산, 대입연산은 왼쪽에서 오른쪽으로 결합하면서 연산한다. 왜 그럴까? 한번 고민해보면 상식선에서 금방 답이 나온다.

2.3.3 자바스크립트에서 사용되는 자료형(type)

변수는 데이터를 담는 그릇이기 때문에 데이터의 형과 밀접한 관계가 있다. 그러나 자바스크립트에서는 변수를 선언할 때엔 데이터 형을 지정해주지 않고 변수에 어떤 값이 들어가는가에 따라서 변수의 형이 결정된다고 볼 수 있다. 이러한 것은 자바스크립트 이외의 액션스크립트나 PHP 스크립트 같은 다른 스크립트 언어에서도 비슷한 특징이라고 볼 수 있다. 자바스크립트에서 변수에 들어갈 수 있는 값에는 문자열, 숫자, 객체 등의 값을 사용할 수 있다. 변수에 저장할 때 사용하는 문자열 상수나 숫자 상수 같은 값들을 리터럴이라고 말하기도 한다.

1 문자열과 문자열의 결합

큰따옴표(" ") 또는 작은따옴표(' ')로 묶여 있는 데이터는 문자열이다. `toUpperCase()`, `toLowerCase()`, `length()`, `indexOf()`, `lastIndexOf()`, `charAt()`, `substring()`, `split()` 등의 메소드를 기본적으로 가지고 있다.

소스 2-21 문자열과 숫자의 결합

CODE js2.3.3(1)strcat.html

```
9   <script type="text/javascript">
10  /*
11   + 연산자에는 두 가지 용도가 있다.
12   첫째, 산술연산자로서 Number형의 데이터를 더하는 용도.
13   둘째, 문자열과 변수의 값을 문자열로 결합하는 기능.
14  */
15  var value1 = "01";
16  var value2 = 200;
17  var value3 = 5;
18
19  console.log(value1 + value2 + value3);
20  console.log(value3 + value2 + value1);
21  </script>
```

19행 문자열에 숫자를 결합하면 결과는 문자열로 포함되므로 012005가 출력된다.

20행 문자열이 결합되기 전까지는 그냥 산술연산을 하므로 205010이 출력된다.

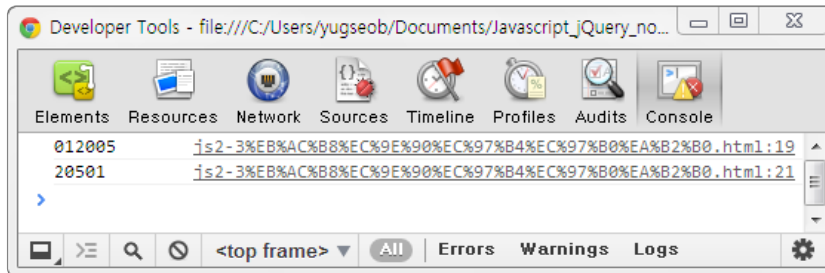


그림 2-29 문자열과 숫자의 결합 결과

2 숫자

정수형 숫자와 부동소수점 숫자로 구분된다. 부동소수점의 연산은 부정확할 수 있다. `Math` 객체인 `Math.abs()`, `Math.random()`, `Math.round()`, `Math.floor()`, `Math.ceil()` 등과 같은 메소드를 사용해서 숫자 데이터를 적절하게 다룰 수 있다.

3 NaN

Not A Number다. 즉, 숫자가 아닌 데이터를 숫자처럼 사용할 때 나타난다.

4 논리값(Boolean)

Boolean이라고 한다. 참 true, 거짓 false의 두 가지 값만을 가진다.

5 객체(Object)

자바스크립트에는 `new Array();`와 같이 `new`로 선언되는 객체와 자바스크립트 내장 객체가 존재한다.

6 undefined와 null

앞에서 언급한 것처럼 `null`은 참조 객체 없음을 뜻한다. `undefined`는 변수명이나 함수명으로 식별자가 선언된 적이 없다는 뜻이다.

7 typeof 연산자

`typeof "korea";`를 출력하면 `String`이 출력된다. 이것은 "korea"라는 문자열 상수 데이터가 `String`형이란 것을 `typeof`가 보여주는 것이다.

8 데이터의 형 변환

문자열에 숫자를 + 연산하면 숫자는 문자열에 포함되어 문자열로 변환된다. 숫자 형태의 문자열 데이터를 연산할 때는 숫자로 인식한다(+ 연산은 예외). `parseInt()`, `parseFloat()`, `toFixed()`와 같은 메소드를 사용해서 문자열 형태의 뉴메릭(numeric) 데이터를 숫자로 변환해서 연산할 수 있다.

2.4 배열 Array 객체

2.4.1 배열의 의미

연속성이 있는 데이터를 저장하고 관리하기 좋도록 메모리 공간을 할당해서 사용한다. 자바스크립트 배열은 각기 다른 데이터를 무엇이든 담아서 사용할 수 있다. 변수가 단독주책이나 과일 하나에 비유된다면, 배열은 연립주책이나 과일 상자 같은 형식이라고 생각할 수 있다.

2.4.2 배열의 선언

| | |
|------|---|
| 형식 | 변수 = new Array(크기 인자) |
| 참고 | 요소의 개수가 Array 메소드의 크기 인자만큼인 배열을 만든다. |
| 사용 예 | var arr = new Array(5);라고 선언하면 요소의 개수가 5개인 배열이 만들어진다. |

소스 2-22 배열의 선언

CODE js2.4.2ArrayDeclaration.html

```
6 <script type="text/javascript">
7   var arr1 = new Array(5); // 배열의 크기가 5인 배열을 생성한다.
8   var arr2 = new Array("오징어", "꿀뚜기", "대구", "명태", "거북이"); //배열 선언과 동시에 요소 값 초기화
9   var arr3 = new Array(); // 배열의 크기가 정해지지 않은 배열을 선언한다.
10  for(var i=0; i<arr1.length; i++) {
11      arr1[i] = (i+1) * 100; // 배열의 정해진 인덱스 요소에 값을 할당.
12      arr3.push((i+1) * 10 ); // 배열에 값을 추가한다.
13  }
14  for(var i=0; i<arr1.length; i++) {
15      document.write("arr1["+i+"] = " + arr1[i] + ", ");
16      document.write("arr2["+i+"] = " + arr2[i] + ", ");
17      //document.write("arr3["+i+"] = " + arr3[i] + "<br>");
18      document.write("arr3.pop() = " + arr3.pop() + "<br>");
19  }
20 </script>
```

7~9행 다양한 형태의 Array 생성자를 이용해서 배열을 선언한다.

10~13행 for 반복문을 이용해서 배열 요소에 순서대로 값을 할당한다.

14~19행 배열 요소에 할당된 값을 출력한다. 18행의 pop()은 값을 뽑아쓰고 제거하는 메소드다. 배열에 push()로 값을 추가하고 pop()으로 뽑아쓴다.

자바스크립트에서 배열은 객체처럼 사용하기 때문에 선언하고 초기화하는 방법도 다양하다. 다음 두 가지 형태가 자바스크립트에서 가장 널리 사용되는 배열 요소 초기화 방법이므로 잘 기억해두길 바란다.

1 배열 선언과 동시에 값 할당

| | |
|------|--|
| 형식 | 변수 = new Array(요소1, 요소2, ... 요소n); |
| 사용 예 | var arr = new Array('오징어', '꼴뚜기', '대구', '명태'); |

2 리터럴을 이용한 배열 초기화

다른 언어에서처럼 중괄호({})를 사용하지 않는 것은 중괄호는 자바스크립트 객체의 리터럴이기 때문이다. 배열은 같은 종류의 타입의 묶음으로 사용하고, 객체 리터럴은 다른 종류 타입의 묶음으로 사용하는 것이 일반적이다.

| | |
|------|---|
| 형식 | 변수 = [요소1, 요소2, ... 요소n]; |
| 참고 | 중괄호({})를 쓰지 않는다. 자바스크립트에서 중괄호는 Object를 뜻한다. |
| 사용 예 | arr = ['오징어', '꼴뚜기', '대구', '명태', '거북이']; |

소스 2-23 배열의 초기화 방법

CODE js2.4.3(2)/ArrayInitialization-0.html

```

6   <script type="text/javascript">
7   window.onload = function() {
8       var arr1 = new Array("오징어", "꼴뚜기", "대구", "명태", "거북이");
9       var arr2 = [100, 200, 300, 400, 500];
10      var arr3 = document.getElementsByTagName("li");
11      console.log(arr1); //
12      console.log(arr2);
13      console.log(arr3);
14  }
15  </script>
16  </head>
17  <body>
18  <ul id="u">
19      <li>울릉도</li>
20      <li>동남쪽</li>
21      <li>백길따라</li>

```

```

22 |         <li>2백리</li>
23 |         <li>독도는 우리땅</li>
24 |     </ul>
25 | </body>

```

8~10행 세 가지 방법의 배열 변수를 초기화하는 방법이다. 10행은 배열을 초기화하는 것은 아니지만, 결과가 NodeList이기 때문에 배열의 기능과 비슷하다. 자바스크립트 배열 컬렉션을 NodeList형으로 본다. NodeList 컬렉션이란 건 console.log()를 이용해서 콘솔에 결과를 찍으면 확인이 가능하다.

11~13행 초기화 값을 콘솔 기능으로 출력해본다. arr3의 출력 결과는 [Object NodeList]인 것을 확인할 수 있다.

18~24행 배열의 내용을 테스트하기 위해 리스트 태그를 준비했다.

2.4.4 배열 요소의 사용

1 배열 요소에 값 할당하기

배열의 첨자(index)는 배열 요소의 순번이다. 배열의 첨자는 0부터 시작하고 변수를 이용해서 배열 요소에 접근이 가능하다. 배열의 마지막 요소의 첨자는 전체 배열 크기에서 1을 뺀 것과 같다. 즉, 배열 요소의 (마지막 첨자 = 배열 크기 - 1)과 같다.

| 형식 | 배열[첨자] = 값; |
|------|--|
| 사용 예 | <pre> var arr = new Array(3); arr[0] = 300; arr[1] = 500; arr[2] = 700; </pre> |

2 배열 요소의 값 사용하기

배열 요소의 값을 사용하거나 변경하기 위해서는 첨자로 해당 요소를 지정한다. 이때 첨자로 는 변수나 상수를 모두 사용할 수 있다.

| 형식 | 변수 = 배열[첨자]; |
|------|--|
| 사용 예 | <pre> var arr = ['오징어', '꼴뚜기', '명태']; var val = arr[1]; 또는 var res = arr[i]; <-- i는 변수 </pre> |

3 배열과 공합이 맞는 for 문

for 문은 배열을 위해 태어났다고 생각될 정도로 배열 요소를 사용하기에 편리하다. for 문에 대한 자세한 설명은 다음 장에서 배운다. for 문은 괄호 안에 초깃값, 조건식, 증감식을 모두 넣어서 실행 가능하기 때문에 편리하다.

| | |
|------|---|
| 사용 예 | <pre>var arr = ['오징어','꿀뚜기','명태']; for(var i=0; i<arr.length; i++) value = arr[i];</pre> |
|------|---|

4 for 문 대신 for in 문을 사용해도 좋다

단순히 배열의 내용을 출력하는 용도로 사용하려면 for in 문을 사용하는 것이 편리하다. for in 문의 사용 예는 다음과 같다. for in 문은 다음 장에서 더 자세하게 다룬다.

| | |
|------|--|
| 사용 예 | <pre>for(var i in arr) val = arr[i];</pre> |
|------|--|

소스 2-24 배열의 초기화 방법

CODE js2.4.3(2)ArrayInitialization-1.html

```
6 <script type="text/javascript">
7 window.onload = function() {
8     var arr1 = new Array("오징어", "꿀뚜기", "대구", "명태", "거북이");
9     var arr2 = [100,200,300,400,500];
10    var arr3 = document.getElementsByTagName("li");
11    console.log(arr1);
12    console.log(arr2);
13    console.log(arr3);
14    var node = (document.getElementById("u")).childNodes;
15    for(var i=0, cnt = 0; i<node.length; i++) {
16        if(node[i].nodeName == "LI") {
17            arr2[cnt] = node[i].innerHTML;
18            node[i].innerHTML = arr1[cnt++];
19        }
20    }
21    var ul = document.getElementById("u");
22    for(var i in arr2) {
23        var li = document.createElement("li");
24        li.innerHTML = arr2[i];
25        ul.appendChild( li );
26    }
27 }
28 </script>
```

| | |
|-----|---|
| 14행 | <body> 태그에 있는 태그의 자식 노드 목록을 태그의 id 속성을 이용해서 가져온다. |
| 16행 | 의 자식 노드들이 가지고 있던 기존의 값을 arr2에 임시 저장하고, arr1 배열의 요소를 이용해서 새로운 값을 채우기 위해 자식 노드들 중에서 nodeName 속성이 "li"인 것만 가려낸다. |
| 21행 | 태그의 DOM 객체를 반환한다. |
| 23행 | 새 목록에 기존의 목록을 추가하기 위해 새로운 태그를 생성시킨다. |
| 24행 | arr2 배열에 임시 저장된 기존의 목록을 새로 만든 태그의 내용으로 만든다. |
| 25행 | 새로운 목록 값으로 변경된 목록 뒤에 기존 값을 가진 목록을 추가한다. |

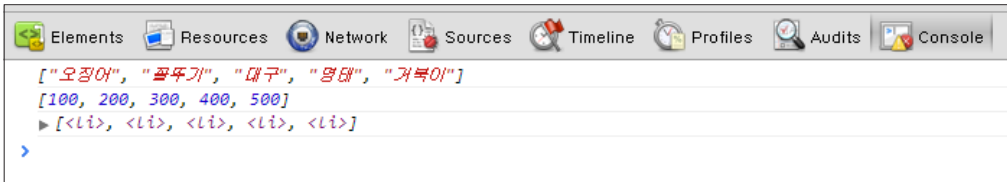


그림 2-30 소스 2-24의 11~13행의 실행 Console_결과 1

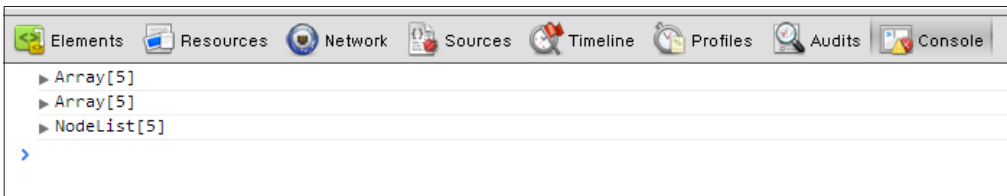


그림 2-31 소스 2-24의 11~13행의 실행 Console_결과 2

그림 2-30과 2-31은 14행부터의 소스를 제외한 윗부분까지의 부분 실행 결과다. 그리고 그림 2-32는 후반부인 14행부터 이어서 실행되었을 때 결과다. 폴더에 두 가지 경우의 소스가 있다. 왜 그런지 한번 생각해보기를 바란다. 책을 제대로 읽고 이해했다면 알 수 있는 내용이다.

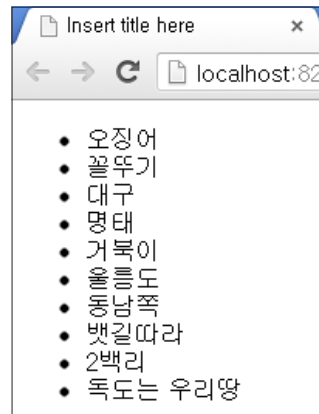


그림 2-32 소스 2-24의 실행 화면

배열도 일종의 객체이기 때문에 유용하게 사용 가능한 내장 메소드가 존재한다. 내장 메소드를 이용하면 제어문을 이용해서 일일이 기능을 만들어야 하는 고풍을 줄일 수 있다. 이번 소절에서는 배열의 내장 메소드의 사용법과 여러 가지 배열의 내장 메소드를 소개하도록 하겠다.

1 배열.indexOf("요소");

| | |
|--------|--|
| 배열 초기화 | arr = ["오징어", "꿀뚜기", "대구", "명태", "거북이"]; |
| 요소 검색 | document.write(arr.indexOf("대구")); |
| 설명 | arr 배열의 요소 값 중 "대구"가 위치한 인덱스 2를 반환한다. |

2 그 밖의 Array 메소드

| 메소드 이름 | 메소드 기능 설명 |
|-----------------|----------------------------|
| concat() | • 두 개 또는 세 개의 배열을 결합한다. |
| join() | • 배열을 결합한다. |
| pop() | • 배열의 맨 뒤에 요소 추가 |
| push() | • 배열의 맨 뒤에 요소 제거 |
| reverse() | • 배열 요소의 위치를 바꾼다(순서 변경). |
| shift() | • 배열에 첫째 요소 제거 |
| slice() | • 배열의 요소 선택 잘라내기 |
| sort() | • 배열 정렬 |
| | • 번호 정렬(내림차순) |
| | • 번호 정렬(오름차순) |
| splice([index]) | • 배열의 index에 해당하는 특정 요소 제거 |
| toString() | • 문자열로 배열을 변환 |
| unshift() | • 새로운 요소 추가 |

2.5 함수의 정의와 호출

함수는 일의 단위다. 함수는 메소드, 모듈, 기능, 프로시저 등으로 말한다. 일을 처리하기 위해서 필요한 인자를 함수에 넣으면 그 일을 함수가 처리하고 그 결과 값을 함수를 호출한 곳으로 돌려주게 된다. 함수라 하면 어린 학창 시절에 다음과 같은 그림 그렸던 걸 떠올릴 것이다. 지금 생각해 보면 함수를 표현함에 가장 명료한 그림일 것 같다.

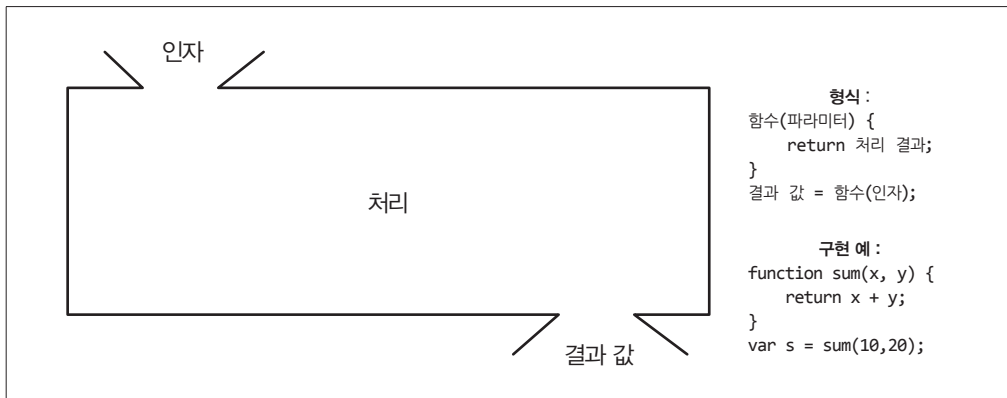


그림 2-33 함수의 구조

2.5.1 자바스크립트 함수 선언 방식

자바스크립트에서는 함수를 선언할 때 반환 타입 대신 **function** 키워드만 사용하면 된다. 일반적으로, **function** 키워드를 앞에 쓰고 선언하는 방법과 함수명을 먼저 선언하고 콜백 함수 형태로 선언하는 방법이 있다. 자바스크립트에서는 함수의 선언도 데이터처럼 사용할 수 있다는 것이 특이한 점이다. 자바스크립트에서는 함수도 배열의 요소 값으로 사용 가능하다. 다음 세 가지 방식은 기능 면에서 모두 비슷한 특징을 가진다. 어떤 방식을 사용해도 자바스크립트 프로그램의 성능과는 무관하다.

1 기본적인 선언 방식

```
function 함수명() { }
```

가장 일반적인 방식의 자바스크립트 함수 선언 방식이다. 자바스크립트는 클래스 키워드가 없다. 자바스크립트에서는 클래스를 만들고자 할 때도 **function** 키워드를 사용한다. 자바스크립트에서 클래스는 기능을 묶는 개념이 더 강하다. 필자의 경우, 함수를 클래스처럼 선언해서 객체를 생성해서 사용할 때 이런 방식으로 선언해서 구분한다.

2 익명 함수를 참조하는 방식

```
함수명 = function() { }
```

함수명을 먼저 선언하고 콜백 함수 형태로 함수를 선언하는 방식이다. 필자의 경우, 보통 일반적인 기능의 멤버 함수 선언은 이와 같이 선언한다. 꼭 이렇게 해야 한다는 것은 아니고 필자만의 클래스와 멤버 함수의 구분 방법이라고 할 수 있다.

3 스스로 동작하는 함수의 선언

```
(function(){ } )();
```

jQuery(제이쿼리) 같은 라이브러리는 내부적으로 스스로 동작하는 함수 선언 방식이 사용된다. 함수를 선언하고 호출해서 사용하는 것이 일반적인 방식인데, 이 방식은 호출하는 부분이 없이 선언부 스스로 동작한다고 볼 수 있다.

2.5.2 함수의 인자와 반환 값

1 파라미터 또는 인자

함수를 호출할 때 재료로 사용할 값을 넘겨준다. 함수를 선언할 때 인자를 실인자라고 하고, 파라미터로 선언하는 인자를 가인자라고 한다. 이것은 함수 선언 시 전달되는 값이 함수의 파라미터로 값이 복사되는 개념이기 때문이다.

2 반환 값

함수의 실행이 끝난 후 함수를 호출한 곳으로 결과를 돌려준다. 함수를 이용하거나 선언할 때 꼭 결과를 돌려줘야 하는 것은 아니다. 그러나 결과를 돌려줘야만 할 경우, `return` 문장에서 보내주는 데이터의 타입과 함수를 호출해서 결과를 돌려받는 쪽의 데이터 타입을 일치시키는 것이 좋다.

2.5.3 자바스크립트 함수 호출

선언된 함수는 함수 이름을 호출해서 써줘야 함수의 기능이 구동된다. 간혹 함수를 선언만 하고 호출하는 것을 빼먹는 경우가 있는데, 이럴 때를 대비해서 미리 호출하는 부분을 먼저 작성해두는 것도 괜찮은 요령이다. 함수는 한 번 선언하고 여러 번 호출해서 사용할 수 있기 때문에 함수는 기능을 재활용할 수 있다는 특징도 있다.

자바스크립트에서는 유용한 내장 함수들이 많이 존재한다. 그리고 필요에 따라 사용자가 함수를 직접 선언해서 작성이 가능하다. 자바스크립트는 `class`라는 키워드가 따로 존재하지 않기 때문에 함수를 선언할 때 쓰는 `function` 키워드를 이용해서 여러 함수를 묶어 줄 수 있다. 즉, 자바스크립트에서는 `class` 선언의 기능이 함수에 포함되어 있는 것이다. 이것은 자바스크립트 특징 중에서 클로저와 관련이 있는 것인데, 함수를 선언하는 것도 어떤 특정 메모리가 잡히기 때문에 함수를 마치 데이터처럼 취급하는 자바스크립트의 객체지향적 특징이다. 자바스크립트도 객체지향 언어이긴 하지만, C++이나 자바와 같은 언어에서의 객체지향적 개념과 철학이 아주 똑같다고 할 수는 없다. 그것은 자바스크립트의 언어적 성격의 뿌리와 자바 언어의 언어적 성격의 뿌리가 다르기 때문이다. 자바나 C++ 언어는 스택토크 언어의 객체지향적 특징을 많이 차용하였다. 다시 한 번 강조하자면, 자바스크립트의 함수는 단순히 기능을 정의하는 특징 외에도 클래스처럼 객체를 생성하거나 변수나 데이터처럼 자기 스스로 동작하는 특징도 있다.

```

1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5  <title>Insert title here</title>
6  <script type="text/javascript">
7  (function() {
8      alert("함수 호출 없이도 자동 실행되는 함수!");
9  })();
10 test01 = function() {
11     alert("test01 함수 호출!");
12 }
13 function test02(x, y) {
14     alert("test02 함수 호출! 결과 = " + (x + y));
15     return x + y;
16 }
17 window.onload = function() {
18     var btn03 = document.getElementById("btn03");
19     btn03.onclick = function() {
20         document.write("결과 = " + test02(50, 60) );
21     }
22 }
23 </script>
24 </head>
25 <body>
26     <button onclick="javascript:test01()">함수1 호출</button><br>
27     <button onclick="test02(10,20)">함수2 호출</button><br>
28     <button id="btn03">함수2의 다른 호출</button>
29 </body>
30 </html>

```

7~9행 ()를 이용해서 함수 호출이 없어도 자동 실행되는 함수를 선언한다.

10~12행 인자가 없는 일반적인 함수를 선언한다.

13~16행 인자도 있고 반환도 있는 함수를 선언한다.

17~22행 id가 btn3인 버튼을 제어하는 부분이다.

19행 btn3 요소에 onclick 이벤트 핸들러를 달아준다.

26행 인자 없는 함수를 태그의 인라인으로 호출한다.

27행 인자 있는 함수를 태그의 인라인으로 호출한다.

2.6 자바스크립트 객체의 멤버 접근

2.6.1 자바스크립트 객체

1 점(.) 연산자와 객체의 멤버 접근 방법

객체와 참조 변수는 다르다. 객체가 이동하는 것이 아니다. 객체의 주소를 알려주는 것이다. 땅을 사도 땅을 잃어지고 가지 않는다. 하지만 주소를 등기소에 등록하면 소유할 수 있는 것과 같은 이치다.

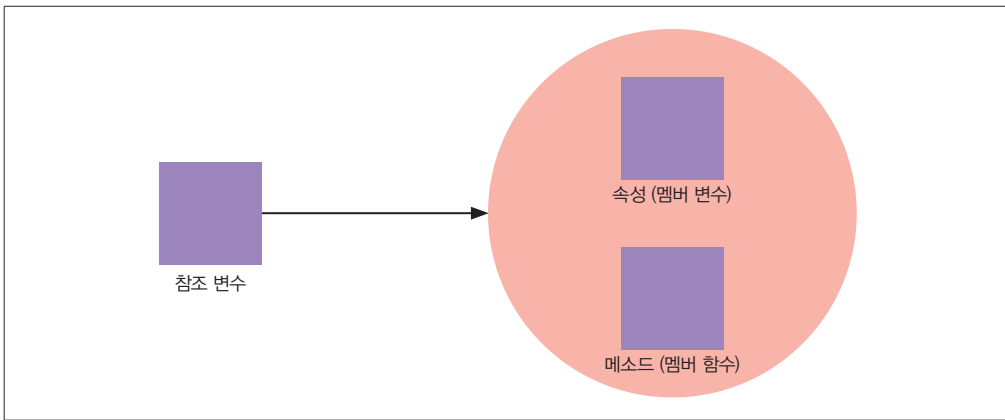


그림 2-34 참조 변수와 객체의 관계도

표 2-7 참조 변수를 통한 객체의 멤버 참조

| | |
|--------|---------------|
| 배열 초기화 | 참조 변수.멤버 변수 |
| 설명 | 참조 변수.멤버 함수() |

소스 2-26 사용자정의 객체 생성 및 객체의 멤버 추가, 참조

CODE js2.6.1(1)object.html

```
4 <script type="text/javascript">
5 function MkDiv() {
6     this.element = null; //멤버 필드
7     this.make = function(id, w, h, c) {
8         //id:엘리먼트의 아이디, w:너비, h:높이, c:색상
9         this.element = document.createElement("div");
10        this.element.id = id;
11        this.element.style.width = w+"px";
12        this.element.style.height = h+"px";
13        this.element.style.backgroundColor = c;
```

```

14         console.log(this);
15         return this.element;
16     } //멤버 메소드
17 }
18 window.onload = function() {
19     var newDiv = new MkDiv().make("newDiv", 300, 300, "Red");
20     document.body.appendChild(newDiv);
21 }
22 </script>

```

5~17행 사용자 객체를 생성하기 위한 클래스를 만들어준다. 자바스크립트에는 class 키워드가 없기 때문에 함수를 선언하는 것처럼 function 키워드를 이용한다.

6행 멤버 필드를 선언한다. 이때 사용하는 this는 생성될 객체의 멤버가 된다. 클래스 외부에서 this를 사용했다면 그것은 window 객체의 멤버가 된다.

7행 사용자 객체의 멤버 메소드를 선언한다.

9행 div 태그의 엘리먼트명을 이용해서 DOM을 얻어온다.

10행 인자로 넘겨져 온 id를 새로 만들어지는 엘리먼트의 id에 대입한다.

14행 클래스 내에서 this가 의미하는 것이 무엇인지 확인하기 위한 출력

15행 make 멤버 함수의 호출이 끝나면 함수가 새롭게 만들어준 결과를 돌려준다.

20행 객체를 이용해 MkDiv의 객체를 이용해서 새롭게 생성한 div 엘리먼트를 doby에 추가한다.

1 객체에 반복해서 접근하기

한번 생성된 이런 객체들은 반복해서 사용 가능하다. 소스 2-26에서 생성한 객체를 재활용하는 예제를 만들어보도록 하겠다. 소스 2-27은 생성된 객체를 반복 사용하는 방식으로 무지개 색 스펙트럼을 만드는 예제다. 이처럼 객체는 한번 만들면 계속해서 재활용할 수 있다는 장점이 있다.

소스 2-27 객체를 활용한 무지개 색 스펙트럼 구현

CODE js2.6.1(2)object.html

```

17 window.onload = function() {
18     var md = new MkDiv();
19     var colors = ["Red", "Orange", "Yellow", "Green", "Blue", "Navy", "Violet"];
20     for(var i=0; i<7; i++) {
21         var newDiv = md.make("id"+i, 50, 300, colors[i]);
22         newDiv.style.float = "left";
23         document.body.appendChild(newDiv);
24     }
25 }

```

- 18행 객체를 반복 사용하기 위해 객체를 참조하는 변수를 선언했다.
- 19행 반복 사용될 색상을 배열에 담아 준비했다.
- 22행 붙여지는 div 요소를 가로 정렬하기 위해서 style을 적용하였다.

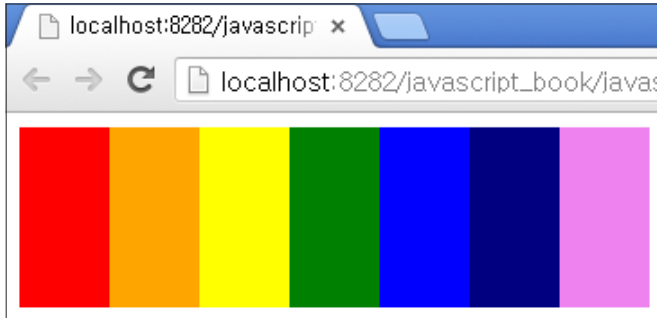


그림 2-35 사용자정의 클래스 MkDiv의 객체를 반복 사용한 결과

객체에 대해서는 이 책의 4장에서 좀 더 자세히 설명하도록 하고, 이번 장에서는 사용자정의 객체를 생성하고 호출하는 방식에 대해서만 이해하도록 한다. 이 예제에서 `make()` 멤버 메소드를 사용하는 방식은 생성자를 사용해서 작성하면 좀 더 간단히 사용할 수 있다. 클래스에 멤버 함수를 추가하는 것도 `prototype` 객체를 이용해서 성능을 더 향상시킬 수 있다.

2.7 이벤트 핸들러와 이벤트 처리

자바스크립트와 같은 스크립트 언어로 만든 프로그램은 대부분 웹 브라우저 안에서 보여지는 페이지의 보조적인 역할을 한다. 이럴 때 사용자는 그저 웹 서버에서 보내지는 데이터를 일방적으로 보는 것만으로는 해당 웹 사이트에 큰 흥미를 가질 수 없을 것이다. 이런 단조로운 웹 페이지에 다이내믹하게 생명력을 불어넣어 주는 것이 바로 자바스크립트인데, 이렇게 사용자의 행동에 즉각적인 반응을 할 수 있게 하는 것은 웹 브라우저와 사용자로부터 발생하는 여러 가지 이벤트를 자바스크립트가 모두 잡아낼 수 있기 때문이다. 이 절에서는 이런 이벤트를 자바스크립트에서 어떤 식으로 잡아내는지를 학습할 것이다.

이번 절에서는 자바스크립트에서 발생하는 이벤트를 어떤 방식으로 잡아서 활용하는지를 보여주는 예제를 만들 것이다. 이벤트가 발생한 것에 대한 단순한 반응을 하는 간단한 예제이지만, 이런 단순한 기능도 자바나 C++과 같은 언어로 구현하려면 상당히 까다로울 것이다. 자바스크립트에서는 미리 내장된 이벤트 기능을 이용해서 간단히 처리할 수 있다.

자바스크립트에서는 표 2-8과 같이 다양한 방식으로 이벤트 핸들러를 처리할 수 있다. 어떤 방식을 이용해도 크게 상관은 없지만, 필자는 2번 방식을 가장 선호한다. 왜냐하면, 3번은 표준이긴 하지만 우리나라 사람들이 아직도 많이 사용하는 인터넷 익스플로러가 표준을 잘 따르지 않기 때문에 크로스 브라우징 처리를 하지 않으면 오히려 문제를 일으키게 된다. 이를 해결하기 위해 예외 처리를 한다는 것은 너무나 번거로운 일이기도 하다.

표 2-8 자바스크립트에서 주로 사용되는 이벤트 처리 방식

| | |
|---|--|
| 1 | 인라인 이벤트 핸들러 - HTML 태그에 직접 이벤트 핸들러 달기 |
| | <code>링크</code> |
| 2 | 콜백 함수를 이용한 자바스크립트 블록 내에서 이벤트 핸들러 제어하기 |
| | <pre>객체.onclick = function() { alert("msg"); }</pre> |
| 3 | DOM Model2의 이벤트 |
| | <code>addEventListener("click", myFunction, false);</code>
<code>removeEventListener()</code> |
| | 인터넷 익스플로러 모델에선 <code>attachEvent</code> 의 반대는 <code>detachEvent</code> 다.
개발자는 고전적인 방식을 더 선호한다. |

자바스크립트의 `addEventListener` 이벤트 처리 방식에는 이벤트 버블링과 캡처링 방식을 지원하기 때문에 이를 직접 지정할 수 있다. 그러나 `attachEvent` 방식은 이런 것을 지원하지 않고 한 가지 방식만을 사용해야 한다.

표 2-9는 자바스크립트에서 자주 사용되는 이벤트 핸들러들이다. 이벤트 핸들러 이름만 봐도 무슨 의미인지 알 정도로 이름이 직관적이다. 자바스크립트는 이벤트 기반 언어라고 할 정도로 이벤트를 사용하는 것이 매우 중요하다. 그리고 대부분 프론트엔트 단의 작업은 사용자의

이벤트를 제어하는 작업이 된다. 아래의 이벤트 핸들러들을 잘 기억해두었다가 필요할 때 제대로 활용하길 바란다. 참고로, 서버 개발자는 서버 측 언어를 주로 사용해서 서버 단의 개발을 하는 개발자이고, 프론트엔드(front-end) 개발자는 자바스크립트, HTML5 등을 이용해서 프론트 단의 개발을 한다.

표 2-9 자바스크립트 작성 시 자주 사용되는 이벤트 핸들러

| 이벤트 핸들러 | 이벤트 핸들러의 작동 방식 설명 |
|--------------|---------------------------------------|
| onkeyup | • 키보드의 키를 눌렀다가 떼었을 때 발생하는 이벤트 |
| onload | • HTML 문서의 특정 요소가 모두 로드되었을 때 발생하는 이벤트 |
| onmousedown | • 마우스의 버튼을 눌렀을 때 발생하는 이벤트 |
| onmousemove | • 마우스를 움직일 때 발생하는 이벤트 |
| onmouseout | • 포커스에서 마우스 포인터가 나갈 때 발생하는 이벤트 |
| onmouseover | • 포커스로 마우스 포인터가 들어갈 때 발생하는 이벤트 |
| onmouseup | • 마우스의 버튼을 눌렀다가 떼었을 때 발생하는 이벤트 |
| onmove | • HTML 내의 특정 요소가 움직일 때 발생하는 이벤트 |
| onreset | • 폼의 리셋 버튼을 눌렀을 때 발생하는 이벤트 |
| onresize | • 문서의 특정 요소의 크기를 변경할 때 발생하는 이벤트 |
| onselect | • 문서 내의 특정 텍스트를 선택할 때 발생하는 이벤트 |
| onsubmit | • 폼의 서브밋 버튼을 눌렀을 때 발생하는 이벤트 |
| onunload | • 문서나 특정 요소가 사라질 때 발생하는 이벤트 |
| onmousewheel | • 마우스의 휠 버튼을 돌릴 때 발생하는 이벤트 |

이벤트가 발생할 때 크롬의 콘솔 창에 이벤트 결과를 출력해보면 이벤트 핸들러가 어떤 값을 사용하는지 확인할 수 있다. 이 값들을 이용해서 문서 내 요소들의 제어가 가능하다.

소스 2-28 이벤트가 발생할 때 전달되는 값 확인하기

CODE js2.7.1.event_value.html

```

3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5  <script type="text/javascript">
6  window.onload = function() {
7      var btn = document.getElementById("btn");
8      btn.onclick = function(e) {
9          console.log(e.target);
10         console.log(e.x); //클릭한 지점의 마우스 포인터 x
11         console.log(e.y); //클릭한 지점의 마우스 포인터 y

```

```

12         console.log(e.target.offsetLeft); //대상 요소의 left
13         console.log(e.target.offsetTop); //대상 요소의 right
14     }
15 }
16 </script>
17 </head>
18 <body>
19 <button id="btn">결과 확인</button>
20 </body>

```

8행 이벤트가 발생할 때 파라미터 e에 이벤트 발생과 관련된 정보가 담긴다.

10~13행 이벤트 발생 지점에서 보내지는 여러 속성들의 값을 확인해본다.

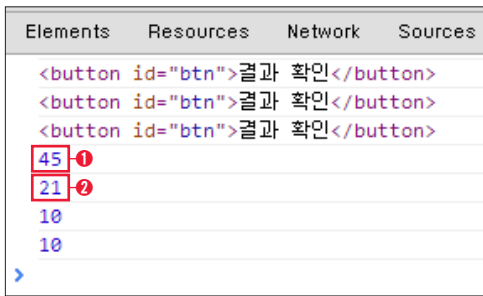


그림 2-36 Console에서 이벤트 발생 결과 확인

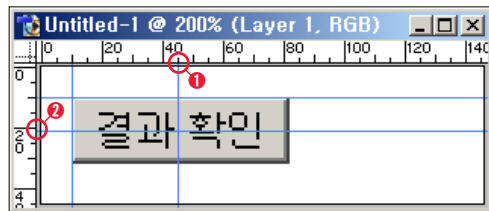


그림 2-37 Console에서 확인된 결과의 좌표의 실제 지점 확인

1 이벤트가 발생할 때 함수 호출하는 예제 1단계

소스 2-29 이벤트 발생 시 함수 호출 HTML 코드

CODE js2.7.1(1)event.html

```

22 <body onload = "start()">
23 <select id="nation">
24     <option value="">번호를 선택하세요.</option>
25     <option value="0">No.0</option>
26     <option value="1">No.1</option>
27     <option value="2">No.2</option>
28     <option value="3">No.3</option>
29 </select>
30 <ul>
31     <li>Korea</li>
32     <li>Japan</li>
33     <li>China</li>
34     <li>Betnam</li>
35 </ul>
36 </body>

```

23~29행 <select> 태그에 있는 <option>을 선택할 수 있도록 준비한다.

30~35행 <select> 태그에서 change 이벤트가 발생하면 피드백 효과를 보여줄 부분을 준비해둔다.

23행 자바스크립트에서 핸들링할 수 있도록 id="nation"으로 id 값을 지정해준다.

소스 2-30 이벤트 발생 시 함수 호출 자바스크립트 코드

CODE js2.7.1(1)event.html

```
5  <script type="text/javascript">
6  start = function() {
7      var li = document.getElementsByTagName("li");
8      var nation = document.getElementById("nation");
9      /* onchange - value 데이터가 바뀌면
10     onclick - 클릭하면
11     onmouseover - 마우스가 올라가면
12     onmouseout - 마우스가 내려가면
13     onfocus - 선택되면
14     onblur - 선택 해제되면
15     keypress, keyup ... 등 */
16     nation.onchange = function() {
17         alert(this.value);
18     }
19 }
20 </script>
```

7행 <select> 태그의 change 이벤트에 반응할 요소의 태그의 DOM을 배열로 받아온다.

8행 16행의 이벤트 핸들링을 하기 위해 id를 이용해서 DOM을 받아온다.

16~18행 <option> 태그를 선택할 때 해당 태그의 값을 보여준다.

9~15행 자주 사용되는 이벤트를 정리한 주석문이다.



번호를 선택 하세요. ▼

- Korea
- Japan
- China
- Bethnam

그림 2-38 이벤트 발생 시 함수 호출 결과 1

일단, 준비 단계에선 <select> 태그에서 change 이벤트가 발생하면 그림 2-39와 같이 다이얼로 그 박스에 선택한 번호의 value 값이 나오도록 했다. 이것을 활용해서 태그의 목록에 효과를 주는 것은 소스 2-31과 같은 방법으로 변경해보았다. 일단은 모든 요소의 스타일을 없애주도록 하고 선택된 요소만 배경색을 다시 지정하는 단순한 방법을 사용해보았지만, 선택 요소의 이웃 요소를 트래버싱(traversing)하여 찾는 방법으로도 할 수 있다. 한번 도전해보길 바란다.

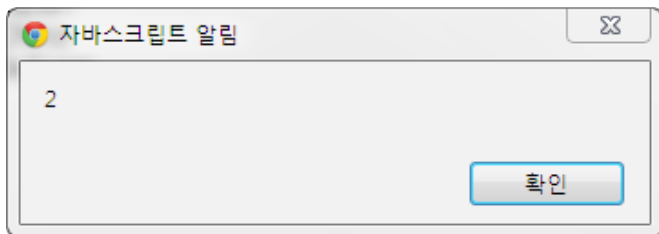


그림 2-39 이벤트 발생 시 함수 호출 결과 1

2 이벤트가 발생할 때 함수 호출하기 2단계

소스 2-30에서 이벤트에 반응하는 것을 확인하였다면 이것을 활용해서 좀 더 구체적으로 작용하는 예제를 만들어보자. 소스 2-31은 이벤트가 발생한 개체의 value 값을 얻어와서 그 값과 일치하는 요소의 배경색을 변경하는 예제다.

소스 2-31 초기화 기능 추가

CODE js2.7.1(2)event.html


```

8   <script type="text/javascript">
9   start = function() {
10      var li = document.getElementsByTagName("li");
11      var nation = document.getElementById("nation");
12
13      nation.onchange = function() {
14          li[0].style.backgroundColor = "";
15          li[1].style.backgroundColor = "";
16          li[2].style.backgroundColor = "";
17          li[3].style.backgroundColor = "";
18          li[this.value].style.backgroundColor = "Yellow";
19      }
20  }
21  </script>

```

14~17행 일단, 모든 태그의 배경색을 없애도록 초기화한다.

18행 선택된 <option> 태그의 value 속성 값과 일치하는 인덱스 요소의 배경색만 노랑색으로 변경하도록 한다.

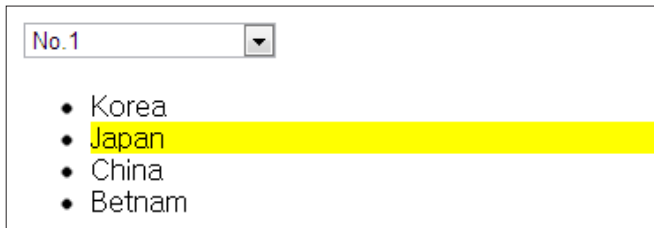


번호를 선택 하세요. ▼

- Korea
- Japan
- China
- Betnam

그림 2-40 초기화 기능 추가_결과 1

실행하면 이번에는 선택한 요소의 배경색이 변경되는 것을 확인할 수 있다. 이처럼 이벤트는 평상시에는 아무 일도 하지 않지만, 어떤 사건이 발생하면 그때 지정된 일을 하도록 할 수 있다. GUI 환경에서의 프로그래밍은 이러한 이벤트 처리가 매우 중요하다는 것은 두말할 나위가 없다.



No.1 ▼

- Korea
- Japan
- China
- Betnam

그림 2-41 초기화 기능 추가_결과 2

지금까지 자바스크립트에 대한 기본 사용법을 정리해보았다. 이번 장에서 자바스크립트에서 많이 사용되는 일반적인 사항들을 정리하였기 때문에 2장에서 설명한 부분만 잘 이해해도 웬만큼 자바스크립트 소스를 이해하는 데에는 크게 막힘이 없을 것이다. 다음 장부터 좀 더 구체적인 결과를 만들어내는 예제를 다루면서 학습해보도록 하자.