

1 HTML5의 탄생과 의미

▶ HTML이란

- ▣ Hyper Text Markup Language 의 약자
- ▣ 웹용 콘텐츠의 구조를 지정하는 컴퓨터 언어
- ▣ HTML은 웹서버에 저장되며 클라이언트 웹 브라우저에 읽혀지고 해석되어 화면에 보여진다.

▶ HTML의 탄생과 발전

- ▣ 1990년 월드 와이드 웹과 함께 탄생
- ▣ 웹의 발전으로 웹 브라우저의 중요성 확대
- ▣ '넷스케이프 네비게이터'와 '마이크로소프트 인터넷 익스플로러'의 시장 점유율 전쟁으로 HTML과 CSS의 비표준 심화
- ▣ 새로운 웹 브라우저가 등장하면서 웹의 표준화 논의
- ▣ 웹 표준을 개발하고 논의하며 제정하는 W3C(World Wide Web Consortium)
- ▣ 1997년 W3C HTML3.2 권고 – 시장 요구에 의해 비표준 HTML 포함
- ▣ 1999년 HTML 4.01 권고 – 시장 요구에 의해 다양한 버전 등장
- ▣ 2000년 XHTML 권고 – XML의 엄격함을 주요 내용으로 한 웹 표준
- ▣ W3C는 XHTML2 개발 추진 하다 돌연 HTML5로 방향 전환

▶ HTML5의 탄생

- ▣ 기존 HTML의 한계
 - 웹 기반 사업과 기술의 발전을 못 따라간 HTML
 - 동적인 웹을 위한 표준화 된 기술을 요구
 - W3C의 XHTML2로의 발전 예고
- ▣ WHATWG((Web Hypertext Application Technology Working Group) 탄생
 - 시장 요구에 부응하지 못하는 W3C에 실망한 웹 관련 업체와 단체가 자체적으로 만든 워킹 그룹
 - 웹 기술과 시장의 요구를 분석하여HTML5 명세서 작업 착수
- ▣ W3C의 HTML5 수용
 - 2008년 W3C HTML5 초안 발표
 - 2009년 XHTML2 개발 중단
 - 2014년 XHTML5 권고 예정
- ▣ HTML5의 특징
 - 기존 HTML과의 호환성 유지
 - 실용적 설계 : 느슨한 문법, 효율적인 추가 요소, 안전한 보안
 - 표현과 내용의 완벽한 분리
 - 플러그인 없이 각종 미디어 처리 및 동적인 작동 : 캔버스
 - 최신 웹 기술 수용 : 지오로케이션, 웹소켓, 웹스토리지, 웹위커 등

2 HTML5 시작하기

▶ Doctype 지정하기

- HTML은 여러 버전이 존재하므로 Doctype을 명시해야 한다.
- 기존 Doctype 은 매우 길고 복잡한 DTD를 명시해야 했었다.
- HTML5의 실용성 원칙으로 인해 짧아졌다.
 - `<!DOCTYPE html>`

▶ HTML 작성 규칙

- HTML의 마크업 명령은 요소라 부른다.
- HTML은 대소문자를 구분하지 않는다.
- 콘텐츠와 구분을 위해서 꺾쇠로 둘러싼다. – 태그
 - `<p>`, `<a>`, `<div>`
- 시작태그와 마침태그로 요소의 범위를 지정한다.
 - `<p>이것은 단락 입니다.</p>`
- 마침태그가 없이 단독으로 사용되는 요소도 있다.
 - `
`, ``, `<meta>` 등
- 요소의 속성은 속성명 = “속성값” 형식으로 기술한다.
 - ``

▶ 이벤트 API

```
<!Doctype html>
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

- `<html>`은 HTML 코드 전체를 감싼다.
- HTML은 `<head>`와 `<body>` 부분으로 나뉜다.
- `<head>`는 메타데이터와 스크립, CSS등이 위치한다.
- `<body>`부분은 콘텐츠가 담기는 곳으로 웹 브라우저에 표시된다.

3 HEAD 설정

▶ 타이틀 지정

- HTML 파일의 제목으로 웹 브라우저 타이틀에 나타난다.
- `<title>웹 페이지 제목</title>`

▶ 문자 인코딩

- 사용하는 텍스트 에디터의 문자 인코딩과 HTML의 문자 인코딩과 동일해야 웹 브라우저에서 올바르게 표시된다.
- 유니코드인 UTF-8로 지정한다.

▶ 메타데이터

- 메타데이터를 기술 하면 웹 검색에 유리하다.
- HTML에 대한 정보를 기록할 수 있다.
- 메타데이터 기술 방법
 - `<meta name="description" content="HTML5와 Javascript 학습콘텐츠">`
 - `<meta name="keywords" content="HTML5, CSS, JavaScript">`
 - `<meta name="author" content="Lee HaeBum">`
 - `<meta name="copyright" content="©2012 Lee HaeBum">`
 - `<meta name="reply-to" content=gomtomi@imacca.com>`
 - `<meta name="date" content="2012-05-30T12:35:20+09:00">`

4 외부 파일 연결

- ▶ HTML과 함께 사용되는 CSS와 자바스크립트는 다른 파일로 분리함이 원칙이다.

▶ 외부 CSS 파일 연결

- `<link rel="stylesheet" href="css/style.css">`

▶ 외부 자바스크립트 파일 연결

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

1. 구조적 마크업의 이해

- 구조적 마크업은 문서의 구조를 정의
- 표현적 마크업은 문서의 외형을 정의
- 구조적 마크업과 표현적 마크업의 혼용
 - 오래된 HTML은 구조적 마크업과 표현적 마크업을 혼용해 사용
 - 현재 웹 표준에서는 표현적 마크업을 HTML에서 퇴출
 - 문서의 구조는 HTML을 사용, 문서의 표현은 CSS를 사용
- 표현적 마크업 사용의 문제점
 - 표현적 요소의 사용은 접근성을 저해
 - 더 높은 유지비용 발생
 - 문서 크기 비대화
- 블록 레벨 요소와 인라인 레벨 요소
 - 블록 레벨 요소: 줄 바꿈이 일어나는 단락 요소
 - 인라인 레벨 요소: 줄 바꿈이 일어나지 않는 행의 일부 요소

2. HTML5의 섹션 요소

- <section> : 보통 제목으로 시작하는 콘텐츠의 의미적 그룹
- <nav> : 문서의 동일 페이지 또는 다른 페이지를 연결하는 네비게이션 링크로 구성되는 섹션
- <article> : 문서나 페이지, 사이트에서 독립적으로 배포 혹은 재사용 할 수 있는 섹션
- <aside> : 문서의 주요 콘텐츠와 별개의 영역 정의, 대체로 사이드바 형태
- <header> : 페이지 또는 섹션의 머릿글 그룹, 제목, 소개, 네비게이션 등이 포함
- 섹션 요소가 아님
- <footer> : 가장 가까운 조상 요소의 푸터, 섹션 요소가 아님
- <h1> ~ <h6> : 섹션의 제목, 헤딩은 명시적 섹션을 생성
- <hgroup> : h1~h6 요소들을 그룹 짓기 위해 사용, <hgroup>의 등급은 포함된 헤딩 요소의 가장 높은 등급
- <address> : 가장 가까운 조상 요소인 article 또는 body 요소의 연락처 정보를 의미

3. 그룹 콘텐츠 요소

- <p> : 문단, p 요소 보다 더 적합한 요소가 있을 때에는 해당 요소를 사용함
- <blockquote> : 다른 소스로부터 가져온 인용 섹션
인용된 소스의 URL 주소가 있다면 cite 속성으로 표시
- <pre> : 형식화된 텍스트 블록을 표현
- <hr> : 문단 레벨에서 주제의 분리, 마침 요소 없이 단독으로 사용
- 목록
 - 순서가 있는 목록: 순서가 바뀌면 의미가 바뀌는 목록
 - 요소로 정의
 - 요소가 목록 아이টে을 정의
 - 순서가 없는 목록: 목록 아이템이 말머리 기호로 시작
 - 요소로 정의

- 요소가 목록 아이템을 정의
- 정의 목록: 사전식 정의를 위해 사용
 - <dl> : 정의 목록 요소
 - <dt> :정의 목록 제목 요소
 - <dd> :정의 목록 데이터(값) 요소
- <figure> :사진, 일러스트, 비디오 등을 표시
- <figcaption> : figure요소 내용에 대한 캡션, <figure> 요소 내부에서 사용
- <div> :스타일 또는 스크립트 목적으로 콘텐츠를 분리하기 위해 사용
- 전역 속성
 - 대부분의 요소에서 속성으로 사용 가능한 전역 속성
 - id 속성은 HTML 코드 내에 유일한 식별자
 - class 속성은 HTML 코드 내에 같은 값을 다수 가질 수 있음
 - class 속성은 css 나 자바스크립트에서 HTML 코드 내 여러 요소에 동일한 스타일 또는 작동을 적용하기 위해 사용
 - title 속성은 요소의 조인 정보를 나타내며 웹 브라우저에서 툴팁으로 표시
- 텍스트 관련 요소 : 인라인 레벨 요소
 - <a> 하이퍼링크
 - href 속성으로 링크 경로를 지정
 - target 속성은 어떤 윈도우에서 링크가 열릴지 결정
 - _self : 현재의 웹 브라우저 창에서 링크가 열림
 - _parent : 현재의 웹 브라우저 창의 부모 창이 있다면 그 부모 창에서 링크가 열림
 - _top : 최상위 웹 브라우저 창에서 링크가 열림
 - _blank : 새로운 웹 브라우저 창을 생성하고 링크가 열림
 - Ii, : 내용을 강조
 - : 내용이 중요함을 나타냄
 - <q> : q 요소는 인용을 나타내는 인라인 요소
 - <cite> : 책, 수필, 시, 대본, 논문, 그림, 연극, 영화 등과 같은 작품의 제목을 언급하거나 참조 했을 때 사용
 - <abbr> : abbr 요소는 약어 및 두문자어를 나타냄
 - title 속성을 이용하여 두문자의 원형을 나타냄
 - <i> : 다른 언어 표시나 영문에서 이탤릭체로 표현하는 속어인용, 분류학 등에 사용
 - :b요소는 의미 있는 중요성을 부가하지 않고 눈에 띄게 하기 위해 사용
 - <sup>, <sub>
 - sup 요소는 위첨자를 나타낸다.
 - sub 요소는 아래첨자를 나타낸다.
 - :인라인 레벨에서 스타일을 적용하거나 스크립트에서 사용하고자 콘텐츠를 분리하는 역할
 -
 : 마침 요소 없이 사용되며 문단 분리가 아닌 단순 줄바꿈을 표시
 - <p> 요소 대용으로 사용하면 안됨
 -
 - img 요소는 이미지를 의미.
 - img 요소는 마침 요소가 없이 단독으로 사용.
 - src 속성에 이미지의 경로를 지정.
 - alt 속성은 이미지를 대체하기 위한 텍스트. 반드시 사용.

1. HTML 표(Table)

➤ HTML 표의 사용시 주의 점

- 표는 접근성이 떨어지므로 표 이외에 더 좋은 표현 방법이 있다면 표를 사용하지 않아야 한다.
- 표를 페이지 레이아웃에 사용하지 말아야 한다.
- 입력 서식을 나타낼 때도 되도록이면 표를 사용하지 말아야 한다.

➤ HTML 표 작성

- table 요소: HTML 표를 의미한다.
- tr 요소: 테이블의 row를 나타낸다.
- th 요소: 테이블 헤더셀을 나타낸다. 제목 셀에 사용한다.
- td 요소: 테이블의 셀(칸)을 의미한다.

➤ 셀 병합

- 가로셀 병합: colspan 속성 사용
- 세로셀 병합: rowspan 속성 사용

➤ 표의 구조화 요소

- 접근성 강화와 데이터 분석을 위해 표를 구조화 한다.
- thead 요소: 표의 제목으로 구성된 row의 집합
표에 두 개 이상의 thead가 정의 되어서는 안된다.
- tbody 요소: 표의 몸 부분으로 표의 내용이 들어간다
여러 개의 tbody설정 가능
- tfoot 요소: 표의 푸터로 구성된 row의 집합
위치와 상관없이 표 마지막에 나타난다.
표에 두 개 이상의 tfoot이 정의 되어서는 안된다
- col 요소: 구조적으로 하나의 column을 대표한다.
caption 요소의 뒤에 thead나 tbody의 앞에 설정한다.
표의 컬럼 갯수 만큼 col 요소를 사용한다.
- colgroup 요소: col 요소의 그룹
구조에 따라 여러개의 colgroup으로 분리할 수 있다.
- scope 속성: 헤더셀의 영향력이 어느쪽으로 향하는지를 지정
scope의 값이 row 또는 col이면 헤더셀은 row 또는 column에 제목
scope의 값이 rowgroup 또는 colgroup이면 헤더셀은 row의 그룹(아래에 있는 여러 줄)
또는 col의 그룹(우측에 있는 여러 컬럼들)의 제목
- caption 요소: 표의 캡션(제목)을 나타낸다.
table 요소의 첫 번째 자식 요소로 가장 먼저 설정되어야 한다.
table이 figure 요소의 유일한 자식요소라면 caption 대신 figcaption요소로 캡션을 표시해야 한다.

2. HTML 서식(Form)

➤ form 요소: HTML 입력 서식을 의미한다.

- action 속성: 서식이 전송되는 서버 파일 URL
- metho 속성: 폼이 전송되는 방식, get 또는 post 값을 가진다.
- input 요소: 다양한 타입을 가지는 입력 데이터 필드
 - <input type="text"> :줄바꿈이 없는 텍스트 입력, 생략가능
 - <input type="password"> :패스워드 입력 폼, 텍스트가 가려진다.
 - <input type="radio"> : 여러 보기 중 하나만 선택하는 라디오 단추
value 속성으로 서버에 전달될 값을 지정
 - <input type="checkbox"> :여러 보기 중 다중 선택이 가능한 체크박스
value 속성으로 서버에 전달될 값을 지정
- select : 팝업 메뉴 방식의 입력 서식
 - option 요소로 메뉴 값을 표시
 - option요소의 select 속성으로 메뉴 중 하나를 미리 선택할 수 있다.
- textarea요소 : 여러줄의 평범한 텍스트를 편집할 수 있는 폼 요소이다.
 - cols속성 : 한 줄단 입력할 수 있는 글자수를 제한
 - rows 속성 :textarea가 몇개의 줄로 보여질지 결정
 - wrap 속성 : 줄바꿈 속성값, soft또는 hard 값을 가진다.
- button요소 : 단추를 만든다.
 - type="submit" :서식을 제출하는 단추
 - type="reset" :서식을 초기화 하는 단추
 - type속성이 정의 되어 있지 않으면 일반적인 단추를 만든다.
- label요소 :서식 입력 요소의 캡션을 나타낸다.
 - for 속성에 label이 적용되는 입력 요소의 아이디를 값으로 지정
 - label 요소 내부에 입력 요소 넣기
- fieldset 요소 :폼 요소들을 공통된 이름으로 그룹화 할때 사용
 - legend요소 :fieldset의 첫 번째 자식요소로 fieldset의 이름을 나타낸다.
- 입력 서식의 공통 속성들
 - name속성 : 서버에 전달될 서식 값의 이름, 반드시 필요
 - required 속성 : 반드시 입력되어야 하는 서식임을 의미
 - placeholder속성 :입력 폼에 짧은 힌트를 나타낸다.
 - maxlength 속성 : 서식 요소에 입력되는 값의 최대 길이 설정
 - autofocus속성 : 폼 로딩이 완료되면 커서가 위치하는 곳 지정

3. HTML 기타 요소들

- mark 요소 : 문서 내에서 다른 문맥과의 관련성을 표시하기 위해 참조 목적으로 표시하거나 하이라이팅 한 텍스트를 나타낸다.
- ruby, rt, rp 요소 : 동아시아의 루비 주석을 표시한다.

- ruby 요소 : ruby 주석을 나타낸다.
- rt 요소 : 루비 주석을 표시한다.
- rp 요소 : 루비 주석을 지원하지 않는 웹 브라우저에서 괄호로 루비 주석을 표시하기 위해 사용한다.
- time 요소 : 시간과 날짜를 표시하는 용도로 사용
 - 정확한 24시간 그리고 그레고리력 날짜만 표시한다.
 - datetime 속성을 사용하여 정확한 날짜와 시간을 명시할 수 있다.
- s 요소 : 더 이상 정확하지 않거나, 관련이 없는 내용을 나타낸다.
- command / menu 요소
 - command 요소 : 사용자가 실행할 수 있는 명령을 나타낸다.
 - menu 요소 : command 목록을 나타낸다.
- details / summary 요소
 - details 요소는 추가적인 정보를 제공하기 위해 정보 일부를 펼쳐 볼 수 있게 하는 요소다.
 - summary 요소는 details 요소 내부에서 요약이나 범례를 나타낸다.

4. HTML5 새로운 서식 요소들

- datalist 요소 : 다른 애플 요소에서 사용하도록 미리 정의 된 option 집합
- 새로운 input 타입 <input type="___">
 - search 타입: 검색을 위한 텍스트 입력 타입
 - tel 타입: 전화번호 입력을 위한 타입
 - url 타입: URL(인터넷 주소) 입력을 위한 타입
 - email 타입: 이메일 입력을 위한 타입
 - 날짜와 시간 입력을 위한 타입들
 - datetime: 국제적 날짜와 시간 입력을 위한 타입
 - date, month, week: 날짜, 달, 주 입력을 위한 타입
 - time: 시간 입력을 위한 타입
 - datetime-local: 지역 날짜와 시간 입력을 위한 타입
 - number 타입: 숫자 입력을 위한 타입
 - max, min, step 속성으로 입력 최대값, 최소값, 증가율을 설정
 - range 타입: 정밀한 숫자 입력이 필요치 않은 입력 타입, 슬라이드 형태
 - color 타입: #으로 시작하는 7글자의 색상 코드 입력을 위한 타입
- progress 요소: 작업의 진척도를 나타낸다.
 - value 속성으로 진척도를 수치로 나타낸다.
 - max 속성으로 완료 수치를 설정한다.
- meter 요소: 알려진 범위 내에서 수치 정도를 나타낸다.
 - 투표율, 디스크 사용현황, 검색빈도 등
 - min, max, value 속성을 이용하여 나타낸다.

5. HTML5 사라진 요소들

➤ 표현 마크업 관련 요소들

- 기존의 표현 마크업 요소들은 CSS사용으로 대체되었다.
- basefont, font, big, blink, spacer, strike, center, tt 등의 요소는 사용하면 안된다.

➤ 프레임 관련 요소들

- 더이상 프레임을 사용하면 안된다.
- iframe 사용은 가능하다.
- frame, frameset, noframes 등의 요소는 이제 사용하면 안된다.

➤ 기타 요소들

- 다른 대체 기술이 개발된 applet이나 bgsound 요소도 사용하면 안된다.
- 기존에 혼란을 일으켰던 acronym와 abbr은 abbr요소로 통일해 사용한다.
- dir, isindex, listing, xmp, plaintext, rb 등의 요소도 더이상 사용하면 안된다.

1. CSS 의 이해

▶ CSS 란?

- **CSS** : Cascading Style Sheets의 약자

• CSS의 특징

- ① **HTML 요소에 스타일 적용**하기 위해 사용
- ② CSS의 장점
 - 디자인을 정의, 관리, 재활용하는데 용이
 - 하나의 CSS 파일은 다수의 HTML에서 사용할 수 있어 체계적이고 경제적
- ③ HTML과 함께 사용되지만 **HTML은 아님**

▶ CSS 작동 방법

- HTML (문서 구조 정의) + CSS (문서 외형 정의) → 짝을 이루어 사용

• CSS 작성 방법

- ① **HTML 요소에 직접 CSS 정의** : HTML 요소에 스타일 속성 이용
- ② **CSS를 모아서 정의** : HTML 헤더 부분에 style 요소 사용
- ③ **CSS 파일을 만들어 HTML에 연결**

▶ CSS 기술방식

- 기본 기술방식

- 형태 : **선택자 { 속성 선언 }**
- 예 : `h1{font-size: 1.5em;}`

• 선택자

- 정의 : HTML의 어떤 요소, 요소들에 속성들이 적용되는지 정의하는 곳
- 형태 : 단순한 요소명, 클래스, ID 또는 복잡한 논리 형 등으로 다양

• 속성 선언

- 정의 : **선택자를 통해 선택된 HTML 요소에 적용할 스타일 속성 내용**
- 구성 : **속성과 값**
- 속성 : 정의하는 스타일의 내용
- 속성 값 : 속성의 내용 (키워드, 단위가 있는 수치)

▶ HTML 문서에 CSS 적용과 연결

• HTML 요소 속성으로 CSS 적용

- 선택자를 통하여 직접 요소에 CSS 스타일 추가
- 방법 : 스타일 속성을 통하여 CSS 스타일 적용
- CSS 속성 선언 : ; (세미콜론) 으로 분리

• Style 요소를 사용한 CSS 적용

- 방법 : <head> 부분에 선택자를 이용하여 style 요소 정의
- CSS 코드가 위치한 HTML 파일에만 적용
- **style 요소의 속성**

- ① **type** : style 언어가 어떤 MIME 타입인지 지정

CSS 파일의 경우 : text/css 지정

HTML5일 경우 : 생략 가능

- ② **media** : 현재 CSS 코드가 어떤 매체일 경우 지정되는지 지정

"all" 지정 : 모든 매체에서 현재 CSS 적용, 미디어쿼리

• 외부 CSS 파일 HTML 문서에 연결

- 장점

- ① 여러 HTML 파일에 공통으로 사용
- ② 수정 시 HTML 파일을 전체 수정할 필요 없음
- ③ 체계적이고 구조적인 스타일 관리 가능
- ④ 전체 전송량이 줄어듦

- HTML에서 외부 CSS 파일의 연결

- ① HTML 코드 작성
- ② 비어있는 파일을 작성하고 확장자 .css로 저장
- ③ CSS 파일 경로 지정
(HTML 코드 <head> 부분에 link요소 사용)
- ④ CSS 파일 열어 CSS 스타일 지정

- **link 요소** : HTML 문서와 외부 리소스 연결을 위해 사용, 대부분 CSS 파일 연결

2. CSS 선택자 1

▶ **선택자** : HTML 요소를 선택하기 위해 사용되는 CSS 구문

▶ 타입 선택자 (Type Selector)

- HTML 요소명, 요소의 클래스 속성 값, ID 값
- 가장 쉽고 선택의 기본이 되는 기초적인 선택자

① HTML 선택자

- 타입 선택자 중에서 **HTML요소가 선택자인** 선택자
- **선택자 형식** : HTML 요소명

② 클래스 선택자

- **CSS를 적용할 대상** : HTML 클래스 속성 값
- 클래스 선택자 속성 : **요소들을 원하는 그룹으로 묶을 수 있음**
- HTML 문서 내의 **동일한 클래스 값을 가지는 모든 요소에 적용**
- **선택자 형식** : .(점)으로 시작하는 클래스 속성 값

③ ID 선택자

- **CSS를 적용할 대상** : HTML ID 속성 값
- HTML ID 속성 : **HTML 문서 내의 유일한 값으로** 고유의 요소 식별
- HTML 문서 내의 **같은 ID를 가진 유일한 요소에 적용**
- **선택자 형식** : #(샵)으로 시작하는 ID 속성 값

④ 그룹 지정

- 여러 요소에 동일한 CSS 스타일 적용 : 클래스 속성 이용
→ **BUT!** 이미 클래스 속성 적용한 경우 **스타일 그룹 위해 클래스 속성 지정 불가**
- **두 개 이상의 요소에 동일한 스타일을 적용하고 싶을 때**
- **그룹 지정 형식** : ,(쉼표)로 동일한 스타일 적용할 선택자들 분리해 나열

▶ 고급 선택자 (계층 위치 선택자)

- **고급 선택자** : HTML 문서의 요소들의 **계층 관계를 이용하여** 선택하는 선택자
- HTML 요소들의 계층 관계 : **자손 요소, 직계 자손 요소, 형제 요소, 인접 형제 요소**

▶ 하위 선택자

- 하위 선택자 : 어떤 요소 하위에 있는 특정 자손 요소 선택 시 사용
- 표현 : (부모 요소) (자손 요소)

▶ 직계 자손 선택자

- 직계 자손 선택자 : 바로 하위에 있는 요소 선택 시 사용
- 표현 : (부모 요소) > (직계 자손 요소)

▶ 형제 선택자

- 형제 선택자 : 특정 요소 다음에 나오는 형제 관계 요소들 선택
- 표현 : (요소1) ~ (요소1의 형제 요소)

▶ 인접 형제 선택자

- 인접 형제 선택자 : 특정 요소의 바로 다음에 오는 형제 요소를 선택
- 표현 : (요소1) + (요소1의 인접 형제 요소)

▶ 전체 선택자

- 전체 선택자 : 모든 요소를 선택하기 위해 사용
- 표현 : *

3. CSS 선택자 2

▶ 의사 클래스(pseudo class)

- 의사 클래스 : 가짜 또는 모조 클래스, 클래스의 특징을 가짐
- 형식 : : (콜론) 의사 클래스 명
 - 선택자 뒤에 붙어 선택자의 상태를 기준으로 선택
 - :link 의사 클래스 : 한 번도 클릭하지 않은 링크 → 히스토리에 없는 링크 상태 의미

▶ 링크 의사 클래스(:link, :visited)

- 링크 의사 클래스 : 링크의 상태 선택자로 이용
- :link 의사 클래스 : 한 번도 방문하지 않은 링크
- :visited 의사 클래스 : 방문한 링크에 CSS 스타일 적용 시 사용

▶ 동적 의사 클래스

- 동적 의사 클래스 : 마우스와 커서에 관한 상태를 의사 클래스로 나타냄
- :active 의사 클래스 : 마우스로 클릭했을 때의 상태 의미
- :hover 의사 클래스 : 마우스 커서가 올라간 상태 의미
- :focus 의사 클래스 : 서식 품과 같은 요소에 마우스 위치하여 입력 또는 선택 상황 의미

▶ 구조적 의사 클래스

- 구조적 의사 클래스
 - HTML 구조에 따른 의사 클래스
 - 형제와 자손 그리고 몇 번째 요소인지로 상태 구분
 - CSS3에서 추가된 내용
- :root 의사 클래스 : 문서의 최상위 요소 의미
 - 단독으로 사용

HTML5&JavaScript 요약본

- **:empty** 의사 클래스 : 비어 있는 요소 의미
- **:only-child** 의사 클래스 : 형제가 없는 요소 → 자신이 속한 부모 요소의 유일한 자손 요소
- **:only-of-type** 의사 클래스 : 같은 타입의 형제가 없을 때 사용
- **:first-child, :last-child** 의사 클래스

:first-child 의사 클래스	:last-child 의사 클래스
• 부모 요소의 첫 번째 자손 요소	• 부모 요소의 마지막 자손 요소

- **:nth-of-type(n), :nth-last-of-type(n)** 의사 클래스

:nth-of-type(n) 의사 클래스	:nth-last-of-type(n) 의사 클래스
<ul style="list-style-type: none"> • 같은 부모 요소의 자손 요소로 특정 요소의 n 번째 요소 • 순서 위에서부터 세어감 	<ul style="list-style-type: none"> • 같은 부모 요소의 자손 요소로 특정 요소의 n 번째 요소 • 순서 아래에서부터 세어감

- **:first-of-type, :last-of-type** 의사 클래스

:first-of-type 의사 클래스	:last-of-type 의사 클래스
• 같은 부모 내에 있는 형제 요소 중 가장 첫 번째 있는 요소	• 같은 부모 내에 있는 형제 요소 중 가장 마지막 요소

▶ 기타 의사 클래스

- **:lang()** 의사 클래스 : 지정한 언어 속성을 가지는 요소
 - 하나의 HTML 문서에 다양한 국가의 언어 사용할 경우 해당 언어가 사용된 요소 적용
- **:not()** 의사 클래스 : 부정을 의미
 - () 안에 선택에서 제외될 선택자 삽입

▶ 의사 엘리먼트

- **의사 엘리먼트** : 가짜 요소, 선택자에 따라 기존 요소에 추가로 새로운 요소 정의
- **:first-letter** 의사 엘리먼트 : 선택자에서 선택한 요소의 첫 번째 글자를 새 요소로 만들
 - 첫 번째 글자에 새로운 스타일 적용 가능
 - 문단의 첫 글자 확대, 들어 쓰기 시 사용
- **:first-line** 의사 엘리먼트 : 선택자에서 선택한 요소의 첫 번째 줄을 새 요소로 만들
- **:after, :before** 의사 엘리먼트 : 선택자에 의해 선택된 요소 앞뒤에 요소를 만들

▶ 속성을 이용한 선택

- **속성 선택자** : HTML 요소들의 다양한 속성 적용
 - 형태 : **a[title] { color: pink; }**
 - title 속성이 있는 <a>요소 선택

▶ 속성과 속성값 선택자

- **속성과 속성값을 모두 이용하는 속성 선택자**
 - 형태 : **선택자[속성="속성값"]**

▶ 속성 선택자의 등식 연산

- 속성과 속성 값 선택자는 속성 값이 정확히 일치하는 요소 선택
- 다양한 등식을 통하여 원하는 선택 가능
- 속성값 선택자 형식

HTML5&JavaScript 요약본

[attr ^= "value"]	<ul style="list-style-type: none"> • 시작 의미 • value로 시작하는 속성 값 가지고 있음
[attr \$= "value"]	<ul style="list-style-type: none"> • 끝 의미 • value로 끝나는 속성 값을 가지고 있음
[attr *= "value"]	<ul style="list-style-type: none"> • 포함 의미 • value를 포함하는 속성 값을 가지고 있음 (띄어쓰기와 상관 없이 value가 속성 값의 일부인 경우 선택)
[attr ~= "value"]	<ul style="list-style-type: none"> • 공백으로 분리되어 포함 의미 • value를 포함하는 속성 값을 가지고 있음 (단, value가 공백으로 분리되어 있어야 함)
[attr = "value"]	<ul style="list-style-type: none"> • 하이픈으로 분리되어 포함 의미 • value가 하이픈으로 고분되어 있는 속성 값을 가지고 있음

▶ 미디어쿼리

- 미디어쿼리 : 현재 HTML 문서가 보여지는 화면이 어떤 것인지 파악
- 미디어쿼리 구문 : link 요소에 속성으로 적용
 - 미디어쿼리 값 + 미디어쿼리 속성
 - 전체 미디어쿼리 속성값 : <http://www.w3.org/TR/css3-mediaqueries/>

▶ 상속과 캐스케이딩

- HTML 요소에 적용된 CSS 스타일도 하위 자식에게 상속
- 예) 계층 구조로 작성된 HTML
- 적용 우선 순위 단계

<div>높음</div> <div>↑</div> <div>낮음</div>	단계	속성
	1	사용자가 선택자를 통해 직접 정의한 스타일
	2	HTML에 스타일 속성으로 적용한 인라인 스타일
	3	미디어타입에서 지정한 속성
	4	!important 구문을 추가한 CSS 속성
	5	구체적인 선택자
	6	뒷 부분에 정의된 스타일
	7	부모로부터 상속된 스타일
	8	웹 브라우저 기본 스타일

1. 텍스트 표현, 컬러와 배경

- 폰트 패밀리(font-family)
 - **font-family**: : 비슷한 모양의 서체를 묶어서 제시
 - 형식 : **font-family** : (사용할 서체),(대안 서체 1),(대안 서체 2);
- ▶ 서체 크기 조절
 - **font-size** : 속성 : 서체 크기 조절
 - 서체 크기 단위 : pt, pc, px, %, em, ex
- ▶ 다양한 변형 서체 (Italic, Bold, Small cap)
 - **Italic** : 약간 비스듬한 모양의 서체
 - 설정 방법 : **font-style** 속성 사용
 - **font-style** 속성 : italic, oblique, normal
 - **Bold** : 일반 텍스트 보 굵은 서체
 - 설정 방법 : **font-weight** 속성 사용
 - **font-weight** 속성 : bold, bolder, lighter, 100~900, normal
 - **Small cap** : 알파벳을 사용하는 문자에 해당하는 서체 스타일, 텍스트의 크기로 대소문자 표기
 - 설정 방법 : **font-variant** 속성 사용
- ▶ 텍스트 스타일 설정
 - 텍스트 간격 설정
 - 텍스트 간격 : 글자와 글자간의 간격 또는 단어와 단어간의 간격 의미
 - 자간 설정 : **letter-spacing** 속성 사용
 - 단어 간격 설정 : **word-spacing** 속성 사용
 - 행간 설정 : **line-height** 속성 사용
- ▶ 컬러
 - 웹 색상 코드 : 16진수 표현 10진수 표현, HSL 표현, 색상 키워드
 - 투명도 표현
 - 컬러에 투명도 설정 가능
 - 투명도 설정 방법 : 10진수 색상 코드로만 가능
 - 텍스트 색 지정 : **color** 속성 사용
- ▶ 배경 설정
 - 배경
 - 배경 색 설정 : **background-color** 속성
 - 배경 이미지 설정 : **background-image** 속성 사용
 - 배경 이미지 반복 설정 : **background-repeat** 속성 사용
 - 배경 이미지 고정 설정 : **background-attachment** 속성 사용
 - 배경 이미지 위치 설정 : **background-position** 속성 사용
 - 배경 이미지 크기 설정 : **background-size** 속성 사용

2. 목록, 표 꾸미기

▶ 불릿 꾸미기

- 불릿 : 목록을 정리하며 예쁘게 보이도록 목록 아이템 앞에 붙는 숫자 또는 특수 문자
- 불릿 설정 : **list-style-type** 속성 사용
- 이미지 불릿 설정 : **list-style-image** 속성 사용
- 불릿 위치 설정 : **list-style-position** 속성 사용

▶ Margin, Padding

기본 설정	<pre>p{ margin: 10px; padding: 10px; }</pre> <ul style="list-style-type: none"> margin과 padding 모두 10px로 설정 	
상하좌우 개별적 설정	<pre>p{ margin-left: 1cm; margin-right: 1cm; margin-top: 1cm; margin-bottom: 1cm; }</pre>	<pre>p{ padding-left: 1cm; padding-right: 1cm; padding-top: 1cm; padding-bottom: 1cm; }</pre>
나열식으로 설정	<pre>margin: 10px, 5px, 12px, 8px;</pre> <ul style="list-style-type: none"> 나열 순서 : top, right, bottom, left (좌측부터) 	

▶ Border 설정 (테두리 선 설정)

- border 생성 속성 : **border-width, border-style, border-color**

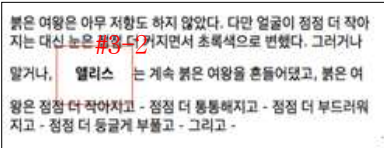
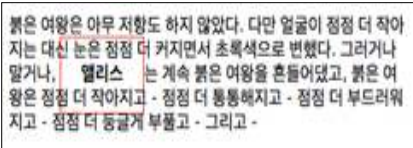
▶ 셀 Padding 과 셀 배경 색 설정

- 셀 padding 설정 : 셀의 여백 조절
- 셀 배경 색 설정 : **background-color** 속성 사용

3. CSS 박스 모델

▶ CSS 박스 모델

- 박스 모델의 margin과 padding : 요소 콘텐츠를 둘러싼 여백 의미
- 블록 레벨 요소와 인라인 레벨 요소의 margin과 padding의 차이점

블록 레벨 요소의 margin과 padding	인라인 레벨 요소의 margin과 padding
<p>상하 좌우 사방으로 margin과 padding 적용</p> 	<p>좌우 만 margin과 padding 적용</p> 

HTML5&JavaScript 요약본

- margin과 padding 초기화 설정

```
* {  
  margin: 0;  
  padding: 0;  
}
```

- 설정 값 '0' : 수치 단위 입력 않아도 됨
 - '0' 이외의 수치는 필수 적으로 단위 입력
- '*' 선택자 : 모든 요소 의미

▶ 요소의 크기 설정 (width, height 속성)

- width, height 속성 : 블록 레벨 요소의 크기 설정 → **인라인 레벨 요소에는 적용 안됨**
- width 속성 : 요소의 넓이 설정
- height 속성 : 요소의 높이 설정

▶ margin 겹침 : 두 개의 margin 연속 적용으로 **margin0** 겹치는 현상 발생

▶ Position

- position 속성 : 특정 요소가 다른 요소들과 어떠한 관계 속에서 위치를 결정하는 지 설정
- position 속성의 속성 값 : 상대 위치, 절대 위치, 고정 위치
- 상대 위치(relative) : position: **relative**;
- 절대 위치(absolute) : position: **absolute**;
 - z-index 속성 : 어떤 요소가 다른 요소 위에 나타나는지 설정
- 고정 위치 (fixed) : position: **fixed**;

1. 레이아웃 정렬

▶ float 속성을 사용한 2단 레이아웃

- **다단 레이아웃** : 화면을 세로로 여러 개의 단으로 나눠 콘텐츠를 보여주는 형태
- HTML 문서는 오로지 위에서 아래로 콘텐츠 제시

→

• float 속성을 사용한 2단 레이아웃 설정 방법

- ① HTML 문서 준비하기
- ② 섹션 요소의 넓이 설정하기
- ③ float 속성 설정하기
- ④ 2단 레이아웃 완성하기

▶ position 속성을 이용한 2단 레이아웃 설정 방법

- float 속성 설정보다 position 속성 사용 시 몇 가지 더 신경 써야 함
- **상대 위치 와 절대 위치**
 - **상대 위치** : 요소 위치 설정 시, 초기 위치에 자신의 볼륨을 그대로 유지
→ 좌표의 원점을 파악하기 어려움
 - **절대 위치** : 요소들의 원점 - 부모 요소의 왼쪽 상단으로 통일
→ 요소의 볼륨에 따른 레이아웃 변화에 적용 되지 않음

▶ display 속성을 이용한 2단 레이아웃 설정하기

블록 레벨 요소들을 **inline-block** 형태로 줄바꿈 없이 나란히 놓을 수 있음

2. CSS 네비게이션

▶ HTML 이미지 버튼

- HTML 이용 시 이미지 버튼 만들 수 있음

▶ 텍스트 네비게이션

- 네비게이션 : 웹 사이트에서 분류된 영역으로 쉽게 갈 수 있는 링크의 모음
 - 초창기 : 텍스트 네비게이션 → 현재 : 그래픽 네비게이션으로 발전
 - 웹 초창기 : 텍스트 네비게이션 - 제작하기 쉽고 로딩이 빨라 현재도 많이 사용

• HTML로 텍스트 네비게이션 구조 작성하기

- **nav** 요소 안에서 작성
- 네비게이션 HTML 작성 방법 : 목록 형태 → **링크의 목록으로 작성**
- 목록 작성 시 순서는 상관 없음 : **(순서 있는 목록), (순서 없는 목록)**

• CSS로 텍스트 네비게이션 스타일 적용 하기

- Key Point!

목록 아이템의 **display** 속성 : inline

- ① 링크에 설정되어 있는 **기본 스타일 초기화**
- ② 목록 요소에 **배경색을 검정으로 설정**

- ③ 웹 브라우저 margin, padding 0 으로 설정
- ④ 마우스 롤 오버 설정하기 : **a: hover** 선택자

3. CSS 변형과 트랜지션

▶ 요소 숨기기

- 요소 숨기기 : 요소가 웹 브라우저에서 보이지 않게 하는 것
 - 웹 브라우저 내 차지하고 있던 영역 사라짐
- 요소를 숨기는 이유
 - ① CSS가 지원되지 않는 웹 브라우저 사용자에게 추가 정보 제공
 - ② 시각 장애를 가진 사용자에게 안내 및 추가 정보 제공
 - ex) 스크린 리더
 - ③ 웹 문서에서 제공하는 정보가 많을 경우 **문서의 가독성을 높이기 위해**
 - ex) 펼쳐지는 콘텐츠 : 사용자 동작으로 콘텐츠가 보였다 가려졌다 하는 기능
 - CSS의 요소 숨기기 속성 이용
- CSS에서 요소 숨김 속성 : ① **visibility: hidden**, ② **display: none**

▶ 요소 클리핑

- clip 속성 : 이미지 또는 요소의 특정 부분 만을 보이게 할 때

JavaScript 기초

1. HTML과 JavaScript

HTML& CSS	JavaScript
문서의 외형 정의	문서의 기능 정의

1) JavaScript 의 특징

- 스크립트 언어이다.
- 함수형 언어이다.
- Java와는 전혀 다른 프로그래밍 언어이다.
- 초보적인 언어가 아니다.
- 웹 표준이다.

2. 데이터 타입과 변수

1) 데이터 타입

- 리터럴(literal) : 데이터 타입이 표현하는 값 자체
- 숫자 : JavaScript 는 정수, 실수 구분하지 않음
- 정수 리터럴, 실수 리터럴, 숫자 상수
- 문자열 : 텍스트를 표현하는 데이터 타입
- Escape sequence : 따옴표('), 쌍 따옴표(")로 인한 문자열 문제 해결
 \ ' , \ " ... 등
- Boolean : 참, 거짓을 구분하는 값
 논리 연산, 판단, 제어 구조에 주로 사용
- Boolean 데이터 값 : True, False
- 단순 데이터 타입 : null, undefined
- 객체 데이터 타입
 - ① 객체(object) : 다양한 값의 집합
 - ② 객체의 값 : property (어떠한 값도 가능)
 - ③ 객체 리터럴 : JavaScript 의 새로운 객체 만들때 사용
 - ④ 배열(array) : 값의 집합
 - ⑤ 배열 생성 : 생성자 (new) 이용, 배열 리터럴 이용


```
ver a = new Object;
ver a = [10];
```
 - ⑥ 함수 : JavaScript 프로그래밍의 기본 단위
 데이터 타입으로 변수에 할당이 가능한 값
 - ⑦ 함수 생성 : 키워드(function) 이용
 - ⑧ 함수 리터럴 : 값으로 변수에 할당 가능

2) 변수

- 변수 선언 : 키워드(ver) 사용


```
ver n ;
ver n = 10; (변수 선언 및 초기화)
```
- 변수의 유효 범위 : 함수를 기준으로 결정

- ①지역 변수 : 함수 안에 선언된 변수
- ②전역 변수 : 모든 함수에서 사용 가능한 변수
최소한의 사용 권장
- 참조 타입 : 문자 열이나 객체의 데이터 타입을 참조하기 위한 데이터 타입

JavaScript 문법

1. 연산자

- 산술 연산자 : 덧셈, 뺄셈, 곱셈, 나눗셈, 나머지(%) 연산
- 할당 연산자 : +=, -=, *=, /=, %=
- 동등 연산자 : ==, === / 부등, 불일치 연산자 : !=, !==
- 비교 연산자 : <, >, <=, >=
- 논리 연산자 : &&, ||, !
- 기타 연산자

2. 조건문과 반복문

① 조건문

- if 문 : if 문, if ~ else 문, if ~ else if 문
- switch 문

```
switch ( 변수 ){  
    case:  
        break;  
}
```

② 반복문

- for 문
for 문 : for (카운트 변수 초기화; 제어 조건; 카운트 실행)
for/in 문 : for (변수 in 배열/객체)
- while 문
while 문 : while (조건)
do/while 문 : do{
 }while(조건)

③ 조건문 반복문 중단/ 이동

- break 문 : 실행을 빠져나감
- continue 문 : 반복의 시작점으로 돌아가 새로운 반복 시작
- 레이블 : 어떤 반복문/조건문 중단하고 이동하는지 표시

내장 객체

1. 객체의 이해와 생성

1) 객체

- 여러 값을 모아 놓은 데이터 타입
- 객체의 종류
 - ① Property : 객체 내부에서 객체 표현하는 변수
 - ② Method : 함수로 된 Property
 - ③ 내장 객체 : JavaScript 에서 제공하는 객체
 - ④ 브라우저 객체 모델, 문서 객체 모델, 사용자 정의 객체 등

2) 객체의 생성

- 생성자 함수와 프로토 타입으로 정의
- 객체 literal, new 연산자 이용

3) 객체 메소드

- 함수를 값으로 가진 프로퍼티
- 값이 함수이므로 작동을 위해 사용

2. Number, String 객체

- 내장 객체 : JavaScript 에서 제공하는 유용한 객체, 기본 데이터 타입에서 처리하지 못하는 데이터 또는 기본 데이터 타입의 객체 타입을 처리하기 위해 제공
- Number : 기본 데이터 타입인 숫자에 대응하는 객체 타입
 - 다양한 프로퍼티와 메소드 존재
 - 생성자를 사용하여 생성할 수 있음
 - 숫자 데이터 타입에 프로퍼티나 메소드를 적용→ 일시적으로 Number 객체
- String : 기본 데이터 타입인 문자열에 대응하는 객체 타입
 - 문자열에 관련된 다양한 메소드를 제공하여 문자열을 가공할 수 있게 해줌

3. Date, Math 객체

- Date : 현재 시간을 얻어내거나 시간 연산 가능
 - 내장 객체와는 다르게 생성자를 통한 인스턴트화가 필요 X
 - 공유 객체 자체에서 메소드와 프로퍼티에 접근 가능
- Math 객체 : 삼각함수 메소드, 반올림 메소드, 올림 값과 내림 값, 절대 값과 역승, 최대 값과 최소 값, 난수발생

배열과 함수

1. 배열

- 배열 : 숫자로 나열된 값의 집합
- 배열의 생성 : ① literal 사용, ② Array 생성자 사용
- 배열의 원소 접근 : 배열 이름 [원소의 인덱스];
- 배열의 원소 순회 : ① for/in 문, ② length 메소드 이용
- 배열의 메소드
 - ① 배열을 문자로 반환 : toString(), join()
 - ② 배열의 원소 정렬 : sort(), reverse()
 - ③ 배열의 원소 조작 메소드 : concat(), slice(), splice()
 - ④ 배열의 원소 추가/ 제거
 - 배열의 마지막에 추가/제거 : pop()/push()
 - 배열의 첫 번째에 추가/제거 : unshift()/shift()

2. 함수

- 함수 : 미리 정의 되어 프로그래밍 여러 부분에서 복수로 호출되어 실행되는 코드 블록
- 선언적 함수 정의

```
function 함수이름(전달인자1, 전달인자2) {  
    함수 실행 코드;  
    return 반환값;  
}
```
- 선언적 함수의 이름 : 함수를 가리키는 변수
- 선언적 함수의 전달인자 : 개 수 및 데이터 타입 상관 없음
- 리턴문과 반환값 : 함수 호출자에게 전달되는 값으로 함수 실행 결과로 발생

문서 객체 모델(DOM)

1. DOM의 이해와 Window 객체

- ① DOM 객체 계층 : 웹 브라우저와 웹 문서의 기능과 내용을 대부분 객체화
- ② 현재 웹 브라우저 윈도우
 - Window 객체
 - window 프레임 객체 배열
 - Navigator 객체
 - Location 객체
 - History 객체
 - Document 객체
 - Screen 객체
- ③ Window 객체 : 다른 DOM 객체에 접근할 수 있는 것 이외에 현재 웹 브라우저 창에 대한 여러 가지 정보 제공
- ④ Window 생성 : 팝업 윈도우 생성 (open 메소드), 윈도우 닫기 (close 메소드), alert

2. 웹 문서 접근

- ① DOM을 통하여 웹 브라우저, HTML 문서의 모든 기능과 요소에 접근 가능
- ② 문서 객체 배열로 접근
 - document.anchors[] : 문서 내 앵커 객체
 - document.applets[] : Java 애플릿
 - document.forms[] : 폼 객체
 - document.images[] : 이미지 객체
 - document.links[] : 링크 객체
- ③ 이름 속성으로 접근 : name 속성
- ④ DOM 문서 트리 : HTML 문서의 모든 요소 접근 가능, 중첩된 요소들 부모, 자손 관계로 계층화

3. 웹 문서의 조작

- ① write/writeln 메소드 : 가장 쉽게 문서 내 콘텐츠 추가
- ② 노드의 생성과 삽입
 - createElement() / createTextNode() 메소드 : 요소와 텍스트 노드 생성
 - appendChild() 메소드 : 자손 노드 삽입
 - insertBefore() 메소드 : 자손 요소 중 기존 요소의 끝에 생성
 - replaceChild() 메소드 : 자손 노드 중 특정 노드 새로운 노드로 치환
- ③ innerHTML 프로퍼티 : createElement() 와 createTextNode() 그리고 appendChild() 를 한꺼번에 처리

이벤트

1. 이벤트의 이해

이벤트: 웹 브라우저에서 웹 문서에 특별한 일이 있을 때 발생하는 신호

이벤트 핸들러: DOM 객체에 할당되어 해당 객체의 이벤트 반응에 호출되어 처리되는 프로퍼티

이벤트 모델의 종류

기본 이벤트 모델(original event model) ? DOM Level 0 이벤트 모델

표준 이벤트 모델(standard event model) ? DOM Level 2 이벤트 모델

인터넷익스플로러 이벤트 모델(Internet Explorer event model) - 기본 이벤트 모델

기본 이벤트 모델의 이벤트 핸들러

마우스 관련: onclick, onmousedown, onmouseup, onmouseover, onmouseout

로딩 관련: onload, onunload

서식 관련: onsubmit, onreset, onfocus, onblur

키보드 관련: onkeydown, onkeypress, onkeyup

기본 이벤트 모델 적용

HTML 요소에 속성으로 이벤트 핸들러 적용

```
<a href="#" onclick="var i=3, j=5; calc=i*j; alert(calc); ">3 x 5 의 값은?</a>
<body onload="popup();" >
```

자바스크립트 DOM 객체 프로퍼티로 이벤트 핸들러와 호출함수 지정

```
window.onload = init;
document.getElementById( "next").onclick = goNext;
```

이벤트 기본 기능 막기

웹 브라우저에 기본으로 정의 되어 있는 이벤트 기능의 작동을 막을 필요성 있음

기본 이벤트 모델에서는 이벤트 핸들러 호출 함수가 false를 리턴하면 기본 기능이 작동을 하지 않음

```
<a href="second.html" onclick="alert('두 번째 페이지로 갈까요?'); return false; ">Second Page</a>
```

위 코드는 'second.html'로 이동하지 않는다.

이벤트 API

이벤트 핸들러 핸들러는 호출함수에 함수 인자로 이벤트 객체를 전달한다.

이벤트 객체를 전달받은 호출 함수는 다양한 이벤트 API를 이용하여 이벤트 정보를 얻을 수 있다.

이벤트 발생 좌표 정보: clientX, clientY, screenX, screenY,

이벤트 발생 키보드 정보: keyCode, altKey, shiftKey, type

2. 표준 이벤트 모델

특징

표준화 된 기능 제공

고급 이벤트 처리 방식

복잡하고 정밀한 이벤트 전파 제어

이벤트 핸들러 적용

```
window.addEventListener('load', init, false);
```

이벤트를 감지하고자 하는 객체에 프로퍼티로 addEventListener 를 적용

첫 번째 인자 'load'는 이벤트 핸들러

기본 이벤트 모델의 이벤트 핸들러에 앞에 on을 제거하고 사용

두 번째 인자 'init'는 호출 함수

세 번째 인자 false는 이벤트 전파 방법

적용 예

```
document.getElementById("drag").addEventListener('mousedown', mouseDown, false);
```

```
document.getElementsByTagName("a")[0].addEventListener(click, linkClick, false);
```

이벤트 기본 기능 막기

표준 이벤트 모델에서는 preventDefault(); 이벤트 API 제공

```
function mouseDown (event) {
    event.preventDefault();
}
```

이벤트 API

기본 이벤트 모델에 비해 많은 수의 이벤트 API 제공

기본 이벤트 모델의 이벤트 API와 동일한 사용법

target : 이벤트 발생 객체

```
event.target.style.display = "none";
```

eventType : 발생한 이벤트타입

timeStamp : 밀리초 단위의 이벤트 발생 시간(날짜와 시간)

Button: 마우스 이벤트 발생시 눌러진 버튼.

1은 왼쪽, 2은 가운데, 3은 오른쪽 버튼

this : 이벤트 핸들러 호출 함수에서 사용되며 이벤트 발생 객체를 의미함

이벤트 API인 target과 동일한 결과

함수를 호출하는 기본 이벤트 모델과 표준 이벤트 모델에서 모두 사용 가능

표준 이벤트 모델에서는 이벤트 전파로 인한 혼란으로 target을 사용하는 것을 권장함

적용된 이벤트 핸들러 함수의 제거

removeEventListener : 적용된 이벤트 핸들러 함수를 제거하는데 사용한다.

이벤트 핸들러 호출 함수 적용 시 사용되었던 addEventListener 와 동일한 인자로 구성된다.

```
document.getElementById("apple").addEventListener('click', bang, true);
```

```
document.getElementById("apple").removeEventListener('click', bang, true);
```

3. 이벤트 전파

이벤트 전파의 이해

이벤트는 발생한 객체에만 머물지 않고 이벤트 발생 객체를 포함하는 모든 상위 객체에 전파된다.

이벤트 전파는 시각적으로 겹쳐 있는 객체에서 일어나는 것이 아니라 부모 자식 관계로 연결된 노드에서 발생하는 것.

이벤트 전파의 3단계

1단계 :document 노드 에서 이벤트 발생 노드까지 내려가면서 이벤트 전파

2단계 : 이벤트 발생 노드

3단계 : 이벤트 발생 노드에서document 노드까지 올라가면서 이벤트 전파

addEventListener 이벤트 전파 조절

addEventListener 의 세번째 인자를 참으로 해 놓으면 이벤트 전파의 전체 과정에서 이벤

트 핸들러가 이벤트를 감지한다.?캡처링 capturing

거짓이면 2단계 3단계에서 이벤트 핸들러가 이벤트를 감지한다. - 버블링 bubbling

이벤트 전파 방지

stopPropagation : 이벤트 전파를 막는 이벤트 API

이벤트 전파에 관련된 이벤트 API

currentTarget : 이벤트 전파로 인해 현재 도달한 객체

eventPhase: 세 단계의 이벤트 전파 과정 중 현재 단계가 몇 단계인지 알려준다.

함수의 고급기능

1. 다양한 함수 정의

생성자를 이용한 함수 정의 : Function 생성자

- 익명함수 : 함수 정의 시 이름 없는 함수
- 동적함수 : 동적으로 함수가 생성되고 처리 시 사라지는 함수

함수 리터럴 : 메소드와 프로퍼티 적용 가능

콜 백 함수 : 메소드 실행 시 자동으로 호출되는 함수

- filter 메소드 : 원하는 요소 정리하여 새로운 배열 생성
- forEach 메소드 : 배열의 엘리먼트 하나 하나를 조작
- every/over 메소드 : 함수에서 제시하는 조건 관련 함수
- map 메소드 : 콜백함수의 처리를 거친 후 새로운 배열을 반환

재귀 함수 : 함수 내부에서 자신을 다시 호출하는 함수

2. 함수의 유효 범위와 클로저

유효 범위

- 어휘적 유효 범위 : 함수가 실행될 때 환경을 유효범위에 포함하지만 함수가 정의 될 당시의 환경도

유효 범위에 포함

클로저(Closure) : 중첩함수

- 함수가 실행되고 내부 함수를 반환한 후에 함수가 종료 됨
 - 함수가 종료되면 JavaScript 가비지 콜렉션에 의해 함수 객체를 제거하고 아무것도 남지 않음
- 커링 (Curring) : 전달인자를 하나 하나 적용하는 것

1. 객체의 생성

① 객체 : property 의 모음

② 객체의 생성 :

- 객체 literal 의 경우 객체 property 를 나열하여 생성
- 빈 객체 : literal 방식의 함수 생성 시 property 를 쉽게 추가할 수 있어 사용
- namespace : 변수와 함수 등의 객체의 property 로 변경하여 전역변수 사용 최소화
- new 연산자와 생성자함수를 이용하여 객체 생성

③ 생성자 함수와 prototype

- this : method 에서 사용되며 method 를 호출한 호출객체를 가리킴
- 생성자 함수 : 생성할 객체에 어떤 property 가 있을 것이며 그 property 는 어떻게 초기화 될 것인지를 정의해 놓은 함수로 일반적인 객체지향 프로그래밍 언어의 Class와 상당히 흡사
- prototype : 객체 literal property 를 생성된 객체에 복사해 오는 것이 아니라 참조하고 있다는 것을 의미하고 생성자 함수의 prototype 객체가 수정되면 이 생성자 함수에 의해 생성된 객체의 property 가 변경되는 것
- prototype 가리기 : 생성된 객체의 property 를 읽으려 할 때 JavaScript 는 먼저 객체 자신의 property 를 검사. 만일 객체 자신의 property 에 읽으려고 하는 property 가 없다면 prototype 객체에 읽으려는 property 가 있는지 검사.

2. Class 흉내내기

① 객체 property 와 method

- instance property : Class라는 객체 명세서로 작성된 객체로 각각의 객체는 독립된 property 들을 가지고 있음
- instance method : instance Class와 비슷하며 instance 객체의 method 를 의미

② Class property 와 method

- Class property : instance 객체의 property 가 아닌 Class 자체의 property 로 정의 되어야 하는 경우
- Class method : JavaScript 에 구현하기 위해서는 this를 사용하지 않는 method 를 정의

③ 비공개 property : 객체 내부의 메서드만 접근이 가능하고 외부 코드에서는 조작할 수 없는 property 를 의미

④ 문자열 또는 기본 데이터 타입으로 자동 변환

- toString() : 객체 method 는 객체를 문자열로 표현
- ValueOf() : 문자열 이외의 기본 데이터 타입으로 변환

1. Class 상속

- 객체 복사 : superclass 와 subclass 의 프로토타입 공유 방식을 위하여 객체를 복사하여 할당

- 객체 복사 방법

① 빈 생성자 함수를 작성

```
var TempFunc = function() {};
```

② 빈 생성자함수의 프로퍼티에 복사하고자 하는 객체를 할당

```
TempFunc.prototype = originalObj;
```

③ 변수에 new 연산자와 생성자를 사용해 새로운 객체를 생성해 할당

```
var duplicatedObj = new TempFunc();
```

- 프로토타입 복사 : 객체 복사 함수 사용

- construtor 프로퍼티설정

constructor : 객체의 생성자

constructor 프로퍼티: 생성자 함수로 객체를 만들 때 생성자 함수의 프로토타입에 생성

2. 기타 다른 상속

- 객체 리터럴 상속 : 객체 리터럴 방식으로 생성된 객체의 모든 프로퍼티와 메소드가 sub 객체에 상속

- 객체 메소드 일부분만 가져다 쓰기

- 여러 객체의 모든 프로퍼티 한 객체로 가져와 사용하는 경우

mixIN : 다수의 부모 객체를 순환하여 모든 프로퍼티 또는 메소드를 빌려와 하나의 객체로 만들어 반환

① 하나 이상의 객체에서 프로퍼티를 모두 빌려오는 함수 : 전달 인자 2개 이상(첫 번째 : 프로퍼티가 가져와 저장되는 객체, 두 번째 이상 : 프로퍼티를 가져올 객체)

② 하나 이상의 Class에서 프로토타입의 메소드를 모두 빌려오는 함수 : 전달 인자 2개 이상(첫 번째 : 메소드를 받을 Class, 두 번째 이상 : 메소드를 빌려줄 class)

1. JavaScript library 의 이해

- JavaScript library : JavaScript 프로그래밍 시 유용하게 쓰일 JavaScript 코드의 덩어리
사용자가 자신의 코드를 추가하거나 library 의 기능 자신의 코드에 추가 가능
- JavaScript library 의 종류
 - ① 다양한 기능 제공
 - prototype.js : 다양한 유틸리티 함수 제공
 - YUI : Ajax, CSS 선택기, 이벤트 처리외 다양한 UI 유틸리티 포함
 - Dojo : 그래픽과 애니메이션 쉽게 구현 가능
 - jQuery: 코딩을 쉽고 편하게 하기 위한 메소드 제공(단, UI에 관한 기능 제공 안함)
 - ② 특화된 기능 제공
 - video.js, SoundManager2, paper.js, Raphael, D3 .. 등

2. jQuery 의 이해

-jQuery 의 특징

- ① 빠른 DOM 객체의 접근과 조작
 - CSS 선택자를 포함하는 강력한 선택자와 요소 속성 조작 방법
 - 브라우저를 고려하지 않아도 되는 쉽고 강력한 이벤트 모델
 - 쉽게 구현할 수 있는 애니메이션 효과
- ② 메소드 체이닝
- ③ Ajax
- ④ jQuery plug-in

3. jQuery 의 선택자

- jQuery 의 선택자 : \$(), jQuery() 함수 사용
- jQuery 함수의 기본 사용법
 - ① 요소 태그 명으로 선택 : `var selectedElements = $("p");`
 - ② 아이디 속성으로 선택 : `$("#firstline").addClass("select");`
 - ③ Class 속성을 선택 : `$(".wise").addClass("select");`
 - ④ CSS 선택자의 혼합 : `$("p a.refer").addClass("select");`
- CSS3 선택자를 이용한 요소 선택
 - ① 첫 번째/ 마지막 요소의 선택 : `:first-child()`, `:last-child()`
 - ② 특정 순서 요소의 선택 : `:nth-child(n)`

1. jQuery 의 기능들

① 속성의 조작

- attr() 메소드 : 선택된 요소의 속성을 읽어오거나 설정하는 메소드
- addClass() 메소드 : 선택된 요소들에 특정 Class 속성 적용
- css() 메소드 : 특정 스타일 적용

② 콘텐츠 조작

- html() 메소드 : 특정 요소에 html 코드를 읽어 오거나 대치
- text() 메소드 : 텍스트 콘텐츠를 읽거나 대치
- append/appendTo() 메소드 : 콘텐츠 추가

③ 이벤트

- bond() 메소드 : 이벤트 적용 메소드
- unbind() 메소드 : 이벤트 삭제 메소드

④ 그 밖의 기능

- 다양한 애니메이션 효과 쉽게 구현하는 메소드 지원
- 객체, 배열, Ajax 쉽게 다루는 메소드 제공
- plug-in 쉽게 확장 가능
- 다양한 plug-in 개발 및 배포

1. HTML5 문서 구조화

<article>

- 독립적으로 배포 혹은 재사용 가능한 섹션
- 독립적인 글
- 예) 신문, 잡지 기사, 블로그 글, 댓 글 하나

2. 이벤트

- 이벤트: 웹 브라우저에서 웹 문서에 특별한 일이 있을 때 발생하는 신호

이벤트 핸들러: DOM 객체에 할당되어 해당 객체의 이벤트 반응에 호출되어 처리되는 프로퍼티

- 기본 이벤트 모델의 이벤트 핸들러

마우스 관련: onclick, onmousedown, onmouseup, onmouseover, onmouseout

로딩 관련: onload, onunload

서식 관련: onsubmit, onreset, onfocus, onblur

키보드 관련: onkeydown, onkeypress, onkeyup

3. DOM 의 이해와 Window 객체

- DOM 객체 계층 : 웹 브라우저와 웹 문서의 기능과 내용을 대부분 객체화

- 현재 웹 브라우저 윈도우

- ① Window 객체
- ② window 프레임 객체 배열
- ③ Navigator 객체
- ④ Location 객체
- ⑤ History 객체
- ⑥ Document 객체
- ⑦ Screen 객체

1. HTML 활용하여 배너만들기

 태그에 onmouseover, onmouseout 효과 적용하기

```

```

- This : 자신에 대한 객체 의미

2. JavaScript 로 변환하기

마우스 이벤트 핸들러로 호출하기

- swapimg_func() 함수

```
function swapimg_func(){
    document.getElementById(obj).setAttribute("src", "images/logo.png ");
```

swapimg_over() 함수

```
function swapimg_over(obj,srcstr){
    if(!document.getElementById(obj).hasAttribute("osrc")){
        var osrcstr = document.getElementById(obj).getAttribute("src");
        document.getElementById(obj).setAttribute("osrc",osrcstr);
    }
    document.getElementById(obj).setAttribute("src",srcstr);
}
```

swapimg_out() 함수

```
function swapimg_out(obj){
    var osrcstr = document.getElementById(obj).getAttribute("osrc");
    document.getElementById(obj).setAttribute("src",osrcstr);
    document.getElementById(obj).removeAttribute("osrc");
}
```

3. CSS 박스 모델 및 트랜지션

CSS 변형과 트랜지션

- 상대 위치 : 요소 위치 설정 시, 초기 위치에 자신의 볼륨을 그대로 유지
 - 좌표의 원점을 파악하기 어려움
- 절대 위치 : 요소들의 원점 - 부모 요소의 왼쪽 상단으로 통일
 - 요소의 볼륨에 따른 레이아웃 변화에 적용 되지 않음
- 요소 숨기기 : 요소가 웹 브라우저에서 보이지 않게 하는 것
 - 웹 브라우저 내 차지하고 있던 영역 사라짐
- clip 속성 : 이미지 또는 요소의 특정 부분 만을 보이게 할 때

1. 텍스트 네비게이션

- 네비게이션 : 웹 사이트에서 분류된 영역으로 쉽게 갈 수 있는 링크의 모음
- HTML로 텍스트 네비게이션 구조 작성하기
 - ① nav 요소 안에서 작성
 - ② 네비게이션 HTML 작성 방법 : 목록 형태 → 링크의 목록으로 작성
 - ③ 목록 작성 시 순서는 상관 없음 : <a>(순서 있는 목록), (순서 없는 목록)
 - ④ CSS로 텍스트 네비게이션 스타일 적용 하기

2. 이미지 네비게이션

- ① HTML 문서 작성
(텍스트 네비게이션 작성과 동일)
- ② 이미지 작업하기
(PhotoShop.. 등 이미지 작업 도구 활용)
- ③ CSS 작성하기
 - 텍스트 스타일과 네비게이션 배경 설정
 - 목록 스타일 설정과 목록 아이템 배경 설정
 - nav 요소의 배경 수정하기
 - 메뉴와 롤오버 설정하기

3. 드롭다운 메뉴의 네비게이션

- 메뉴 확장 함수 작성(expandMenu())


```
function expandMenu(num,dir){
  var mtobj = document.getElementById('mt_'+num);
  var smobj = document.getElementById('sm_'+num);
  clearInterval(smobj.timer);
  if(dir == "on"){
    ...

  }else{
    ...

  }
}
```