Chapter 2

CSS: Desinging HTML

Contents

2.1 Introducing CSS

Proposal of CSS

1장을 통해 HTML을 이용하여 간단한 형태의 웹 페이지를 작성할 수 있었다. 그러나 HTML만 사용하여 작성한 웹 페이지는 우리가 평소에 보는 잘 디자인 된 웹 페이지와는 다르게 매우 단순하고 어떠한 디자인도 할 수 없다. 인터넷의 초창기에 웹 페이지가 단순했을 때에는 디자인 역시 단순했기 때문에 HTML 문서에 웹 페이지의 구조뿐만 아니라 디자인 요소까지 작성하였다. 예를 들어, **Code 2.1**과 같이 `li` 태그는 정보를 저장하는 태그에 지나지 않았고, 텍스트에 스타일을 저장하기 위해 `font`, `b` 등의 태그를 사용하여 스타일을 저장하곤 했다. 그러나 시간이 흐르면서 스타일 및 레이아웃에 관한 정보를 훨씬 많이 저장하게 되었고, HTML 문서에는 본래의 목적인 "문서의 구조 서술"과는 거리가 먼, 디자인과 관련된 부가적인 정보가 지나치게 많이 작성되게 되었다. 이로 인해 HTML은 인간이 읽었을 때에도 문서의 구조를 파악하기 힘들고, 웹 브라우저가 사용자에게 웹 페이지를 렌더링하기 위해 분석하는 작업조차 힘든, 비효율적인 언어가 되었다.

Code 2.1 Example of early HTML

```
<body>
    <li><font color="red">HTML before CSS existence.</font></li>
    <b>This is a bold text.</b>
    <i>This text is italicized.</i>
</body>
```

HTML이 가지는 이러한 비효율적인 면을 개선하기 위해 1996년 CSS(Cascading Style Sheet)가 발표되었고, HTML과 CSS가 분리되면서 HTML에는 가급적 문서에 대한 구조만 서술하고, CSS에는 각 요소에 대한 스타일이나 레이아웃만을 서술하도록 권고되었다. CSS의 도입으로 HTML은 본연의 목적을 되찾아 문서의 구조를 표현하는 효율적인 언어가 되었으며, 더 나아가 웹 브라우저가 여러 웹 페이지에서 공통으로 사용되는 CSS 문서를 서버로부터 여러 번 다운로드할 필요가 없어져 웹 페이지 로딩 역시 빨라지게 되었다.

이번 장에서는 이러한 CSS를 이용하여 HTML 문서를 디자인하는 방법에 대해 다룰 것이다.

2.2 Basic Structure of CSS

Basic Structure of CSS

CSS는 문서의 구조를 체계적으로 서술하는 언어인 HTML과 유사하게 스타일이나 레이아웃을 체계적으로 서술하는 컴퓨터 언어이므로 지켜서 작성해야 하는 규칙이 있다.

```
selector { property1: value1; property2: value2 }
```

Figure 2.1 Basic Structure of CSS

CSS의 구조는 **Figure 2.1**과 같이 표현될 수 있다. **선택자(selector)**는 HTML 요소를 태그 이름, 클래스 이름, 아이디, 상태, 속성 등을 기준으로 선택하는 방법을 서술하는 문자열이며, 선택자에 의해 선택된 요소들에 일괄적으로 적용할 스타일을 중괄호({}) 내부에 작성한다.

스타일은 key-value pair의 형태인 속성과 속성값의 집합으로 표현한다. **속성(property)**은 HTML 요소에 적용하고자 하는 디자인 요소로, 너비, 높이, 글자의 색, 폰트의 크기 및 굵기 등 250가지가 넘는 다양한 속성들이 존재하며, 각 속성에는 그에 대응하는 **속성값(value)**을 지정할 수 있다. 예를 들어, 글자의 색상과 관련된 속성은 `color`이며, 속성값으로는 `red`나 `blue`와 같은 색상 이름을 지정할 수 있다. 속성과 속성값 pair를 `property: value`의 형태로 쓰고, 각 pair를 세미콜론(`;`)으로 구분하여 나열한다.

Code 2.2 Simple example of CSS

```
.main-panel {
    width: 800px;
    height: 450px;
    border: 1px solid black;
}

.ball {
    width: 80px;
    height: 80px;
    border: 1px solid red;
    border-radius: 40px;
    position: absolute;
}
```

하나의 요소는 여러 선택자에 의해 선택될 수 있으며, 이러한 선택자들에 의해 속성값의 충돌(conflict)이 일어날 수 있다. 이때 실제로 적용되는 속성값은 아래의 우선순위 규칙에 따라 우선순위가 가장 높은 선택자에 의해 적용되는 속성값이 적용된다.

1. 속성값의 뒤에 `!important`가 붙은 속성
2. HTML에서 `style` 속성을 사용하여 정의한 속성

3. ID > 클래스나 추상클래스의 이름 > 태그 이름 선택자의 속성
4. 상위 요소에 의해 상속된 속성

우선순위가 같은 경우, 부모-자식의 관계가 많을수록 우선순위가 높고, 그 다음으로는 나중에 작성된 속성이 적용된다.

Application of CSS on HTML

먼저 HTML 문서에 CSS 문서를 적용하는 세 가지 방법을 알아보자.

첫 번째 방법은 inline style으로, HTML 요소에 `style` 속성의 값으로 property-value pair를 직접 열거하는 방법이며, HTML 요소에 개별적으로 디자인을 적용하는 방식이기 때문에 선택자를 쓰지 않는다. 이 방법은 각 요소가 어떠한 디자인을 가지는지 쉽게 알 수는 있으나, 웹 페이지의 구조를 표현한다는 HTML의 목적에 위배되며, 요소마다 스타일을 작성해주어야 하므로 동일한 스타일을 적용하고자 하는 HTML 요소가 많아지면 문서가 불필요하게 길어지며 유지 및 보수 역시 번거로워진다. 따라서 inline style은 지양되는 스타일이지만, 예외적으로 서식이 있는 텍스트(rich text)를 표현할 때에는 자주 사용된다.

Code 2.3 Applying CSS with inline style

```
<div>
    Already member? <span style="font-weight: bold">Sign In</span>
</div>
<div>
    <span style="color: red">Sign Up</span>
</div>
```

두 번째 방법은 internal style sheet으로, `head` 태그 내부에 `<style type="text/css">` 요소를 삽입하고, 그 내부에 CSS 코드를 작성한다. 선택자를 사용할 수 있으므로 inline style보다는 효율적으로 작성할 수 있으나, HTML 본연의 목적에는 여전히 위배되며, 여러 HTML 문서에 동일한 CSS 문서를 적용할 때에는 여전히 번거롭고 비효율적이다.

Code 2.4 Applying CSS with internal style sheet

```
<style type="text/css">
    .title {
        font-weight: bold;
        font-size: 16px;
    }
    #article-list {
        list-style-type: none;
        font-size: 12px;
    }
</style>
<div class="title">KWEB Front-end Study: </div>
<ul id="article-list">
    <li>Ch 1. Introduction to Front-end</li>
    <li>Ch 2. HTML: The Basic Structure</li>
    <li>Ch 3. CSS: Designing HTML</li>
```

```
    <li>Ch 4. Basics of JavaScript</li>
    <li>Ch 5. JavaScript: Dynamic Frontend</li>
</ul>
```

가장 추천
마지막 방법은 **external style sheet**으로, HTML 문서와 CSS 문서를 서로 다른 파일에 작성하고, HTML 문서의 head 태그 내부에 link 태그를 이용하여 CSS 파일을 연동한다. link에는 다음과 같은 속성을 지정하여야 한다.

- type="text/css" - 연결하고자 하는 문서가 CSS 형태임을 명시한다.
- rel="stylesheet" - 연결하고자 하는 문서가 HTML 문서의 stylesheet임을 명시한다.
- href - 연결하고자 하는 CSS 문서의 주소를 명시한다.

먼저 index.html과 style.css를 같은 폴더 내에 생성하고, **Code 2.5**와 같이 style.css를 작성한다.

Code 2.5 Applying CSS with external style sheet - style.css

```
.title {
    font-size: 16px;
    font-weight: bold
}

#article-list {
    padding: 0;
    list-style-type: none
}

#article-list > li {
    font-size: 12px
}
```

이후, index.html을 **Code 2.6**과 같이 작성한다. link 태그의 구조를 확인해보자.

Code 2.6 Applying CSS with external style sheet - index.html

```
<head>
    <link type="text/css" rel="stylesheet" href="./style.css">
</head>
<body>
    <div class="title">KWEB Study So Far: </div>
    <ul id="article-list">
        <li>Ch 0. Introduction to Front-end</li>
        <li>Ch 1. HTML: The Basic Structure</li>
        <li>Ch 2. CSS: Designing HTML</li>
    </ul>
</body>
```

이제 index.html 파일을 웹 브라우저에 열어서 확인해보면, CSS 파일에 작성된 디자인이 적용되었음을 확인할 수 있다. 이처럼 external style sheet 방식은 웹 페이지의 구조를 표현하고, 스타일을 표현한다는 HTML과 CSS 각각의 목적을 모두 달성하면서도 유지 및 보수가 용이하다는 장점이 있다.

2.3 Style Properties

2.2절에서 설명한 바와 같이 CSS의 속성은 매우 많다. 모든 속성과 속성값의 역할을 모두 알고 있는 것이 나쁘지는 않으나, 주로 사용되는 속성들을 잘 습득하고 그 외에 필요한 속성은 필요할 때 찾아서 사용하는 것이 바람직하다. 이번 절에서 스타일과 관련된 속성 중 자주 사용되는 속성들을 알아보자.

Text Properties

- `text-align` – 텍스트를 수평적으로 어떻게 정렬할지에 관한 속성이다. 속성값으로는 `center`, `left`, `right`, `justify`가 있으며, 각각 가운데, 왼쪽, 오른쪽, 양쪽 정렬을 뜻한다.
- `text-decoration` – 텍스트를 선(line)을 이용하여 꾸민다. `text-decoration-line`, `text-decoration-color`, `text-decoration-style`의 3가지 세부 속성이 있고, 각 속성의 값을 `text-decoration` 속성의 값으로 차례대로 나열하여 표현하여도 된다.
- `line-height` – 줄 간격에 관한 속성. 100%는 `1`, 150%는 `1.5` 등 단위 없이 표현한다.
- `letter-spacing` – 글자 간 간격에 관한 속성. 단위는 %, em, px 등을 사용한다.
- `vertical-align` – 텍스트를 세로 방향으로 어떻게 정렬할지에 관한 속성. 속성값으로는 `baseline`, `top`, `bottom`, `text-top`, `text-bottom`, `middle` 등이 있다.

Font Properties

- `font-size` – 글자의 크기에 관한 속성. 단위는 em, px 등을 사용하여 표현한다.
- `font-weight` – 글자의 두께에 관한 속성. `100`, `200`, ⋯, `900`의 값이 가능하다. 이 외에도 `normal`, `bolder` 등의 값이 가능하며, 각각 `400`, `700`에 해당하는 값이다.
- `font-family` – 글자의 서체를 지정하는 속성. 웹 브라우저에서 지원하는 서체뿐만 아니라 로컬 컴퓨터나 외부 서버에 있는 서체도 가능하다.
- `color` – 글자의 색상에 관한 속성. `red`, `blue` 등의 색상 이름이나, `rgb(26, 219, 158)` 또는 `rgba(26, 219, 158, .5)`와 같이 RGB(A) 형태로 나타낸 값, `hsl(161, 79%, 48%)`나 `hsla(161, 79%, 48%, .5)`와 같이 HSL(A) 형태로 나타낸 값, `#1ADB9E`와 같이 HEX 형태로 나타낸 값 모두 가능하다.

Background Properties

- `background-color` – 배경의 색상에 관한 속성으로, 앞의 `color`과 동일한 형태의 값을 가질 수 있다.
- `background-image` – 배경에 이미지를 삽입할 수 있는 속성
- `background-repeat` – 배경 이미지가 반복되는 형태를 지정할 수 있다. 속성값으로는 `repeat`, `repeat-x`, `repeat-y`, `no-repeat` 등이 있다.
- `background-size` – 배경 이미지의 크기에 관한 속성

2.4 Selectors

2.2절에서 CSS의 구조에 관해 다루면서 선택자(selector)를 소개하였다. CSS에서는 선택자에 따라 원하는 HTML 요소에 원하는 스타일을 지정할 수 있다. 이번 절에서는 선택자를 작성하는 방법을 학습한다.

Universal Selector

전체 선택자(universal selector)는 HTML 문서의 모든 요소를 선택하며, *로 작성한다.

Code 2.7 Universal selector

```
1 * { width: 80% }
```

Tag, Class, ID Selector

태그 이름, 클래스 이름, 아이디를 기준으로 요소를 선택하는 선택자로, 태그 이름은 `tag-name`, 클래스 이름은 `.class-name`, 아이디는 `#id`의 형태로 작성한다. 또한, 태그 이름, 클래스 이름, 아이디 등으로 표현된 선택자 `element`로 선택되는 요소 중 클래스 이름이 `class-name`인 요소를 선택하는 선택자는 `element.class-name`의 형태로 작성한다.

Code 2.8 Examples of tag, class, ID selector

```
1 ul { list-style: none }
2 .title { font-size: 20px }
3 #article-list { padding: 0 }
4 .title.recent-article { font-weight: bold }
```

Child Selector and Descendants Selector

자식 선택자(child selector)와 자손 선택자(descendants selector)는 두 개 이상의 선택자를 이용하여 요소를 선택하는 선택자이다. 자식 선택자는 `parent > child`의 형태로 쓰며, `parent` 선택자로 선택된 각 요소의 바로 밑에 있는 자식 요소 중 `child` 선택자를 기준으로 선택한다. 반면, 자손 선택자는 `parent child`의 형태로 쓰며, `parent` 선택자로 선택된 요소들의 밑에 있는 모든 자식 요소 중 `child` 선택자를 기준으로 선택한다.

Code 2.9 Understanding child selector and descendants selector

```
1 <div class="class1"><span id="span1"></span></div>
2 <div class="class2"><span id="span2"></span></div>
3 <div class="class1">
4     <div><span id="span3"></span></div>
5 </div>
```

Code 2.9를 통해 이해해보자. 자식 선택자인 `.class1 > span`은 `#span1`만 선택하나, 자손 선택자인 `.class1 span`은 `#span1`과 `#span3`를 선택한다.

Pseudo-class Selector

가상 클래스 선택자(pseudo-class selector)는 특정한 상태에 놓여있는 요소들을 선택하는 선택자이다. 예를 들어 `button` 태그로 구현한 버튼에 일반적인 상태, 호버된 상태(`:hover`), 비활성화된 상태(`:disabled`) 등의 상태를 클래스의 형태로 나타낸 것을 가상 클래스(pseudo-class)라고 하며, 각각 다른 스타일을 적용할 수 있다. 이러한 가상 클래스 선택자는 `:pseudo-class`의 형태로 쓰고, `element:pseudo-class`로 표현된 선택자는 `element` 선택자로 선택된 요소 중 `pseudo-class`에 해당하는 요소들을 선택한다. `a` 태그는 `:link`, `:visited`, `:hover`, `:active` 등의 가상 클래스를 가질 수 있다.

Code 2.10 Example of pseudo-class selector (1)

```
<style>
    button { color: black }
    button:hover { color: red }
    button:disabled { background-color: yellow }
</style>
<button>Enabled Button</button>
<button disabled>Disabled Button</button>
```

가상 클래스에는 HTML 요소의 상태뿐만 아니라 구조와 관련된 가상 클래스도 있다.

- `:first-child` – 선택된 요소 중 가장 첫 번째 요소를 선택하는 가상 클래스
- `:last-child` – 선택된 요소 중 가장 마지막 요소를 선택하는 가상 클래스
- `:nth-child(e(n))` – n으로 표현된 식 $e(n)$에 대해, 선택된 요소 중 $e(0), e(1), e(2), \cdots$ 번째 요소를 모두 선택하는 가상 클래스; `e(n)`은 $an+b$의 꼴만 가능하며, `e(n)` 대신 `odd`나 `even`을 사용할 수 있다.
- `:nth-last-child(e(n))` – `nth-child`와 유사하게 동작하나, 뒤에서부터 선택하는 가상 클래스

이 외에도, `:not(selector)`은 `selector`에 의해 선택되지 않는 요소들만 선택하는 가상 클래스이다.

Code 2.11 Example of pseudo-class selector (2)

```
<style>
    li:first-child { color: red }
    li:last-child { color: blue }
    li:nth-child(4n+3) { color: yellow }
    li:nth-last-child(4n+3) { color: green }
    li:not(#current) { font-style: italic }
</style>
<ul>
    <li>01</li>
    <li>02</li>
    <li>03</li>
    <li>04</li>
    <li>05</li>
    <li id="current">06</li>
```

```
15      <li>07</li>
16      <li>08</li>
17      <li>09</li>
18      <li>10</li>
19  </ul>
```

가상 클래스의 종류는 매우 많으므로, 자주 쓰이는 몇 가지를 제외하고는 필요할 때마다 찾아서 사용하면 된다. 더 많은 가상 클래스는 아래의 MDN reference에서 확인할 수 있다.

- https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

이번 절에서 다양한 선택자에 대하여 알아보았으며, 이 절에서 언급한 선택자 외에도 속성 선택자, 동위 선택자, 가상 요소 선택자 등 다양한 선택자가 존재한다. 교재에서 언급되지 않은 선택자들은 아래 제시된 링크들에서 확인할 수 있다.

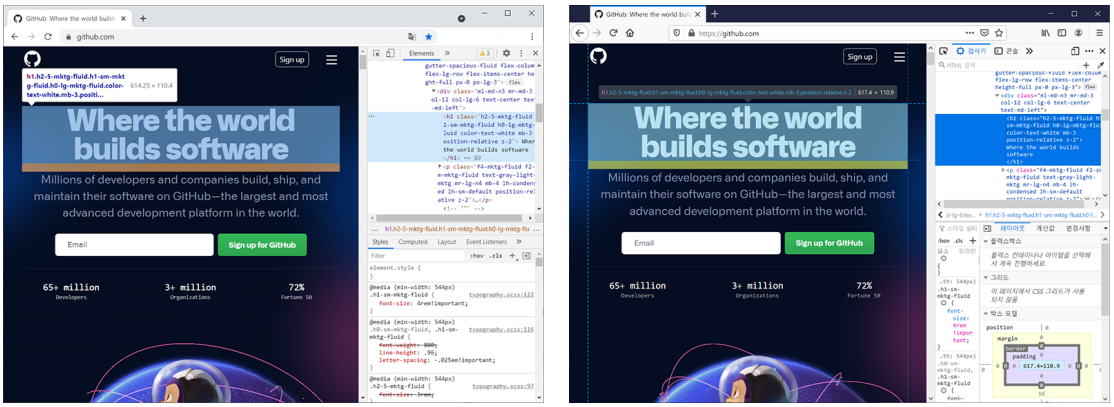
- https://www.w3schools.com/cssref/css_selectors.asp
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors
- https://css-tricks.com/almanac/selectors

2.5 Layouts

2.1절에서 CSS는 스타일과 레이아웃을 표현하는 데에 목적이 있는 문서라고 소개하였다. 지금까지는 CSS를 이용하여 텍스트의 크기나 색상, 배경의 색상이나 이미지를 디자인하는 방법을 주로 다루었고, 이번 절에서는 CSS를 이용하여 레이아웃을 설정하는 방법을 다룰 것이다.

Layout

레이아웃(layout)은 직역하면 "배치"라는 뜻을 갖는 단어이다. CSS에서 레이아웃을 설정하는 것은 웹 페이지에서 사용자가 정보를 원활하게 주고받을 수 있게끔 HTML 요소들을 적절하게 배치하고 정돈하는 작업이다. HTML 요소들을 웹 페이지 상에 원하는 대로 배치하는 것은 문서가 복잡해질수록 어려운 일이다. 따라서 레이아웃을 정확히 설정하기 위해서는 각 CSS 속성들과 속성값의 성질을 정확히 알아야 한다.



(a) Chrome **(b)** Firefox

Figure 2.2 Checking layout of HTML element using DevTools

다행히도, 웹 브라우저의 개발자 도구는 특정한 HTML 요소가 웹 페이지 내에서 영역을 어떻게 차지하고 있으며, 어떻게 배치되었는지 확인할 수 있는 기능을 제공한다. Chrome과 Firefox에서 F12를 눌러 개발자 도구를 열고, Chrome은 Elements 탭을, Firefox는 검사기 탭을 눌러 열려있는 HTML 문서의 코드를 띄운다. 그리고 레이아웃을 확인하고자 하는 요소 위에 마우스를 hover하면 해당 요소의 레이아웃이 **Figure 2.2**와 같이 나타난다. 반대로, 웹 페이지에서 HTML 코드를 확인하고 싶은 부분을 우클릭한 후 [검사]를 누르면 해당 요소의 코드를 찾아준다. 이렇게 HTML 문서를 작성할 때 개발자 도구를 이용하면 특정 요소가 어떠한 레이아웃을 갖는지, 어떻게 수정해야 할지 등의 정보를 보다 쉽게 얻을 수 있다.

Box Model

HTML의 각 요소는 **사각형 형태**의 레이아웃을 가지며, 이러한 레이아웃을 box model이라고 한다. **Figure 2.3**을 참고하여 box model이 어떻게 구성되는지 알아보자.

- **Content**: HTML 요소의 내용에 해당하는 영역
  - 예: `img` 태그의 이미지, 제목 태그의 내부 텍스트

- **Border**: 요소의 테두리가 차지하는 영역
- **Padding**: Border과 content 사이의 영역
  - 예: 배경은 content 영역과 padding 영역에 적용됨
- **Margin**: 테두리 바깥쪽에서 요소가 차지하는 영역; 원칙상 다른 요소와는 겹치지 않음.

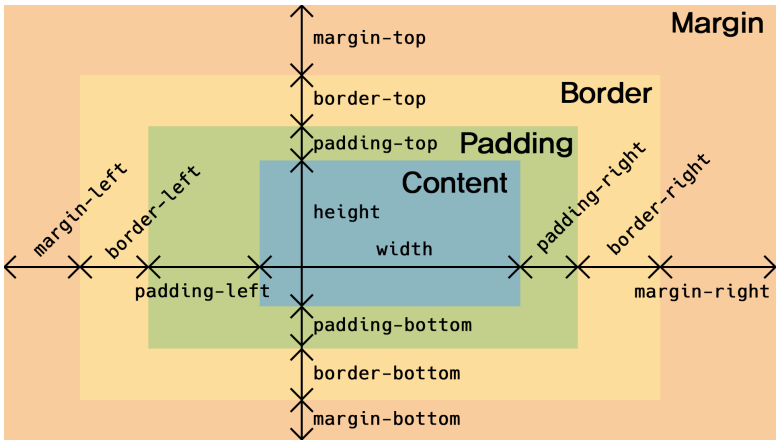


Figure 2.3 Box model

이 box model은 HTML 요소 배치의 기본이 되기 때문에 제대로 이해하는 것이 매우 중요하다. 앞으로 나오는 예제 코드를 직접 작성해보고, 브라우저의 개발자 도구를 이용하여 padding과 border, margin이 어떻게 나타나는지 각각 살펴보자.

먼저, content에 관여하는 속성으로는 `width`와 `height`가 있다. 두 속성은 각각 content 영역의 너비와 높이를 지정하는 속성이며, px, em 등의 단위나 %를 이용하여 명시할 수 있다. %로 지정하는 경우는 상위 요소의 너비나 높이를 기준으로 한다.

Code 2.12 Content area

```
<style>
    #small-box {
        width: 400px;
        height: 200px;
        background-color: red;
    }

    #large-box {
        width: 800px;
        height: 400px;
        background-color: blue;
    }
</style>

<div id="small-box"></div>
<div id="large-box"></div>
```

Border에 관여하는 속성으로는 `border-width`, `border-style`, `border-color` 등이 있다. `border-height`는 테

두리의 굵기, border-style은 테두리의 모양, border-color는 테두리의 색상을 나타내는 속성으로, 테두리의 모양은 solid, dashed, dotted 등의 값을 지정하여 테두리의 모양을 결정할 수 있다. 이 세 가지 속성은 각각 따로 작성하여도 되고, border 속성에 굵기, 모양, 색상 순으로 나열하여 작성할 수도 있다. **Code 2.13**에서 #box 요소의 border 속성이 이러한 방법으로 작성되었다.

또한, 방향에 관한 키워드[1]를 사용하여 네 면 중 하나의 면에만 특정 스타일을 지정해줄 수 있다. 예를 들어, 왼쪽 면에만 스타일을 적용하고 싶은 경우 border-left 속성에 스타일을 작성한다.

border-radius는 테두리의 모서리를 둥글게 만들 수 있는 속성으로, 속성값이 하나인 경우 주어진 값을 반지름으로 하는 원형으로, 두 개인 경우 두 값을 짧은 반지름과 긴 반지름으로 하는 타원형으로 만든다. border와 마찬가지로 네 모서리에 모두 적용되며, 한쪽 모서리에만 적용하고 싶은 경우 역시 방향과 관련된 키워드를 사용하여 기술한다. 예를 들어, 좌상단 모서리의 스타일은 border-top-left-radius 속성에 작성한다.

Code 2.13 Border area

```
<style>
    #box {
        width: 300px; height: 300px;
        border: 3px solid black;
        border-radius: 20px;
    }
</style>

<div id="box"></div>
```

Padding 속성은 border 영역의 각 면이 content 영역의 각 면으로부터 어느 정도 떨어져 있는지, margin 속성은 요소 가장자리의 각 면이 border 영역의 각 면으로부터 어느 정도 떨어져 있는지 명시하여 작성해준다. 앞의 border 영역과 마찬가지로 방향 키워드를 사용하여 한쪽 면에만 특정 padding/margin 값을 적용할 수 있다. (예: padding-left, margin-bottom)

다만 padding과 margin 속성은 방향마다 속성과 속성값을 모두 작성하는 번거로운 방식 대신, **Table 2.1**과 같이 padding, margin 속성의 값에 차례대로 각 방향의 값을 나열하여 shortened form으로 작성하는 것이 가능하다.

Table 2.1 Shortened padding and margin

Shortened Form	top	right	bottom	left
padding/margin: A;	A	A	A	A
padding/margin: A B;	A	B	A	B
padding/margin: A B C;	A	B	C	B
padding/margin: A B C D;	A	B	C	D

Code 2.14를 참고하여 padding과 margin을 정하는 방법을 이해하고, 개발자 도구를 이용하여 box model의 각

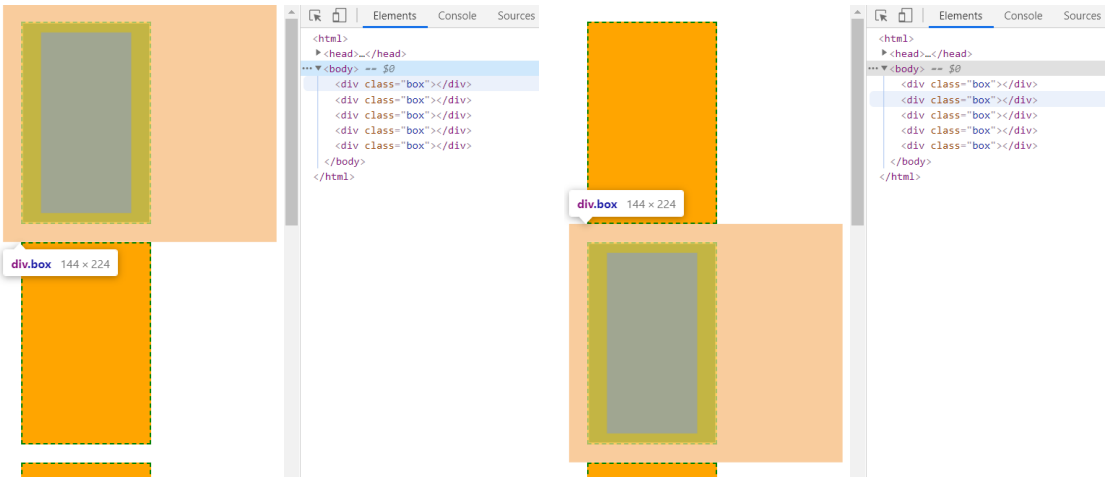
[1] CSS에서 문서의 위, 아래, 왼쪽, 오른쪽 방향의 키워드는 각각 top, bottom, left, right이다.

부분이 웹 브라우저상에 어떻게 표시되는지 확인해보자.

Code 2.14 Examining box model

```
<style>
    .box {
        width: 100px;
        height: 200px;
        border: 2px dashed green;
        padding: 10px 20px;
        margin: 20px;
        background-color: orange;
        display: inline-block;          /* We will learn this later */
    }
</style>

<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
<div class="box"></div>
```

Code 2.14에서 `display: inline-block;`을 지우고 웹 페이지를 열어보면 각 `.box` 요소가 세로로 나열된다. 이때 **Figure 2.4**와 같이 두 요소의 margin이 겹친 것을 확인할 수 있으며, 이렇게 HTML 요소 간에 margin이 겹치는 현상을 마진 상쇄 또는 마진 겹침(margin collapsing)이라고 한다. Margin 겹침 현상은 인접한 형제 요소 간의 상하 margin이 겹칠 때, 빈 요소의 상하 margin이 겹칠 때, 부모 요소의 top(bottom) margin과 첫 번째(마지막) 자식 요소의 top(bottom) margin이 겹칠 때 발생한다. **Code 2.14**의 경우는 형제 요소 간의 상하 margin이 겹친 경우로, 이러한 현상에 유의하여 디자인하여야 한다.



(a) Upper box **(b)** Lower box

Figure 2.4 Margin collapsing

Inline-level and Block-level Elements

지금까지 모든 HTML 요소는 box model에 따라 영역이 결정되고, 영역에 따라 다른 요소들과 맞물려 배치됨을 학습하였다. 앞의 **Code 2.14**에서는 HTML 요소가 이전 형제 요소의 오른쪽에 충분히 빈 공간이 있다면 오른쪽에, 그렇지 않으면 아래쪽에 배치되었다. 그런데 지금까지 HTML을 다루면서, 이전 형제 요소에 따로 margin이나 padding을 지정해주지 않아도 이전 형제 요소와 같은 줄에 배치되지 않는 현상을 보았다.

Code 2.15 Example of inline-level, block-level elements

```
1 <h1>HTML Element Layout</h1>
2 <h2>This is about <span style="color: red">HTML element layout</span>.</h2>
```

Code 2.15에서 h1 태그와 h2 태그에 padding이나 margin이 전혀 부여되지 않았음에도 불구하고 서로 다른 줄에 배치되었다. 반대로, h2 태그 내부의 텍스트와 span 태그 내부의 텍스트는 서로 같은 줄에 배치되어 있다. 앞의 1.2절에서 non-semantic 태그는 다른 요소와 같은 줄에 있으려고 하는지에 따라 div 태그와 span 태그로 나뉜다고 했던 것을 기억하는가? 이처럼 같은 줄에 다른 요소가 배치되는 것을 허용하는지에 따라 inline-level 요소와 block-level 요소로 구분할 수 있고, 이 성질은 display 속성을 이용하여 명시할 수 있다.

먼저 inline-level 요소는 다른 요소와 같은 줄을 공유하고, content의 너비만큼만 가로 폭을 차지한다. Inline-level 요소에는 width, height, margin-top, margin-bottom, padding 등의 속성을 적용할 수 없으며, 그 내부에 inline-level 요소를 포함할 수 없다. display 속성값이 inline인 요소는 inline-level 요소의 성질을 가지며, span, a, strong, img, br, input, select, textarea, button 등의 태그는 기본적으로 inline-level 요소이다.

Block-level 요소는 다른 요소와 같은 줄을 공유하지 않으며, 화면의 가로 폭을 모두 차지하며 여러 block-level 요소들을 배치하면 수직 방향으로 배치된다. Block-level 요소에는 width, height, margin, padding 등의 속성을 적용할 수 있으며, 그 내부에 inline-level 요소를 포함할 수 있다. display 속성값을 block인 요소는 block-level 요소의 성질을 갖게 되며, div, h1~h6, p, ol, ul, li, hr, table, form 등의 태그는 기본적으로 block-level 요소이다.

그런데, 레이아웃을 구성하다 보면 요소들을 같은 줄에 배치하면서 width, height, margin, padding 등의 속성을 사용해야 하는 경우가 있다. 이러한 문제는 해당 요소의 display 값을 inline-level 요소의 성질과 block-level 요소의 성질을 모두 갖는 inline-block으로 지정하여 해결할 수 있다. 또한, display 값을 none으로 지정하면 해당 요소가 보이지 않아, 사용자로부터 숨길 수 있다.[2] 이제 **Code 2.14**에서 display 속성의 값을 바꿔보며 이해해보자.

Position

지금까지는 이전의 형제 요소가 배치되고 난 다음 공간에 다음 요소가 차례대로 배치되는 레이아웃 배치 방식에 대해 알아보았다. 이러한 방식이 일반적인 배치 방식이지만, 간혹 웹페이지에서 다른 요소들과의 위치와는 관계없이 고정된 위치에 배치되는 요소들이 있고, 다른 요소와 겹치게끔 배치되는 요소들이 있다. 대표적으로는 뉴스 기사에서 스크롤을 내려도 계속 화면상에 표시되는 광고들이 이러한 배치를 갖는다. 이렇게 특정 위치에 HTML

[2] 유사한 속성으로는 visibility: hidden이 있으나, display: none과는 달리 요소가 공간을 차지하되 보이지만 않게끔 한다.

요소를 배치하고자 할때 position 속성을 이용한다.

position 속성에는 static, relative, absolute, fixed의 네 가지 값이 존재한다. 기본값은 static으로, static 배치 방식은 HTML 요소가 지금까지 학습해왔던 배치 방식, 즉 빈 공간에 왼쪽에서 오른쪽으로, 위에서 아래로 차례대로 배치된다. 나머지 세 값은 네 개의 방향 속성을 함께 사용하여 요소의 위치를 정할 수 있다. 방향 속성, 즉 top, left, right, bottom 속성은 기준점이 요소로부터 어떤 방향으로 얼마나 떨어져 있는지 나타내며, 속성값이 명시되지 않은 경우 기본값은 0이다.

먼저 relative는 static으로 지정되었을 때를 기준으로 방향 속성의 값에 따라 이동되어 배치되며, 네 방향 속성이 모두 지정되지 않은 경우 static과 동일하게 배치된다. 요소의 위치가 relative로 인해 static이었을 때와 비교하여 벗어났다고 해서 다른 요소들의 위치가 바뀌지는 않는다.

absolute는 원래 요소가 배치되었어야 할 공간과 관계없이 position 값이 static이 아닌 부모나 조상 요소를 기준으로 배치되며, 이러한 부모나 조상 요소를 찾지 못하면 body 태그를 기준으로 배치된다. absolute 속성값을 갖는 요소가 원래 배치되었어야 하는 공간에는 다른 요소가 들어올 수 있으며, 요소의 display 속성이 block이더라도 너비가 content에 맞게 바뀌기 때문에 적절한 너비를 지정해주어야 한다.

fixed는 absolute와 유사한 성질을 갖지만, 위치를 지정하는 기준이 viewport, 즉 화면에 보여지는 웹페이지의 영역이다. 따라서 화면을 스크롤하여 올리거나 내렸을 때도 화면에서 고정된 위치에 요소를 배치할 때 사용된다.

Code 2.16에서 #pos-element에 position과 관련된 속성과 그 값을 지정하고 바꿔가며 position에 대해 정확히 이해해보자.

Code 2.16 Example of position property

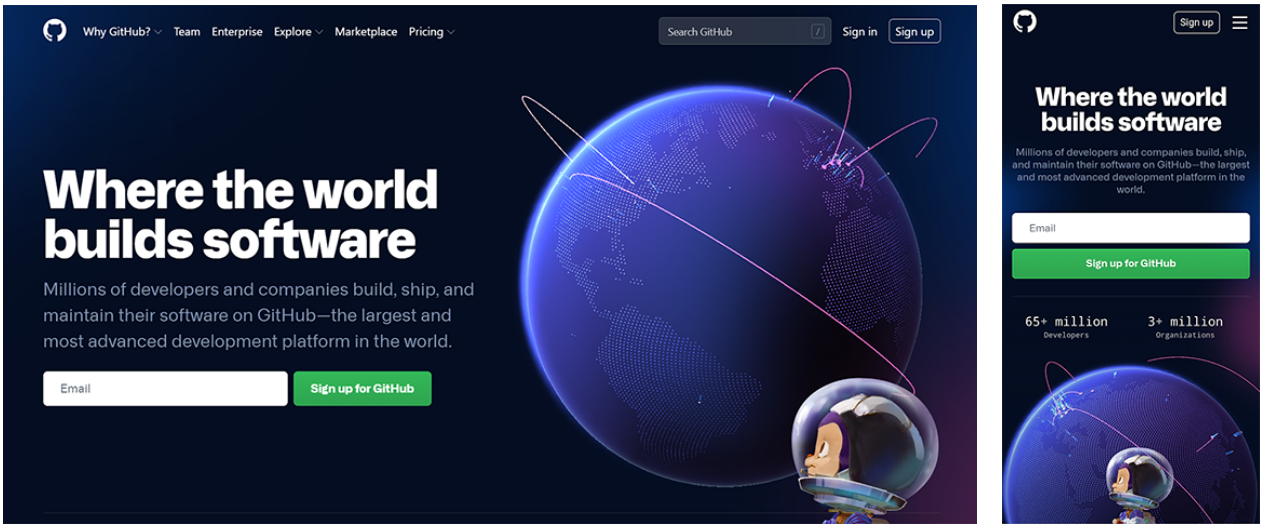
```
<style>
    .box {
        width: 100px;
        height: 3000px;
        border: 1px solid;
        display: inline-block;
        background-color: skyblue;
        border-color: blue;
    }

    .red-box {
        height: 100px;
        background-color: pink;
        border-color: red;
    }

    #pos-element {
        /* Your CSS code here */
    }
</style>

<div class="box">1</div>
<div class="box red-box" id="pos-element">2</div>
<div class="box">3</div>
```

```
25 <div class="box">4</div>
```

여담으로, position 속성을 다루다 보면 요소 간에 겹침이 발생하여 특정 요소를 다른 요소 위에 오도록 할 필요가 있다. 이때 z-index 속성을 이용하면 요소 간의 쌓임 순서를 조절할 수 있다. z-index의 값은 정수(integer)이며, 0이 기본값이고, 값이 높은 요소가 위에 쌓인다.

2.6 Responsive Web

지금까지 CSS를 이용하여 문서를 디자인할 때 주로 가로 폭이 넓은, PC의 웹 브라우저를 기준으로 작업해왔다. 그러나 모든 사용자가 항상 웹 페이지를 PC와 같이 가로 폭이 넓은 디바이스에서 열람하는 것은 아니다. 휴대폰과 같은 모바일 디바이스는 가로 폭이 좁아서 PC를 기준으로 설계한 웹 페이지는 모바일에서 열람했을 때 가독성이 심각하게 저하될 수 있다. HTML 요소들이 의도와는 다르게 배치될 수 있고, 이를 방지하고자 요소의 너비 등을 정해진 값으로 딱딱하게 정하면 모바일 디바이스에서는 좌우로 스크롤하면서 웹 페이지를 읽어야 한다. 웹 페이지는 가급적 좌우 방향으로는 움직이지 않고, 상하 방향으로만 움직여 정보를 전달하게끔 설계하였을 때 가독성이 좋은데, 위와 같이 PC를 기준으로 웹 페이지를 설계하면 가독성이 매우 떨어진다. 따라서, 웹 페이지가 렌더링되는 화면의 크기에 따라 디자인이 바뀌는, **반응형 웹 페이지(Responsive Web)**를 디자인할 필요가 있다.



(a) Desktop Browser **(b)** Mobile Browser

Figure 2.5 Webpage view of GitHub homepage

Viewport

반응형 웹 페이지는 웹 페이지가 렌더링되는 화면, 즉 뷰포트에 따라 디자인이 바뀌므로 먼저 뷰포트를 설정해주어야 한다. 뷰포트의 크기는 **Code 2.17**과 같이 HTML 문서의 `meta` 태그에서 설정한다.

Code 2.17 Viewport setting

```
1 <meta name="viewport" content="width=device-width, initial-scale=1" >
```

Code 2.17의 코드는 페이지의 너비를 디바이스 화면의 너비로 설정(`width=device-width`)하고, 원래 페이지의 크기를 그대로 사용(`initial-scale=1`)하는 코드이다. 이 코드가 가장 기본적인 설정이며, 추가 설정이 가능하지만 이 교재에서는 기본 설정만 사용한다. 이러한 설정은 1.1절의 **Code 1.1**에서도 확인할 수 있다.

뷰포트는 개발자 도구를 이용하여 조절할 수도 있다. Chrome은 개발자 탭에서 Elements 탭 왼쪽에, Firefox는 오른쪽 상단에 모바일 디바이스와 유사한 아이콘(Toggle device toolbar)을 눌러 뷰포트를 조절할 수 있다.

Media Query

이제 CSS를 이용하여 웹 페이지를 반응형으로 디자인해보자. 반응형 웹페이지를 디자인하기 위해서 `@media query`라는 구문을 사용한다.

Code 2.18 Media query statement

```
@media only screen and (min-width: 800px) {
    /* CSS code goes here */
}
```

Code 2.18은 `@media query` 구문의 예시이다. 중괄호 내부에는 일반적인 CSS 코드를 작성하고, `@media query` 구문은 중괄호 내부의 디자인을 적용할 조건을 제시한다. 코드에서 `min-width: 800px`은 화면의 "너비가 800px 이상"이라는 조건이며, 이 외에도 다양한 조건을 제시할 수 있으나 **Code 2.18**이 가장 기본적인 형태이다.

`@media query` 구문을 이용하여 반응형 웹페이지를 디자인할 때 너비가 작은 화면에서 큰 화면의 순서로 작성하는 것이 원칙이다. 예를 들어, 뷰포트의 너비가 (1) 400px 미만일 때, (2) 400px 이상 800px 미만일 때, (3) 800px 이상 1200px 미만일 때, (4) 1200px 이상일 때의 디자인을 각각 다르게 설계하는 상황을 가정해보자. 먼저 (1)의 디자인을 먼저 작성하고, (2)의 디자인 중 (1)과 다른 부분을 `@media query` 구문을 이용하여 400px 이상의 뷰포트에 대해 작성한다. 이렇게 작성하면 (1)에서 작성한 디자인 중 (2)에 의해 덮어씌워지지 않는 디자인은 그대로 유지된다. (3), (4)도 마찬가지의 방법으로 작성하여 완성한다.

2.7 CSS Exercises

Exercise 2.1: Text Styling

Code A.1이 **Figure 2.6**과 같이 나타나도록 text-styling.css를 작성하여라.

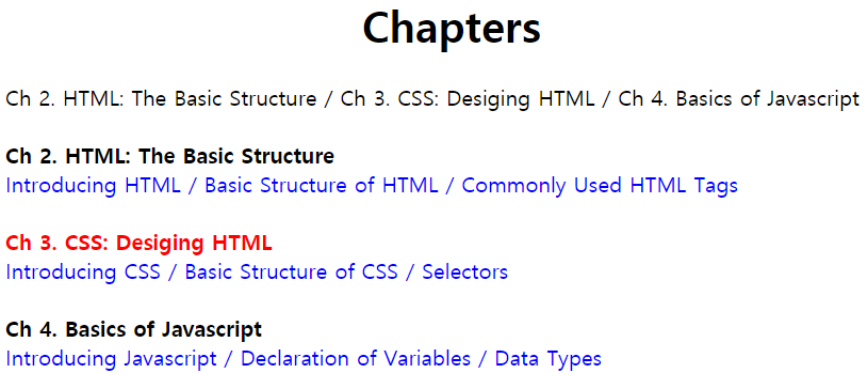


Figure 2.6 Exercise 2.1 example

Exercise 2.2: ARS Buttons

Code A.2가 **Figure 2.7**과 같이 나타나도록 ars-buttons.css를 작성하여라.

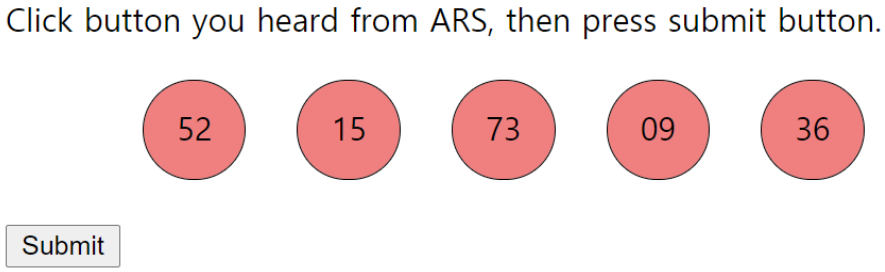


Figure 2.7 Exercise 2.2 example

Exercise 2.3: Scroll Button

Code A.3을 참고하여, **Figure 2.8**과 같이 우측 하단에 고정되어 있으면서, 클릭하였을 때 웹 페이지의 제일 상단과 제일 하단으로 이동하는 버튼을 `#scroll-button` 내부를 수정하고, scroll-button.css를 작성하여 구현하여라. `a` 태그의 `href` 속성값을 HTML 요소의 id로 설정하면 해당 위치로 이동할 수 있으며, 위쪽 삼각형(▲)과 아래쪽 삼각형(▼)의 개체 번호는 각각 9650, 9660이다.

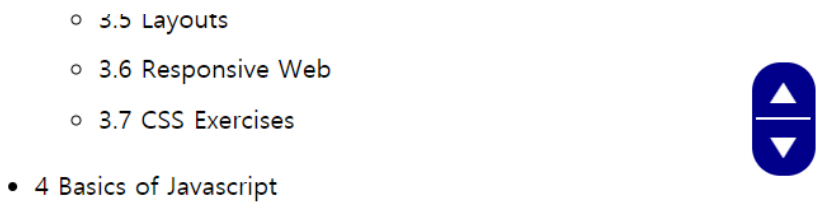


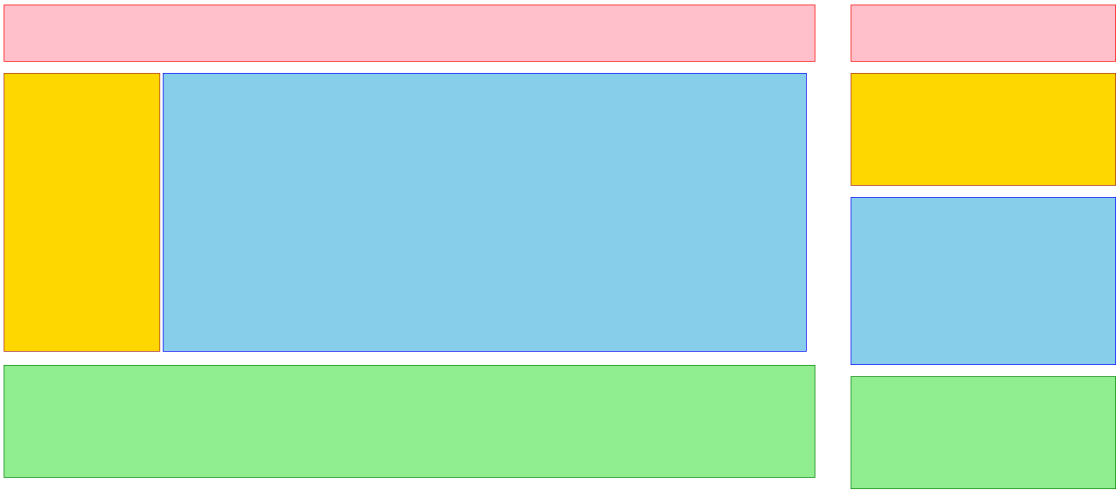
Figure 2.8 Exercise 2.3 example

Exercise 2.4: Responsive Color Page

Code A.4를 참고하여 responsive-color-page.css를 작성하여라. `body` 태그의 배경색이 600px 미만일 때는 `skyblue`, 600px 이상 1200px 미만일 때는 `blue`, 1200px 이상일 때는 `darkblue`이어야 하며, `h1` 태그의 색은 900px 미만일 때는 `white`, 900px 이상일 때는 `yellow`이어야 한다.

Exercise 2.5: Responsive Layout

Code A.5를 참고하여 웹 페이지의 너비가 800px 이상이면 **Figure 2.9a**, 800px 미만이면 **Figure 2.9b**와 같이 렌더링 되도록 responsive-layout.css 파일을 작성하여라.



(a) Wider than 800px **(b)** Narrower than 800px

Figure 2.9 Webpage view of Exercise 2.5