HTML & CSS & Javascript

Junhyeok Lee (wnsx0000@gmail.com)

August 18, 2024

참고 문헌 : 김기수, *코딩 자율학습 HTML+CSS+자바스크립트*, 도서출판 길벗(2022), pp.1-612.

목차

Part I

HTML

HTML(Hypertext Markup Language)은 웹사이트의 모습을 기술하기 위한 마크업 언어임.

1. 서론

1.1. 서론

1.1.1. html과 마크업 언어

html은 조판을 위한 마크업 언어임.

html 파일은 .html 확장자를 가지며, 해당 파일을 웹 브라우저 등으로 열면 조판 결과를 확인할 수 있음.

html 문서는 index.html 파일을 기본으로 만듦. 웹브라우저 주소를 입력할 때 파일을 명시하지 않으면 index.html 파일을 요청하기 때문. 그래서 html 코드를 작성할 때 첫 번째 파일의 이름을 index.html로 하는 것이 관습임.

웹브라우저는 html 문서를 첫 번째 줄에서부터 차례대로 해석함.

1.1.2. 인터넷, 서버와 클라이언트

인터넷이란 TCP/IP 프로토콜을 사용하는 네트워크 또는 네트워크의 집합체. 전 세계의 컴퓨터들을 하나로 연결하는 통신망을 말함.

인터넷은 여러 컴퓨터가 클라이언트(고객)와 서버(사업자)로서 연결된 상태인 것으로, 중앙통제장치가 존재하지 않음. 각각의 컴퓨터가 서로 정보를 주고받을 뿐. 클라이언트는 브라우저를 통해 서버에게 정보를 요청하고, 서버는 정보를 제공함. 그렇기에 최소 2대의 컴퓨터만 있어도 인터넷이 작동한다고 할 수 있음.

1990년경 팀 버너스 리는 최초의 웹브라우저(웹클라이언트)와 웹서버를 만들었음. 웹 서버가 설치된 컴퓨터는 info,cern,ch라는 주소를 가지고, 그 컴퓨터에는 index.html이라는 파일이 저장되어 있다고 하자. 이때 웹브라우저에서 http://info,cern.ch/index.html이라는 주소로 접근하면 해당 파일을 열 수 있는 것.

1.1.3. 네트워크 기초

컴퓨터와 컴퓨터의 통신에서의 규약을 프로토콜(protocol)이라고 함. 인터넷은 TCP/IP 프로토콜을 따름. TCP/IP는 여러 계층(응용, 전송, 네트워크, 링크, 하드웨어)으로 이루어져 있고, 각각의 계층마다 규약이 정해져 있음.

HTTP(Hypertext Transfer Protocol)는 웹클라이언트와 웹서버 간 통신을 위한 프로토콜임. HTTPS(Hypertext Transfer Protocol Secure)는 HTTP에서 보안이 강화된 프로토콜임.

컴퓨터와 컴퓨터가 통신하기 위해서는 각각의 주소가 존재해야 함. 컴퓨터에서 사용되는 주소에는 MAC, IP, 도메인 네임 등이 있음.

1.1.4. 웹서버 게시

컴퓨터에 웹서버를 만드는 것에는 크게 두 가지 방법이 있음. 하나는 웹서버를 제공하는 웹호스팅 등의 서비스를 이용하는 것이고, 다른 하나는 직접 서버 프로그램을 설치하는 것임. 웹호스팅이 훨씬 간단하기에 주로 사용됨. 웹호스팅 회사로는 대표적으로 github가 있음.

github pages 이용하여 웹서버를 게시할 수 있음. github의 서버컴퓨터에 작성한 html 코드가 올라가고, 웹사이트가 만들어지는 것. (https://www.opentutorials.org/course/3084/18891)

직접 서버 프로그램 설치해서 서버 만드는 법. (https://www.opentutorials.org/course/3084/31144)

2. 기본 문법

2.1. 기본 구성 요소

2.1.1. 태그(tag)

Definition 1 *웹페이지의 구성요소를 정의하는 문법.*

<태그명> 으로 작성함.

HTML을 이루는 가장 기본적이고 작은 단위.

태그끼리는 부모, 자식, 형제, 조상, 자손 관계가 성립함. 디렉토리에서의 그것과 동일한 의미.

가독성을 위해 태그 사용 시 줄바꿈과 들여쓰기를 적절히 해야 함.

2.1.2. 속성(attribute)

Definition 2 *태그에 의미나 기능을 추가하는 옵션으로서의 문법. 속성명과 속성값으로 구성됨.*

<태그명 속성명="속성값"> 로 작성함. 속성명이 여러 개인 경우 공백 문자로 구분하고, 속성값이 여러 개인 경우 큰따옴표 안에 공백 문자로 구분하여 작성함.

속성값을 작성하지 않는 속성도 있음.

속성값에 다른 값을 지정하는 형태도 존재함. <태그명 속성명="속성값=값"> 의 형태인 것.

2.1.3. 콘텐츠(content)

Definition 3 *태그 사이의 내용.*

HTML의 문법은 태그와 성분으로 구성됨. 문법은 콘텐츠의 유무에 따라 콘텐츠가 있는 문법, 콘텐츠가 없는 문법으로 구분됨.

콘텐츠가 있는 경우 2개의 태그를 사용함. 앞 태그를 시작 태그(open tag), 뒷 태그를 종료 태그(close tag)라고 함.

콘텐츠가 없는 경우 하나의 태그만 사용함. 이때의 태그를 빈 태그(empty tag)라고 함.

2.1.4. 요소(element)

Definition 4 *시작 태그, 콘텐츠, 종료 태그를 합쳐서 요소(element)라고 함. 콘텐츠가 없는 경우에는 시작 태그가 곧 요소(element)임.*

사용 시마다 줄바꿈이 되어 새로운 줄에서 시작되며, 해당 줄의 모든 공간을 사용하는 태그가 들어간 요소를 블록 요소(block element)라고 함.

사용 시에 새로운 줄에서 시작하지 않고, 콘텐츠만큼만 공간을 사용하는 태그가 들어간 요소를 인라인 요소(inline element)라고 함.

2.1.5. 주석

Definition 5 *<!– 내용 –>으로 작성함.*

2.2. 기본 구조

2.2.1. DTD(Document Type Definition)

Definition 6 *문서형 정의. 문서 맨 앞에서 웹브라우저가 처리할 HTML 문서가 어떤 형식을 따르는지 명시하는 것.*

<!DOCTYPE html> 로 작성하면 HTML5를 따른다는 것임. 콘텐츠가 없는 태그.

2.2.2. <html></html>

Definition 7 *HTML 문서의 시작과 끝을 나타내는 태그.*

DTD를 제외한 모든 태그는 이 태그 안에 작성함.

속성.
lang : 언어 지정. 한글은 ko로 지정함.

2.2.3. <head></head>

Definition 8 *메타데이터(metadata)를 정의하는 영역을 나타내는 태그.*

메타데이터는 HTML 문서에 대한 정보로서, 웹브라우저에는 조판되지 않음.

메타데이터를 정의하는 태그에는 meta, title, link, style, script 등이 있음. 이 태그들은 head 태그 내부에 작성함.

1. meta : 메타데이터를 정의하는 콘텐츠가 없는 태그.
속성.
charset : 인코딩 방식 지정. 주로 UTF-8로 지정함.
http-equiv="X-UA-Compatible" content="IE=edge" : 인터넷 익스플로러(IE)의 렌더링 엔진을 강제로 최신 렌더링 엔진으로 지정.
name="viewport" content="width=device-width initial-scale=1.0" : 기기 화면 넓이를 맞춤.

2. title : HTML 문서 제목을 지정하는 콘텐츠가 있는 태그.

모든 HTML 문서는 제목을 지정해 줘야 하고, 이때 제목명이 중복되면 안됨. (검색 엔진 노출 시 불이익.)

2.2.4. <body></body>

Definition 9 *웹브라우저에 노출되는 영역을 나타내는 태그.*

2.2.5. 예시

HTML 코드의 기본 구조를 정리하면 아래와 같음.

```
<!DOCTYPE html>
<html lang="ko">
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <title>title</title>
    </head>
    <body>
        <!-- content -->
    </body>
</html>
```

3. 필수 태그들

3.1. 텍스트

3.1.1. <h1~h6> </h1~h6>

: 제목 작성. 숫자가 작을수록 크기가 큼. (header)

검색 엔진은 본 태그의 내용을 키워드로 반영하므로 적절한 제목을 작성해야 함. 또한 태그가 순차적으로 검색되기 때문에 h1 h2 h5 등으로 중간 단계를 건너뛰면 h3이 존재하지 않아 그 이후 단계는 반영되지 않음. 숫자를 건너뛰지 말고 단계적으로 사용해야 함.

3.1.2.

: 하나의 문단으로 내용 작성. (paragraph)

물론 p 태그를 사용하지 않아도 문장은 출력되지만, 웬만하면 묶어주는 것이 좋음.

3.1.3.

: 줄바꿈. (line break)

3.1.4. <blockquote> </blockquote>

: 인용문 작성. 해당 문단이 들여쓰기됨. 내부에 p 태그로 문단을 작성해줘야 함.

cite="출처 URL" : 출처 명시 속성.

출처가 확실한 경우 속성으로 명시해주는 것이 좋음.

3.1.5. <q> </q>

: 문단 내에서 인용문 작성. 해당 문장이 따옴표로 묶임. (quote)

cite="출처 URL" : 출처 명시 속성.

3.1.6. <ins> </ins>, <del> </del>

: ins는 추가된 내용 표시(밑줄), del은 삭제된 내용 표시(줄 그어짐). (insert, delete)

3.1.7. ,

: sub는 앞 문자에 아랫첨자 추가, sup는 앞 문자에 윗 첨자 추가. (subscript, superscript)

3.2. 공간 분할

관련 요소들끼리 묶어 그룹을 짓는 것을 공간 분할이라고 함. 이런 기능을 수행하는 태그를 공간 분할 태그라고 함. 공간 분할을 통해 레이아웃을 쉽게 구성하고 구조를 깔끔하게 할 수 있음.

3.2.1.

: 내부의 블럭 요소와 인라인 요소 모두를 묶음. 출력 시 사각형의 영역을 가짐. (division)

3.2.2.

: 내부의 인라인 요소들을 묶음. 출력 시 문장 단위의 영역을 가짐.

3.3. 목록

3.3.1.

: 순서가 없는 목록 생성. 목록 내용은 li 태그로 작성함. 목록 내용마다 글머리 기호가 붙음. (unordered list)

3.3.2.

: 순서가 있는 목록 생성. 목록 내용은 li 태그로 작성함. 목록 내용마다 앞에 번호가 붙음. (ordered list)

3.3.3. <dl> <dt> </dt> <dd> </dd> </dl>

: 정의형 목록 생성. dt 태그(description term)로 용어를, dd 태그(description details)로 용어 설명 작성. (description list)

정의형 목록은 용어와 용어 설명을 나열한 목록임.

3.4. 링크, 이미지

3.4.1. <a>

: 링크 생성. 콘텐츠가 링크의 형태가 됨. href 속성을 반드시 사용해야 함.
href="대상 경로" : 경로 지정.

target="링크 연결 방식" : _blank으로 지정하면 새 창으로 열림.
title="링크 설명" : 마우스를 올릴 경우 뜨는 설명 지정.

지정할 링크가 불분명한 경우, #으로 작성해둘 수 있음.

콘텐츠로 img 태그를 작성하면 이미지 링크로 사용할 수 있음. 이미지를 클릭하면 해당 링크로 이동하는 것.

3.4.2. <img>

: 이미지 삽입. src, alt 속성을 반드시 사용해야 함.

src="대상 경로" : 삽입할 이미지의 경로 지정.
alt="이미지 설명" : 마우스를 올릴 경우 뜨는 설명 지정. (alternate)

img에서의 경로 지정은 항상 웹브라우저에서 실행되는 HTML 파일 위치가 기준이고, 리눅스에서처럼 현재 디렉토리는 ".", 상위 디렉토리는 ".."으로 표현할 수 있음.

3.5. 텍스트 강조

3.5.1.

: 볼드체 적용과 의미 강조.

볼드체가 적용되는 것만이 아니라, 웹브라우저에서 중요한 부분임을 알리는 역할도 함. 본 태그를 여러 번 중첩해 사용하면 더 굵어지지는 않지만 중요하다는 의미가 더해짐.

b 태그도 볼드체를 적용하지만, 의미를 강조하지는 않는다는 점에서 차이가 있음.

3.5.2.

: 이텔릭체 적용과 의미 강조. (emphasis)

이텔릭체가 적용되는 것만이 아니라, 웹브라우저에서 중요한 부분임을 알리는 역할도 함. 본 태그를 여러 번 중첩해 사용하면 더 굵어지지는 않지만 중요하다는 의미가 더해짐.

i 태그도 이텔릭체를 적용하지만, 의미를 강조하지는 않는다는 점에서 차이가 있음.

3.6. 폼

폼(form)은 사용자와 상호작용해서 정보를 입력받고 서버로 전송하기 위한 양식을 의미함.

3.6.1. <form> </form>

: 폼 구성 요소들을 묶고, 폼 양식을 지정함. action, method 태그를 사용해야 함.

action="서버 url" : 입력받은 값들을 전송할 서버의 URL 주소를 지정.
method="송신 방식" : 데이터 조회 등의 읽기 작업인 경우 get, 데이터 전송 등의 작업인 경우 get으로 지정함. 송신 방식에 따라 백엔드에서 다르게 처리됨.

HTML의 폼 구성 태그들은 모두 본 태그 내부에 작성함.

지정할 링크가 불분명한 경우, #으로 작성해둘 수 있음.

3.6.2. <input>

: 사용자로부터 정보를 입력받는 부분 생성. type 속성은 반드시 사용해야 함.

type="종류" : 입력받는 방식 지정. 아래에 따로 정리함.
name="이름" : 서버로 정보 송신 시에 노출될 이름 지정.
value="초깃값" : 초깃값 지정.

type에 지정할 수 있는 내용은 아래와 같음.

text : 한 줄 텍스트를 입력할 수 있는 요소를 생성.
password : 비밀번호를 입력할 수 있는 요소를 생성.
tel : 전화번호 형식을 입력할 수 있는 요소를 생성.
number : 숫자만 입력할 수 있는 요소를 생성.
url : URL 주소 형식을 입력할 수 있는 요소를 생성.
search : 검색용 텍스트를 입력할 수 있는 요소를 생성.
email : 이메일 형식을 입력할 수 있는 요소를 생성.
checkbox : 체크박스 요소를 생성.
radio : 라디오버튼 요소를 생성.
file : 파일 업로드 요소를 생성.
button : 버튼 요소를 생성.
image : 이미지로 버튼 요소를 생성. 따라서 img 태그처럼 src 속성을 사용해야함. 단, alt 속성은 사용하지 않음. hidden : 사용자 눈에 보이지 않는 입력 요소를 생성.
date : 날짜(연, 월, 일)를 선택할 수 있는 입력 요소를 생성.
datetime-local : 사용자 시간대에 맞는 날짜(연, 월, 일, 시, 분)를 선택할 수 있는 입력 요소를 생성.
month : 날짜(연, 월)를 선택할 수 있는 입력 요소를 생성.
week : 날짜(연, 주차)를 선택할 수 있는 입력 요소를 생성.
time : 시간을 선택할 수 있는 입력 요소를 생성.
range : 숫자 범위를 선택할 수 있는 슬라이드 요소를 생성.
color : 색상을 선택할 수 있는 요소를 생성.
submit : 폼 전송 역할을 하는 버튼 요소를 생성.
reset : 폼 요소에 사용자가 입력한 값을 초기화하는 버튼 요소를 생성.

3.6.3. <label> </label>

: form 태그 내에서 상호작용 요소(input, textarea 등)에 이름을 붙임. 암묵적 방법과 명시적 방법이 있음.

for="이름 지정" : 명시적 방법을 위한 이름 지정.

이름 상호작용 요소까지가 한 덩어리가 되어 이름을 클릭해도 상호작용 요소가 클릭됨.

암묵적 방법은 label 태그의 콘텐츠로 상호작용 요소를 포함하는 것.

```
<label>
  아이디
```

```
  <input type="text">
</label>
```

명시적 방법은 label 태그에서 input 태그의 아이디에 대해 이름을 지정하는 것.

```
<label for="userpw">비밀번호</label>
<input type="password" id="userpw">
```

3.6.4. <fieldset> <legend> </legend> </fieldset>

: fieldset 태그로 상호작용 요소(input, textarea 등등)들을 하나의 그룹으로 묶고 박스 형태로 출력함. 해당 그룹의 이름은 legend 태그로 지정하고, 상호작용 요소들과 함께 출력됨.

3.6.5. <textarea> </textarea>

: 여러 줄을 입력받는 부분 생성. 콘텐츠로는 초깃값 생성.

여러 줄을 입력받을 때는 input 태그 대신 textarea 태그를 사용함.

3.6.6. <select> <option> </option> </select>, <optgroup> </optgroup>

: 체크박스 생성. select 태그로 체크박스를 생성하고, option 태그의 콘텐츠로 항목을 작성함. optgroup으로 항목들을 그룹으로 묶으면 체크박스 내에서 그룹별로 분류됨.

optgroup으로 묶을 때는 label 속성으로 optgroup의 이름(그룹명)을 지정해줘야 함.

select 속성
size="숫자" : 콤보박스에서 화면에 노출되는 항목 갯수 지정.

option 속성
value="서버에 전송할 값" : 서버에 전송할 값 지정. 지정하지 않으면 콘텐츠가 전송됨.
selected : 콤보박스의 기본 선택값 지정. 지정하지 않으면 맨 위 항목이 선택됨.

optgroup 속성
label="그룹 이름" : 그룹 이름 지정. 반드시 지정해줘야 함.

```
<select>
  <optgroup label="그룹 이름">
    <option value="서버에 전송할 값">웹 브라우저에 표시할 값</option>
  </optgroup>
</select>
```

3.6.7. <button> </button>

: 버튼 생성. 콘텐츠는 버튼 내용을 의미함.
type="종류" : 버튼의 역할을 지정함. submit은 폼을 서버에 전송하고, reset은 입력받은 내용을 초기화하고, button은 단순 버튼을 의미함.

input 태그에서 type 속성을 지정하여 버튼을 생성할 수 있지만, 본 태그로도 생성할 수 있음. 본 태그를 사용하면 버튼 내용을 원하는대로 편하게 지정할 수 있다는 장점이 있음. (button 태그가 더 나중에 만들어짐.)

3.6.8. 공통 속성들

폼 관련 태그에서 사용할 수 있는 공통 속성들.

disabled : 해당 상호작용 요소 비활성화. 출력은 됨.
readonly : 해당 상호작용 요소를 읽기 전용으로 전환.
maxlength="숫자" : 입력 글자 수 제한.
checked : 체크박스 등에서 해당 요소를 선택된 상태로 표시.
placeholder="내용" : 입력 요소 예시 문장 추가. 입력 전 연한 글씨로 내용이 출력됨.

3.7. 표

표는 행, 열, 셀로 구성됨.

3.7.1. <table> </table>

: 표 생성. 표 관련 태그들은 모두 본 태그 내부에 작성함.

3.7.2. <caption> </caption>

: 표 제목 지정.

table 태그에서 주로 맨 위에 작성함.

3.7.3. <tr> </tr>

: 하나의 행 생성. (table row)

3.7.4. <th> </th>, <td> </td>

: 열 생성. tr 태그 안에 작성함. th는 표에서 제목에 해당되는 열을 생성할 때, td는 데이터를 나타내는 열을 생성할 때 사용. (table header, table data)

rowspan="병합할 셀 개수" : 행끼리 병합.
colspan="병합할 셀 개수" : 행끼리 병합.

해당 위치에서부터 병합할 셀 개수를 지정하여 하나의 셀로 만듦. 이때 병합당하는 셀은 비어있어야 함.

아래는 예시.

```
<table>
  <tr>
    <th>번호</th>
    <th>상품명</th>
    <th>수량</th>
    <th>가격</th>
  </tr>
  <tr>
    <td>1</td>
    <td>콜라</td>
    <td>1개</td>
    <td>1,500원</td>
  </tr>
  <tr>
    <td>2</td>
    <td>사이다</td>
    <td>2개</td>
    <td rowspan="2">1,000원</td> <!-- 행 병합 -->
  </tr>
  <tr>
    <td>3</td>
    <td>탄산수</td>
    <td>3개</td>
    <!-- 4행 4열은 3행 4열과 병합했으므로 생성하지 않습니다. -->
  </tr>
  <tr>
    <td>총 금액</td>
    <td colspan="3">6,500원</td> <!-- 열 병합 -->
    <!-- 5행 2열부터 열 3개를 병합했으므로 나머지 열은 생성하지 않습니다. -->
```

```
  </tr>
</table>
```

3.7.5. thead, tfoot, tbody 태그

: 행을 묶어 그룹화함.

3.7.6. <col> </col>, <colgroup> </colgroup>

: 열을 묶어 그룹화함. 스타일을 지정할 때 주로 사용. caption 태그보다 뒤에, tr 태그보다 전에 사용해야 함. style은 하나의 열을 그룹화하고, colgroup은 span 속성을 사용해 여러 개의 열을 그룹화함.

span="숫자" : colgroup에서 몇 개의 행을 그룹화할지 지정.
style="width:숫자" : 해당 열의 가로 길이를 지정함. 단위는 px 등.

3.8. 멀티미디어

오디오, 비디오 등 집어넣기.

3.8.1. <audio> </audio>

: 오디오 파일 출력. 반드시 src 속성을 사용해야 함.

src="오디오 파일 경로" : 출력할 오디오 파일 지정.
controls : 오디오 컨트롤 패널 화면에 출력.

웹브라우저마다 지원하는 오디오 파일 포맷이 다르므로 적절한 포맷 선택 필요. MP3는 대체로 지원함. (https://thebook.io/080313/0112/ 참고.)

3.8.2. <video> </video>

: 비디오 파일 출력. 반드시 src 속성을 사용해야 함.

src="비디오 파일 경로" : 출력할 비디오 파일 지정.
controls : 비디오 컨트롤 패널 화면에 출력.

웹브라우저마다 지원하는 오디오 파일 포맷이 다르므로 적절한 포맷 선택 필요. MP4는 대체로 지원함. (https://thebook.io/080313/0114/ 참고.)

3.8.3. <source> </source>

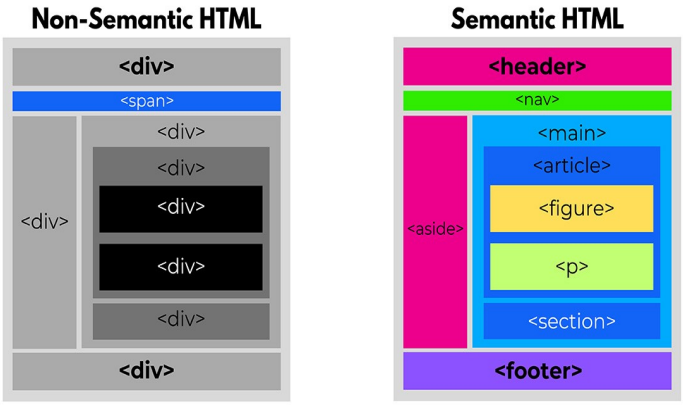
: audio/video 태그에서 파일의 경로와 미디어 타입 표시. 웹브라우저별 포맷과 미디어 타입 설정을 위해 주로 사용.

```
<audio controls>
  <source src="sample.wav" type="audio/wav">
  <source src="sample.mp3" type="audio/mp3">
  지원하지 않는 웹 브라우저입니다.
</audio>
```

위와 같이 코드를 작성하면 맨 윗 source 태그에서부터 내려오면서 현재 웹브라우저에서 해당 포맷과 미디어 타입을 지원하는지 확인함. 지원한다면 확인이 끝나고, 전부 지원하지 않는다면 맨 아래의 텍스트가 출력됨.

3.9. 시맨틱 태그

시맨틱 웹(sementic web)은 웹을 더 의미론적으로 명확하게 한 것을 의미함. 이를 위한 태그들이 시맨틱 태그임. table, form, a 등은 의미론적으로 명확하기에 시맨틱 태그라고 할 수 있음. 여기서는 웹페이지의 구조를 명확히 표기하는 시맨틱 태그들을 정리함.



3.9.1. <header> </header>

: 헤더 영역. 주로 로고, 검색, 메뉴 등을 포함함.

3.9.2. <nav> </nav>

: 링크 영역. (navigation)

링크가 모두 링크 영역 안에 있어야 하는 것은 아님. 주요 탐색 링크에만 주로 사용.

3.9.3. <section> </section>

: 논리적 내용 구분 영역. 논리적으로 관련 있는 내용 영역을 구분함.

3.9.4. <article> </article>

: 독립적 부분 영역. 로그인 영역 등 어떤 웹페이지에서든 독립적으로 사용될 수 있을 만한 부분에 사용.

3.9.5. <aside> </aside>

: 주 내용이나 독립적 내용으로 보기 어려워서 article, section으로 구분하기 어려울 때 사용. 광고 영역 등.

3.9.6. <footer> </footer>

: 푸터 영역. 주로 페이지의 최하단에서 저작권 정보, 연락처, 사이트맵 등을 모아놓은 부분.

3.9.7. <main> </main>

: 메인 영역. 주요 내용 지정.

article, aside, footer, header, nav의 내부에 들어갈 수 없음. 자세한 사용법은 나중에 정리함.

3.10. 글로벌 속성

태그에 상관없이 전반적으로 사용하는 속성인 글로벌 속성 정리.

class="값" : 요소에 클래스 값 지정.
여러 요소가 같은 클래스명을 가질 수 있음. 클래스 값은 CSS에서 클래스 선택자로 활용함.

id="값" : 요소에 아이디 값 지정.
아이디 값은 CSS에서 아이디 선택자로 활용함.

style="스타일" : 요소에 인라인 스타일을 지정.
css 코드를 인라인으로 작성할 때 사용.

title="텍스트" : 요소에 추가 정보를 지정.
마우스를 요소 위에 올리면 툴팁(tooltip)으로 해당 텍스트가 표시됨.

lang="언어 코드" : 요소에 사용한 텍스트의 언어 정보를 지정함.
html 태그에서 문서 전체의 언어를 지정하고, 문서 특정 부분에서 다른 언어가 나오면 해당 태그에서는 그 언어를 지정하는 것이 좋음.

hidden="hidden" : 요소를 화면에서 감춤.

data-이름="값" : 사용자가 임의의 속성을 만들 수 있음.

Part II

CSS

초창기 HTML에서는 sytle이라는 속성으로 스타일을 전부 지정했었는데, 웹이 발전하며 이 방식의 한계가 드러났음. 이에 W3C는 웹 문서에서 구조와 디자인을 분리하고 디자인을 위한 언어인 CSS(Cascading Style Sheets)라는 언어가 탄생하게 됨.

HTML이 태그로 웹 문서의 구조를 설계했다면, CSS는 웹 문서의 디테일과 디자인을 적용함.

1. 기본 문법

1.1. 기본 문법

1.1.1. 기본 문법 형식

Definition 10 *CSS 문법은 선택자와 선언부로 구분됨. 선택자는 스타일을 적용할 HTML 태그(요소)를 지정하는 부분이고, 선언부는 해당 태그(요소)에 적용될 스타일을 지정하는 부분임.*

선택자{속성:값;} 으로 작성함. 여기서 속성과 값 부분이 선언부임.

세미콜론으로 구분해서 여러 스타일을 계속 작성할 수 있음.

추가로, CSS에는 at rule이라는 것이 존재하여 부가적인 규칙을 지정할 수 있음. 해당 문법은 @로 시작하며 특정 속성들에 한해 사용할 수 있음.

CSS 코드는 주로 아래와 같은 꼴로 작성함.

```
h1{
    font-size:24px;
    color:red;
}
```

1.1.2. 주석

Definition 11 *CSS에서의 주석은 /* 내용 */ 로 작성함.*

1.1.3. CSS 적용 방법

HTML 문서에 CSS를 적용하는 대표적인 방법은 3가지가 있음.
주로 외부 스타일 시트 방식을 사용하고, 인라인 스타일 방식은 거의 사용되지 않음.

1. 내부 스타일 시트(internal style sheet) 사용
HTML의 style 태그의 콘텐츠로 CSS 코드를 작성하는 것.
위치는 상관 없지만 주로 head 태그 안에 작성.

아래와 같이 작성함.

```
<head>
<title>내부 스타일 시트(Internal Style Sheet)</title>
    <style>
        h1{
            color:blue;
        }
    </style>
</head>
<body>
    <h1>내부 스타일 시트</h1>
</body>
```

2. 외부 스타일 시트(external style sheet) 사용
CSS 코드를 위한 별도 파일을 생성하는 것.
이때 파일 확장자는 .css여야 함.
CSS 파일을 HTML 파일에 연결할 때는 HTML의 link 태그를 사용함. (head 태그 안에.)

아래와 같이 작성함.

```
<link rel="stylesheet" href="파일 경로">
```

3, 인라인 스타일(inline style) 사용
HTML 태그에서 style 속성에 CSS 코드를 작성하는 것.
태그를 이미 선택한 것이므로 선택자 부분이 필요 없음.

아래와 같이 작성함.

```
<body>
    <h1 style="color:red; font-size:24px">인라인 스타일</h1>
</body>
```

2. 선택자

선택자를 사용하는 방법을 정리함.

2.1. 기본 선택자

기본적 선택자들.

2.1.1. 전체 선택자

Definition 12 *전체 선택자는 HTML의 모든 가능한 요소를 한 번에 선택자로 지정하는 방법임.*
**{* CSS 코드 *\}* 로 작성함.*

2.1.2. 태그 선택자

Definition 13 *태그 선택자는 해당 태그명과 일치하는 요소 모두를 선택자로 지정하는 방법임.*
태그명{ CSS 코드 *\}* 로 작성함.*

2.1.3. 아이디 선택자

Definition 14 *아이디 선택자는 아이디 속성값을 이용하여 일치하는 요소를 선택자로 지정하는 방법임.*

♯id속성값{ CSS 코드 *\} 로 작성함.*

아이디는 태그에 부여된 고유한 값임. 한 페이지에서 단 한 번의 정의로 고유한 하나의 태그에서만 사용할 수 있음. c에서의 식별자 같은 느낌.

아래는 예시.

```
<style>
    #title{
        color:green;
    }
</style>
...
<h1 id="title">아이디 선택자</h1>
```

2.1.4. 클래스 선택자

Definition 15 *클래스 선택자는 class 속성값을 이용하여 일치하는 요소를 선택자로 지정하는 방법임.*

.class속성값{ CSS 코드 *\} 로 작성함.*

아이디 선택자와는 달리, class 속성값은 중복이 가능함.
실무에서 가장 자주 쓰이는 방법.

아래는 예시.

```
<style>
  .red{
    color:red;
  }
  .blue{
    color:blue;
  }
</style>
...
<h1 class="red">클래스 선택자</h1>
<p class="blue">class 속성값으로 선택자를 지정합니다.</p>
<p class="blue">class 속성은 id 속성과 다르게 속성값을 중복해서 사용할 수 있습니다.</p>
```

2.1.5. 속성 선택자

Definition 16 *속성 선택자는 HTML의 속성과 값을 이용하여 일치하는 요소를 선택자로 지정하는 방법임.*

[속성]{ CSS 코드 *\} 로 작성함.*
[속성="값"]{ CSS 코드 *\} 로 작성함.*

아이디/클래스 선택자 또한 속성과 속성값을 기준으로 요소를 선택하므로 속성 선택자와 유사하다고 볼 수도 있음음.

속성 선택자는 전체, 태그, 아이디, 클래스 선택자와 함께 사용할 수 있음. [] 앞에 뭐가 추가로 있으면 제한된 범위에서 선택되는 것이고, 아무것도 없다면 해당 속성/값 전부가 선택되는 것. 아래는 그 예시임.

```
<style>
  a[target="_blank"]{
```

```
    color:red;
  }
</style>
...
<a href="#">기본 a 태그</a>
<a href="#" target="_blank">새 창으로 열리는 a 태그</a>
```

추가로, 태그 선택자와 아이디/클래스 선택자 또한 함께 사용할 수 있음.

2.1.6. 문자열 선택자

Definition 17 *문자열 선택자는 태그가 가진 속성값을 이용하여 선택자로 지정하는 방법임.*

속성값을 문자열 선택자에서 지정한 문자열과 비교하는 것인데 잘 쓰이지 않아 특별히 정리하지 않음.

2.2. 조합 선택자

기본 선택자와 함께 사용하여 선택자의 의미를 더 풍부하게 하는 방법들.

각 방법들은 서로 혼용하여 사용할 수 있음.

2.2.1. 그룹 선택자

Definition 18 *그룹 선택자는 여러 선택자들을 하나로 묶을 때 사용하는 방법임.*

선택자1, 선택자2, ... , 선택자n{ CSS 코드 *\} 로 작성함.*

2.2.2. 자식 선택자

Definition 19 *자식 선택자는 부모 요소와 자식 요소를 지정하여 해당 부모 요소의 자식 요소만 선택자로 지정하는 방법임.*

부모선택자 > 자식선택자{ CSS 코드 *\} 로 작성함.*

아래와 같이 작성하면 class 속성값이 box인 요소와 자식 관계인 p 태그로 작성된 요소만 선택자로 지정됨.

```
.box > p{
    color:red;
}
```

2.2.3. 하위 선택자

Definition 20 *하위 선택자는 부모 요소와 하위 요소를 지정하여 해당 부모 요소의 하위 요소만 선택자로 지정하는 방법임.*

선택자1 선택자2 선택자3 ...{ CSS 코드 *\} 로 작성함.*

선택자1의 하위 요소들 중 선택자2에 해당하는 하위 요소들 중 선택자3에 해당하는 하위 요소 ... 가 선택되는 것.

2.2.4. 인접 형제 선택자

Definition 21 *인접 형제 선택자는 지정한 요소 바로 다음에 있는 형제 요소를 선택자로 지정하는 방법임.*

이전선택자 + 대상선택자{ CSS 코드 *\} 로 작성함.*

이전 선택자 바로 다음에 있는 형제 요소 중 대상선택자에 해당되는 것이 선택되는 것.

2.2.5. 일반 형제 선택자

Definition 22 *일반 형제 선택자는 지정한 요소 뒤에 오는 형제 요소 모두를 선택자로 지정하는 방법임.*

이전선택자 ~ 대상선택자{ CSS 코드 **\} 로 작성함.*

2.3. 가상 요소 선택자

가상 요소 선택자는 여러 개가 존재하지만, 주로 사용되는 ::before, ::after 선택자만 정리함.

가상 요소 선택자를 사용할 때는 ::를 붙임.

2.3.1. ::before, ::after

Definition 23 *::before 선택자는 기준 선택자 요소 바로 앞의 공간을 선택하고, ::after 선택자는 기준 선택자 요소 바로 앞의 공간을 선택함.*

기준선택자::가상요소선택자{ CSS 코드 **\} 로 작성함.*

예를 들어 아래와 같이 코드를 작성하면 <before>abc<after>가 출력됨.

```
<style>
  p::before{
    content:'<before>';
  }
  p::after{
    content:'<after>';
  }
</style>
...
<p>abc</p>
```

2.4. 가상 클래스 선택자

2.4.1. 가상 클래스 선택자

Definition 24 *요소의 상태를 이용해 선택자를 지정하는 방법. 즉, 특정 이벤트에 따른 스타일을 지정하는 것.*

요소:가상클래스선택자{ CSS 코드 **\}*

2.4.2. 주로 쓰이는 것들

다양한 가상 클래스 선택자가 존재하지만, 자주 사용되는 것들만 정리함.

:link : a 태그에 대해, 한 번도 방문하지 않은 링크일 때 선택.
:visited : a태그에 대해, 한 번이라도 방문한 링크일 때 선택.

:hover : 요소에 마우스를 올리면 선택.
:active : 요소를 마우스로 클릭하고 있는 동안 선택.

:focus : 입력 요소(input, textarea)에 커서가 활성화되면 선택.
:checked : 체크박스가 표시되어 있으면 선택.
:disabled : 상호작용 요소가 비활성화(disable)되면 선택.
:enabled : 상호작용 요소가 활성화(disable 사용x)되면 선택.

:first-child : 해당 요소의 첫 번째 자식 요소를 선택.
:last-child : 해당 요소의 마지막 자식 요소를 선택.

:nth-child(숫자) : 해당 요소의 n(숫자) 번째 자식 요소를 선택.
:nth-last-child(숫자) : 해당 요소의 뒤에서 n(숫자) 번째 자식 요소를 선택.
:nth-of-type(숫자) : 해당 요소의 형제 요소들 중에서 n(숫자) 번째로 등장하는 해당 요소를 선택.
:nth-last-of-type(숫자) : 해당 요소의 형제 요소들 중에서 뒤에서 n(숫자) 번째로 등장하는 해당 요소를 선택.
:first-of-type : 해당 요소의 형제 요소들 중에서 첫 번째로 등장하는 해당 요소를 선택.
:last-of-type : 해당 요소의 형제 요소들 중에서 마지막으로 등장하는 해당 요소를 선택.

3. 필수 속성 다루기

속성 전부에 대한 구체적 정리는 시간상 하지 않음. 필요한 내용은 그때그때 검색해서 알아내자. (https://thebook.io/)

3.1. CSS의 특징

3.1.1. 기본 스타일 시트

HTML의 각 태그에는 스타일에 관한 역할이나 기능이 존재하지 않음. 스타일을 지정해주기 전에도 기본 스타일이 존재하는 것은, 웹브라우저 자체에 기본 스타일 시트가 존재하기 때문임. 이 기본 스타일 시트는 웹브라우저마다 상이할 수 있음.

3.1.2. 적용 우선순위와 개별성

CSS(Cascading Style Sheets)의 Cascading(폭포처럼 흐르다.)은 CSS가 계층성을 가지고 단계적 적용을 기본으로 하기 때문에 붙은 이름임. CSS는 스타일을 중복 지정하더라도 우선순위에 따라 단계를 나누어 적용함.

하나의 요소에 다양한 CSS 스타일이 지정되는 경우, 스타일 적용 우선순위는 CSS의 개별성(specificity) 규칙에 의해 결정됨. 각 선택자하다 점수가 존재하여 점수가 가장 높은 스타일을 반영함. 점수 체계는 아래와 같고, 해당 선택자에 사용된 문법 수만큼 점수를 더해 합을 구함.

점수가 같은 경우, 뒤쪽에 작성한 코드가 우선순위가 높음.

전체 : 0점
태그, 가상 요소 : 1점
클래스, 속성, 가상 클래스 : 10점
아이디 : 100점
인라인 : 1000점

어떤 선택자를 사용하든 기본 스타일 시트보다는 점수가 높음.

실제로 매번 점수를 계산하는 것은 매우 번거로우므로, 선택자를 구체적으로 지정할수록 점수가 높다고 이해하면 됨.

점수를 계산해 주는 사이트도 존재함.

3.1.3. 상속

CSS 속성은 상속됨. 즉, CSS로 스타일을 지정하면 해당 요소의 하위 요소들에게도 스타일이 적용됨.

다만 상속되는 속성도 있고 그렇지 않은 속성도 존재함. 상속이라는 개념이 존재한다는 것만 기억하자.

3.1.4. 단위

1. 절대 단위 : 어떤 환경에서도 동일한 크기를 가지는 단위.
px : 모니터 화면을 구성하는 사각형 1개의 크기. (pixel)
CSS에서 사용할 수 있는 가장 기본적인 단위이며, 유일한 절대 단위임.

2. 상대 단위 : 환경(부모 요소, 웹브라우저 등)에 따른 크기를 가지는 단위.
% : 해당 속성의 상위 요소 속성값에 대한 상대적인 크기를 가짐.
(ex. 80%이면 80퍼센트 크기인 것.)

em : 부모 요소의 텍스트 크기에 상대적인 크기를 가짐. (emphemeral)
텍스트만을 기준으로 삼음.
(ex. 2em이면 텍스트 크기가 2배인 것.)
em은 유용할 수 있지만, CSS 파일 내에서는 해당 폰트 사이즈를 알기 어렵고 em 단위가 여러 번 중첩되면 계산 결과를 직관적으로 예측하기 어렵다는 단점이 있음.

rem : 루트 요소(html 태그로 이루어진 요소)의 텍스트 크기에 상대적인 크기를 가짐. (root em)
루트 요소의 텍스트 크기는 웹브라우저의 기본 텍스트 크기로 결정되며(주로 16px), html 태그에서 지정할 수도 있음.
(ex. 3rem이면 루트 요소의 텍스트 크기의 3배인 것.)
사용하기 불편할 수 있어도 웹브라우저마다의 호환성이 좋고, 전체 크기를 조정하기 편리함.

vw : 뷰포트 너비를 기준으로 상대적인 크기를 가짐.
뷰포트는 웹브라우저 창의 넓이를 말함.
(ex. 1vw이면 뷰포트 너비의 1/100인 것.)

vh : 뷰포트 높이를 기준으로 상대적인 크기를 가짐.
(ex. 1vh이면 뷰포트 높이의 1/100인 것.)

3.1.5. 색상 표기법

구체적인 색 값은 인터넷에서 금방 찾을 수 있음.

1. 키워드 표기법 : 색상을 영문명으로 표기
실무에서는 잘 사용하지 않음.

2. RGB 표기법 : 색상을 rgb, 또는 투명도를 나타내는 alpha값까지 포함하는 rgba로 표기.
rgb(숫자, 숫자, 숫자) 로 작성함.
rgba(숫자, 숫자, 숫자, 숫자) 로 작성함.
rgb 숫자는 0 255로 표기하고, alpha값은 0과 1 사이의 유리수를 소수로 표기함. (이때 소수의 0은 생략 가능)

3. HEX 표기법 : rgb에 해당하는 값을 16진수로 변환해 표기.
♯RRGGBB 로 작성함. R, G, B에는 해당되는 16진수 숫자가 들어감.

3.2. 텍스트

3.2.1. 텍스트 속성

font-family : 글꼴을 지정. 한글 또는 공백 문자가 들어간 경우 큰따옴표로 감싸야 함. 반드시 마지막 속성값으로 글꼴 유형을 작성해야 함. 해당 글꼴이 웹브라우저에서 제대로 출력되지 않아도 해당 글꼴 유형에 맞게 자동 출력됨.

font-size : 텍스트 크기를 지정.
font-weight : 텍스트 굵기를 지정.
font-style : 글꼴 스타일을 지정. (이탤릭, 기울임 등)
font-variant : 영문 소문자를 크기가 작은 대문자로 변경.
color : 텍스트 색상을 지정. inherit으로 지정하면 부모 요소로부터 상속받음.
text-align : 텍스트 정렬을 지정. (왼쪽, 중앙, 오른쪽, 양쪽 정렬 등)
text-decoration : 텍스트 꾸밈을 지정. (텍스트 위, 아래, 중간에 선 긋는 것 등)
letter-spacing : 자간을 지정. (글자 사이 좌우 간격을 지정하는 것)
line-height : 행간을 지정. (윗줄 아랫줄 사이 상하 간격을 지정하는 것)

3.2.2. 웹/아이콘 폰트 사용

인증된 기관이나 회사에서 웹서버에 게시해 놓은 폰트를 웹 폰트라고 함. 대표적인 것은 구글 폰트(Google Fonts). 구글 폰트 웹사이트에 들어가서 원하는 폰트를 고르고 해당 폰트를 사용할 수 있음.

구글 폰트를 사용하는 방법으로는 link 태그와 @import 문법이 있음. 해당 코드는 구글 폰트에도 자동완성되기 때문에 복붙해 사용하면 됨. @import를 사용한다면 코드를 css파일 최상단에 붙여넣고 font-family 속성으로 사용하면 됨.

웹 폰트처럼 아이콘 폰트도 웹에서 불러와 사용할 수 있음. 아이콘 폰트는 다양한 아이콘을 이미지 파일을 사용하지 않고도 간편하게 사용할 수 있게 함.

대표적인 아이콘 폰트로는 Font Awesome과 Material Icon이 있는데, 여기서는 전자의 것으로 정리함. Font Awesome을 적용하는 방법은 Font Awesome 웹사이트에서 아이콘 폰트 라이브러리를 다운받는 방법과, CDN(Content Delivery Network)을 사용해 서버에 올려진 파일을 참조하는 방법이 있음. CDN을 이용하는 방법은 다음 링크 참고. (https://thebook.io/080313/0333/) CDN을 연결하고 원하는 아이콘에 대한 코드를 찾아 붙여넣는 것.

3.3. 배경

3.3.1. 배경 속성

요소의 배경을 지정할 수 있음.

배경 속성은 박스 모델 중 padding과 content에 적용됨.

background-color : 배경색을 지정.
background-image : 배경에 이미지를 삽입.
background-repeat : 배경 이미지의 반복 여부를 지정.
(공간이 남으면 이미지를 반복 출력함.)
background-size : 배경 이미지의 크기를 지정.
(크기를 직접 지정하거나 공간이 남을 경우의 처리 방법을 지정함.)
background-position : 배경 이미지의 위치를 지정. 배경 내에서의 위치를 지정하는 것.
background-attachment : 배경 이미지를 스크롤할 때의 모습을 결정.
background : 모든 배경 속성을 한 번에 사용할 수 있는 단축 속성.
(보기 헷갈릴 수 있으므로 따로 작성하는 것이 좋음.)

3.4. 박스 모델

3.4.1. 박스 모델

박스 모델(box model)은 HTML의 모든 요소가 박스로 둘러 쌓여 있다는 개념임. 이 박스에 대한 값을 지정하여 웹브라우저에 표시되는 방식을 결정하는 것.

박스 모델은 아래와 같이 4가지 영역으로 구성됨.
margin : 요소의 외부 여백.
border : 요소의 테두리(경계선).
padding : 요소의 내부 여백.
content : 요소의 내용. 요소의 콘텐츠 그 자체가 들어가는 부분임.

각 영역은 사격형 형태이므로 위, 아래, 왼쪽, 오른쪽에 속성에 대해 값을 각각 지정할 수 있음. 속성에 대한 것들은 뒤에 따로 정리함.

3.4.2. margin 겹침 현상

margin 영역을 다룰 때는 margin 겹침 현상(margin collapse)을 잘 신경써야 함. 두 요소의 margin이 맞닿아 있으면 둘 중 큰 값을 가진 margin 영역값이 두 요소의 경계선 사이의 거리가 됨. (ex. 20, 30인 경우 거리가 30이 됨.)

3.4.3. 박스 모델 관련 속성들

대부분 상하좌우 길이를 지정하는 속성들임.

1. margin 관련 속성들.
margin-top/bottom/right/left:<크기>;
margin:<margin-top> <margin-right> <margin-bottom> <margin-left>;
margin:<margin-top> <margin-right&left> <margin-bottom>;
margin:<margin-top&bottom> <margin-right&left>;
margin:<margin-top&right&bottom&left>;

2. border 관련 속성들.
border-top/right/bottom/left:<border-width> <border-style> <color>;
border:<border-width> <border-style> <color>;

<border-width> : 테두리 굵기 지정.
<border-style> : 테두리 모양 지정. 여러 가지 키워드가 존재함.
<color> : 테두리 색 지정.

3. padding 관련 속성들. (margin과 사용법 동일함.)
margin-top/bottom/right/left:<크기>;
margin:<margin-top> <margin-right> <margin-bottom> <margin-left>;
margin:<margin-top> <margin-right&left> <margin-bottom>;
margin:<margin-top&bottom> <margin-right&left>;
margin:<margin-top&right&bottom&left>;

4. content 관련 속성들.
width:<크기>;
height:<크기>;

box-sizing:<속성값>; : content 영역 크기를 더 쉽게 지정하도록 하는 속성.
웹브라우저에 실제로 표시되는 크기는 border+padding+content인데 매번 세 값을 계산하는 것은 불편하므로 사용하는 것.
속성값으로는 content-box, border-box가 들어갈 수 있음.
content-box : width, height의 적용 범위를 content 영역으로 제한. 이게 기본값임.
border-box : width, height의 적용 범위를 border 영역으로 제한. 지정한 값이 border, padding, content를 합한 길이가 됨.

3.4.4. 박스 모델의 성격

각 요소들의 박스 모델은 세 가지 성격(블록, 인라인, 인라인 블록)으로 분류할 수 있음.

1. 블록 성격 : 요소의 너비가 콘텐츠 유무에 상관없이 가로 한 줄을 전부 차지하는 것들.
(ex. h1 h6, p, div 등)
width, height, margin, padding 속성이 전부 적용 가능함.

2. 인라인 성격 : 요소의 너비가 콘텐츠 크기만큼만 차지하는 것들.
(ex. a, span, strong 등)
padding, margin 속성은 왼쪽/오른쪽만 적용되고, width, height 속성은 적용되지 않음.

3. 인라인 블록 성격 : 요소의 너비가 콘텐츠 크기만큼만 차지하지만, 그 외의 성격은 블록 성격을 가지는 것들.
(img 등)
width, height, margin, padding 속성이 전부 적용 가능함.

3.4.5. 박스 모델 성격의 변경

요소의 박스 모델 성격은 display 속성으로 변경할 수 있음.
속성값으론 block, inline, inline-block을 지정할 수 있음.

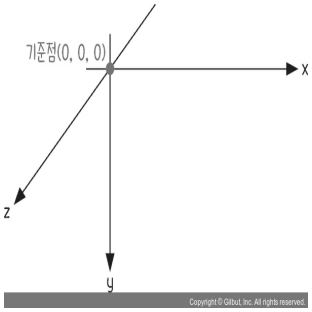
3.5. 위치 지정

요소를 기본 흐름에서 벗어나 원하는 곳에 배치하는 속성들을 정리함.

3.5.1. 좌표계

좌표계는 위치 지정 등에서 필수적임. 웹브라우저에서의 좌표계는 기준점을 기준으로 오른쪽으로 x축(왼쪽), 아래로 y축(위쪽), z축이 존재함.

CSS에서 좌표는 4가지 속성으로 지정함.
(ex. left:100px;)
top : 위 방향 좌표값 지정. (사용자 기준 아래쪽)
right : 오른쪽 방향 좌표값 지정. (사용자 기준 오른쪽)
bottom : 아래 방향 좌표값 지정. (사용자 기준 위쪽)
left : 왼쪽 방향 좌표값 지정. (사용자 기준 왼쪽)



3.5.2. 좌표로 위치 지정

position : 요소를 특정 좌푯값에 배치하는 속성. 속성값으로는 static, relative, absolute, fixed, sticky 등이 있음.

1. static : 기본 흐름에 따라 배치(변화x).

2. relative : 좌표 속성으로 지정한 값에 따라 요소의 왼쪽 위(상대적)를 기준으로 위치 지정.
요소의 왼쪽 위 점이 움직인다고 생각하면 편함.
조판된 결과에서 해당 요소의 위치만 바뀌는 것으로, 원래 요소가 있던 위치에 다른 요소가 채워지지 않음.
즉, 원래 위치를 보장함.

3. absolute : 좌표 속성으로 지정한 값에 따라 특정 지점의 왼쪽 위(절대적)를 기준으로 위치 지정. 상위 요소 등이 특정 지점이 됨.
요소의 왼쪽 위 점이 브라우저의 왼쪽 위를 기준으로 움직인다고 생각하면 편함.
원래 요소가 있던 위치에 다른 요소가 채워짐. 즉, 원래 위치를 보장하지 않음.
top, bottom을 지정하지 않으면 원래 위치에서 좌우로만 움직임. 원래 위치가 top/bottom으로 지정되어 있다고 생각하면 편함.

3. fixed : 요소를 처음 위치에 고정함. 스크롤해도 처음 출력된 위치에 있는 것.

4. sticky : 좌표 속성으로 지정한 값이 임계점이 되어, 스크롤하면 해당 값을 넘어가지 않고 위치가 고정됨.
표준 문법은 아니지만 최신 웹브라우저에서 유용하게 사용할 수 있어 정리함.

z-index : position으로 배치한 요소의 z축 위치를 지정하는 속성. 속성값으로 하나의 정수값을 작성함.
여러 요소들이 서로 겹칠 경우 정수값이 큰 요소가 위에 표시됨.
모든 요소는 z-index 기본 속성값이 0임.

3.5.3. 기타 위치 속성

1. float : 요소를 지정한 위치로 적절하게 배치함. 속성값은 아래와 같음.
none : 적용하지 않음.
left : 왼쪽으로 배치함.
right : 오른쪽으로 배치함.

float은 주로 이미지와 텍스트를 함께 배치할 때 사용함.
float을 사용했을 때 width 속성이 지정되어 있지 않으면 콘텐츠만큼 너비가 조정됨. 즉, 블록 성격 요소에 float을 사용하면 너비가 콘텐츠만해짐.
float을 사용하면 요소의 원래 위치가 보장되지 않음.
float을 사용하면 부모 요소가 해당 요소를 제대로 인식하지 못하게 됨.

2. clear : float 속성을 해제함. 속성값은 float에 맞게 left, right, both임.
float 속성을 해제하면 다른 요소의 float 속성의 영향을 받지 않음. (정확한 원리는 더 찾아봐야 함.)

3.6. 전환 효과

3.6.1. 전환 효과

CSS에서 어떤 속성의 속성값을 다른 속성값으로 변하게 하는 것을 전환(transition)이라고 함.
(ex. 마우스를 올리면 색이 변하게 하는 것)

모든 요소에 전환 효과를 적용할 수 있는 것은 아님. (https://thebook.io/080313/0302/ 참고.)

지금으로서 전환 효과 속성, 애니메이션 속성을 적용하는 이벤트의 지정은 가상 클래스 선택자로 (주로 :hover를 사용하는 듯.) 하는 것으로 보임. (Javascript 부분에 정리한 내용을 사용할 수도 있음.)

3.6.2. 전환 효과 속성

전환 효과 관련 속성들이 있음.

transition-property : 전환 효과의 대상 속성명을 값으로 지정. 지정한 속성에만 전환이 적용됨. 기본값은 all임.
transition-duration : 전환 효과가 진행되는 시간을 지정. 단위는 초임.(ex. 2s, 2000ms)
transition-delay : 전환 효과가 지연되는 시간을 지정. 단위는 초임. 해당 시간 이후에 전환이 적용됨.
transition-timing-function : 전환 효과 속도를 지정. 키워드 또는 직접 지정 가능. 직접 지정할 때는 계산하기 귀찮으므로 크롬 개발자 도구나 웹사이트 등을 사용하여 구할 수 있음.
transition : 모든 전환 효과 속성을 한 번에 지정할 수 있는 단축 속성.

아래는 예시임.

```
.container .bar{
  width:10px;
  transition-property:width;
  transition-duration:5s;
}
.container:hover .bar{
  width:110px;
}
```

3.6.3. 애니메이션 속성

전환 효과 속성보다 더 부드럽고 정확하게 전환 효과를 적용할 수 있게 하는 속성들.

애니메이션 속성을 사용하기 위해서는 키 프레임(@keyframes)을 지정해야 함. 키 프레임은 애니메이션이 진행되는 과정에서 특정 시점에 발생해야 하는 작업을 정의하는 문법임. 이는 아래와 같이 작성함.

```
@keyframes <키 프레임명>{
  0%{/* CSS 코드1 */}
  ...
  n%{/* CSS 코드2 */}
  ...
  100%{/* CSS 코드3 */}
}
```

시작할 때는 코드1의 상태, n% 진행됐을 때는 코드2의 상태, 끝날 때는 코드3의 상태가 됨. 각 상태는 부드럽게 전환됨.

0%는 from으로, 100%는 to으로 작성해도 됨.

작성한 키 프레임은 animation-name 속성으로 지정해줘야 함.

아래는 애니메이션 속성들임.

animation-name : 애니메이션을 지정할 키 프레임명 지정.
animation-duration : 애니메이션의 지속 시간을 지정. 단위는 초임.
animation-delay : 애니메이션의 지연 시간을 지정. 단위는 초임.
animation-fill-mode : 애니메이션 실행 전과 종료 후의 상태를 지정. 애니메이션 실행 이후에는 원래 상태로 돌아가는 것이 default임.
animation-iterator-count : 애니메이션의 반복 횟수를 지정.
animation-play-state : 애니메이션의 진행/정지 상태를 정의. Javascript를 사용해야 함.
animation-direction : 애니메이션의 진행 방향을 지정. 역순, 짝/홀수번째 방향 등을 지정할 수 있음.
animation-timing-function : 애니메이션의 속도를 지정.
animation : 모든 애니메이션 관련 속성을 지정.

3.7. 변형

3.7.1. 변형

요소의 크기나 위치, 각도를 바꾸는 것을 변형(transform) 효과라고 함. 변형 효과 속성으로는 transform과 tranform-origin을 주로 사용함.

3.7.2. 변형 효과 지정

transform : 변형 효과를 지정함. 속성값으로는 함수를 지정함. 함수는 아래와 같음. 이때 길이와 좌표계는 이전에 정리한 것과 동일함.

translate(a, b) : 요소를 현재 위치에서 x축으로 a만큼, y축으로 b만큼 이동.
translateX(n) : 요소를 현재 위치에서 n만큼 x축으로 이동.
translateY(n) : 요소를 현재 위치에서 n만큼 y축으로 이동.
scale(a, b) : 요소를 x축으로 a배, y축으로 b배 확대 또는 축소.
scaleX(n) : 요소를 n만큼 x축으로 확대 또는 축소.
scaleY(n) : 요소를 n만큼 y축으로 확대 또는 축소.
skew(xdeg, ydeg) : 요소를 x축과 y축으로 xdeg, ydeg(각도)만큼 기울임.
skewX(deg) : 요소를 deg(각도)만큼 x축 방향으로 기울임.
skewY(deg) : 요소를 주어진 deg(각도)만큼 y축 방향으로 기울임.
rotate(deg) : 요소를 deg(각도)만큼 회전.

기울이기, 회전에서 각도의 단위는 도(deg)임. 기울이는 것은 해당 방향으로 밀었다고 생각하면 이해가 쉬움. 회전 시에 속성값이 양수이면 오른쪽, 음수이면 왼쪽으로 회전함.

3.7.3. 변형 기준점 지정

transform-origin : 변형 시작 기준점 지정. 요소의 어느 지점을 기준으로 삼을 것인지를 지정할 수 있음.

4. 레이아웃 설계하기

HTML, CSS에서의 레이아웃(layout)은 요소를 제한된 공간 안에 효과적으로 배열하는 것을 의미함.

이전에는 HTML에서 div, span 태그, CSS에서 position, float 태그로 레이아웃을 구성했음. 현재에는 HTML5에서는 시맨틱 태그, CSS3에서는 플렉스 박스와 그리드 레이아웃을 사용할 수 있음.

여기서 정리하는 신규 레이아웃 속성들은 CSS3에 추가되어 표준으로 정해진 것은 아니지만, 최신 웹브라우저에서 유용하게 사용할 수 있음. 구 버전 웹브라우저에서는 지원하지 않는 경우가 많음.

4.1. 플렉스 박스 레이아웃

4.1.1. 플렉스 박스 레이아웃

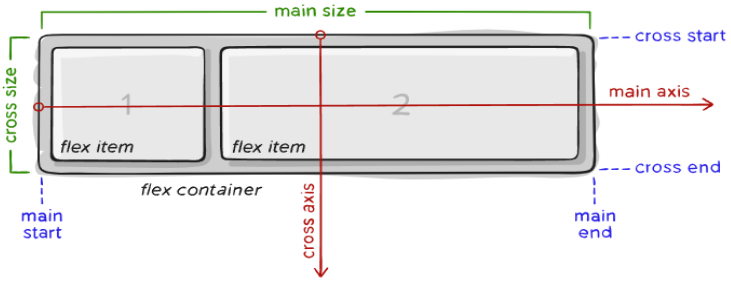
Definition 25 *플렉스/플렉서블 박스 레이아웃(flex/flexible box layout)은 1차원 방식으로 효과적으로 레이아웃을 설계할 수 있도록 고안된 속성임.*

1차원 방식은 가로(row) 또는 세로(column) 중 한 방향으로만 레이아웃을 설계하는 방식.

4.1.2. 플렉스 박스 레이아웃의 구성

주축(main axis) : 플렉스 박스의 진행 방향과 수평한 축.
교차축(cross axis) : 주축과 수직한 축.
플렉스 컨테이너(flex container) : 플렉스 아이템을 담는 그릇. display 속성값으로 flex나 inline-flex가 적용된 요소.
플렉스 아이템(flex item) : 플렉스 박스 레이아웃에서 배열되는 요소들. 플렉스 컨테이너와 자식 관계(하위x)를 이루는 태그 요소들.

플렉스 박스 레이아웃은 크롬의 개발자 도구 등으로 확인할 수 있음.



4.1.3. 플렉스 박스 레이아웃 속성

플렉스 박스 레이아웃 생성하기.

1. display:flex/inline-flex;
모든 플렉스 박스 레이아웃은 display 속성값을 flex나 inline-flex로 지정하는 것에서 시작함.
flex : 적용된 요소가 항상 줄바꿈됨.
inline-flex : 적용된 요소가 주변에 배치됨.
해당 속성값이 지정된 요소는 플렉스 컨테이너가 되고, 자식 요소들은 플렉스 아이템이 됨.
이때 플렉스 아이템이 가지고 있던 원래의 박스 성격(블록, 인라인, 인라인 블록)은 무시됨.

2. flex-direction : 플렉스 박스 레이아웃의 주축 방향을 지정.
속성값은 아래와 같음.
row : 주축 방향을 왼쪽에서 오른쪽으로 지정. 기본값.
row-reverse : 주축 방향을 오른쪽에서 왼쪽으로 지정.
column : 주축 방향을 위쪽에서 아래쪽으로 지정.
column-reverse : 주축 방향을 아래쪽에서 위쪽으로 지정.

3. flex-wrap : 플렉스 컨테이너 영역을 벗어나게 되는 플렉스 아이템의 자동 줄 바꿈 여부를 지정.
속성값은 아래와 같음.
nowrap : 플렉스 아이템이 플렉스 컨테이너를 벗어나도 무시. 줄바꿈하지 않음.
wrap : 플렉스 아이템이 플렉스 컨테이너를 벗어나면 줄 바꿈.
wrap-reverse : 플렉스 아이템이 플렉스 컨테이너를 벗어나면 wrap의 역방향으로 줄 바꿈.

4. flex-flow : flex-direction과 fled-wrap 속성을 한 번에 사용할 수 있는 단축 속성.

4.1.4. 플렉스 박스 레이아웃 정렬 속성

플렉스 박스 내부의 정렬 방법 지정하기. 텍스트의 정렬과 동일한 형태임.

어떤 위치로 정렬시키면 박스가 해당 위치에서 최소화됨.

1. justify-content : 플렉스 아이템을 주축 방향으로 정렬.
속성값은 아래와 같음.
flex-start : 주축 방향의 시작을 기준으로 정렬.
flex-end : 주축 방향의 끝을 기준으로 정렬.
center : 주축 방향의 중앙에 정렬.
space-between : 플렉스 아이템 사이의 간격이 균일하도록 정렬.
space-around : 플렉스 아이템의 둘레(around)가 균일하도록 정렬.
space-evenly : 플렉스 아이템 사이와 양끝의 간격이 균일하도록 정렬.

2. align-items : 플렉스 아이템을 교차축 방향으로 정렬.
속성값은 아래와 같음.
stretch : 교차축 방향으로 플렉스 아이템의 너비나 높이가 늘어남. 기본값.
flex-start : 교차축 방향의 시작을 기준으로 정렬.
flex-end : 교차축 방향의 끝을 기준으로 정렬.
center : 교차축 방향의 중앙을 기준으로 정렬.
baseline : 플렉스 아이템의 baseline을 기준으로 정렬.

align-content는 플렉스 아이템이 두 줄 이상으로 배열됐을 때만 사용하고, align-items와 동일한 사용법을 가짐.

align-self는 플렉스 아이템을 각각 따로 정렬할 때 사용함. 각 플렉스 아이템에 각각 사용하고, align-items와 동일한 사용법을 가짐.

4.2. 그리드 레이아웃

4.2.1. 그리드 레이아웃

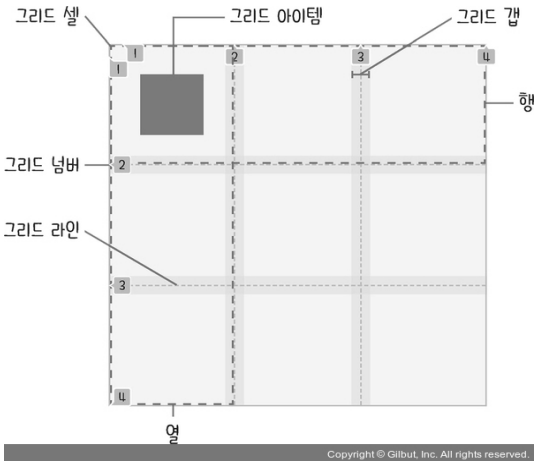
Definition 26 *그리드 레이아웃(Grid layout)은 2차원 방식으로 효과적으로 레이아웃을 설계할 수 있도록 고안된 속성임.*

2차원 방식은 가로(row)와 세로(column) 모두를 사용해 레이아웃을 설계하는 방식.

4.2.2. 플렉스 박스 레이아웃의 구성

행(row): 그리드 레이아웃에서의 가로줄.
열(column): 그리드 레이아웃에서의 세로줄.
그리드 셀(grid cell): 행과 열이 만나 이루어지는 하나의 공간.
그리드 갭(grid gap): 그리드 셀과 그리드 셀 사이의 간격.
그리드 아이템(grid item): 그리드 셀 안에서 표현되는 콘텐츠.
그리드 라인(grid line): 그리드 행과 열을 그리는 선.
그리드 넘버(grid number): 그리드 라인에 붙는 번호. 몇 번째 행/열인지가 아니라, 행/열 방향의 몇 번째인지로 세야 함. 즉, 아래 그림에서 가로 방향이 column이고 세로 방향이 row임.
그리드 컨테이너(grid container): 그리드 레이아웃의 전체 내용을 담고 있는 최상위 부모 요소. 그리드와 관련한 내용은 모두 그리드 컨테이너 안에 표현됨.

그리드 레이아웃은 크롬의 개발자 도구 등으로 확인할 수 있음.



4.2.3. 그리드 레이아웃 속성

그리드 레이아웃 생성하기.

1. display:grid/inline-grid;
모든 그리드 레이아웃은 display 속성값을 grid나 inline-grid로 지정하는 것에서 시작함.
grid : 적용된 요소가 항상 줄바꿈됨.
inline-grid : 적용된 요소가 주변에 배치됨.
해당 속성값이 지정된 요소는 그리드 컨테이너가 되고, 자식 요소들은 그리드 아이템이 됨.

2. grid-template-columns, grid-template-rows : 행과 열의 크기 지정.
아래와 같이 작성하여 행과 열의 크기를 지정함. 이때 행 크기는 가로 길이, 열 크기는 세로 길이를 의미함.
속성값으로 auto를 작성하면 그리드 컨테이너의 크기에 맞춰 자동으로 지정됨.
repeat()로 반복 지정할 수 있고, minmax로 해당 행/열을 범위로 지정할 수도 있음.

```
grid-template-columns:<1열값> <2열값> ...;
grid-template-rows:<1행값> <2행값> ...;

grid-template-columns:repeat(2, 100px); /* 100px 100px */
grid-template-rows:repeat(3, 50px); /* 50px 50px 50px */

grid-template-columns:repeat(2, minmax(50px, 100px)); /* 두 열 최소 50px~최대 100px */
grid-template-rows: repeat(2, minmax(10px, 50px)); /* 두 행 최소 10px~최대 50px */
```

3. row-gap, column-gap : 그리드 갭 지정.
행과 행 사이의 간격은 row-gap, 열과 열 사이의 간격은 column-gap으로 지정함.

4.2.4. 그리드 레이아웃 정렬 속성

각 그리드 셀 내부에서의 그리드 아이템들의 정렬 방법을 지정하기. 텍스트의 정렬과 동일한 형태임.

어떤 위치로 정렬시키면 박스가 해당 위치에서 최소화됨.

1. align-items : 셀의 높이가 그리드 아이템보다 클 때, 그리드 아이템을 세로 방향으로 정렬함.
속성값은 아래와 같음.
stretch: 그리드 아이템이 그리드 셀을 꽉 채우도록 크기를 늘림.
start : 그리드 아이템을 그리드 셀의 맨 위에 배치.
center : 그리드 아이템을 그리드 셀의 세로 방향 중간에 배치.
end : 그리드 아이템을 그리드 셀의 맨 아래에 배치.

align-self는 그리드 아이템을 각각 따로 정렬할 때 사용함. 각 그리드 아이템에 각각 사용하고, align-items와 동일한 사용법을 가짐.

2. justify-items : 셀의 너비가 그리드 아이템보다 클 때, 그리드 아이템을 가로 방향으로 정렬함.
속성값은 아래와 같음.
stretch: 그리드 아이템이 그리드 셀을 꽉 채우도록 크기를 늘림.
start : 그리드 아이템을 그리드 셀의 맨 왼쪽에 배치.
center : 그리드 아이템을 그리드 셀의 가로로 방향 중간에 배치.
end : 그리드 아이템을 그리드 셀의 맨 오른쪽에 배치.

justify-self는 그리드 아이템을 각각 따로 정렬할 때 사용함. 각 그리드 아이템에 각각 사용하고, justify-items와 동일한 사용법을 가짐.

4.2.5. 그리드 레이아웃 배치 속성

각 아이템을 그리드 레이아웃 내에서 특정 셀에 배치하기.

1. grid-template-areas, grid-area : 각 셀에 이름을 붙여 아이템을 배치함.
grid-template-areas 속성으로 그리드 컨테이너의 각 영역에 이름을 지정하고, grid-area로 각 아이템에 이름을 해당 이름을 지정하여 영역에 넣는 것. 아래는 예시임.

```
...
<style>
    .grid-container{
        display:grid;
        grid-template-areas:
        "header header header" /* 첫번째 행 */
        "sidebar content content" /* 두번째 행 */
        "footer footer footer"; /* 세번째 행 */
    }
    #header{
        grid-area:header;
    }
</style>
</head>
<body>
      <div class="grid-container">
            <p id="header">header</p> <<!-- header로 지정. 3칸 다 들어감. -->>
...
```

2. grid-column-start, grid-column-end, grid-row-start, grid-row-end
: 시작과 끝 그리드 넘버를 지정하여 해당 범위 안에 아이템을 지정하는 것.
각 속성별로 숫자 하나씩을 지정함. 각 아이템에 지정하고, 지정하지 않으면 해당 방향(행 또는 열)은 첫 번째 칸(1번부터 2번)이 지정됨.

grid-column, grid-row 속성으로 아래와 같이 단축 작성할 수도 있음.

```
grid-column:1/3; /* 또는 grid-column:1/span 2; */
grid-row:1/3; /* 또는 grid-row:1/span 2; */
```

4.3. 미디어 쿼리

4.3.1. 반응형 웹

각 기기의 디스플레이마다 최적화되어 UI 배치 등이 조정되는 웹을 반응형 웹(responsive web)이라고 함. 반응형 웹을 위해서는 뷰포트와 미디어 쿼리가 필수적임.

4.3.2. 뷰포트

Definition 27 *접속한 기기에서 보이는 웹페이지의 실제 영역의 크기를 뷰포트(viewport)라고 함.*

기기마다 뷰포트가 다르기 때문에, 뷰포트를 기기 너비로 지정해 줘야 기기에 상관없이 웹페이지가 제대로 출력됨. (자세한 내용은 더 찾아볼 필요가 있음.)

뷰포트는 meta 태그로 지정함. 주로 사용하는 설정은 아래와 같음.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

뷰포트 관련 속성값은 아래와 같음.
width : 뷰포트의 너비를 설정합니다. 보통 device-width로 설정함.
height : 뷰포트의 높이를 설정합니다. 잘 사용하지 않음.
initial-scale : 뷰포트의 초기 배율을 설정합니다. 1이 기본값이며 1보다 작으면 축소 값, 1보다 크면 확대 값으로 설정함.
minimum-scale : 뷰포트의 최소 축소 비율을 설정합니다. 기본으로 0.25가 적용되어 있음.
maximum-scale : 뷰포트의 최대 확대 비율을 설정합니다. 기본으로 5.0이 적용되어 있음.
user-scalable : 뷰포트의 확대 또는 축소 여부를 설정합니다. yes 또는 no로 지정하는데, no로 지정하면 화면을 확대 또는 축소할 수 없음.

4.3.3. 미디어 쿼리

Definition 28 *각 기기가 가진 디스플레이의 미디어 타입, 해상도 등에 따라 다른 스타일 속성을 적용하는 기술을 미디어 쿼리(media query)라고 함. 즉, 디스플레이의 속성을 조건으로 CSS 코드를 적용하는 것.*

미디어 쿼리의 기본 문법은 아래와 같음.

```
@media <not|only> <mediatype> and (<media feature>) <and|or|not> (<media feature>){
  /* CSS 코드; */
}
```

not/only : not은 뒤에 오는 모든 조건을 부정하는 것, only는 미디어 쿼리 지원 기기만 미디어 쿼리를 해석하라는 것.

mediatype : 미디어 쿼리를 적용할 미디어 타입을 지정하는 부분. 생략할 경우 all로 지정됨. 생략하지 않은 경우 반드시 뒤에 and가 와야 함. 주로 지정하는 값은 아래와 같음.
all : 모든 기기
print : 인쇄 장치(ex. 프린터기)
screen : 컴퓨터 화면 장치 또는 스마트 기기
speech : 스크린 리더기 같은 보조 프로그램으로 웹 페이지를 소리 내어 읽어 주는 장치

media feature : 미디어 쿼리를 적용할 미디어 조건을 지정하는 부분. 주로 지정하는 값은 아래와 같음.
min-width:<화면 너비> : 미디어 쿼리가 적용될 최소 너비
max-width:<화면 너비> : 미디어 쿼리가 적용될 최대 너비
orientation:portrait : 세로 모드, 뷰포트의 세로 높이가 가로 너비보다 큰 경우
orientation:landscape : 가로 모드, 뷰포트의 가로 너비가 세로 높이보다 큰 경우

즉, 아래와 같이 지정하면 미디어 쿼리를 적용할 수 있는 기기(only)이면서 컴퓨터 화면 장치 또는 스마트 기기(screen)일 때 그리고(and) 뷰포트의 너비가 최소 360px(min-width)이고 최대 720px(max-width)이면 해당 CSS 코드가 실행되는 것.

```
@media only screen and (min-width:360px) and (max-width:720px){
  /* CSS 코드 */
}
```

추가로, 미디어 쿼리는 CSS에서 우선순위를 결정하는 개별성(specificity) 규칙의 점수에 영향을 주지 않음.

Part III

Javascript

HTML과 CSS로는 정적 웹(static web)만을 구현할 수 있음. 사용자가 상호작용이 가능한 동적 웹(dynamic web)을 만들기 위해서는 Javascript라는 스크립트 언어를 추가로 사용해야 함.

1. 기본 문법

전반적으로 Javascript 문법은 C 문법과 유사한 점이 많음.

1.1. 서론

1.1.1. HTML 파일과의 연결 방법

내부 스크립트 방법과 외부 스크립트 방법이 있는데, 대체로 외부 스크립트 방식을 사용함.

이때 script 태그는 항상 body 태그의 종료 태그 바로 직전에 사용해야 함.
Javascript 코드는 실행 시에 시간을 꽤 잡아먹을 수 있는데, HTML 코드의 앞쪽에서 이를 실행하면 실행하는 동안 뒤쪽 HTML 코드가 해석되지 못하기 때문.

1. 내부 스크립트 방법 : HTML 파일에서 script 태그의 콘텐츠로 Javascript 코드를 작성하는 방법.

2. 외부 스크립트 방법 : .js 확장자를 가진 파일에 Javascript 코드를 작성하고, script 태그의 src 속성으로 파일 경로를 지정해 연결하는 방법.
관습적으로, 첫 번째로 만드는 메인 Javascript 코드 파일의 이름은 main.js로 함.
아래는 예시.

```
<body>
    <script src="script.js"></script>
</body>
```

1.1.2. Javascript 코드 실행하기

HTML 코드에 사용하는 것이 아니라도, Javascript 코드는 그 자체로 실행이 가능함.

1. 웹브라우저로 실행하기
웹브라우저 주소창에 about:blank를 입력하면 빈 창이 생성됨. 개발자 도구를 열어 콘솔(Console)창으로 이동하면 Javascript 코드를 작성하여 실행할 수 있고, 실행 결과와 반환값 등을 확인할 수 있음.

콘솔은 웹페이지를 동적으로 확인할 때 값을 띄워 테스트하거나 오류를 잡는 데에 사용하면 상당히 유용함.

2. VSC의 확장 프로그램 Code Runner 사용하기
간단한 실행 결과를 확인할 수 있음. 특정 메서드를 사용해야 하거나 글자가 깨지는 등 제약사항이 많지만 쉽고 빠르게 확인이 가능하다는 장점이 있음. Node.js도 설치야줘야 함.

1.1.3. 인터프리터 방식

Javascript는 인터프리터(interpreter) 방식을 사용함. 코드를 한 번에 한 줄씩 번역하여 실행함.

1.1.4. 주석

Definition 29 *Javascript에서 주석은 단일 줄 주석과 다중 줄 주석이 있음. (C와 동일)*

단일 줄 주석은 // 로 작성함.

다중 줄 주석은 / */ 로 작성함.*

1.1.5. 용어

키워드(keyword)(예약어, reserved word) : 프로그래밍 언어에서 기능이나 역할이 정해져 있는 단어.
식별자(identifier) : 변수, 함수 등에 부여되는 이름. 메모리에 이름을 붙이는 것.
연산자(operator) : 연산 작업을 하는 데에 사용되는 기호.
표현식(expression) : 평가되어 하나의 값을 반환하는 식이나 코드. (ex. 10 + 20)
평가(evaluation) : 표현식을 실행해 하나의 값을 만드는 과정.
값(value) : 더 이상 평가할 수 없는 데이터.
변수 선언 : 변수 선언 키워드로 식별자를 지정하는 행위.
값의 할당 : 할당 연산자로 값을 변수 공간에 대입하는 행위.
변수 초기화 : 변수 선언과 동시에 값을 할당하는 것.

1.1.6. 식별자 명명 규칙

Javascript 식별자 명명에는 아래의 규칙이 적용됨.
1. 키워드 사용 불가.
2. 공백 문자 사용 불가.
3. 식별자의 첫 글자로는 영문 소문자, _, $만 사용 가능.

식별자를 작성하는 관용적 표기법에는 다음과 같은 것들이 있음.
카멜 표기법 : 변수명, 함수명에 사용 (firstName)
언더스코어 표기법 : 상수명에 사용 (FIRST_NAME, last_name)
파스칼 표기법 : 생성자 함수명에 사용 (FirstName)

또한 식별자는 영문으로만 작성하고, 의미를 알아보기 쉽게 작성함.

1.1.7. 이스케이프 문자

문자의 원래 역할을 없애거나 특수한 기능을 위해 사용하는 문자.
이스케이프 문자에는 아래와 같은 것들이 있음.

\' : 작은따옴표(single quotes)
\" : 큰따옴표(double quotes)
\n : 줄 바꿈(new line)
\t : 수평 탭(horizontal tab)
\\: 역슬래시(backslash)

1.1.8. 블록문

{}와 내부의 코드로 구성된 것을 블록문이라고 함.

다른 문장들은 ;로 끝남을 보이지만, 관용적으로 블록문은 ;을 작성하지 않아도 됨.

1.1.9. console.log()와 document.write()

C에 printf()가 있다면 Javascript에는 console.log()와 document.write()가 있음. console.log()는 웹브라우저의 콘솔창에 내용을 출력하고, document.write()는 웹브라우저에서 출력하는 웹페이지에 내용을 출력함.

1.2. 변수와 상수

1.2.1. 변수와 상수

Definition 30 *메모리에 이름(식별자)을 붙여 데이터를 저장하고 관리하기 위한 공간을 변수(variant)라고 함.*

Javascript의 변수 선언 키워드로는 var, let, const가 있음.

값이 변하지 않는 수를 상수(constant)라고 함.

변수 선언, 할당, 초기화는 아래의 형태를 가짐. (C와 동일)

```
var num;
num = 10;
var num = 10;
```

1.2.2. var, let, const 키워드

let이 좀 더 C 문법에 가까운 느낌.

1. var : 해당 식별자로 변수를 선언함.
변수명 중복 선언 가능. 이때 뒤쪽 선언문에서 초기화한 내용이 할당됨. 그냥 할당으로 취급되는 것.
호이스팅(hoisting)됨.
블록 스코프 적용되지 않음.

2. let : var에서 여러 기능이 개선된 키워드.
같은 스코프에서 변수명 중복 불가능.
호이스팅(hoisting)되지 않음.
블록 스코프 적용됨.

3. const : let과 동일하지만, 재할당이 불가능함. 초기화 시에만 할당 가능.
const로 선언한 변수는 값을 바꿀 수 없기 때문에 상수 변수(constant variable)라고 부르기도 함.

1.3. 연산자

1.3.1. 연산자

Definition 31 *어떤 연산을 처리하는 데에 사용하도록 미리 정의된 기호를 연산자(operator)라고 함.*

1.3.2. 대입(할당) 연산자

변수에 데이터를 대입(할당)하는 연산자.

대입 연산자
= : x에 y를 대입함. (ex. x = y)

복합 대입 연산자
+= : x에 x + y를 대입함. (ex. x += y)
-= : x에 x - y를 대입함. (ex. x -= y)
= : x에 x * y를 대입함. (ex. x *= y)
/= : x에 x / y를 대입함. (ex. x /= y)
%= : x에 x % y를 대입함. (ex. x %= y)
**= : x에 x ** y를 대입함. (ex. x **= y)

1.3.3. 산술 연산자

수학 연산을 하는 연산자.

이항 산술
+ : 더하기. (ex. x + y)
- : 빼기. (ex. x - y)
: 곱하기. (ex. x * y)
/ : 나누기. (ex. x / y)
% : 앞의 값을 뒤의 값으로 나누어 나머지를 구함. (ex. x % y)
** : 앞의 값을 뒤의 값만큼 거듭제곱함. (ex. x ** y)

단항 산술(증감 연산자)
++ : x를 1 증가시킵니다. 전치와 후치 존재. (ex. ++x, x++)
−− : x를 1 감소시킵니다. 전치와 후치 존재. (ex. −−x, x−−)

단항 부정
- : x의 부호를 전환. (ex. -x)

1.3.4. 비교 연산자

피연산자를 비교하여 논리형 값인 ture, false를 반환하는 연산자.

== : x와 y의 값이 같으면 true를 반환. (ex. x == y)
=== : x와 y의 값과 자료형이 같으면 true를 반환. (ex. x === y)
!= : x와 y의 값이 다르면 true를 반환. (ex. x != y)
!== : x와 y의 값과 자료형이 다르면 true를 반환. (ex. x !== y)
< : x가 y보다 작으면 true를 반환. (ex. x < y)
<= : x가 y보다 작거나 같으면 true를 반환. (ex. x <= y)
> : x가 y보다 크면 true를 반환. (ex. x > y)
>= : x가 y보다 크거나 같으면 true를 반환. (ex. x >= y)

참고로, 숫자 문자열(ex."10")의 값은 해당 숫자 그 자체임.(ex. 10)

1.3.5. 논리 연산자

피연산자를 논리적으로 평가하여 조건에 맞는 피연산자를 반환하는 연산자.

&& : x가 참이면 y를 반환하고, 거짓이면 x를 반환. (ex. x && y)
|| : x가 참이면 x를 반환하고, 거짓이면 y를 반환. (ex. x || y)
! : x가 참이면 false를 반환하고, 거짓이면 true를 반환. (ex. !x)

논리 연산자는 ""(빈 문자열), undefined, null, 0을 false로 평가하고, 그 외의 모든 것들은 true로 평가함.

&&는 왼쪽부터 평가를 시작해 거짓인 것이 하나라도 있으면 해당 값(false)을 반환하고, 전부 참이면 마지막에 평가되는 가장 오른쪽 값(true)을 반환함.

||는 왼쪽부터 평가를 시작해 참인 것이 하나라도 있으면 해당 값(true)을 반환하고, 전부 거짓이면 마지막에 평가되는 가장 오른쪽 값(false)을 반환함.

즉, C에서와 동일한 의미의 연산자임.

1.3.6. 삼항 연산자

C의 삼항 연산자와 동일한 의미의 연산자임.

?: : x가 참이면 y를 반환하고, x가 거짓이면 z를 반환함. (ex. x ? y : z)

적절히 사용하여 조건문을 대체할 수 있음.

1.3.7. 연산자 우선순위

모든 연산자에는 우선순위가 존재하여 여러 연산자를 사용한 경우 해당 순위대로 연산이 적용됨. 우선순위가 같은 연산자들에 대해서는 결합 순서를 기준으로 연산함.

순위를 반영하여 수식을 작성하기보다는 ( )를 잘 사용하여 묶어주는 것이 좋음.

연산자 우선순위는 인터넷에서 참고하자.

1.3.8. 형변환

자료형이 바뀌거나 바꾸는 것. (자료형에 대해서는 아래에 정리함.)

1. 암시적 형변환 : Javascript에서 규칙에 따라 자체적으로 자료형을 바꾸는 것.
(ex. 문자열 + 숫자형 계산 시 문자열이 숫자형으로 형변환됨.)

2. 명시적 형변환 : 직접 코드로 자료형을 명시하여 자료형을 바꾸는 것.
(ex. String(num) 메서드를 사용하면 num의 문자열을 숫자형으로 바꿀 수 있음.)

암시적 형변환은 개발자가 놓친 부분으로 보이거나 실제로 그런 것일 수 있으므로 웬만하면 명시적 형변환을 사용하는 것이 좋음.

1.4. 조건문

범위를 다룰 때는 if문, 이산적인 값을 다룰 때는 case문이 유리함.

1.4.1. if, if else, else 문

if문은 해당 조건식을 평가하여 참이면 {} 안의 코드를 실행하는 조건문임.
if else문은 if문 뒤에 작성되어 해당 조건 외의 경우에서의 조건식을 평가하여 {} 안의 코드를 실행함.
else문은 모든 경우 이외의 경우에 대해 {} 안의 코드를 실행함.

아래는 그 예시임.

```
if(num > 10)
{
    document.write("num > 10");
}
else if(num < 5)
{
    document.write("num < 5");
}
else
{
    document.write("5 < num < 10");
}
```

C에서의 문법과 동일함.

1.4.2. switch문

switch문은 주어진 값과 일치하는 case를 실행하는 조건문임.
이때 값과 자료형 모두 일치해야 일치하는 것으로 봄.
일치하는 case가 없을 경우 default의 코드가 실행됨.
일치하는 case부터 그 아래의 코드가 전부 실행되고, break으로 그만 실행할 지점을 지정할 수 있음.

아래는 그 예시임.

```
switch(key){
  case value1:
    // key가 value1일 때 실행할 블록문
    break;
  case value2:
    // key가 value2일 때 실행할 블록문
    break;
  default:
    // 아무것도 일치하지 않을 때 실행할 블록문
    break;
}
```

1.5. 반복문

1.5.1. while, do-while문

조건식이 참이면 해당 블록문을 실행하는 반복문.
break을 사용하여 반복문을 끝낼 수 있고, contine를 사용하여 해당 회차를 넘길 수 있음.

아래는 그 예시.

```
while(조건식){
  // 조건식이 참이면 실행
}
```

한 번 실행한 후 조건을 평가하는 do-while문도 존재함.

```
do{
  // 한 번 실행하고, 두 번째부터는 조건식이 참이면 실행
}while(조건식)
```

C에서와 동일함.

1.5.2. for문

초깃값, 조건식, 증감식을 사용해 조건을 구성하는 반복문.
초깃값-조건식(평가)-블록문-증감식-조건식-...의 순서를 가짐.
break을 사용하여 반복문을 끝낼 수 있고, contine를 사용하여 해당 회차를 넘길 수 있음.

```
for(초깃값; 조건식; 증감식){
  // 블록문
}
```

1.5.3. for-in문

객체 리터럴, 배열에 반복 접근할 수 있게 하는 반복문. ES6에서 추가됨.
해당 변수에 객체 리터럴의 경우 키 값이, 배열의 경우 인덱스가 할당됨.
루프가 돌 때마다 바로 다음 키 또는 인덱스 값이 배정됨.

```
for(가변수 in 배열/객체 리터럴){
  // 블록문
}
```

예를 들어 아래와 같이 작성하면
name: 철수
age: 20
이 출력됨.

```
let obj = {name:"철수", age:"20"};
for(let key in obj)
{
    console.log(key + ": " + obj[key]);
}
```

2. 자료형

Javascript의 자료형은 기본 자료형(객체 빼고 전부)과 참조 자료형(객체)으로 나뉨.

Javascript에서는 어떤 자료형을 가지는 값이라도 변수에 할당할 수 있음.

2.1. 기본 자료형

2.1.1. 문자열

문자열 자료형은 큰따옴표 또는 작은따옴표로 묶어 작성함. 단, 시작과 끝의 따옴표가 같아야 함. C에서와 같이 문자열 데이터는 []를 사용하여 인덱스로 접근이 가능함.

문자열에 따옴표가 존재할 경우 다른 종류의 따옴표로 묶어줘야 함.
(ex. 큰따옴표가 포함되어 있으면 작은따옴표로 묶음.)
또는 그냥 이스케이프 문자 사용.

문자열은 문자열 연결 연산자(+)로 묶어줄 수 있음.
(ex. '문자열' + "연결" 로 작성하면 "문자열 연결"로 취급됨.)

ES6에서는 템플릿 문자열이 추가되었음. 템플릿 문자열은 `(백틱)으로 문자열을 묶음.
문자열 내부에서 Enter 키를 눌러 개행한 것이 \n으로 개행한 것으로 취급됨.
문자열 중간에 ${} 문법을 사용할 수 있음. {} 사이에 변수나 식을 작성하면 문자열 사이에 해당 값을 넣을 수 있음.

2.1.2. 숫자형

Javascript에서는 정수와 실수를 하나의 숫자 자료형(숫자형)으로 취급함.

이때 실수는 부동소수점 방식으로 저장되는데, 연산 시 결과가 정확하지 않은 것 주의.

2.1.3. 논리형

논리값은 논리 자료형(논리형)을 가짐. 논리값으로는 true와 false만이 존재함.
true와 false는 키워드임.

논리 연산자의 값은 논리형을 가짐.

2.1.4. 특수 자료형

1. undefined : Javascript가 내부적으로 임시 데이터를 할당했을 때 가지는 자료형.
undefined는 값이 undefined이고 자료형도 undefined임.
변수 등을 선언했을 때 아무 값도 할당하지 않으면 undefined를 값으로 가지는 것.

2. null : 빈 공간을 나타내는 자료형.
null은 값이 null이고 자료형도 null임.
변수 등을 선언한 뒤 빈 공간임을 나타내기 위해 할당함.

2.2. 객체(참조 자료형)

기본 자료형을 제외한 모든 데이터와 자료구조는 객체(참조 자료형)라고 할 수 있음.

객체 자료형에서 파생되는 자료형으로는 배열, 객체 리터럴, 함수가 있음. 즉, 배열, 객체 리터럴, 함수는 모두 자료형이고, 변수에 할당할 수 있음.

객체에 대한 자세한 정리는 뒤에서 다시 함.

2.2.1. 배열

복수의 데이터를 정의하는 자료형. 배열에 저장한 데이터를 배열 요소라고 하고, 각 배열 요소를 가리키는 숫자를 인덱스(index)라고 함.
[]로 요소를 묶어 작성하고, 0부터 시작하는 인덱스를 이용하여 해당 요소를 사용함.

배열에는 Javascript의 모든 자료형을 정의할 수 있음.

배열의 선언과 사용은 아래와 같이 함.

```
let studentScore = [80, 70, 60, 50];

document.write(studentScore[2]); // 60
```

추가로, 배열의 length 속성으로 배열의 길이를 얻을 수 있음.

```
const color = ['white', 'red', 'black', 'yellow'];

document.write(color.length); // 4
```

2.2.2. 객체 리터럴

{}를 사용하여 키(key)와 값(value)의 한 쌍으로 이뤄진 속성(property)들을 정의함. 객체를 정의하는 가장 기본적이고 간단한 방법.
배열에서 인덱스로 배열 요소에 접근했다면, 객체 리터럴에서는 키를 이용하여 속성의 값에 접근함.
이때 배열처럼 [ ]를 사용하거나 .를 사용하여 접근함.

아래는 그 예시임.

```
let studentScore = {
      koreanScore:80,
      englishScore:70,
      mathScore:90,
      scienceScore:60
};
console.log(studentScore.koreanScore);      // 80
console.log(studentScore['englishScore']);  // 70
```

3. 함수

C에서의 그것과 다르지 않음.

3.1. 함수 정의

함수를 생성하는 것을 함수를 정의한다고 함.

Javascript에서 함수를 정의하려면 아래의 방법들 중 한 가지를 사용함.

3.1.1. 함수 선언문

함수 선언문(function declaration statement)으로 함수를 정의하는 방법에서는 아래의 형식을 사용함.

```
// 정의
function 식별자()
{
    // 내용
}

// 호출
식별자();
```

C에서의 그것과 동일한 형태.

3.1.2. 함수 표현식

함수 또한 자료형이므로 변수에 할당할 수 있음. 이를 이용한 것이 함수 표현식(function expression)으로 함수를 정의하는 방법임.

함수 표현식으로 함수를 정의하는 방법에서는 아래의 형식을 사용함.
식별자가 있으면 네이밍 함수(naming function), 식별자가 없으면 익명 함수(anonymous function)라고 함. 주로 네이밍 함수 방식을 사용하고, 함수명을 변수명과 동일하게 함.

함수의 {}는 블록문이지만 여기서는 변수에 할당하는 식으로 취급해 세미콜론을 붙임.

이렇게 정의한 경우 호출 시에는 해당 변수명을 사용함.

이때 변수 선언 시에는 const 키워드를 주로 사용함.

```
// 정의
const 변수명 = function 식별자() // 네이밍 함수
{
    // 내용
};
const 변수명 = function () // 익명 함수
{
    // 내용
};

// 호출
변수명();
```

3.1.3. 화살표 함수

화살표 함수(arrow function)로 함수를 정의하는 방법에서는 아래의 형식을 사용함.

함수의 {}는 블록문이지만 여기서는 변수에 할당하는 식으로 취급해 세미콜론을 붙임.

```
// 정의
const 변수명 = () =>
{
    // 내용
};

// 호출
변수명();
```

{}를 작성하지 않으면 화살표 바로 다음에 오는 코드가 return문으로 취급됨. 아래의 코드 참고.

```
const sum = (num1, num2) => num1 + num2; //return num1 + num2; 으로 취급됨.
const result = sum(10, 20); // 30
```

3.2. 매개변수, 인수, return문

3.2.1. 매개변수와 인수

함수 정의 시 외부의 데이터를 받아오기 위한 변수를 매개변수(parameter), 함수 호출 시 보낼 데이터를 작성한 것을 인수(argument)라고 함. 즉, 함수 호출 시 인수로 작성한 데이터가 매개변수에 할당되는 것.

매개변수와 인수는 아래의 형식으로 작성함.

```
// 함수 선언문
```

```
function 함수명(매개변수1, 매개변수2, ..., 매개변수N){}
// 함수 표현식
const 함수명 = function 식별자(매개변수1, 매개변수2, ..., 매개변수N){};
// 화살표 함수
const 함수명 = (매개변수1, 매개변수2, ..., 매개변수N) => {};

// 호출
함수명(인수1, 인수2, ..., 인수N);
```

인수로 작성한 값의 개수보다 매개변수가 많이 전부 할당되지 못한 경우, 할당받지 못한 매개변수는 할당받지 못한 일반 변수처럼 undefined 값이 할당됨.

인수로 작성한 값의 개수보다 매개변수가 적어 남는 인수가 존재하는 경우, 남는 인수는 오류 없이 그냥 사라짐.

3.2.2. default값 지정

매개변수 작성 시 데이터를 할당해 놓으면, 인수로 데이터를 전달해 주지 않았을 때 해당 데이터가 default 값이 됨.

3.2.3. return문

return문으로 작성하여 함수 실행을 종료하고 반환값을 지정할 수 있음. 단순히 return만 작성하면 undefined가 반환됨.

3.3. 변수와 함수의 적용 범위

3.3.1. 스코프

Definition 32 *스코프(scope, 범위)는 변수나 함수와 같은 참조 대상 식별자를 찾아내기 위한 규칙. 즉, 변수나 함수의 사용 가능 범위.*

Javascript의 스코프는 함수 스코프(function scope) 방식이냐 블록 스코프(block scope) 방식이냐에 따라, 전역 스코프(global scope)와 지역 스코프(local scope)의 참조 범위가 달라짐.

일반적으로 함수는 전역 스코프에 정의됨.

3.3.2. 함수 스코프

Definition 33 *함수에서 정의한 블록문만 스코프의 유효 범위로 인정하는 방식. 함수 블록문 내부가 지역 스코프, 외부가 전역 스코프가 됨.*

전역 스코프의 변수나 함수는 지역 스코프에서 참조가 가능하지만, 지역 스코프의 변수나 함수는 지역 스코프에서만 참조가 가능함.

3.3.3. 블록 스코프

Definition 34 *모든 블록문을 스코프의 유효 범위로 인정하는 방식. 블록문 내부가 지역 스코프, 외부가 전역 스코프가 됨.*

단, let과 const로 선언한 변수에 대해서만 적용됨.

3.3.4. 참조 우선순위

let, const로 변수를 선언할 때, 하나의 스코프 내에서 동일한 식별자를 사용하여 중복 선언하는 것이 불가능함.

단, 블록 외부에서 선언한 변수와 같은 식별자를 블록 내부에서 사용하면 외부의 변수는 무시되고 내부의 변수가 정상적으로 작동함.

3.3.5. 호이스팅

Definition 35 *변수나 함수 등의 코드를 선언과 할당으로 분리해 선언부를 소속된 스코프의 맨 위로 옮기는 것을 호이스팅(hoisting)이라고 함.*

변수는 할당하는 부분을 제외한 선언부가 스코프의 맨 위로 옮겨지고, 함수 선언문의 함수는 정의하는 부분 전체가 스코프의 맨 위로 옮겨짐.

함수 표현식과 화살표 함수는 변수의 할당하는 형태로 함수를 정의하는 방법임. 즉, 할당되는 부분으로 취급되어 호이스팅되지 않음.

3.4. 즉시 실행 함수

3.4.1. 즉시 실행 함수

Definition 36 *정의하면서 동시에 실행까지 바로 하는 함수를 즉시 실행 함수(IIFE, Immediately Invoked Function Expression)라고 함.*

즉시 실행 함수는 실행 후 바로 메모리에서 삭제되고 해당 식별자도 다른 함수에서 다시 사용 가능함.

즉시 실행 함수는 아래의 형식을 가짐. 식별자는 작성하지 않아도 됨.

```
(function 식별자() // 매개변수
{
    // 내용
})(); // 인수
```

더 사용하지 않는 변수나 함수가 메모리에 계속 남아 있어 해당 식별자를 사용할 수 없게 되는 현상을 전역 스코프가 오염됐다고 함. 즉시 실행 함수로 이 문제를 해결할 수 있음.

4. 객체

4.1. 객체

4.1.1. 객체

Definition 37 *Javascript에서 객체(object)란 포괄적인 개념이지만, 자료형의 관점에서는 키(key)와 값(value)으로 구성된 속성들의 집합이라고 할 수 있음.*

객체는 모든 자료형의 데이터를 가질 수 있고, 여러 개의 값을 서로 다른 자료형으로도 가질 수 있음. 즉, 함수, 배열, 다른 객체 등을 값으로 가질 수 있음.

객체 속성의 값으로 들어간 함수를 메서드(method)라고 함.

위에서 정리한 것처럼 객체는 리터럴(literal) 방식으로 정의가 가능하고, 빈 객체를 생성할 수도 있음. 이때 객체의 키는 문자열로 작성하지만 따옴표는 작성할 필요 없음. 단, 문자열에 공백 문자가 들어가면 따옴표로 묶어줘야 함.

4.1.2. 객체의 데이터 관리 방법

Javascript에서 기본 자료형은 해당 데이터 자체가 메모리 공간에 저장됨. 그래서 다른 변수 등에 복사하여 저장한 후 데이터를 변경해도 서로 영향을 미치지 않는 깊은 복사(deep copy)가 적용됨.

반면에 객체(참조 자료형)는 해당 데이터의 메모리 주소가 메모리 공간에 저장됨. Javascript에서는 이를 참조(reference)한다고 함. 그래서 다른 변수 등에 복사하여 저장한 후 데이터를 변경하면 다른 쪽 데이터도 변경되어 서로 영향을 주고받는 얕은 복사(shallow copy)가 적용됨.

객체를 저장하는 변수에는 주로 const 키워드를 사용하는데, 객체에 접근하여 값을 변경하는 것이 가능한 것은 const 키워드를 사용한 변수의 값(메모리 주소)에는 변화가 없기 때문임.

4.2. 객체 속성 다루기

4.2.1. 속성에 접근하기

대괄호 연산자 또는 마침표 연산자를 사용함.

1. 대괄호 연산자로 접근하는 방법
대괄호 안에 키를 따옴표로 묶어 작성함. 여러 개의 키를 작성하려면 대괄호를 여러 개 작성함.

```
// person 객체의 name 속성에 접근
console.log(person["name"]);

// person 객체의 name 속성에 값으로 할당된 객체의 firstName 속성에 접근
console.log(person["name"]["firstName"]); // Gildong
```

2. 마침표 연산자로 접근하는 방법
객체명과 키를 마침표로 연결해 작성함. 이때 따옴표를 사용하면 오류 발생함.
키에 공백 문자가 있으면 마침표 연산자로는 접근할 수 없음.

4.2.2. 속성 변경, 추가, 삭제

1. 속성 변경
해당 속성에 키로 접근해 값을 재할당함.

2. 속성 추가
존재하지 않는 속성에 접근해 데이터를 할당하면 해당 식별자와 데이터로 속성이 추가됨.
함수, 배열, 객체 등도 동적으로 추가할 수 있음.

3. 속성 삭제
delete 키워드로 속성을 삭제함.

```
delete person.name; // name 속성이 삭제됨
```

4.3. 표준 내장 객체 사용하기

Javascript에는 기본으로 내장된 객체들이 있음. 이 객체들은 스코프에 상관없이 어디서든 사용이 가능함.

4.3.1. 문자열 처리 객체, String

문자열의 객체로써 사용할 수 있음.

속성.
length : 문자열의 길이를 반환함. (ex. "abc".length는 3임.)

메서드.
includes() : 메서드의 매개변수에 인자로 전달되는 문자열이 대상 문자열에 포함되어 있으면 true, 아니면 false를 반환함.
replace() : 대상 문자열에서 메서드의 매개변수에 인자로 전달되는 문자열과 일치하는 한 부분을 찾아서 다른 데이터로 변경한 새로운 문자열을 반환함.
replaceAll() : 대상 문자열에서 메서드의 매개변수에 인자로 전달되는 문자열과 일치하는 모든 부분을 찾아서 다른 데이터로 변경한 새로운 문자열을 반환함.
split() : 메서드의 매개변수에 인자로 전달되는 구분자를 이용해 대상 문자열을 여러 개의 문자열로 분리하고, 분리한 문자열을 새로운 배열로 반환함.
toUpperCase() : 대상 문자열을 대문자로 변경해 반환함.

trim() : 대상 문자열의 앞뒤 공백을 제거한 값을 반환함.
indexOf() : 대상 문자열과 일치하는 첫 번째 문자의 인덱스를 반환함.

아래는 예시.

```
const pw = "124";
if(pw.length < 4)
{
    console.log("비밀번호는 최소 4자리 이상 입력해 주세요.");
}
```

4.3.2. 배열 처리 객체, Array

배열의 객체로써 사용할 수 있음.

속성.
length : 배열의 요소 개수를 반환함.

파괴적 메서드.
push() : 배열의 맨 뒤에 데이터를 추가함.
pop() : 배열의 맨 뒤에서 데이터를 추출함.
unshift() : 배열의 맨 앞에 데이터를 추가함.
shift() : 배열의 맨 앞에서 데이터를 추출함.
sort() : 배열의 요소를 정렬함.
reverse() : 배열의 요소를 역순으로 정렬함.

비파괴적 메서드.
forEach() : 배열의 요소를 하나씩 순회하면서 요소마다 콜백(callback) 함수를 호출함.
filter() : 배열의 요소를 하나씩 순회하면서 요소마다 콜백 함수를 호출해 true를 반환하는 요소만 추출합니다. 추출한 요소로 새로운 배열을 만들고 만들어진 배열을 반환함.
find() : 배열의 요소를 탐색하면서 주어진 판별 함수를 만족하는 첫 번째 값을 반환함.
findIndex() : 값 대신 인덱스 숫자를 반환한다는 것만 빼면 find() 메서드와 같음.
includes() : 배열에 특정 값이 포함되어 있는지 확인해서 포함됐으면 true, 아니면 false를 반환함.
join() : 배열의 모든 요소를 주어진 구분자로 합함.

Javascript에서 함수는 객체로부터 파생된 자료형임. 함수는 다른 함수의 매개변수로도 사용이 가능한데, 이때 이 함수를 콜백 함수(callback function)라고 함.

예를 들어 forEach() 메서드는 아래와 같이 작성함. 배열의 각 요소가 콜백 함수의 인자로 들어감.

```
const arr = [10, 20, 30];
arr.forEach(function(v)
{
    console.log(v);
}); // 10 20 30이 출력됨.
```

4.3.3. 날짜/시간 처리 객체, Date

Date 객체의 메서드를 사용하려면 우선 시간 정보에 대한 인스턴스(instance)를 만들어야 함. (인스턴스와 그 문법에 대한 자세한 설명은 따로 찾아보자.) 인스턴스는 new 키워드를 사용하여 아래와 같이 생성함. 인스터스를 할당한 변수로 날짜/시간 데이터를 다루는 것.

```
const date = new Date(); // 시간 정보가 객체의 형태(메모리 주소)로 date 변수에 할당됨
```

Date()의 인자로 시간을 지정할 수도 있음. 이때 월은 0부터 시작함. 자세한 내용은 검색해보자.

```
const date1 = new Date(2022, 11, 25); // Sun Dec 25 2022 00:00:00 GMT+0900 (한국 표준
시)
```

아래의 메서드 중 get으로 시작하는 메서드는 날짜/시간 정보를 가져오고, set으로 시작하는 메서드는 날짜/시간 정보를 지정함.

메서드.
getFullYear()/setFullYear() : 연도를 4자리 숫자로 표시함.
getMonth()/setMonth() : 월을 0부터 11까지의 숫자로 표시함.(월은 0부터 시작.)
getDate()/setDate() : 일을 1부터 31까지의 숫자로 표시함.
getDay() : 요일을 0부터 6까지의 숫자로 표시함.(월은 0부터 시작)
getTime()/setTime() : 1970년 1월 1일 12:00 이후의 시간을 밀리초(1/1000초) 단위로 표시함.
getHours()/setHours() : 시를 0부터 23까지의 숫자로 표시함.
getMinutes()/setMinutes() : 분을 0부터 59까지의 숫자로 표시함.
getSeconds()/setSeconds() : 초를 0부터 59까지의 숫자로 표시함.
getMilliseconds()/setMilliseconds() : 밀리초를 0부터 999까지의 숫자로 표시함.

두 가지 시간을 getTime()으로 각각 얻어 그 차를 구하면 날짜/시간 간격을 알 수 있음.

4.3.4. 수학 연산 처리 객체, Math

Math 객체의 메서드는 Math 객체에 바로 사용함.

메서드.
Math.floor() : 주어진 숫자와 같거나 작은 정수 중에서 가장 큰 수를 반환함(내림).
Math.ceil() : 주어진 숫자와 같거나 큰 정수 중에서 가장 작은 수를 반환함(올림).
Math.round() : 주어진 숫자를 반올림한 수와 가장 가까운 정수를 반환함(반올림).
Math.random() : 0 이상 1 미만의 난수를 반환함.

아래는 예시.

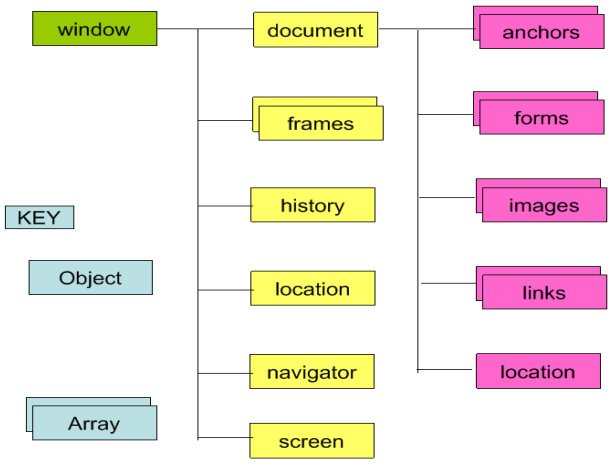
```
const random = Math.random();
console.log(random);
```

특정 범위의 난수를 구하고 싶으면 아래와 같이 적당한 수를 곱해주면 됨.

```
function getMinMaxRandom(min, max){
  return Math.floor(Math.random() * (max - min)) + 1 + min;
}
const maxRandom = getMinMaxRandom(10, 20);
console.log(maxRandom); // 10 이상 20 이하의 무작위 정수
```

4.4. 브라우저 객체 모델 사용하기

Javascript 언어 사양에 포함되지는 않지만 웹브라우저에서 제공하는 객체들을 브라우저 객체 모델(BOM, Browser Object Model)이라고 함.



window : 웹 브라우저가 열릴 때마다 생성되는 최상위 관리 객체
document : 웹 브라우저에 표시되는 HTML 문서 정보가 포함된 객체
location : 웹 브라우저에 현재 표시된 페이지에 대한 URL 정보가 포함된 객체
history : 웹 브라우저에 저장된 방문 기록이 포함된 객체
navigator : 웹 브라우저 정보가 포함된 객체
screen : 웹 브라우저의 화면 정보가 포함된 객체

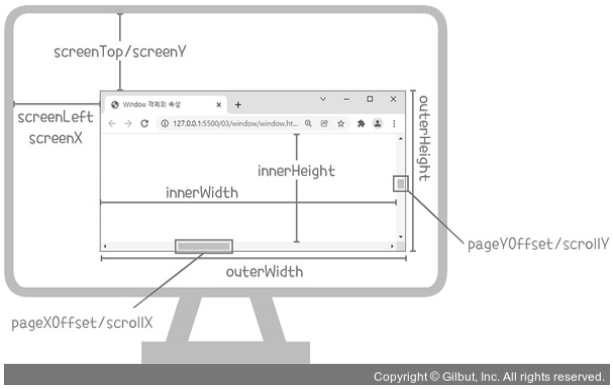
이 필기에서는 window와 document에 대해서만 정리함. document는 나중에 따로 정리함.

4.4.1. window 객체의 기본 속성과 메서드

window 객체의 속성/메서드는 window 객체에 바로 사용함. 주로 웹브라우저의 창과 관련된 속성/메서드가 많음.

속성.
innerWidth : 웹 브라우저 화면의 너비를 px(픽셀) 단위로 나타냄.
innerHeight : 웹 브라우저 화면의 높이를 px 단위로 나타냄.
outerWidth : 웹 브라우저 창의 너비를 px 단위로 나타냄.
outerHeight : 웹 브라우저 창의 높이를 px 단위로 나타냄.
screenTop/screenY : 웹 브라우저 위쪽 면과 모니터의 간격을 px 단위로 나타냄.
screenLeft/screenX : 웹 브라우저 왼쪽 면과 모니터의 간격을 px 단위로 나타냄.
pageXOffset/scrollX : 웹 브라우저의 수평 스크롤 위치를 px 단위로 나타냄.
pageYOffset/scrollY : 웹 브라우저의 수직 스크롤 위치를 px 단위로 나타냄.

이때 /로 구분된 동일한 기능의 두 속성들은, 왼쪽 속성이 더 호환성이 좋음.



메서드.
alert() : 알림창을 표시함.
confirm() : 확인창을 표시함.
prompt() : 입력창을 표시함.
open() : 새로운 웹 브라우저 창을 엶.
close() : 웹 브라우저 창을 닫음.
setTimeout() : 일정 시간(ms) 뒤에 콜백 함수를 한 번만 실행함.
setInterval() : 일정 시간(ms)마다 콜백 함수를 반복적으로 실행함.
clearInterval : setInterval() 메서드로 반복 실행되는 함수를 중지함.
scrollTo() : 웹 브라우저의 스크롤을 특정 위치만큼 이동함.
scrollBy() : 웹 브라우저의 스크롤을 현재 위치에서 상대적 위치로 이동함.

4.4.2. 새로운 창 열기

window 객체의 open() 메서드를 아래와 같이 작성하여 새로운 창을 열 수 있음.

```
window.open(경로, 이름, 제어속성);
```

경로에는 새로운 창으로 띄울 파일(html)의 경로를 작성함.
이름에는 해당 창의 이름을 작성함. 이 이름은 open() 메서드에서 내부적으로 창을 구분하기 위한 것으로,

open() 메서드는 이름당 한 번씩만 새로운 창을 띄움.
제어 속성에는 아래와 같은 것들이 있음.

width : 웹 브라우저의 너비를 px 단위로 지정함. (ex. width=400)
height : 웹 브라우저의 높이를 px 단위로 지정함. (ex. height=400)
left : 웹 브라우저 왼쪽에서의 위치를 px 단위로 지정함. (ex. left=400)
top : 웹 브라우저 위쪽에서의 위치를 px 단위로 지정함. (ex. top=400)

띄운 창은 close() 메서드로 닫을 수 있음.

4.4.3. 웹브라우저의 스크롤 이동하기

window 객체의 scrollTo(), scrollBy() 메서드로 스크롤 위치를 지정할 수 있음.
scrollTo()는 해당 좌표로 이동하고, scrollBy()는 상대적인 위치로 해당 값만큼 이동함.
형식은 아래와 같음.

```
window.scrollTo(x좌표, y좌표);
window.scrollBy(x좌표, y좌표);
```

특정 속성을 추가하여 움직임을 제어할 수도 있음. 자세한 것은 검색해보자.

5. 문서 객체 모델

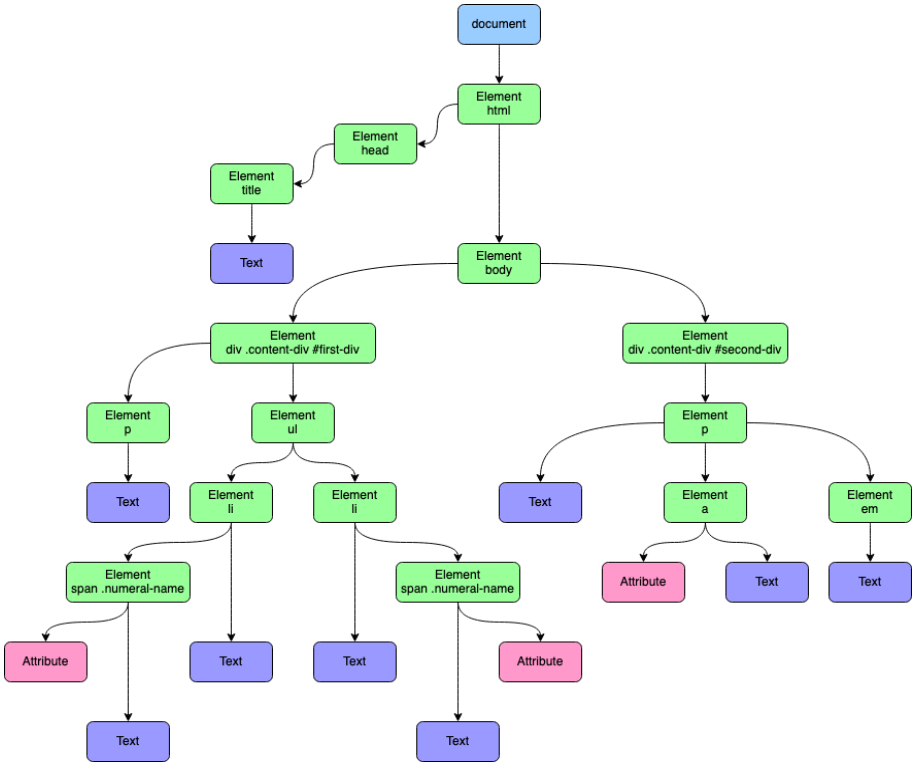
5.1. 문서 객체 모델

5.1.1. 문서 객체 모델

웹브라우저는 HTML 문서를 해석하여 문서 객체 모델(DOM, Document Object Model)을 생성함. 이렇게 생성한 문서 객체 모델을 다시 해석하여 웹페이지로 출력하는 것. 이때 웹브라우저는 HTML 문서의 모든 구성 요소를 객체로 인식하여 문서 객체 모델을 생성하는데, Javascript로 이 객체들을 조작하여 HTML 문서 구조를 변경, 추가, 삭제하는 등의 작업을 할 수 있음.

추가로, 문서 객체 모델은 CSS를 포함하지 않음.

문서 객체 모델은 아래와 같은 트리(tree) 구조를 가지는데, 이를 DOM 트리라고 함.



5.2. 노드

DOM 트리는 요소, 속성, 텍스트, 주석 등으로 구성되는데, 이 각각의 구성 요소를 노드(node)라고 함. 또한 DOM 트리의 최상위 노드를 루트 노드(root node)라고 함. 루트 노드는 주로 html 태그가 포함된 요소임.

각 노드들끼리는 부모, 자식, 형제 등의 관계가 형성됨.

5.2.1. 노드 타입

DOM 트리에서는 구성요소의 종류에 따라 다른 타입의 노드가 됨. 현대적인 DOM 트리에서는 아래의 5가지 노드 타입이 존재함.

문서 노드(Node.DOCUMENT_NODE) : 최상위 document 객체의 노드 타입.
요소 노드(Node.ELEMENT_NODE) : 요소의 노드 타입. (ex. h1, p 태그)
속성 노드(Node.ATTRIBUTE_NODE) : 속성의 노드 타입. (ex. href, src 속성)
텍스트 노드(Node.TEXT_NODE) : 텍스트에 해당하는 노드 타입.
주석 노드(Node.COMMENT_NODE) : 주석에 해당하는 노드 타입.

5.3. 노드 선택

브라우저 객체 모델에서의 window 객체의 document 객체를 사용해 Javascript로 문서 객체 모델을 조작할 수 있음. document 객체로 문서 객체 모델의 노드를 조작하려면, 우선 어떤 노드를 조작할지를 지정할 수 있어야 함.

document의 속성/메서드로 노드를 선택했으면, 해당 데이터를 상수 변수 (const 키워드 사용) 등에 저장하여 사용할 수 있음.

다수의 노드를 선택한 경우 반복문 등을 사용하여 전부 처리해야 하는 것 염두하자.

5.3.1. document의 속성으로 선택하기

document 객체는 노드를 탐색하며 선택할 수 있도록 하는 속성들을 가지고 있음.

document 객체부터 시작해서 DOM 트리를 탐색할 수 있음. 예를 들어 html 요소에 접근하려면 아래와 같이 작성함.

```
document.firstElementChild; //html
```

모든 노드 탐색.
parentNode : 부모 노드를 반환함.
childNodes : 모든 자식 노드를 반환함.
firstChild : 첫 번째 자식 노드를 반환함.
lastChild : 마지막 자식 노드를 반환함.
previousSibling : 이전 형제 노드를 반환함.
nextSibling : 다음 형제 노드를 반환함.

요소 노드만 탐색.
parentElement : 부모 요소 노드를 반환함.
children : 자식 요소 노드를 반환함.
firstElementChild : 첫 번째 자식 요소 노드를 반환함.
lastElementChild : 마지막 자식 요소 노드를 반환함.
previousElementSibling : 이전 요소 노드를 반환함.
nextElementSibling : 다음 요소 노드를 반환함.

DOM 트리가 복잡하고 규모가 커질수록 원하는 노드를 찾기 번거롭다는 단점이 있음.

5.3.2. document의 메서드로 선택하기

document 객체는 노드를 탐색하며 선택할 수 있도록 하는 메서드들을 가지고 있음.

선택된 노드에 대한 데이터는 나름의 형식을 가지고 있음. 여러 개의 노드가 선택된 경우 HTMLCollection(get 메서드 사용 시), NodeList(query 메서드 사용 시) 등의 객체에 담겨 반환됨. 이 객체에는 배열처럼 인덱스로 접근할 수 있음. 배열처럼 사용이 가능하지만 배열에는 없는 여러 속성을 가지고 있어 유사 배열이라고 부름. 자세한 내용은 검색해보자.

1. get 메서드 : 요소의 속성과 태그를 이용하는 방법.
getElementById("<id 속성값>") : 해당 id 속성값과 일치하는 요소 노드를 1개만 선택.
getElementsByClassName("<class 속성값>") : 해당 class 속성값과 일치하는 요소 노드를 전부 선택.
getElementsByTagName("<태그명>") : 해당 HTML 태그명과 일치하는 요소 노드를 전부 선택.

2. query 메서드 : CSS 선택자를 이용하는 방법.
querySelector("<CSS 선택자>") : 지정한 CSS 선택자에 해당하는 노드를 1개만 선택.
querySelectorAll("<CSS 선택자>") : 지정한 CSS 선택자에 해당하는 노드를 전부 선택.

5.4. 노드 조작

DOM 트리의 노드들이 모두 객체인 것을 생각하면 조작에 대한 이해가 간단함.

5.4.1. 콘텐츠 조작하기

선택한 노드의 타입이 요소이면 콘텐츠를 조작할 수 있음.

속성.
textContent : 요소 노드의 모든 텍스트에 접근. (HTML 태그 등은 포함x.)
innerText : 요소 노드의 텍스트 중 웹 브라우저에 표시되는 텍스트에만 접근.
innerHTML : 요소 노드의 텍스트 중 HTML 태그를 포함한 모든 텍스트에 접근.

해당 속성으로 텍스트를 가져올 수 있고, 속성에 값을 할당하면 해당 노드의 콘텐츠를 바꿀 수 있음. 이때 textContent, innerText에는 태그를 작성해 추가해도 단순 텍스트로 취급되고, innerHTML에 태그를 작성해 추가하면 태그는 태그로 적용됨.

5.4.2. 스타일 조작하기

선택한 노드의 타입이 요소이면 스타일을 조작할 수 있음. style 속성을 사용하는데, 형식은 아래와 같음. 특수한 형식인 것은 아니고, style 객체의 특정 CSS 속성을 객체로서 접근하는 것.

```
<노드>.style.<CSS 속성명> = <속성값>;
```

이때 속성명에 대시(-)가 있는 속성은 Javascript에서 뺄셈 연산자로 인식하기 때문에 카멜 표기법으로 변경해서 작성해야 함. (ex. background-color를 backgroundColor로.)

5.4.3. class 속성 조작하기

선택한 노드의 타입이 요소이면 classList 속성으로 해당 요소의 class 속성을 조작할 수 있음. 여러 스타일을 지정하는 경우 등에서는 CSS 선택자로 class를 지정하고 요소에 해당 class를 추가하는 것이 편리함.

add(), remove(), toggle() 메서드를 사용하고, 그 형식은 아래와 같음.

```
<노드>.classList.add("class 속성값"); // 추가. 한 번에 여러 개 작성 가능.
<노드>.classList.remove("class 속성값"); // 삭제. 한 번에 여러 개 작성 가능.
<노드>.classList.toggle("class 속성값"); // 추가와 삭제 반복. 사용법은 따로 검색해보자.
```

5.4.4. data 속성 조작하기

사용자 정의(custom) 속성인 data-* 속성을 조작할 수 있음.

속성.
dataset : data-* 속성에 대한 정보를 가져옴. DOMStringMap 객체에 담겨 반환됨.

data-*에서 *에 해당되는 data 속성명을 dataset의 속성으로 지정하여, data 속성의 값을 가져오거나 바꿀 수 있음. 아래는 그 예시임.

```
<button data-cnt="10">가방 구매</button>
<button data-cnt="0">신발 구매</button>

const buttonEls = document.querySelectorAll("button");
buttonEls.forEach((el) => {
    el.dataset.cnt = 50; // 두 요소 각각에 50이 지정됨.
})
```

5.4.5. 메서드로 속성 조작하기

아래의 메서드들을 사용하면 어떤 속성도 지정하여 편리하게 조적할 수 있음.

<노드>.getAttribute("속성명"); : 속성값을 가져옴.
<노드>.setAttribute("속성명", "속성값"); : 속성을 지정함.
<노드>.removeAttribute("속성명"); : 속성을 삭제함.

단, class 속성을 조작할 경우 classList가 더 나은 선택일 수 있음. classList는 기존의 속성값을 보존한 상태에서 새로운 값을 추가하고, 지정하지 않은 속성값은 삭제하기 않고 유지함. setAttribute()는 속성값을 완전히 새로 지정하고, removeAttribute() 속성 자체를 완전히 지워버림.

5.5. 노드 추가/삭제

5.5.1. 노드 추가

DOM 트리에 노드를 추가하기 위해서는 1. 노드 생성 2. 노드 연결의 과정을 거쳐야 함. 이때 사용할 수 있는 메서드들은 아래와 같음.

노드 생성.
createElement() : 요소 노드를 생성함.

createTextNode() : 텍스트 노드를 생성함.
createAttribute() : 속성 노드를 생성함. 값을 지정하여 속성 노드를 완성할 수 있음.

노드 연결.
<기준 노드>.appendChild(<노드>) : 해당 노드를 기준 노드에 자식 노드로 연결함.
<기준 노드>.setAttributeNode(<속성 노드>) : 기준 노드에 속성 노드를 연결함.

우선 요소 노드를 생성해 연결함. 이때 body 요소 등은 DOM 트리에서 제공하는 요소인 document.body로 바로 접근이 가능함. 이후 텍스트와 속성을 넣으려면 해당 요소 노드에 추가하면 됨. 이때 속성은 자식 노드가 아니라 속성 노드로서 추가해야 함.

5.5.2. 노드 삭제

노드 삭제에는 removeChild() 메서드를 사용함.

<기준 노드>.removeChile(<자식 노드>) : 해당 기준 노드의 자식 노드를 지정하여 삭제함.

5.6. 폼 조작

문서 객체 모델을 이용해 HTML의 폼(form)을 제어할 수 있음.

5.6.1. 폼 선택하기

폼을 조작하기 위해서는 우선 조작할 폼을 선택해야 함. forms 속성 또는 name 속성을 사용할 수 있음.

1. forms 속성 사용하기. (document 객체의 속성)
document.forms로 모든 form 요소들에 대한 데이터를 얻을 수 있음. document.forms의 값은 유사 배열인 HTMLCollection 객체임. 인덱스로 접근하여 사용할 수 있지만 form의 순서가 바뀌면 값도 바뀐다는 단점이 존재함.

2. name 속성 사용하기. (HTML 속성)
form 태그에 name 속성으로 이름을 지정하면, 해당 속성값을 사용하여 document.속성값의 형태로 form 요소 노드에 접근할 수 있음. 아래는 그 예시임.

```
<body>
    <form name="frm1">
        <input type="text">
    </form>
</body>
...
document.frm1; // form 태그의 name 속성값이 frm1인 노드
```

5.6.2. 폼 요소 선택하기

폼을 선택했으면 그 내부의 폼 요소를 선택해야 함. 그 원리는 폼 선택과 유사함.

1. elements 속성 사용하기. (폼 객체의 속성)
<폼>.elements로 모든 폼 요소들에 대한 데이터를 얻을 수 있음. <폼>.elements의 값은 유사 배열인 HTMLFormControlsCollection 객체임.

2. name 속성 사용하기. (HTML 속성)
폼 요소 태그에 name 속성으로 이름을 지정하면, 해당 속성값을 사용하여 <폼>.속성값의 형태로 form 요소 노드에 접근할 수 있음. 또한 이때 HTMLFormControlsCollection 객체에도 해당 속성값으로 인덱스가 형성되기 때문에 대괄호로도 작성이 가능함. 아래는 그 예시임.

```
document.frm1.uname; // name 속성값이 uname인 노드
document.frm1.elements['uname']; // name 속성값이 uname인 노드
```

5.6.3. 폼 요소의 입력값 다루기

1. 한 줄/여러 줄 입력 요소 다루기
value 속성을 사용해 입력받은 값을 가져오거나 지정할 수 있음. 아래는 그 예시임.

```
document.frm1.pw.value; // document.폼.폼요소.value로 작성한 것
document.frm1.pw.value = "jscodeing"; // 값 지정
```

아무것도 입력하지 않은 입력 요소에는 말 그대로 아무것도 없음. undefined도 아님. 이 경우 value.length == 0 인 것으로 판정 가능. 또는 jquery라는 것을 사용한다는데 필요하면 찾아보자.

2. 체크박스, 라디오버튼, 콤보박스 다루기
value 속성으로 값을 가져올 수 있지만, checked/selected 속성으로 해당 항목이 체크되어있는지를 확인하거나 지정하는 방법을 주로 사용함. checked/selected 속성은 논리형 데이터를 가짐.

체크박스, 라디오버튼에서는 checked를, 콤보박스에서는 selected 속성을 사용함.

아래는 체크박스 노드에 하나씩 접근하여 선택되어 있는지를 확인하는 예시임.

```
const checkboxEls = document.querySelectorAll("[type='checkbox']");
for(let i = 0; i < checkboxEls.length; i++)
{
	if(checkboxEls[i].checked === true)
	{
		console.log(checkboxEls[i].value);
	}
}
```

3. 파일 업로드 요소 다루기
파일 업로드 요소(input 태그의 type 속성값을 file로 지정한 요소)는 files 속성을 사용하여 FileList 객체를 반환받아 다룰 수 있음.

Files 객체의 속성은 아래와 같음.

```
const files = document.frm.upload.files;
files[0].name; // 파일 이름
files[0].size; // 파일 크기
files[0].type; // 파일 타입
files[0].lastModifiedDate; // 파일 마지막 수정일
```

5.6.4. 폼 관련 기타 메서드들

1. submit() : 폼 요소의 값을 전송함.

2. focus() : 폼 요소에 포커스(커서)를 이동함.

5.7. 이벤트 조작

5.7.1. 이벤트

Definition 38 *웹브라우저와 사용자 사이에 상호작용이 발생하는 특정 시점. 자바스크립트로 이벤트 종류에 따른 동작을 지정할 수 있음.*

5.7.2. 이벤트 종류

대표적인 이벤트들만 정리함.

마우스 이벤트
onclick : 마우스로 클릭하면 발생함.

ondblclick : 마우스로 빠르게 두 번 클릭하면 발생함.
onmouseover : 마우스 포인터를 올리면 발생함.
onmouseout : 마우스 포인터가 빠져나가면 발생함.
onmousemove : 마우스 포인터가 움직이면 발생함.
onwheel : 마우스 휠(wheel)을 움직이면 발생함.

키보드 이벤트
onkeypress : 키보드 버튼을 누르고 있는 동안 발생함.
onkeydown : 키보드 버튼을 누른 순간 발생함.
onkeyup : 키보드 버튼을 눌렀다가 뗀 순간 발생함.

포커스 이벤트
onfocus : 요소에 포커스가 되면 발생함.
onblur : 요소가 포커스를 잃으면 발생함.

폼 이벤트
onsubmit : 폼이 전송될 때 발생함.

리소스 이벤트
onload : 웹 브라우저의 리소스 로드가 끝나면 발생함.

5.7.3. 이벤트 등록

이벤트 발생 시 수행할 작업을 Javascript 코드로 작성하는 것을 이벤트 등록이라고 함. 이벤트 등록에는 크게 3가지 방법이 있음.

1. 인라인 방식 : HTML 태그에 속성으로 이벤트를 등록하는 방법.
HTML 태그에 이벤트 종류를 속성으로, 실행할 함수를 속성값으로 작성함. 해당 함수는 Javascript 코드를 작성하는 부분에 작성함.

아래는 그 예시임.

```
<button onclick="clickEvent()">클릭</button>
<script>
    function clickEvent()
    {
        alert("click");
    }
</script>
```

참고로, 포커스 이벤트를 사용할 때 경고창을 띄우는 alert() 메서드를 사용해선 안 됨. 경고창이 무한으로 뜨는 현상이 발생할 수 있음.

2. 프로퍼티 리스너(property listener) 방식 : 객체로서 속성에 접근하여 함수를 지정하는 방법.
함수를 할당문에서 작성해도 되고, 따로 작성한 함수를 할당해도 됨.

```
<button>클릭</button>
<script>
  const btnEl = document.querySelector("button");
  btnEl.onclick = function()
  {
    alert("click");
  }
</script>
```

```
<button>클릭</button>
```

```
<script>
  const btnEl = document.querySelector("button");
  btnEl.onclick = clickEvent;
  function clickEvent(){
    alert('click');
  }
</script>
```

3. 이벤트 등록 메서드 사용 : DOM에서 제공하는 addEventListener() 메서드를 사용한 방법.
매개변수에 이벤트 타입과 이벤트 함수를 작성하는데, 이벤트 타입은 위에 정리한 이벤트 종류에서 on을 뺀 것임. 이벤트 함수로는 해당 함수의 정의를 바로 작성해도 되고, 정의는 따로 작성하고 함수명만 지정해도 됨. 형식은 아래와 같음.

```
<노드>.addEventListener("<이벤트 타입>", <이벤트 함수>);
```

이때 이벤트 함수의 인자를 작성하면 해당 노드가 객체로서 들어감.

3가지 방법 중 가장 권장되는 방법임.

아래는 그 사용 예시임.

```
<button>클릭</button>
<script>
    const btnEl = document.querySelector("button");
    btnEl.addEventListener("click", function()
    {
        alert("button Click");
    });
</script>
```

5.7.4. this 키워드

이벤트 함수 내부에서 this 키워드를 사용하면 이벤트가 발생한 요소 노드를 바로 가리킬 수 있음. 즉, 이벤트 함수의 인자로 들어가는 해당 노드를 가리킬 수 있는 것.

이벤트 함수로 화살표 함수가 작성된 경우 this가 의도대로 작동하지 않을 수 있음. 잘 알아보고 사용하자.

5.7.5. 이벤트 삭제

preventDefault() 메서드를 사용하면 default값으로 지정되어 있는 이벤트(a, form 등에서.)를 삭제할 수 있음. 아래는 그 예시임.

```
<a href="https://www.naver.com">네이버 이동</a>
<a href="https://www.daum.net">다음 이동</a>
<script>
  const aEls = document.querySelectorAll("a");
  for(let i = 0; i < aEls.length; i++){
    aEls[i].addEventListener("click", function(e){
      // 기본 이벤트 취소
      e.preventDefault();
    });
  }
</script>
```

자세한 사용법은 검색해보자.

5.7.6. 이벤트 객체

Definition 39 *이벤트 타입에 따라 발생하는 이벤트의 정보들이 담긴 객체를 이벤트 객체라고 함.*

이벤트 객체는 이벤트 발생 시 내부적으로 이벤트 함수로 전달되기에 작성자가 전달해 줄 필요는 없음. 다만 이벤트 객체를 매개변수로 받아와 내부의 여러 정보를 활용할 수 있음. 아래는 그 예시임.

```
<button>클릭</button>
<script>
  const btnEl = document.querySelector("button");
  btnEl.addEventListener("click", function(event){ // 이벤트 객체
    console.log(event); // 내부의 여러 정보들이 출력됨
  })
</script>
```

클릭 이벤트에서는 PointerEvent 객체가 전달되고, 아래와 같은 속성들이 담겨 있음. 이를 유용하게 활용할 수 있음.
clientX : 마우스가 클릭된 x좌표(수평 스크롤 포함 X)
clientY : 마우스가 클릭된 y좌표(수직 스크롤 포함 X)
pageX : 마우스가 클릭된 x좌표(수평 스크롤 포함 O)
pageY : 마우스가 클릭된 y좌표(수직 스크롤 포함 O)
screenX : 모니터의 왼쪽 위 모서리를 기준으로 마우스가 클릭된 x좌표
screenY : 모니터의 왼쪽 위 모서리를 기준으로 마우스가 클릭된 y좌표

키보드 이벤트에서는 KeyboardEvent 객체가 전달되고, 아래와 같은 속성들이 담겨 있음.
keyCode : 키보드에서 눌린 키의 유니코드 값을 반환함.
ctrlKey : Ctrl 키가 눌렸으면 true, 그렇지 않으면 false를 반환함.
altKey : Alt 키가 눌렸으면 true, 그렇지 않으면 false를 반환함.
shiftKey : Shift 키가 눌렸으면 true, 그렇지 않으면 false를 반환함.

Part IV

Project

최종 프로젝트로서 개인 웹페이지 만들기.

1. 최종 프로젝트

1.1. 최종 프로젝트

1.1.1. 개요

실무에서 웹페이지를 만들 때에는 디자인 파일을 받아서 구현하는 경우가 많은데, 이걸 위해서는 포토샵, Adobe XD 등의 디자인 프로그램을 이해하고 있어야 함. 여기에서는 그 대신 아래의 웹페이지를 모델 삼아 프로젝트를 진행함.
(https://pensive-kowalevski-fdf325.netlify.app/)

아래의 내용들은 웹페이지 작성 순서에 맞춰 정리함.

1.1.2. HTML 기본 구조 작성

!DOCTYPE, html, head, body, 메타데이터 등 HTML의 기본 구조를 만듦.

이때 임의의 텍스트를 작성하기 위해 html 태그의 lang 속성은 zxx로 지정할 수 있음.

1.1.3. 헤더 영역

1. HTML 코드 작성
헤더 영역과 그 내부의 네비게이션 영역을 만들 것이므로 head 태그와 nav 태그를 사용함.

head 태그의 자식 요소로 바로 div 태그를 사용하는데, container라는 이름의 class를 지정했음. 이 container는 전체 코드 이곳저곳에 사용되며 전체 프로젝트의 기준 너비(보통 1140px)를 지정하기 위한 래퍼(wrapper)임.

2. CSS 코드 작성
CSS 코드를 작성하기에 앞서 웹브라우저의 기본 스타일 시트를 초기화해야 함. 이때 margin, padding을 0으로 지정하고, border-box 등을 지정함.

색 지정을 inherit으로 해주면 body 등의 색과 동기화할 수 있음.

container를 만들어 줌.

1.1.4. 메인 영역

메인 영역을 어떤 것으로 할 것인지는 작성자의 선택임. 여기서는 웹페이지에 접속했을 때 사용자에게 보여지는 화면으로 함. 화면에 꽉 차게 하려면 뷰포트 넓이를 기준으로 단위를 입력하면 됨.

각 요소들은 웬만하면 div 태그로 묶고, class를 지정해야 하는 경우에도 div에 하는 것이 좋음. 한 요소에는 class를 최소로 작성하는 것이 보기에 편함.

1.1.5. 섹션 영역

섹션 태그는 h2 태그부터 h6 태그 중에서 하나를 반드시 사용해야 함.

영역을 나눌 때는 그 너비를 %로 지정하면 편리함.

1.1.6. 배경 영역

배경을 지정할 때는 div 태그를 사용함. section 태그를 사용할 경우 h2 태그부터 h6 태그 중에서 하나를 사용하도록 권장되기 때문.

background-attatchment 속성을 사용하면 괜찮은 느낌을 줄 수 있음.

1.1.7. 반응형 코드 적용

폰트 사이즈를 rem으로 작성해 왔다면, 디스플레이 크기에 따라 html에 font-size를 지정해 줌으로써 글자 크기를 조정할 수 있음.

또한 디스플레이 크기에 따라 container와 main, section 등의 크기를 지정할 수 있음.

1.1.8. Javascript 적용

부드러운 이동 등 적용.

1.1.9. 유효성 검증하기

HTML에 대한 유효성 검증은 https://validator.w3.org/#validate_by_input 등에서, CSS에 대한 유효성 검증은 https://jigsaw.w3.org/css-validator/#validate_by_input 등에서 하면 됨.