4. CSS 이해하기

1) CSS 소개

- CSS와 HTML : CSS는 간단히 이야기하면 HTML(마크업 언어)을 꾸며주는 표현용 언어입니다. HTML이 문서의 정보, 구조를 설계한다면 CSS는 문서를 보기 좋게 디자인합니다. CSS는 분명히 HTML과는 독립된 다른 언어지만 마크업 문서 자체가 존재하지 않으면 CSS는 무용지물이나 마찬가지입니다. 왜냐하면, CSS는 표현을 위한 언어인데 마크업 문서가 없다면 표현할 대상이 없기 때문입니다. 그래서 CSS는 보통 마크업 언어인 HTML과 같이 묶어서 이야기합니다.

- CSS로 표현한 다양한 웹 사이트들 : 전 세계에 많은 웹 사이트들이 있습니다. 그리고 그것들 모두 HTML 태그를 이용해서 만들어졌고, 자주 사용되는 태그의 개수는 10여 개밖에 되지 않습니다. 결국, 모든 사이트가 비슷한 HTML 태그를 사용해서 만들어졌음에도 불구하고 각각 다양하고 독창적인 디자인으로 표현 가능한 이유가 바로 CSS 덕분입니다. CSS는 문서를 디자인하는 강력한 힘을 가졌습니다.

http://www.csszengarden.com/

2) CSS 문법과 적용

- CSS 문법 : h1{color: yellow; font-size: 2em;}

CSS는 HTML 요소를 꾸며주는 역할을 합니다.

CSS는 꾸밀 대상이 되는 요소와 그에 대한 스타일 내용으로 이루어져 있습니다.

- CSS 구문

선택자(selector) : “h1”

속성(property) : “color”

값(value) : “yellow”

선언(declaration) : "color: yellow", "font-size: 2em"

선언부((declaration block) : "{ color: yellow; font-size:2em; }"

규칙(rule set) : "h1 { color: yellow; font-size:2em; }“

CSS 파일은 여러 개의 규칙으로 이루어져 있습니다. 선택자와 선언부 사이, 선언과 선언 사이는 앞뒤로 개행을 해도 상관이 없습니다. 하지만 속성이름과 속성값 사이에는 개행을 하면 안 됩니다.

[올바른 CSS]

```
h1
{ color: yellow; font-size:2em; }

h1 {
    color: yellow;
    font-size:2em;
}
```

[잘못된 CSS]

```
h1 {
    color:
    yellow;
}
```

- CSS의 속성(Property)과 HTML의 속성(Attribute) : HTML에도 속성이 있고, CSS에도 속성이 있습니다. 두 가지는 전혀 다른 것입니다. HTML의 속성은 영어로 attribute이고, CSS의 속성은 property입니다. 둘 다 한국어로 번역할 때 "속성"이라고 하므로 잘 구분하셔야 합니다.

- CSS 주석

```
/* 주석 내용 */
```

/*
    주석은 여러 줄로도
    선언 할 수 있습니다.
*/
 - CSS의 적용
 1. Inline : Inline은 해당 요소에 직접 스타일 속성을 이용해서 규칙들을 선언하는 방법입니다. 해당 요소에 직접 입력하기 때문에 선택자는 필요하지 않게 되고, 선언부에 내용만 스타일 속성의 값으로 넣어주면 됩니다. Inline 스타일 방식이라고 부릅니다.
  내용
 Inline 스타일 방식은 코드의 재활용이 되지 않기 때문에 자주 사용하지 않습니다.
 2. Internal : Internal은 문서에 <style>을 활용한 방법입니다. <style>은 <head>내부에 들어가며 <style>안에 스타일 규칙이 들어갑니다.
 <style> div {color: red;} </style>
 위의 코드로 모든 에 같은 스타일을 줄 수 있습니다. 하지만 이것도 한계가 있습니다. 많은 페이지가 있는 경우에는 모든 페이지에 저마다의 규칙을 선언해줘야 합니다. 페이지가 많고 스타일 규칙 내용이 많아지면 결코 쉬운 일은 아닙니다.
 3. External : External은 외부 스타일 시트 파일을 이용한 방법입니다. 외부 스타일 시트는 스타일 규칙들을 별도의 외부 파일을 만들어 넣는 방식입니다. 외부 파일은 확장자가 .css가 되며 css 파일이라고 부릅니다.
 div {color: red;}
 우선 CSS 파일을 하나 만들고 스타일 규칙을 선언합니다. 그다음 <link>을 이용해서 CSS 파일을 연결하면 됩니다.
 <link rel="stylesheet" href="css/style.css">
 <head> 내부에 <link>를 선언한 후 href 속성을 이용해 CSS 파일의 경로를 적습니다. rel 속성은 연결되는 파일이 문서와 어떤 관계인지를 명시하는 속성으로, CSS 파일은 'stylesheet' 라고 적어야 합니다. 외부 스타일 시트 방식으로 스타일을 선언하면 많은 페이지가 있더라도 이 한 줄로 모든 페이지에 같은 스타일을 적용할 수 있습니다. 또한, 수정이 필요할 때도 CSS 파일을 수정하면 연결된 모든 페이지에 반영할 수 있습니다. 외부 스타일 시트 방식은 파일 관리가 편하면서도 용량이 작기 때문에 주로 사용되는 방법입니다.
 4. Import : Import는 스타일 시트 내에서 다른 스타일 시트 파일을 불러오는 방식입니다.
 @import url("css/style.css");
 <style> 내부 상단이나 외부 스타일 시트 파일 상단에 선언하는데 성능상 좋지 않아서 거의 쓰이지 않습니다. 이에 본 강의에서는 다루지 않습니다. 궁금하신 분들은 따로 찾아보시길 바랍니다.
<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
    <meta charset="UTF-8">
    <title>css</title>
    <link rel="stylesheet" href="./style.css">
</head>
<body>
    <p>Hello, CSS</p>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa debitis facere fuga
```

laboriosam placeat provident, quibusdam quidem quo sapiente totam?</p>
</body>
</html>
https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_is_structured
https://www.w3schools.com/css/css_howto.asp

3) 선택자1

- 요소 선택자: 선택자 중에 가장 기본이 되는 선택자이며, 태그 선택자라고도 합니다.

```
h1 { color: yellow; }
h2 { color: yellow; }
h3 { color: yellow; }
h4 { color: yellow; }
h5 { color: yellow; }
h6 { color: yellow; }
```

요소 선택자는 위 코드처럼 선택자 부분에 태그 이름이 들어갑니다. 문서 내에 선택자에 해당하는 모든 요소에 스타일 규칙이 적용됩니다.

- 그룹화 : 선택자는 쉼표를 이용해서 그룹화를 할 수 있습니다.

```
h1, h2, h3, h4, h5, h6 { color: yellow; }
```

위 코드는 요소 선택자의 예제 코드와 같은 코드입니다. 그리고 전체 선택자 라고 불리는 간단한 선택자도 있습니다.

```
* { color: yellow; }
```

*(별표, asterisk) 기호로 문서 내에 모든 요소를 선택할 수 있습니다. 이렇게 선언하면, 한 번의 선언만으로 문서 내에 모든 요소에 스타일 규칙이 적용됩니다. 전체 선택자는 매우 편리하지만, 성능에 좋지 않으므로 될 수 있으면 사용을 지양합니다. 선택자뿐만 아니라 선언들도 그룹화가 가능합니다.

```
h1 { color: yellow; font-size: 2em; background-color: gray; }
```

그리고 마지막으로 선택자와 선언이 동시에 그룹화도 가능합니다.

```
h1, h2, h3, h4, h5, h6 { color: yellow; font-size: 2em; background-color: gray; }
```

<코드실습 >

```
<!DOCTYPE html>
<html lang="ko">
<head>
	<meta charset="UTF-8">
	<title>css</title>
	<style>
		h1 { color: yellow; font-size: 2em; background-color: gray; }
	</style>
</head>
<body>
	<h1>Hello, CSS</h1>
	<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Culpa debitis facere fuga laboriosam placeat provident, quibusdam quidem quo sapiente totam?</p>
</body>
</html>
```

https://developer.mozilla.org/en-US/docs/Glossary/CSS_Selector

https://www.w3schools.com/cssref/css_selectors.asp

4) 선택자2

- class 선택자 : 요소에 구애받지 않고 스타일 규칙을 적용할 수 있는 가장 일반적인 방법은 class 선택자를 활용하는 것입니다. class 선택자를 사용하기 위해서는 HTML을 수정해 class 속성을 추가해야 합니다. class 속성은 글로벌 속성이므로 어느 태그에서도 사용할 수 있습니다. class 속성에 값을 넣게 되면, class 선택자를 이용해서 해당 요소에 스타일 규칙을 적용할 수 있습니다.

```
.foo { font-size: 30px; }
<p class="foo"> ... </p>
```

<p>의 class 속성의 값으로 "foo"라는 값을 넣었다면, CSS에서 그 값("foo")을 선택자로 지정하면 됩니다. 클래스 선택자를 쓸 때는, 맨 앞에 .(마침표)를 찍어주셔야 합니다. 이렇게 되면 어느 요소든지 class 속성값이 "foo"로 선언된 요소가 있다면 해당 스타일 규칙을 적용받게 됩니다. 그럼 이제 클래스 선택자를 이용해서 인트로에 제시했던 퀴즈를 풀어보겠습니다.

```
.html { color: purple; }
.css { text-decoration: underline; }
<dl>
    <dt class="html">HTML</dt>
    <dd><span class="html">HTML</span>은 문서의 구조를 나타냅니다.</dd>
    <dt class="css">CSS</dt>
    <dd><span class="css">CSS</span>는 문서의 표현을 나타냅니다.</dd>
</dl>
```

- 다중 class : class 속성은 꼭 하나의 값만 가질 수 있는 것은 아닙니다. 공백으로 구분하여 여러 개의 class 값을 넣을 수 있습니다.

```
.foo { font-size: 30px; }
.bar { color: blue; }
<p class="foo bar"> ... </p>
```

<p>에 class 속성에 "foo" 와 "bar" 2개의 값을 넣었습니다. 그리고 foo class 선택자에는 폰트의 크기를 30px로, bar class 선택자에는 글자를 파란색으로 적용하는 스타일 규칙이 선언되어있습니다. 그렇게 되면 이 <p>에는 2개의 규칙이 모두 적용이 됩니다.

- id 선택자 : id 선택자는 class 선택자와 비슷합니다. 선택자를 쓸 때는, .(마침표) 기호 대신 #(해시) 기호를 써주시면 되고, 요소에는 class 속성 대신 id 속성만 써주면 됩니다.

```
#bar { background-color: yellow; }
<p id="bar"> ... </p>
```

이 <p>는 id 선택자의 스타일 규칙이 적용됩니다.

- class 선택자와의 차이점 : 기호가 아닌 #기호 사용, 태그의 class 속성이 아닌 id 속성을 참조, 문서 내에 유일한 요소에 사용, 구체성

가장 큰 차이점은 class와 달리 id는 문서 내에서 유일해야 한다는 점입니다. 클래스 선택자는 여러 요소에 같은 클래스를 넣고 같은 규칙을 적용 할 수 있었습니다. 그리고 그것이 클래스 선택자의 장점이기도 합니다. 하지만 id 속성값은 문서 내에 유일하게 사용이 되어야 합니다. 결국, id 선택자로 규칙을 적용할 수 있는 요소는 단 하나뿐입니다. 그리고 마지막으로 구체성의 값이 다릅니다.

<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
```

```
        <meta charset="UTF-8">
        <title>css</title>
        <style>
                .item { background: gray; }
                .a { color: yellow; }
                .b { color: blue; }
                #special { color: red; }
        </style>
</head>
<body>
        <ul>
                <li class="item a">first</a>
                <li class="item b">second</a>
                <li class="item" id="special">third</a>
        </ul>
</body>
</html>
```

https://css-tricks.com/the-difference-between-id-and-class/

5) 선택자 3

- 선택자의 조합

```
/* 요소와 class의 조합 */
p.bar { ... }

/* 다중 class */
.foo.bar { ... }

/* id와 class의 조합 */
#foo.bar { ... }
```

첫 번째는 요소와 클래스를 조합한 경우입니다. 이 경우에는 <p>이면서 class 속성에 bar가 있어야 적용됩니다.

두 번째는 다중 클래스의 경우입니다. 이 경우에는 class 속성에 foo와 bar가 모두 있어야 적용됩니다.

마지막은 id와 class를 조합한 경우입니다. 이 경우에는 id가 foo이며 class가 bar인 요소에 적용됩니다.

- 속성 선택자 : 단순 속성으로 선택

```
p[class] { color: silver; }
p[class][id] { text-decoration: underline; }
<p class="foo">Hello</p>
<p class="bar">CSS</p>
<p class="baz" id="title">HTML</p>
```

속성 선택자는 대괄호를 이용해서 선언하며 대괄호 안에 속성 이름이 들어갑니다. 요소에 해당 이름의 속성이 있다면 해당 사항이 적용됩니다. 위 CSS 코드는 요소 선택자와의 조합으로 이루어진 코드입니다. 첫 번째는 <p>이면서 class 속성이 있는 요소이면 color: silver 규칙이 적용됩니다. 두 번째는 <p>이면서 class 속성과 id 속성이 함께 있어야 text-decoration: underline 규칙이 적용됩니다. 바로 위 HTML 코드에는 3개의 <p>가 있습니다. 그렇다면 이 3개의 <p>에는 각자 어떤 스타일이 적용될까요? 먼저 예측

을 하시고 직접 실습을 하는 게 좋습니다. p[class] 선택자의 규칙은 class 속성만 존재하면 적용이 되기 때문에 3가지 요소 모두에 적용됩니다. p[class][id] 선택자의 규칙은 class 속성과 id 속성 모두 있는 요소만 해당하기 때문에 마지막 요소에만 적용됩니다. 두 규칙 모두 속성의 값은 상관하지 않습니다.

: 정확한 속성값으로 선택

정확한 속성값으로 선택은 제목 그대로 속성의 값으로 요소를 선택합니다. 선택자는 대괄호 안에 속성 이름과 속성값을 다 적으면 됩니다.

```
p[class="foo"] { color: silver; }
p[id="title"] { text-decoration: underline; }
```

p[class="foo"]는 <p>이면서 class 속성의 값이 foo이면 적용되고, p[id="title"]는 <p> 이면서 id 속성의 값이 title이면 적용됩니다.

: 부분 속성값으로 선택

부분 속성값으로 선택은 속성 이름과 속성값 사이에 사용되는 기호에 따라 동작이 조금 다릅니다.

[class~="bar"] : class 속성의 값이 공백으로 구분한 "bar" 단어가 포함되는 요소 선택

[class^="bar"] : class 속성의 값이 "bar"로 시작하는 요소 선택

[class$="bar"] : class 속성의 값이 "bar"로 끝나는 요소 선택

[class*="bar"] : class 속성의 값이 "bar" 문자가 포함되는 요소 선택

```
<p class="color hot">red</p>
<p class="cool color">blue</p>
<p class="colorful nature">rainbow</p>
p[class~="color"] { font-style: italic; }
p[class^="color"] { font-style: italic; }
p[class$="color"] { font-style: italic; }
p[class*="color"] { font-style: italic; }
```

위의 코드에서는 모두 class 속성값으로 "color"를 선택합니다. 요소 순서대로 기호에 따라 규칙이 적용되는 결과는 다음과 같습니다.

```
p[class~="color"] { font-style: italic; } /* 1, 2번째 요소 */
p[class^="color"] { font-style: italic; } /* 1, 3번째 요소 */
p[class$="color"] { font-style: italic; } /* 2번째 요소 */
p[class*="color"] { font-style: italic; } /* 1, 2, 3번째 요소 */
```

<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
	<meta charset="UTF-8">
	<title>css</title>
	<style>
		p[class$="color"] { font-style: italic; }
	</style>
</head>
<body>
	<p class="color hot">red</p>
	<p class="cool color">blue</p>
	<p class="colorful nature">rainbow</p>
</body>
```

</html>
https://www.w3schools.com/css/css_attribute_selectors.asp
https://developer.mozilla.org/ko/docs/Web/CSS/Attribute_selectors

6) 문서 구조 관련 선택자

- 문서 구조의 이해 : 선택자와 문서의 관계를 이해하기 위해 먼저 어떻게 문서가 구조화되는지를 다시 한 번 살펴보겠습니다.

```
<html>
<body>
    <div>
        <h1><span>HTML</span>: Hyper Text Markup Language</h1>
    </div>
    <p>HTML과 CSS와 JAVASCRIPT를 이용해서 멋진 웹 사이트를 제작할 수 있습니다.</p>
</body>
</html>
```

부모와 자식 : 부모 요소는 그 요소를 포함하는 가장 가까운 상위 요소로, 그 요소의 부모 요소는 단 하나뿐입니다. 자식 요소는 부모 요소와 반대라고 생각하면 되며 자식 요소는 여러 개일 수도 있습니다.

<body>의 부모 요소: <html> ↔ <html>의 자식 요소: <body>
의 부모 요소: <body> ↔ <body>의 자식 요소: , <p>
<h1>의 부모 요소: ↔ 의 자식 요소: <h1>
의 부모 요소: <h1> ↔ <h1>의 자식 요소:
<p>의 부모 요소: <body> ↔ <body>의 자식 요소: , <p>

조상과 자손 : 조상과 자손의 관계는 부모와 자식의 관계와 비슷합니다. 정확히 얘기하면 부모와 자식의 관계를 포함한 확장된 관계라고 생각하면 됩니다. 조상 요소는 그 요소를 포함하는 모든 요소로, 부모 요소를 포함하여 여러 개일 수도 있습니다. 자손 요소는 그 반대로, 그 요소가 포함하고 있는 모든 요소가 자손 요소입니다.

<body>의 조상 요소: <html> ↔ <html>의 자손 요소: <body>, , <h1>, , <p>
의 조상 요소: <html>, <body> ↔ <body>의 자손 요소: , <h1>, , <p>
<h1>의 조상 요소: <html>, <body>, ↔ 의 자손 요소: <h1>,
의 조상 요소: <html>, <body>, , <h1> ↔ <h1>의 자손 요소:
<p>의 조상 요소: <html>, <body> ↔ <body>의 자손 요소: , <h1>, , <p>

보통 문서의 요소들은 모두 이처럼 어느 요소의 자식(자손) 요소이자 부모(조상) 요소가 되는 경우가 많습니다.

형제 : 부모와 자식, 조상과 자손 말고도 형제 관계도 있습니다. 같은 부모를 가지고 있는 요소들은 서로 형제 관계에 있습니다. 위 코드에서는 , <p>가 형제 요소입니다. 형제 관계 중에는 인접한 관계도 있습니다. 형제 관계에 있는 요소 중 바로 뒤에 이어 나오는 요소를 인접해 있다고 합니다. 여기서 <p>가 에 인접한 형제 요소가 됩니다. 문서의 구조는 흔히 가계도나 조직도의 관계와 비슷하다고 생각하면 이해하기가 쉽습니다.

- 문서 구조 관련 선택자 : 문서 구조를 이용한 선택자는 3가지 있습니다.

자손 선택자 : 자손 선택자는 선택자 사이에 아무 기호 없이 그냥 공백으로 구분을 합니다.

```
div span { color: red; }
```

이 선택자는 의 자손 요소인 를 선택하는 선택자 입니다.

자식 선택자 : 자식 선택자는 선택자 사이에 닫는 꺽쇠 기호(>)를 넣습니다. 꺽쇠 기호와 선택자 기호 사이에는 공백은 있거나 없어도 상관이 없습니다.

```
div > h1 { color: red; }
```

이 선택자는 의 자식 요소인 <h1>를 선택하는 선택자입니다.

인접 형제 선택자 : div + p { color: red; }
인접 형제 선택자는 선택자 사이에 + 기호를 넣습니다. 자식 선택자와 마찬가지로 공백은 있거나 없어도 상관이 없습니다. 인접 형제 선택자는 형제 관계이면서 바로 뒤에 인접해 있는 요소를 선택하는 선택자입니다.

```
/* body 요소의 자식인 div 요소의 자손인 table 요소 바로 뒤에 인접한 ul 요소 선택! */

body > div table + ul { ... }
```

문서 구조 관련 선택자는 더 복잡하게 사용할 수 있습니다. 유의할 점은 요소들이 많이 나열되어 있더라도 제일 우측에 있는 요소가 실제 선택되는 요소라는 것입니다.

https://www.w3schools.com/cssref/sel_gen_sibling.asp
https://developer.mozilla.org/en-US/docs/Web/CSS/Adjacent_sibling_combinator
https://developer.mozilla.org/ko/docs/Web/CSS/General_sibling_selectors
https://developer.mozilla.org/en-US/docs/Web/CSS/Child_combinator
https://developer.mozilla.org/en-US/docs/Web/CSS/Descendant_combinator
https://www.w3schools.com/cssref/sel_gen_sibling.asp

7) 가상 선택자 1

- 가상 클래스(pseudo class) : 가상 클래스는 미리 정의해놓은 상황에 적용되도록 약속된 보이지 않는 클래스입니다. 우리가 직접 요소에 클래스를 선언하는 것은 아니고, 약속된 상황이 되면 브라우저 스스로 클래스를 적용해줍니다.

가상 클래스가 없다면 이런 과정을 거치게 됩니다.

1. 임의의 클래스 선택자를 선언해 특정 스타일 규칙을 만든다.
2. p 요소에 커서 올라가면 p 요소에 클래스를 집어넣는다.
3. p 요소에서 커서가 빠지면 p 요소에 클래스를 삭제한다.

여기서 2, 3번은 동적으로 변화되어야 하는데, HTML과 CSS는 정적인 언어이기 때문에 처리할 수 없습니다. 어쩔 수 없이 다른 언어를 사용해야 하는데, 이는 개발 비용이 들어가는 일입니다. 그래서 CSS에서는 흔하게 사용되는 패턴에 대해서 미리 정의해놓고, 가상 클래스로 제어할 수 있게 했습니다.

```
:pseudo-class {
    property: value;
}
```

가상 클래스는 :(콜론) 기호를 써서 나타냅니다. 가상 클래스를 이용하면 아래의 경우에도 CSS만으로 처리가 가능하므로 훨씬 효율적입니다. ":hover 가상 클래스 선택자를 이용해서 스타일 규칙을 만든다. (hover는 마우스 커서가 올라갔을 때 적용이 되도록 정의되어 있습니다.)" 가상 클래스에는 여러 가지가 있습니다. https://developer.mozilla.org/ko/docs/Web/CSS/Pseudo-classes

- 문서 구조와 관련된 가상 클래스 : 문서 구조와 관련된 가상 클래스는 first-child와 last-child 가상 클래스 선택자입니다.

:first-child : 첫 번째 자식 요소 선택
:last-child : 마지막 자식 요소 선택

```
<ul>
    <li>HTML</li>
    <li>CSS</li>
    <li>JS</li>
</ul>
li:first-child { color: red; }
```

li:last-child { color: blue; }
첫 번째 와 마지막 에 가상 클래스가 적용됩니다. 실제 에는 class 속성이 없지만 내부적으로 가상 클래스가 적용되어 마치 아래의 코드와 같이 동작하게 됩니다.

```
<ul>
    <li class="first-child">HTML</li>
    <li>CSS</li>
    <li class="last-child">JS</li>
</ul>
```

- 앵커 요소와 관련된 가상 클래스 : 앵커 요소와 관련된 가상 클래스로는 :link와 :visited가 있습니다.
:link : 하이퍼 링크이면서 아직 방문하지 않은 앵커
:visited : 이미 방문한 하이퍼링크를 의미
하이퍼 링크는 앵커 요소에 href 속성이 있는 것을 의미합니다.

```
a:link { color: blue; }
a:visited { color: gray; }
```

- 사용자 동작과 관련된 가상 클래스 : 이 클래스들도 <a>에 주로 많이 쓰입니다. <a>에만 쓸 수 있는 것은 아니며, 이 조건에 맞는 상황이 되는요소들을 다 사용이 가능합니다.
:focus: 현재 입력 초점을 가진 요소에 적용
:hover: 마우스 포인터가 있는 요소에 적용
:active: 사용자 입력으로 활성화된 요소에 적용

```
a:focus { background-color: yellow; }
a:hover { font-weight: bold; }
a:active { color: red; }
```

:focus는 현재 입력 초점을 가진 요소에 적용됩니다.
focus(초점)는 지금 현재 선택을 받는 것을 의미합니다. 예를 들면, 입력 폼 요소에 텍스트를 입력하려고 마우스 클릭해서 커서를 입력 폼 위에 올려놓으면 그때 입력 폼 요소가 초점을 받는 상태입니다. 또 키보드의 탭 키를 이용해서 요소를 탐색하다 보면 링크나 버튼에 점선 테두리가 이동하는 것을 볼 수 있는데, 점선 테두리가 위치하는 것도 초점을 받은 상태입니다.
:hover는 마우스 커서가 있는 요소에 적용됩니다. (마우스를 올렸을 때를 의미합니다.)
:active는 사용자 입력으로 활성화된 요소를 의미하는데, <a>를 클릭할 때 또는 <button>를 눌렀을 때처럼 순간적으로 활성화됩니다.

<코드실습>

```
<!DOCTYPE html>
<html lang="ko">
<head>
        <meta charset="UTF-8">
        <title>css</title>
        <style>
                a:focus { background-color: yellow }
                a:hover { font-weight: bold }
                a:active { color: red }
        </style>
</head>
<body>
        <a href="http://www.naver.com">네이버</a>
```

```
        <a href="http://www.google.com">구글</a>
        <a href="http://www.daum.net">다음</a>
</body>
</html>
```

https://www.w3schools.com/css/css_pseudo_classes.asp

8) 가상 선택자 2

- 가상 요소(pseudo element) : 가상 요소는 HTML 코드에 존재하지 않는 구조에 스타일을 부여할 수가 있습니다. 가상 요소도 가상 클래스처럼 문서 내에 보이지 않지만, 미리 정의한 위치에 삽입되도록 약속이 되어있습니다. 선언 방법은 가상 클래스와 같이 콜론을 사용하며, CSS3부터는 가상 클래스와 가상 요소를 구분하기 위해 가상 요소에는 ::(더블 콜론) 기호를 사용하기로 했습니다. 하지만, 하위 브라우저에서 ::문법을 지원하지 않는 문제가 있으므로 상황에 따라 : 기호를 사용하셔야 합니다.

```
::pseudo-element {
    property: value;
}
```

https://developer.mozilla.org/ko/docs/Web/CSS/Pseudo-elements

:before : 가장 앞에 요소를 삽입

:after : 가장 뒤에 요소를 삽입

:first-line : 요소의 첫 번째 줄에 있는 텍스트

:first-letter : 블록 레벨 요소의 첫 번째 문자

```
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
p::before { content: "###" }
p::after { content: "!!!" }
p::first-line { ... }
p::first-letter { ... }
```

before와 after 가상 요소는 애초에 내용이 없는 상태로 생성되기 때문에 내용을 넣기 위해 content 속성을 이용해야 합니다. 실제 HTML 코드에는 나타나지 않지만, before와 after가 어떻게 동작하는지 이해를 돕기 위해 코드를 아래와 같이 변경했습니다.

```
<p>
    <before>###</before>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    <after>!!!</after>
</p>
```

눈에 보이지 않지만, 내부에서 이처럼 요소가 생성됩니다. first-line과 first-letter도 마찬가지로 아래 코드와 같은 것으로 생각하시면 됩니다.

```
<p>
    <first-letter>L</first-letter>orem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
</p>
<p>
    <!-- 모니터 가로 해상도에 따라 요소가 포함하는 내용이 변동됩니다. -->
    <first-line>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiu ..(..어딘가
```

쯤..) </first-line>... unt ut labore et dolore magna aliqua.

https://code.tutsplus.com/ko/tutorials/the-30-css-selectors-you-must-memorize—net-16048
https://developer.mozilla.org/en-US/docs/Web/CSS/content
https://www.w3schools.com/cssref/pr_gen_content.asp

9) 구체성

- 구체성 : 요소를 선택하는 데는 여러 방법이 있습니다. 따라서 서로 다른 선택자를 이용해 같은 요소를 선택할 수도 있습니다.

```
h1 { color: red; }
body h1 { color: green; }
```

서로 다른 규칙들이 상반된 스타일을 가지고 있다. 두 규칙은 모두 <h1>을 선택하게 됩니다. 하지만, 두 규칙이 지정하는 스타일은 서로 다릅니다. <h1>에는 color: green이 적용되는데 이는 구체성과 연관이 있습니다. 선택자에는 어떤 규칙이 우선으로 적용되어야 하는지에 대해 정해진 규칙이 있습니다. 이 규칙을 '구체성'이라고 합니다. 구체성은 선택자를 얼마나 명시적으로(구체적으로) 선언했느냐를 수치화한 것으로, 구체성의 값이 클수록 우선으로 적용이 됩니다.

0, 0, 0, 0

구체성은 4개의 숫자 값으로 이루어져 있습니다. 값을 비교할 때는 좌측에 있는 값부터 비교하며, 좌측 부분의 숫자가 클수록 높은 구체성을 갖습니다. 구체성은 아래의 규칙대로 계산됩니다.

0, 1, 0, 0 : 선택자에 있는 모든 id 속성값
0, 0, 1, 0 : 선택자에 있는 모든 class 속성값, 기타 속성, 가상 클래스
0, 0, 0, 1 : 선택자에 있는 모든 요소, 가상 요소
전체 선택자는 0, 0, 0, 0을 가진다.
조합자는 구체성에 영향을 주지 않는다. (>, + 등)

```
h1 { ... }  /* 0,0,0,1 */
body h1 { ... }  /* 0,0,0,2 */
.grape { ... }  /* 0,0,1,0 */
*.bright { ... }  /* 0,0,1,0 */
p.bright em.dark { ... }  /* 0,0,2,2 */
#page { ... }  /* 0,1,0,0 */
div#page { ... }  /* 0,1,0,1 */
```

선택자의 구체성 값을 잘 알아야 많은 스타일 규칙들을 정의할 때 의도하지 않은 일이 생기지 않습니다.

- 인라인 스타일

p#page { color: red; } 0, 1, 0, 1의 구체성을 가지는 선택자로 스타일 선언
Lorem impusm dolor sit. 요소에 직접 인라인 스타일 방식으로 스타일을 선언

결과적으로 에는 color: blue가 적용됩니다. 인라인 스타일의 구체성 값은 1, 0, 0, 0이며 규칙들 중 가장 큰 구체성을 갖기 때문입니다.

- Important : important 키워드는 별도의 구체성 값은 없지만, 모든 구체성을 무시하고 우선권을 갖습니다. important 키워드는 속성값 뒤 한 칸 공백을 주고 느낌표 기호와 함께 씁니다.

```
p#page { color: red !important; }
<p id="page" style="color:blue">Lorem impusm dolor sit.</p>
```

에는 important로 인해 color: red가 적용됩니다.
https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance

10) 상속

- 상속되는 속성

```
h1 { color: gray; }
<h1>Hello, <em>CSS</em></h1>
```

은 부모인 <h1>의 color: gray를 상속받습니다. 상속은 자연스러운 현상처럼 보이지만, 모든 속성이 다 상속되는 것은 아닙니다. 아직 속성에 대해 다 배우지는 않았지만, margin, padding, background, border 등 박스 모델 속성들은 상속되지 않는다는 것을 알고 계시면 됩니다. 상속되는 속성들은 보통 상식적으로 구분될만한 속성들이며, 후에 속성들에 대해 배우게 되면 자연스럽게 이해할 수 있습니다.

- 상속되는 속성의 구체성

```
* { color: red; }
h1#page { color: gray; }
<h1 id="page">Hello, <em>CSS</em></h1>
```

전체 선택자를 이용해 color: red를 적용하고 id 선택자를 이용해 color: gray를 선언했습니다. 전체 선택자의 구체성은 0, 0, 0, 0 이며 id 선택자의 구체성은 0,1,0,1입니다. 에는 color: red가 적용되는데 그 이유는 바로 상속된 속성은 아무런 구체성을 가지지 못하기 때문입니다.

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

11) 캐스케이딩(cascading)

```
h1 { color: red; }
h1 { color: blue; }
```

<h1>에는 같은 구체성을 가진 두 규칙이 적용되었습니다. <h1>에는 color: blue가 적용되며 이는 cascading 규칙에 의해 적용된 결과입니다.

- cascading 규칙 :중요도(!important)와 (CSS)출처, 구체성, 선언 순서

CSS 출처는 제작자와 사용자, 그리고 사용자 에이전트(user agent) 경우로 구분합니다. 제작자의 경우는 사이트를 실제 제작하는 개발자가 작성한 CSS를 의미합니다. (대부분이 여기에 해당합니다.) 그리고 사용자의 경우는 웹 페이지를 방문하는 일반 사용자들이 작성한 CSS를 의미합니다. 마지막으로 사용자 에이전트의 경우는 일반 사용자의 환경, 즉 브라우저에 내장된 CSS를 의미합니다. 현재의 브라우저에서는 사용자 스타일을 지원하지 않는 추세이기 때문에 이와 관련해서는 생략하도록 하겠습니다.

모든 스타일은 규칙에 따라 단계적으로 저용된다.

1. 스타일 규칙들을 모아서 중요도가 명시적으로 선언되었는지에 따라 분류합니다.
   중요도가 명시적으로 선언된 규칙들은 그렇지 않은 규칙들보다 우선합니다.
   중요도가 있는 규칙들끼리는 아래 다른 규칙들을 적용받습니다.
2. 스타일 규칙들을 출처에 따라 분류합니다.
   제작자 스타일 규칙이 사용자 에이전트 스타일 규칙보다 우선합니다.
3. 스타일 규칙들을 구체성에 따라 분류합니다.
   구체성이 높은 규칙들이 우선합니다.
4. 스타일 규칙들을 선언 순서에 따라 분류합니다.
   뒤에 선언된 규칙일수록 우선합니다.

```
<p id="bright">Hello, CSS</p>
p#bright { color: silver; }
p { color: red; }
```

구체성에 따라 color: silver가 적용

p { color: silver; }
p { color: red; }
선언 순서에 따라 color: red가 적용
https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance
https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade

12) 선택자 정리

https://www.w3schools.com/cssref/css_selectors.asp