



***2**

웹프로그래밍 복습

UT-NodeJS / 03.10.2023

ut-nodejs.github.io



Contents / 내용

Prerequisite knowledge / 전제 지식

설문: <https://forms.gle/S16WqFdeax5fTMuv6>

위를 해본 적이 있어요?

01. HTML

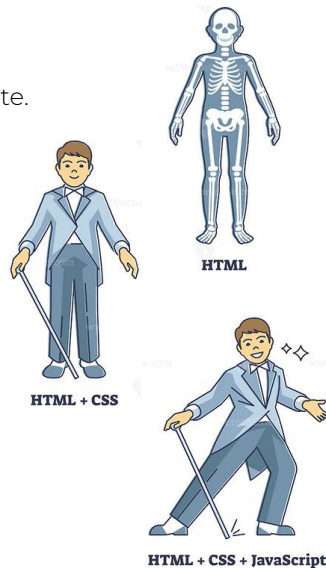
Structural: providing the frame or skeleton of the website.
구조: 웹사이트의 프레임 또는 골격을 제공합니다.

02. CSS

Presentational: providing the paint or skin and style.
외모: 페인트 또는 스킨 및 스타일 제공.

03. JavaScript

Behavioral: providing the interactivity and functionality.
행동: 상호 작용 및 기능 제공.



01

HTML

Structural: providing the frame
or skeleton of the website.
구조: 웹사이트의 프레임 또는 골격을 제공합니다.



HTML



HTML를 얼마나 알아야 한다?

How much HTML do I need to know?



Basic Concepts

1. Structure / 구조
2. Text / 텍스트
3. Lists / 목록
4. Links / 링크
5. Images / 이미지
6. Tables / 테이블
7. Forms / 폼
8. Extra Markup / 추가 마크업
9. Flash, Video, & Audio / 플래시, 비디오, 오디오



HTML은 요소를 사용해 페이지의 구조를 설명한다

앞 페이지에 나온 코드를 좀 더 자세히 살펴보자.
여러 가지 요소가 있으며, 각 요소에는
열기 태그와 닫기 태그가 있다.

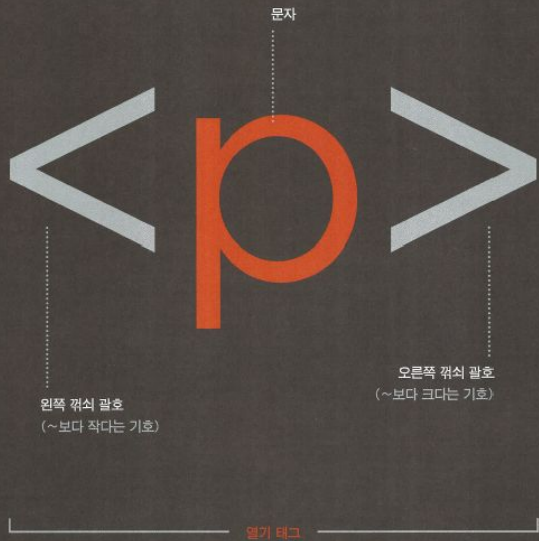
태그는 컨테이너와 같다. 열기 태그와 닫기 태그
사이에 위치하는 정보에 대한 사항을 알려준다.

코드

설명

| | |
|---|---|
| <pre><html></pre> | <pre><html>태그는 열기 태그로 닫기 태그인</html> 사이에 HTML 코드를 나타낸다.</pre> |
| <pre><body></pre> | <pre><body>태그는 열기 태그로 닫기 태그인 </body>사이에 메인 브라우저 창에 표시되는 내용을 나타낸다.</pre> |
| <pre><h1>메인 제목</h1></pre> | <pre><h1>와</h1>사이에 있는 단어는 메인 제목이다.</pre> |
| <pre><p>전체 페이지를 소개하는 내용을 작성한다. 메인 제목이 될 경우에는 여러 개의 부 제목으로 나눈다.</p></pre> | <pre><p>와</p>사이에 있는 내용은 텍스트의 단락이다.</pre> |
| <pre><h2>첫 번째 부 제목</h2></pre> | <pre><h2>와</h2>사이에 있는 단어는 부 제목이다.</pre> |
| <pre><p>긴 기사의 경우 대부분 부 제목을 작성한다. 경우에 따라서는 부 부 제목(부 제목의 부 제목)을 작성하기도 한다.</p></pre> | <pre><p>와</p>사이에 있는 내용은 또 다른 단락이다.</pre> |
| <pre><h2>두 번째 부 제목</h2></pre> | <pre><h2>와</h2>사이에 있는 단어는 또 다른 부 제목이다.</pre> |
| <pre><p>두 번째 부 제목 부분이다.</p></pre> | <pre><p>와</p>사이에 있는 내용은 또 다른 단락이다.</pre> |
| <pre></body></pre> | <pre>닫기 태그인</body>는 메인 브라우저 창에 표시되는 내용의 끝을 나타낸다.</pre> |
| <pre></html></pre> | <pre>닫기 태그인</html>은 HTML 코드의 끝을 나타낸다.</pre> |

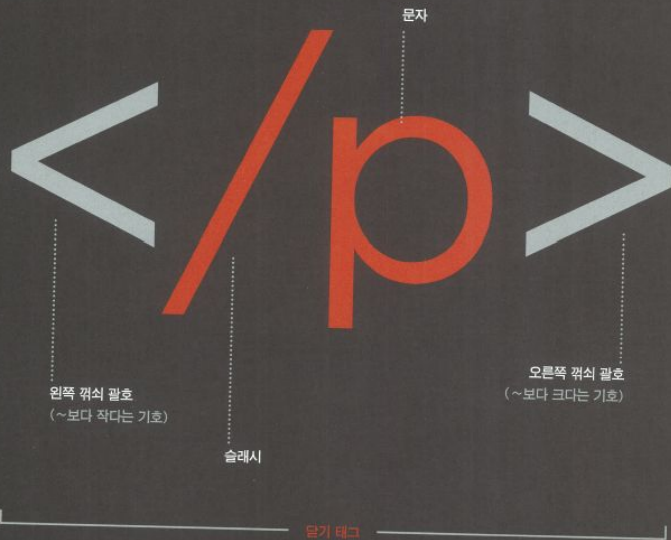
태그 상세 분석



괄호 안에 있는 문자는 태그의 목적을 나타낸다.

예컨대 위에 있는 p는 paragraph(단락)를 의미한다.

닫기 태그는 < 기호 뒤에 슬래시가 있다.



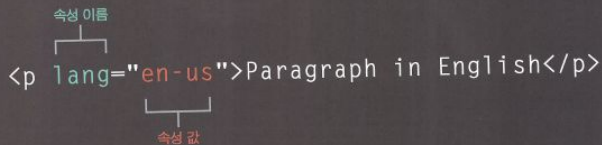
'태그'와 '요소'라는 용어는 종종 같은 의미로 사용한다.

하지만 엄밀히 말해 요소는 열기 태그와 닫기 태그 그리고 이 사이

에 있는 모든 콘텐츠를 포함한다.

속성은 요소에 대해 자세히 설명해 준다

속성은 요소의 콘텐츠에 대한 추가 정보를 제공한다. 속성은 요소의 열기 태그에 위치하며, 등호(=)로 구분한 이름^{name}과 값^{value}이라는 두 부분으로 구성돼 있다.

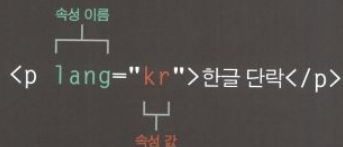
A diagram showing the HTML tag <p lang="en-us">Paragraph in English</p>. A bracket above the text "lang="en-us"" is labeled "속성 이름" (Attribute Name). A bracket below the text "en-us" is labeled "속성 값" (Attribute Value).

```
<p lang="en-us">Paragraph in English</p>
```

속성 이름은 요소의 콘텐츠에 대해 제공하는 추가 정보의 종류를 나타낸다. 속성 이름은 소문자로 작성해야 한다.

같은 속성에 대한 정보 또는 설정이다. 값은 큰따옴표("") 안에 위치해야 한다. 다른 속성은 다른 값을 가질 수 있다.

위 예제에 있는 lang이라는 속성은 이 요소에서 사용되는 언어를 나타내는 데 사용한다. 이 속성의 값은 이 페이지에 미국식 영어를 지정한다.

A diagram showing the HTML tag <p lang="kr">한글 단락</p>. A bracket above the text "lang="kr"" is labeled "속성 이름" (Attribute Name). A bracket below the text "kr" is labeled "속성 값" (Attribute Value).

```
<p lang="kr">한글 단락</p>
```

비록 lang 같은 일부 속성은 모든 요소에서 사용할 수 있지만 대부분 속성은 특정 요소에서만 사용할 수 있다.

대부분 속성 값은 미리 정의돼 있거나 규정된 형식을 따른다. 각각의 새로운 속성을 소개할 때 허용 값을 살펴보겠다.

HTML5에서는 대문자 속성 이름을 사용하고 따옴표를 생략할 수도 있지만 권장하지는 않는다.



HTML를 얼마나 알아야 한다?

How much HTML do I need to know?



Basic Concepts

1. Structure / 구조
 - a. `<body>`
 - b. `<head>`
 - c. `<title>`
2. Text / 텍스트
 - a. `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`
 - b. `<p>`
 - c. ``, ``, `<i>`, ``
 - d. `<sup>`, `<sub>`
 - e. `
`, `<hr>`
 - f. `<blockquote>`, `<q>`, `<cite>`
 - g. `<abbr>`, `<dfn>`
 - h. `<address>`
 - i. `<ins>`, ``, `<s>`, `<mark>`



HTML를 얼마나 알아야 한다?

How much HTML do I need to know?



Basic Concepts

3. Lists / 목록
 - a. <a>, ,
 - b. <dl>, <dt>, <dd>
4. Links / 링크
 - a. <a>
 - i. href="<https://www.google.com>"
 - ii. href="<mailto:aaronkr.trainer@gmail.com>"
 - iii. target="_blank"
5. Images / 이미지
 - a.
 - i. src, alt, title, height, width
 - b. <figure>, <figcaption>
6. Tables / 테이블
 - a. <table>, <tr>, <td>, <th>
 - b. <thead>, <tbody>, <tfoot>





HTML를 얼마나 알아야 한다?

How much HTML do I need to know?



Basic Concepts

7. Forms / 폼
 - a. <form>, action, method, id
 - b. <input>, type, name, size, maxlength
 - i. type="text", "password", "email", "url", "search", placeholder
 - ii. type="radio", "checkbox", value, checked
 - iii. type="file", "image", "submit"
 - iv. type="hidden", "date",
 - c. <textarea>
 - d. <select>, <option>, selected, multiple
 - e. <button>, <input>
 - f. <label>, for, <fieldset>, <legend>
8. Extra Markup / 추가 마크업
 - a. <!-- Comment -->
 - b. id, class
 - c. <div>, , <iframe>, <meta>
 - d. 확장 문자





HTML를 얼마나 알아야 한다?

How much HTML do I need to know?



Basic Concepts

9. Flash, Video, & Audio / 플래시, 비디오, 오디오
 - a. `<video>`
 - b. `<source>`
 - c. `<audio>`
 - d. `src, controls, autoplay, preload, loop, type, codecs, poster, width, height`



02

CSS

Presentational: providing the
paint or skin and style.
외모: 페인트 또는 스킨 및 스타일 제공.



HTML + CSS



CSS를 얼마나 알아야 한다?

How much CSS do I need to know?



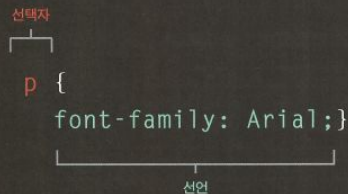
Basic Concepts

1. Color / 색상
2. Text / 텍스트
3. Boxes / 박스
4. Lists / 목록
5. Tables / 테이블
6. Forms / 폼
7. Layout / 레이아웃
8. Images / 이미지
9. HTML5 Layout / HTML5 레이아웃



CSS는 HTML 요소와 스타일 규칙을 연결한다

CSS는 HTML 요소와 규칙의 연결에 의해 동작한다. 이러한 규칙은 지정된 요소의 콘텐츠가 표시되는 방식을 결정한다. CSS 규칙은 **선택자^{selector}**와 **선언^{declaration}**이라는 두 부분으로 구성돼 있다.



이 규칙은 모든 <p> 요소가 예리얼^{Arial} 서체로 표시돼야 함을 나타낸다.

선택자는 규칙을 적용할 요소를 나타낸다. 링크 요소 이름을 구분하면 동일한 규칙을 하나 이상의 요소에 적용할 수 있다.

선언은 선택자로 지정한 요소에 스타일을 적용하는 방법을 나타낸다. 선언은 프로퍼티와 값이라는 두 부분으로 분리하며 콜론으로 구분한다.

CSS 프로퍼티는 요소가 표시되는 방식에 영향을 미친다

CSS 선언은 중괄호 안에 작성하며 **프로퍼티**와 **값**이라는 두 부분으로 구성하고 콜론으로 구분한다. 또한 하나의 선언에서 세미콜론으로 구분하여 여러 프로퍼티를 지정할 수 있다.

```
h1, h2, h3 {
  font-family: Arial;
  color: yellow;
}
┌───┐ ┌───┐
프로퍼티  값
```

이 규칙은 모든 <h1>, <h2>, <h3> 요소가 노란색의 예리얼 서체로 표시돼야 함을 의미한다.

프로퍼티는 변경하려는 요소의 부분을 나타낸다. 예컨대 색상, 서체, 너비, 높이, 테두리 등이 있다.

값은 선택한 프로퍼티에 대해 사용하려는 설정을 지정한다. 예컨대 color 프로퍼티를 지정한 경우 값은 해당 요소에 있는 텍스트의 색상이 될 수 있다.

chapter-10/cascade.htm

HTML

```
<h1>감자(Potato)</h1>
<p id="intro"><i>수십 가지</i>의 <b>감자</b> 품종이 존재한다.</p>
<p>재배작형에 따라서는 1년에 한 번 심는 1기작과 두 번 심는 2기작 품종이 있다.</p>
```

CSS

```
* {
  font-family: Arial, Verdana, sans-serif;
}
h1 {
  font-family: "Courier New", Courier, monospace;
}
i {
  color: green;
}
i {
  color: red;
}
b {
  color: pink;
}
p b {
  color: blue !important;
}
p b {
  color: violet;
}
p#intro {
  font-size: 100%;
}
p {
  font-size: 75%;
}
```

결과

감자 (Potato)

*수십 가지*의 감자 품종이 존재한다.

재배작형에 따라서는 1년에 한 번 심는 1기작과 두 번 심는 2기작 품종이 있다.

HTML

chapter-10/inheritance.html

```
<div class="page">
  <h1>감자(Potato)</h1>
  <p>수십 가지의 감자 품종이 존재한다.</p>
  <p>재배작형에 따라서는 1년에 한 번 심는 1기작과 두 번 심는 2기작 품종
  이 있다.</p>
</div>
```

CSS

```
body {
  font-family: Arial, Verdana, sans-serif;
  color: #665544;
  padding: 10px;
}
.page {
  border: 1px solid #665544;
  background-color: #efefef;
  padding: inherit;
}
```

결과

감자(Potato)

수십가지의 감자 품종이 존재한다.

재배작형에 따라서는 1년에 한 번 심는 1기작과 두 번 심는 2기작 품종이 있다.

| 선택자 | 의미 | 예제 |
|-----------|---|---|
| 범용 선택자 | 문서에 있는 모든 요소를 의미한다. | * {} 페이지에 있는 모든 요소를 대상으로 한다. |
| 타입 선택자 | 요소 이름을 의미한다. | h1, h2, h3 {} <h1>, <h2>, <h3> 요소를 대상으로 한다. |
| 클래스 선택자 | class 속성의 값이 마침표(.) 이후의 값과 같은 요소를 의미한다. | .note {} class 속성의 값이 note인 요소를 대상으로 한다. p.note {} class 속성의 값이 note인 p 요소를 대상으로 한다. |
| id 선택자 | id 속성의 값이 해시 기호(#) 이후의 값과 같은 요소를 의미한다. | #introduction {} id 속성의 값이 introduction인 요소를 대상으로 한다. |
| 자식 선택자 | 직계 자식 요소를 의미한다. | i>a {} <i> 요소의 바로 아래에 있는 <a> 요소를 대상으로 한다(페이지에 있는 다른 <a> 요소가 아니라). |
| 자손 선택자 | 직계 자식뿐만 아니라 특정 요소의 자손 요소를 의미한다. | p a {} <p> 요소 내에 있는 <a> 요소를 대상으로 한다. 두 요소 사이에 다른 요소가 증첩 돼 있어도 상관 없다. |
| 인접 형제 선택자 | 바로 옆에 있는 요소를 의미한다. | h1+p {} <h1> 요소 바로 다음에 오는 <p> 요소를 대상으로 한다. |
| 일반 형제 선택자 | 형제 관계에 있는 요소를 의미한다(바로 옆에 없어도 된다). | h1-p {} <h1> 요소와 두 <p> 요소가 형제 관계에 있다면 두 <p> 요소에 모두 적용된다. |





CSS를 얼마나 알아야 한다?

How much CSS do I need to know?



Basic Concepts

1. Color / 색상
 - a. color, background-color
 - i. RGB 값, hexa 코드, 색상 명, HSL (색조, 채도, 명도)
 1. opacity, rgba, hsl, hsla
2. Text / 텍스트
 - a. serif, sans-serif, monospace, cursive, fantasy
 - b. @font-face (eot, woff, ttf/otf, svg), font-family, font-size (px, %, em, rem)
 - c. font-weight, font-style
 - d. text-transform, text-decoration
 - e. line-height, letter-spacing, word-spacing
 - f. text-align, vertical-align
 - g. text-indent, text-shadow
 - h. :first-letter, :first-line, :link, :visited, :hover, :active, :focus





CSS를 얼마나 알아야 한다?

How much CSS do I need to know?



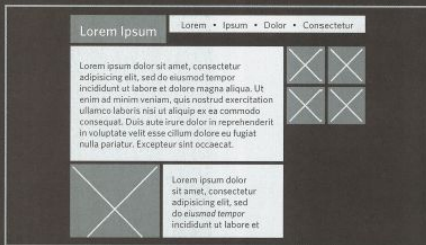
Basic Concepts

3. Boxes / 박스
 - a. width, min-width, max-width, height, min-height, max-height, overflow
 - b. border,
 - i. border-width, border-style, border-color, border-image, border-radius
 - c. margin, padding, box-shadow
 - d. display, visibility
4. Lists / 목록
 - a. list-style-type, list-style-image, list-style-position, list-style
5. Tables / 테이블
 - a. empty-cells, border-spacing, border-collapse
6. Forms / 폼
 - a. input, input:focus, input:hover
 - b. fieldset, legend
 - c. cursor



고정 너비 레이아웃

고정 너비 레이아웃
width layout 디자인은 사용자 브라우저 창의 크기를 늘리거나 줄이더라도 크기가 변경되지 않는다. 대체로 단위는 픽셀을 사용한다.



장점

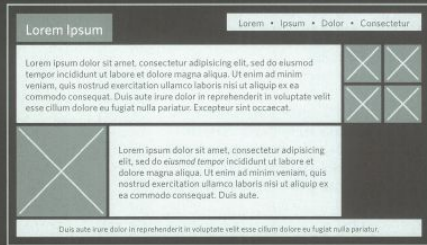
- 픽셀 값을 사용하여 요소의 크기와 위치를 정확히 제어할 수 있다.
- 유동 레이아웃보다 항목의 모양이나 위치의 제어가 용이하다.
- 화면 크기와 상관 없이 텍스트 줄의 길이를 제어할 수 있다.
- 이미지의 크기는 페이지의 나머지 부분에서도 항상 동일하게 유지된다.

단점

- 페이지의 가장자리에 큰 여백이 생길 수 있다.
- 사용자의 화면 해상도가 디자인 너의 화면보다 더 높을 경우 페이지가 더 작아 보일 수 있으며 텍스트는 읽기 더 어려워질 수 있다.
- 사용자가 폰트 크기를 늘리면 텍스트가 할당된 영역에 맞지 않을 수 있다.
- 디자인은 데스크톱이나 노트북 컴퓨터와 유사한 크기나 해상도를 제공하는 기기에 최적화 돼 있다.
- 페이지는 동일한 콘텐츠에 대해 유동 레이아웃보다 세로 영역을 더 많이 차지한다.

유동 레이아웃

유동 레이아웃
liquid layout 디자인은 사용자가 브라우저 창의 크기를 늘리거나 줄이면 커지거나 작아진다. 대체로 단위는 퍼센트를 사용한다.



장점

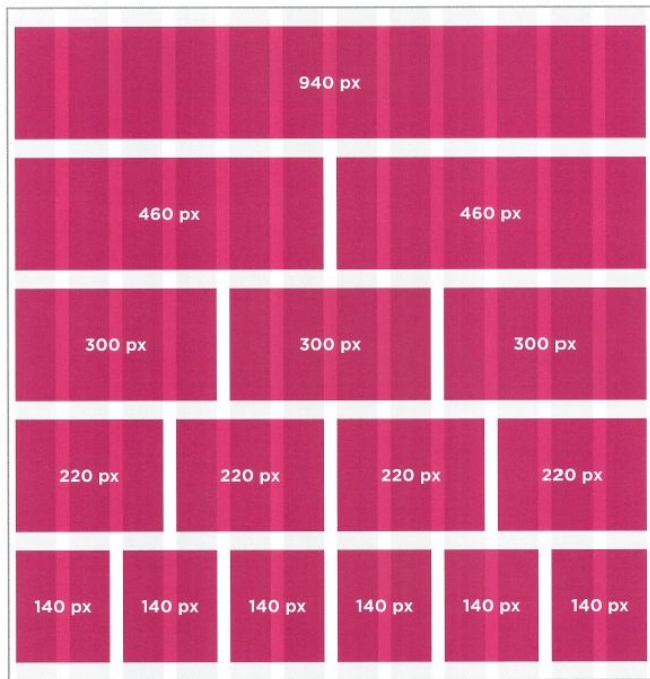
- 페이지는 전체 브라우저 창을 채우기 위해 확대된다. 따라서 큰 화면의 페이지 주위에 여백이 생기지 않는다.
- 사용자의 창이 작으면 사용자가 옆으로 스크롤하지 않아도 페이지에 맞게 축소된다.
- 페이지가 늘어날 수 있으므로 사용자는 디자인의 의도보다 폰트 크기를 더 크게 설정할 수 있다.

단점

- 페이지의 선택 너비를 제어하지 않으면 특정 요소 주위에 예기치 못한 여백이나 밀린 항목으로 인해 의도한 디자인과 많이 달라질 수 있다.
- 사용자의 창이 넓으면 텍스트의 줄이 매우 길어져 가독성이 떨어질 수 있다.
- 사용자의 창이 좁으면 단어가 밀릴 수 있으며 일부 단어는 여러 줄에 걸쳐 나타날 수도 있다.
- 사용자의 창을 너무 작게 만들면 이미지 같은 고정 너비 항목이 표현하기에 너무 작은 박스에 있게 되면 이미지가 텍스트 영역을 침범할 수 있다.

유동 레이아웃은 브라우저의 전체 너비로 확대할 수 있으므로 결과적으로 텍스트의 줄이 길어져 가독성이 떨어질 수 있다. 따라서 유동 레이아웃은 페이지의 일부뿐만 확대하거나 축소시키도록 적용한다. 페이지의 나머지 부분은 최소나 최대 너비를 설정한다.

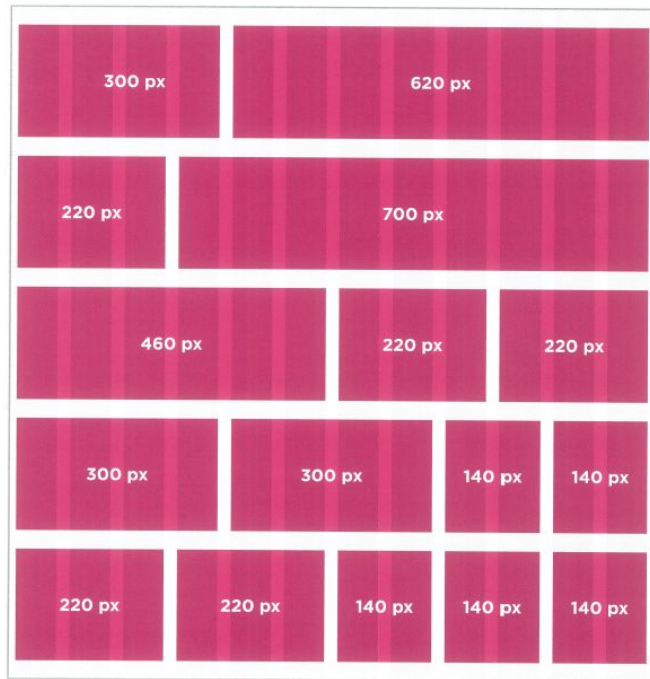
가능한 레이아웃: 960픽셀 폭, 12열 그리드



두 페이지에 걸쳐 960픽셀의 너비, 12열의 그리드를 볼 수 있다. 이는 하나의 그리드를 사용하여 다양한 형태의 열 레이아웃을 만들 수 있다는 사실을 보여준다.

페이지의 너비는 960픽셀이고 12개의 동일한 크기의 열이 있는데 (회색 표시), 각 열의 너비는 60픽셀이다.

각 열은 마진을 10픽셀로 설정했으므로 각 열 사이에는 20픽셀의 여백이, 페이지의 왼쪽과 오른쪽 면에는 10픽셀의 여백이 생긴다.





CSS를 얼마나 알아야 한다?

How much CSS do I need to know?



Basic Concepts

7. Layout / 레이아웃
 - a. 요소의 위치 제어 / position:
 - i. static, relative, absolute, fixed, z-index, float, clear
 - b. @import, link
8. Images / 이미지
 - a. background, background-image, background-repeat, background-attachment, background-position
9. HTML5 Layout / HTML5 레이아웃
 - a. <header>, <footer>, <nav>, <section>, <article>, <aside>, <hgroup>
 - b. <figure>, <figcaption>, <div>



전통적인 HTML 레이아웃

오랜 세월 웹 페이지 제작자들은 헤더, 기사, 푸터, 사이드바 요소처럼 페이지에 있는 관련 요소들을 그룹화하는 데 <div> 요소를 사용했다. 또한 페이지의 구조에서 <div> 요소의 역할을 나타내는 데에는 class나 id 속성을 사용했다.

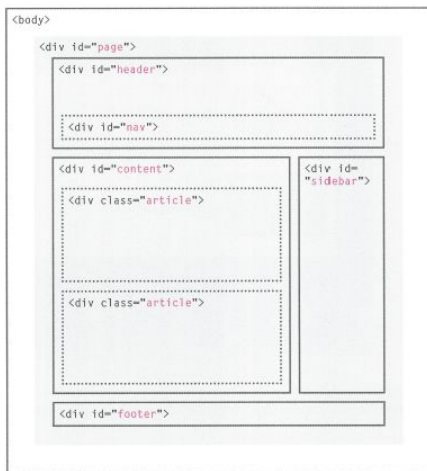
오른쪽 그림은 주로 블로그 사이트에서 볼 수 있는 매우 일반적인 레이아웃이다.

페이지의 상단은 헤더로, 로고와 주요 내비게이션이 위치한다.

헤더 아래에는 여러 가지 기사나 글이 위치한다. 때로는 각각의 글로 링크되는 요약문이 위치하기도 한다.

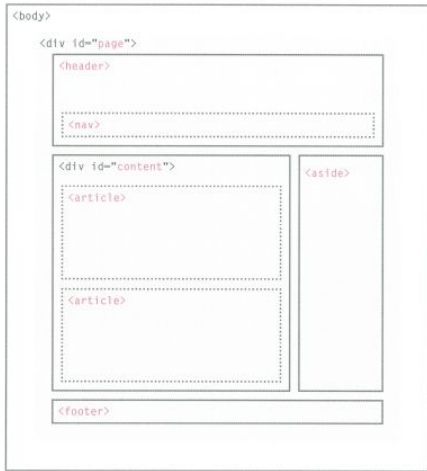
우측에는 사이드바가 위치한다. 대체로 검색 창이나 다른 최근 글, 사이트의 다른 섹션, 또는 광고에 대한 링크가 존재한다.

이와 같은 방식으로 사이트들 구축할 경우 개발자는 일반적으로 페이지의 메인 섹션을 <div> 요소 내에 배치하고 class와 id 속성을 사용하여 페이지의 해당 부분에 대한 목적을 나타낸다.



새로운 HTML5 레이아웃 요소

HTML5는 페이지를 여러 부분으로 나눌 수 있는 새로운 요소들을 추가했다. 새로운 요소의 이름은 콘텐츠의 종류를 나타낸다. 새로운 요소 역시 계속 변경되고 있지만 이미 수많은 웹 페이지 제작자가 사용하고 있다.

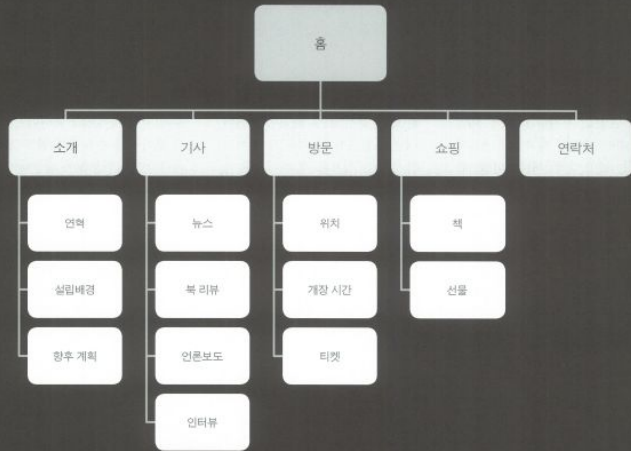


이번 예제는 이전 페이지와 동일한 구조로 구성돼 있다. 하지만 많은 <div> 요소를 새로운 HTML5 레이아웃 요소로 대체했다.

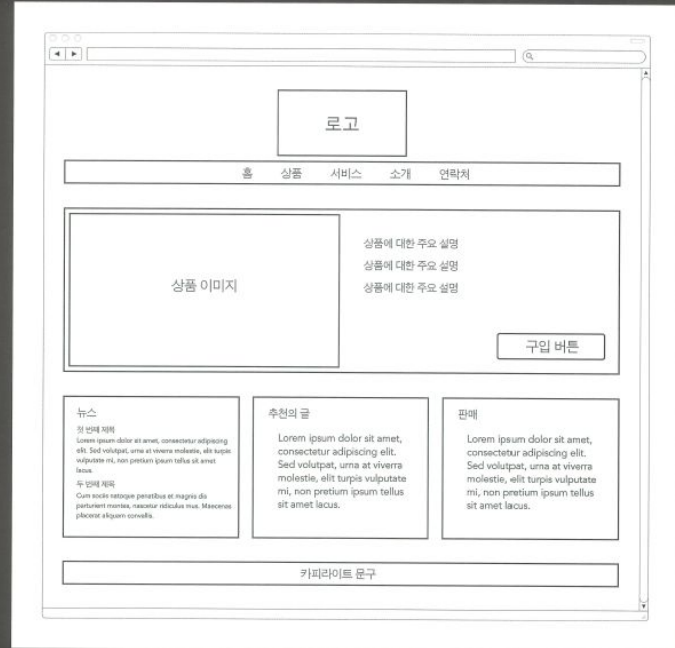
예컨대 헤더는 새로운 <header> 요소 안에 있으며 내비게이션은 <nav> 요소에 있다. 그리고 기사는 각각의 <article> 요소에 있다.

이러한 새로운 요소 생성의 핵심은 웹 페이지 제작자가 이들 요소를 사용하여 페이지의 구조를 표시할 수 있다는 점이다. 한 예로 스크린 리더 소프트웨어는 헤더와 푸터를 무시하고 바로 콘텐츠로 이동하게 할 수 있으며, 미친기저코 검색 엔진은 <header>나 <footer> 요소보다 <article> 요소의 콘텐츠에 더 높은 가중치를 부여할 수 있다. 그리고 코드에 대한 가독성도 높아진다.

사이트맵 예제



예제 와이어프레임



03

JavaScript

Behavioral: providing the
interactivity and functionality.

행동: 상호 작용 및 기능 제공.

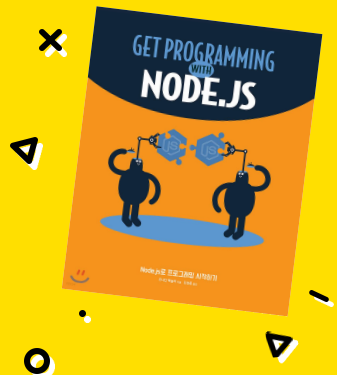


HTML + CSS + JavaScript



JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



Basic Concepts

1. [Lexical Structure](#) / 여휘 문법
2. [Expressions](#) / 식 및 연산자
3. [Data Types](#) / 타입과 자료구조
4. [Classes](#) / class
5. [Variables](#) / 변수
6. [Functions](#) / 함수
7. [this operator](#) / this
8. [Arrow Functions](#) / 화살표 함수
9. [Loops](#) / 루프와 반복
10. [Scopes](#) / 스코프
11. [Arrays](#) / array
12. [Template Literals](#) / 템플릿 문자열
13. [Strict Mode](#) / 엄격 모드
14. [ECMAScript 2015 \(ES6\) and beyond](#) / JavaScript 기술 개요



Asynchronous Programming

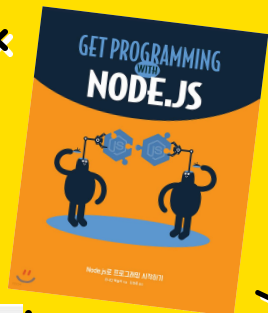
15. [Async programming](#) / 동기 프로그래밍
16. [Timers](#) / [setTimeout\(\)](#)
17. [Promises](#) / [promise](#)
18. [Async and Await](#) / [async function](#)
19. [Closures](#) / 클로저
20. [The Event Loop](#) / 이벤트 루프





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



1. Lexical Structure / 어휘 문법

- Format-control characters / 제어 문자
- White space / 공백
- Line terminators / 개행 문자
- Comments / 주석
- Identifiers / 해시뱅 주석
- Keywords / 키워드
- Literals / 리터럴
 - null, 불리언
 - 숫자: 10진법, 2진법, 8진법, 16진법, BigInt, _
 - 객체, 배열, 문자열, 이스케이프 시퀀스
 - 정규 표현식, 템플릿
- Automatic semicolon insertion / 자동 세미콜론 삽입
- ECMAScript Language Specifications / 명세
- Browser compatibility / 브라우저 호환성
- See also / 같이 보기

```
function comment() {  
  /* 자바스크립트 각주 한 줄입니다. */  
  console.log("Hello world!");  
}  
comment();
```

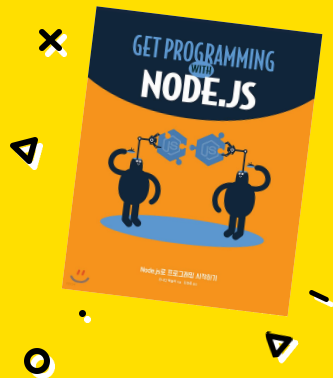
```
// separators in decimal numbers  
1_000_000_000_000  
1_050.95  
  
// separators in binary numbers  
0b1010_0001_1000_0101  
  
// separators in octal numbers  
0o2_2_5_6  
  
// separators in hex numbers  
0xA0_B0_C0  
  
// separators in BigInts  
1_000_000_000_000_000_000_000n
```





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



2. Expressions / 식 및 연산자

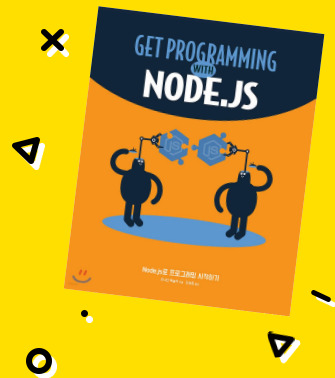
- 기본 식
 - this, function, class, function*, yield, yield*, async function, await, [], {}, /ab+c/i, (,)
- 좌변 식
 - object.property, new, new.target, super, ...obj
- 증가 및 감소
 - A++, A--, ++A, --A
- 단항 연산자
 - delete, void, typeof, +, -, ~, !
- 산술 연산자
 - +, -, /, *, %, **
- 관계 연산자
 - in, instanceof, <, >, <=, >=
- 같은 연산자
 - ==, !=, ===, !==
- 비트 시프트 연산자
 - <<, >>, >>>
- 이진 비트 연산자
 - &, |, ^
- 이진 논리 연산자
 - &&, ||, ??
- 조건부(삼항) 연산자
 - condition ? ifTrue : ifFalse
- 선택적 연결 연산자
 - ?. (nullish, null, undefined)
- 할당 연산자
 - =, *=, **=, /=, %=, +=, -=, <<=, >>=, >>>=, &=, |=, &=, &=, ||=, ??=, [a,b] = [1,2], {a,b} = {a:1, b:2}
- 쉼표 연산자
 - ,





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



3. Data Types / [타입과 자료구조](#)

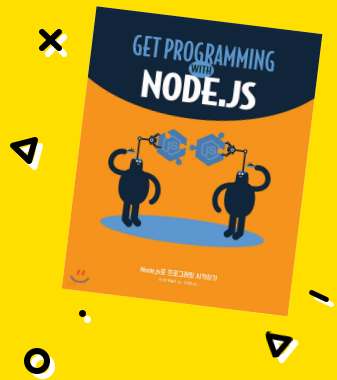
- 원시 값 (언어의 최고 로우레벨에서 직접 표현되는 불변 데이터)
 - Boolean 타입
 - Null 타입
 - Undefined 타입
 - Number 타입
 - BigInt 타입
 - String 타입
 - Symbol 타입
- 객체 (속성의 컬렉션)
 - 데이터 속성
 - [[Value]], [[Writable]], [[Enumerable]], [[Configurable]]
 - 접근자 속성
 - [[Get]], [[Set]], [[Enumerable]], [[Configurable]]





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



4. Classes / class

```
class Point {
  constructor(x, y) {
    this.x = x;
    this.y = y;
  }

  static displayName = "Point";
  static distance(a, b) {
    const dx = a.x - b.x;
    const dy = a.y - b.y;

    return Math.hypot(dx, dy);
  }
}

const p1 = new Point(5, 5);
const p2 = new Point(10, 10);
p1.displayName; // undefined
p1.distance; // undefined
p2.displayName; // undefined
p2.distance; // undefined

console.log(Point.displayName); // "Point"
console.log(Point.distance(p1, p2)); // 7.0710678118654755
```

```
class Animal {
  constructor(name) {
    this.name = name;
  }

  speak() {
    console.log(`${this.name} makes a noise.`);
  }
}

class Dog extends Animal {
  constructor(name) {
    super(name); // super class 생성자를 호출하며 name 매개변수 전달
  }

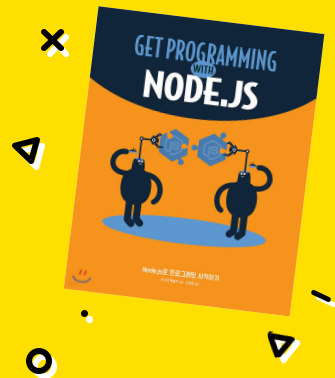
  speak() {
    console.log(`${this.name} barks.`);
  }
}

let d = new Dog('Mitzie');
d.speak(); // Mitzie barks.
```



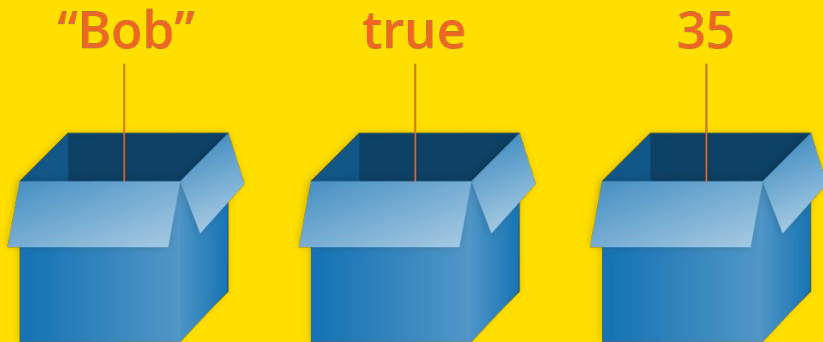
JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



5. Variables / 변수

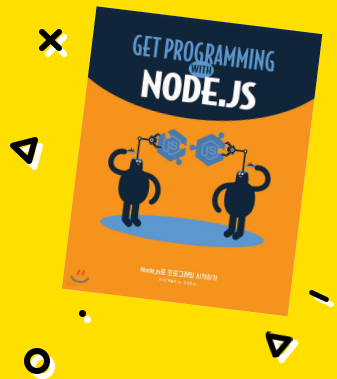
- 변수란?
 - 변수란, 숫자(합계나 계산에 사용되는) 또는 문자열(문장의 일부로 사용되는)과 같은 값의 컨테이너입니다. 그러나 변수에 대한 한 가지 특별한 점은 포함된 값이 변경될 수 있다는 것입니다. 간단한 예를 살펴 보겠습니다:





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



6. Functions / 함수

```
function square(number) {  
  return number * number;  
}
```

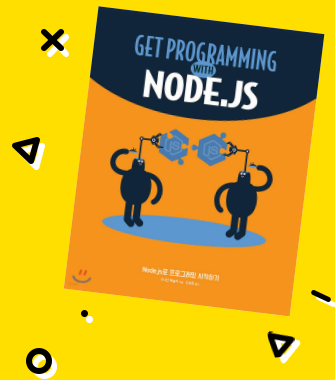
```
function addSquares(a, b) {  
  function square(x) {  
    return x * x;  
  }  
  return square(a) + square(b);  
}  
  
a = addSquares(2, 3); // 13  
b = addSquares(3, 4); // 25  
c = addSquares(4, 5); // 41
```

```
function A(x) {  
  function B(y) {  
    function C(z) {  
      console.log(x + y + z);  
    }  
    C(3);  
  }  
  B(2);  
}  
A(1); // logs 6 (1 + 2 + 3)
```



JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



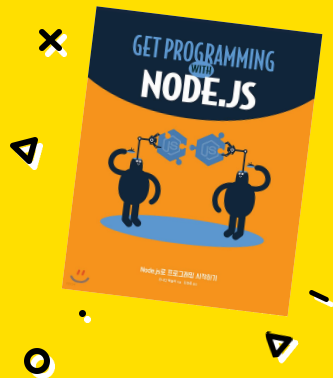
7. this operator / this

```
// call 또는 apply의 첫 번째 인자로 객체가 전달될 수 있으며 this가  
그 객체에 묶임  
var obj = {a: 'Custom'};  
  
// 변수를 선언하고 변수에 프로퍼티로 전역 window를 할당  
var a = 'Global';  
  
function whatsThis() {  
  return this.a; // 함수 호출 방식에 따라 값이 달라짐  
}  
  
whatsThis(); // this는 'Global'. 함수 내에서 설정되지 않았  
으므로 global/window 객체로 초기값을 설정한다.  
whatsThis.call(obj); // this는 'Custom'. 함수 내에서 obj로 설정한  
다.  
whatsThis.apply(obj); // this는 'Custom'. 함수 내에서 obj로 설정한  
다.
```




JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



8. Arrow Functions / [화살표 함수](#)

기본 구문

```
(param1, param2, ..., paramN) => { statements }  
(param1, param2, ..., paramN) => expression  
// 다음과 동일함: => { return expression; }  
  
// 매개변수가 하나뿐인 경우 괄호는 선택사항:  
(singleParam) => { statements }  
singleParam => { statements }  
  
// 매개변수가 없는 함수는 괄호가 필요:  
() => { statements }
```

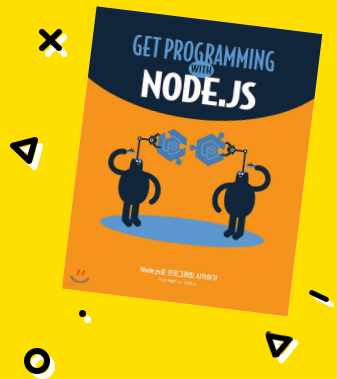
고급 구문

```
// 객체 리터럴 표현을 반환하기 위해서는 함수 본문(body)을 괄호  
// 속에 넣음:  
params => ({foo: bar})  
  
// 나머지 매개변수 및 기본 매개변수를 지원함  
(param1, param2, ...rest) => { statements }  
(param1 = defaultValue1, param2, ..., paramN = defaultValueN) =>  
{ statements }  
  
// 매개변수 목록 내 구조분해할당도 지원됨  
var f = ([a, b] = [1, 2], {x: c} = {x: a + b}) => a + b + c;  
f(); // 6
```



JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



9. Loops / 루프와 반복

- [for 문](#)
- [do..while 문](#)
- [while 문](#)
- [레이블 문](#)
- [break 문](#)
- [continue 문](#)
- [for...in 문](#)
- [for...of 문](#)

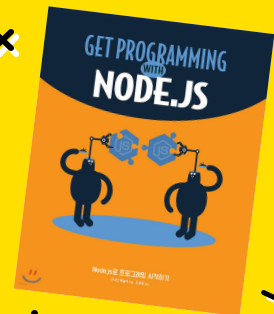
```
var step;
for (step = 0; step < 5; step++) {
  // Runs 5 times, with values of step 0 through 4.
  console.log('Walking east one step');
}
```





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



10. Scopes / 스코프

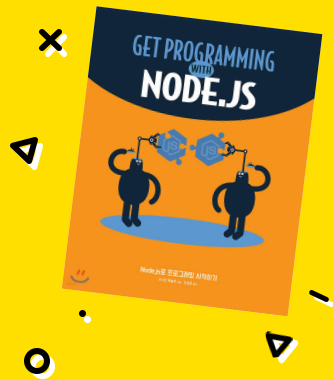
```
function exampleFunction() {  
  var x = "declared inside function";  
  // x는 오직 exampleFunction 내부에서만 사용 가능.  
  console.log("Inside function");  
  console.log(x);  
}  
  
console.log(x); // 에러 발생
```

```
var x = "declared outside function";  
  
exampleFunction();  
  
function exampleFunction() {  
  console.log("Inside function");  
  console.log(x);  
}  
  
console.log("Outside function");  
console.log(x);
```



JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



11. Arrays / [array](#)

배열 만들기

```
let fruits = ['사과', '바나나']  
  
console.log(fruits.length)  
// 2
```

인덱스로 배열의 항목에 접근하기

```
let first = fruits[0]  
// 사과  
  
let last = fruits[fruits.length - 1]  
// 바나나
```

배열 끝에 항목 추가하기

```
let newLength = fruits.push('오렌지')  
// ["사과", "바나나", "오렌지"]
```

배열 끝에서부터 항목 제거하기

```
let last = fruits.pop() // 끝이었던 '오렌지'를 제거  
// ["사과", "바나나"]
```

배열 앞에서부터 항목 제거하기

```
let first = fruits.shift() // 제일 앞의 '사과'를 제거  
// ["바나나"]
```

배열 앞에 항목 추가하기

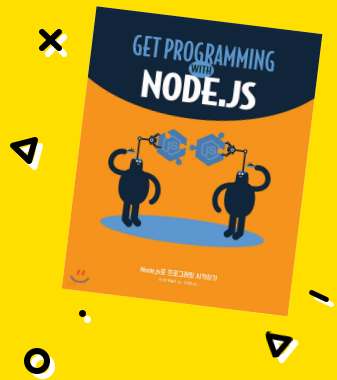
```
let newLength = fruits.unshift('딸기') // 앞에 추가  
// ["딸기", "바나나"]
```





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



12. Template Literals / [템플릿 문자열](#)

```
`string text`
```

```
`string text line 1  
string text line 2`
```

```
`string text ${expression} string text`
```

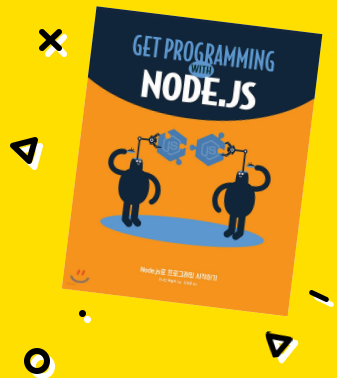
```
tag `string text ${expression} string text`
```





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



13. Strict Mode / 엄격 모드

엄격 모드는 평범한 JavaScript 시맨틱스 몇가지 변경이 일어나게 합니다.

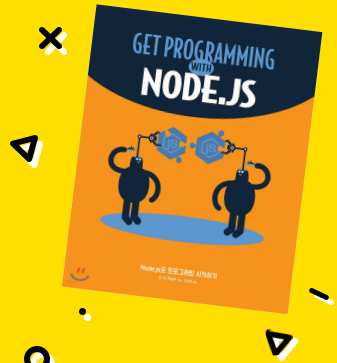
- 기존에는 조용히 무시되던 에러들을 throwing합니다.
- JavaScript 엔진의 최적화 작업을 어렵게 만드는 실수들을 바로잡습니다. 가끔씩 엄격 모드의 코드는 비-엄격 모드의 동일한 코드보다 더 빨리 작동하도록 만들어집니다.
- 엄격 모드는 ECMAScript의 차기 버전들에서 정의 될 문법을 금지합니다.





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



14. ECMAScript 2015 (ES6) and beyond / [JavaScript 기술 개요](#)

ECMAScript는 다음 항목을 포함하고 있습니다.

- 언어 구문 (구문 분석 규칙, 키워드, 흐름 제어, 객체 리터럴 초기화 등)
- 오류 처리 방법 (throw, try...catch, 사용자 정의 Error 유형 등)
- 자료형 (불리언, 숫자, 문자열, 함수, 객체, ...)
- 전역 객체. 브라우저에서 전역 객체는 window 객체지만, ECMAScript는 브라우저에 국한되지 않는 API(parseInt, parseFloat, decodeURI, encodeURI 등)만 정의합니다.
- 프로토타입 기반 상속 구조
- 내장 객체 및 함수 (JSON, Math, Array.prototype 메서드, Object 내성검사 메서드 등)
- 엄격 모드

브라우저 지원

2016년 10월 기준 주요 브라우저의 현버전은 ECMAScript 5.1과 ECMAScript 2015를 구현하지만, (여전히 사용 중인) 오래된 브라우저는 ECMAScript 5만 구현합니다.

미래

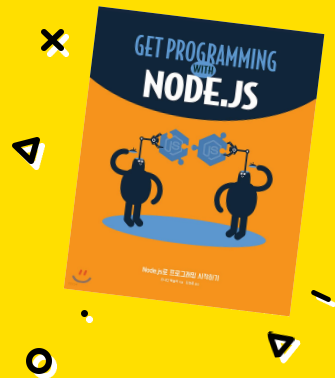
2015년 7월 17일, ECMAScript 6판이 ECMA General Assembly에 의해 공식으로 채택되고 표준으로 출간 되었습니다. 이후 ECMAScript는 매년 새로운 판을 출판하고 있습니다.





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



15. Async programming / 동기 프로그래밍

비동기 프로그래밍은 작업이 완료될 때까지 기다리지 않고 잠재적으로 오래 실행되는 작업을 시작하여 해당 작업이 실행되는 동안에도 다른 이벤트에 응답할 수 있게 하는 기술입니다. 작업이 완료되면 프로그램이 결과를 제공합니다.

브라우저가 제공하는 많은 기능, 특히 가장 흥미로운 것들은 시간이 오래 걸릴 가능성이 있으므로 비동기적입니다. 예를 들어 다음과 같습니다.

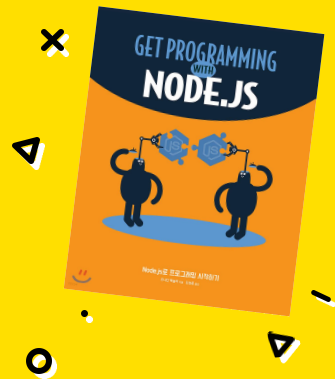
- `fetch()` (en-US)를 이용해 HTTP 요청 만들기
- `getUserMedia()`를 사용해 사용자의 카메라 또는 마이크에 접근하기
- `showOpenFilePicker()` (en-US)를 통해 사용자에게 파일 선택을 요청





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



16. Timers / setTimeout()

```
var timeoutID = setTimeout(function[, delay, arg1, arg2, ...]);  
var timeoutID = setTimeout(function[, delay]);  
var timeoutID = setTimeout(code[, delay]);
```

```
setTimeout(() => {console.log("첫 번째 메시지")}, 5000);  
setTimeout(() => {console.log("두 번째 메시지")}, 3000);  
setTimeout(() => {console.log("세 번째 메시지")}, 1000);
```

// 콘솔 출력:

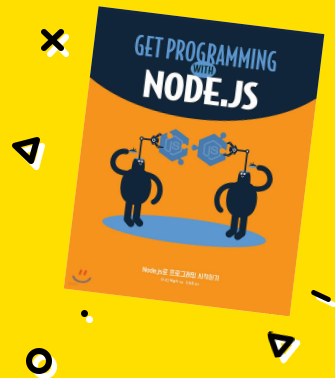
```
// 세 번째 메시지  
// 두 번째 메시지  
// 첫 번째 메시지
```





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



17. Promises / promise

Promise는 비동기 작업의 최종 완료 또는 실패를 나타내는 객체입니다. 대부분 여러분은 이미 만들어진 promise를 사용했었기 때문에 이 가이드에서는 어떻게 promise를 만드는지 설명하기에 앞서 promise의 사용법에 대해 설명합니다.

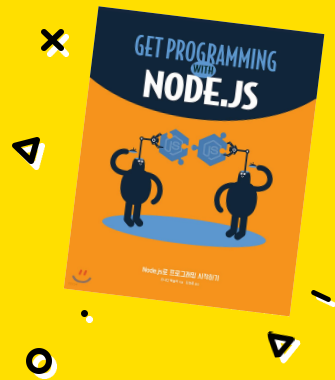
```
doSomething().then(function(result) {  
  return doSomethingElse(result);  
})  
.then(function(newResult) {  
  return doThirdThing(newResult);  
})  
.then(function(finalResult) {  
  console.log('Got the final result: ' + finalResult);  
})  
.catch(failureCallback);
```

```
doSomething()  
.then(result => doSomethingElse(result))  
.then(newResult => doThirdThing(newResult))  
.then(finalResult => {  
  console.log(`Got the final result: ${finalResult}`);  
})  
.catch(failureCallback);
```



JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



18. Async and Await / async function

async function 선언은 AsyncFunction 객체를 반환하는 하나의 비동기 함수를 정의합니다. 비동기 함수는 이벤트 루프를 통해 비동기적으로 작동하는 함수로, 암시적으로 Promise를 사용하여 결과를 반환합니다. 그러나 비동기 함수를 사용하는 코드의 구문과 구조는, 표준 동기 함수를 사용하는 것과 많이 비슷합니다.

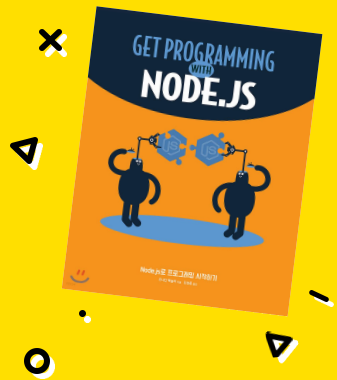
```
async function getProcessedData(url) {  
  let v;  
  try {  
    v = await downloadData(url);  
  } catch (e) {  
    v = await downloadFallbackData(url);  
  }  
  return processDataInWorker(v);  
}
```





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)



19. Closures / 클로저

클로저는 함수와 함수가 선언된 어휘적 환경의 조합이다. 클로저를 이해하려면 자바스크립트가 어떻게 변수의 유효범위를 지정하는지 (Lexical scoping)를 먼저 이해해야 한다.

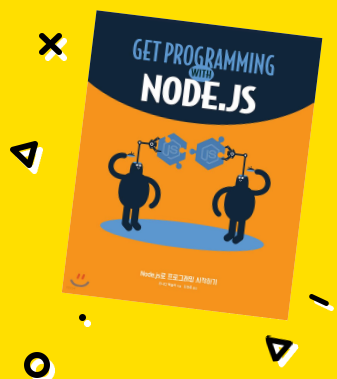
```
function init() {  
  var name = "Mozilla"; // name은 init에 의해 생성된 지역 변수이다.  
  function displayName() { // displayName() 은 내부 함수이며, 클로저다.  
    alert(name); // 부모 함수에서 선언된 변수를 사용한다.  
  }  
  displayName();  
}  
init();
```





JavaScript를 얼마나 알아야 한다?

[How much JavaScript do I need to know to use NodeJS?](#)

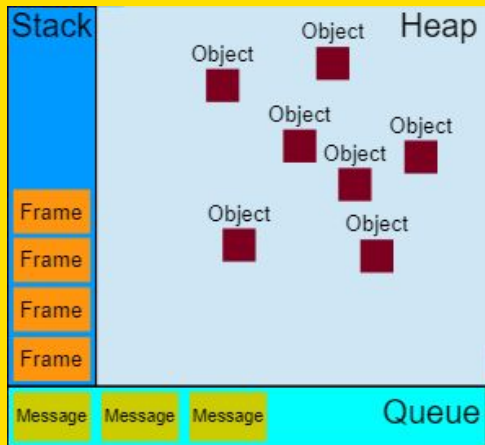


20. The Event Loop / 이벤트 루프

- 스택
 - 함수의 호출들은 '프레임' 스택을 형성합니다.
- 힙
 - 객체는 힙에 할당됩니다. 힙은 단순히 메모리의 큰 (그리고 대부분 구조화되지 않은) 영역을 지칭하는 용어입니다.
- 큐
 - JavaScript 런타임은 메시지 큐, 즉 처리할 메시지의 대기열을 사용합니다. 각각의 메시지에는 메시지를 처리하기 위한 함수가 연결돼있습니다.

이벤트 루프는 이 기능을 구현할 때 보통 사용하는 방식에서 그 이름을 얻었으며, 대략 다음과 같은 형태입니다.

```
while(queue.waitForMessage()){
  queue.processNextMessage();
}
```





Thanks !

설문조사를 보자~