221205_2반_실습

JavaScript

JavaScript 개요

JavaScript란?

- 웹 페이지에서 복잡한 기능을 구현할 수 있도록 하는 **스크립팅 언어** 또는 **프로그래밍 언어**입니다.
- JavaScript는 개발자가 대화식 웹 페이지를 만들기 위해 사용하는 프로그래밍 언어입니다. 소셜 미디어 피드 새로 고침부터 애니메이션 및 대화형 지도 표시에 이르기까지, JavaScript 기능은 웹사이트의 사용자 경험을 개선할 수 있습니다. 클라이언트 측 스크립팅 언어로서 JavaScript는 월드 와이드 웹의 핵심 기술 중 하나입니다. 예를 들어 인터넷 탐색 시 웹페이지에서 이미지 슬라이드쇼, 클릭하면 표시되는 드롭 다운 메뉴 또는 객체 색상의 동적 변화를 보게 된다면 이는JavaScript의 효과를 보는 것입니다.
- 자바스크립트(JavaScript)는 **객체(object)** 기반의 **스크립트 언어**입니다.

  > 💡 스크립트 언어 : **응용 소프트웨어를 제어하고 소스 코드를 컴파일(Compile)하지 않고도 실행할 수 있는 컴퓨터 프로그래밍 언어**를 가리킨다.

- HTML로는 웹의 내용을 작성하고, CSS로는 웹을 디자인하며, 자바스크립트로는 웹의 동작을 구현할 수 있습니다.
- 자바스크립트는 주로 웹 브라우저에서 사용되나, Node.js와 같은 프레임워크를 사용하면 서버 측 프로그래밍에서도 사용할 수 있습니다.
- 현재 컴퓨터나 스마트폰 등에 포함된 대부분의 웹 브라우저에는 자바스크립트 인터프리터가 내장되어 있습니다.

JavaScript 특징

- HTML/CSS와 완전히 통합할 수 있음
- 간단한 일은 간단하게 처리할 수 있게 해줌
- 모든 주요 브라우저에서 지원하고, 기본 언어로 사용됨
- 자바스크립트는 객체 기반의 스크립트 언어입니다.

- 자바스크립트는 동적이며, 타입을 명시할 필요가 없는 인터프리터 언어입니다.
- 자바스크립트는 객체 지향형 프로그래밍과 함수형 프로그래밍을 모두 표현할 수 있습니다.

JavaScript 사용 목적

1. Javascript의 사용 효과
    - 모던 자바스크립트는 '안전한' 프로그래밍 언어입니다. 메모리나 CPU 같은 저수준 영역의 조작을 허용하지 않습니다. 애초에 이러한 접근이 필요치 않은 브라우저를 대상으로 만든 언어이기 때문이죠.
    - 자바스크립트의 능력은 실행 환경에 상당한 영향을 받습니다. Node.js 환경에선 임의의 파일을 읽거나 쓰고, **네트워크 요청을 수행하는 함수를 지원**합니다.
    - 브라우저 환경에선 **웹페이지 조작**, **클라이언트와 서버의 상호작용**에 관한 모든 일을 할 수 있습니다.
    - 브라우저에서 자바스크립트로 할 수 있는 일은 다음과 같습니다.
        - 페이지에 새로운 HTML을 추가하거나 기존 HTML, 혹은 스타일 수정하기
        - 마우스 클릭이나 포인터의 움직임, 키보드 키 눌림 등과 같은 사용자 행동에 반응하기
        - 네트워크를 통해 원격 서버에 요청을 보내거나, 파일 다운로드, 업로드하기 (AJAX나 COMET과 같은 기술 사용)
        - 쿠키를 가져오거나 설정하기. 사용자에게 질문을 건네거나 메시지 보여주기
        - 클라이언트 측에 데이터 저장하기(로컬 스토리지)

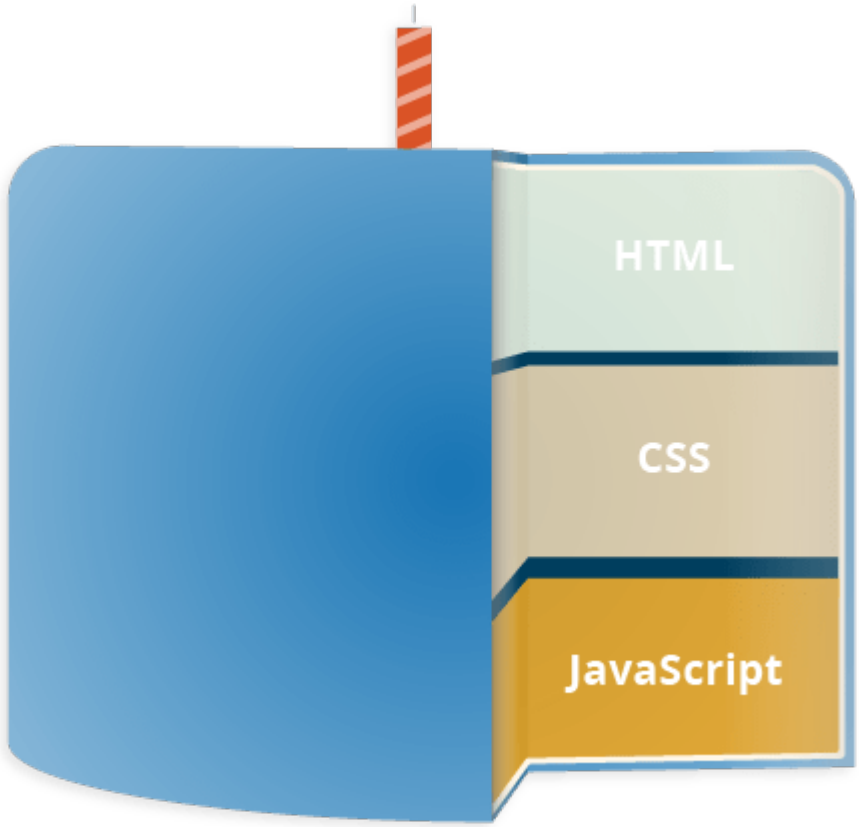
API(Application Programming Interface)

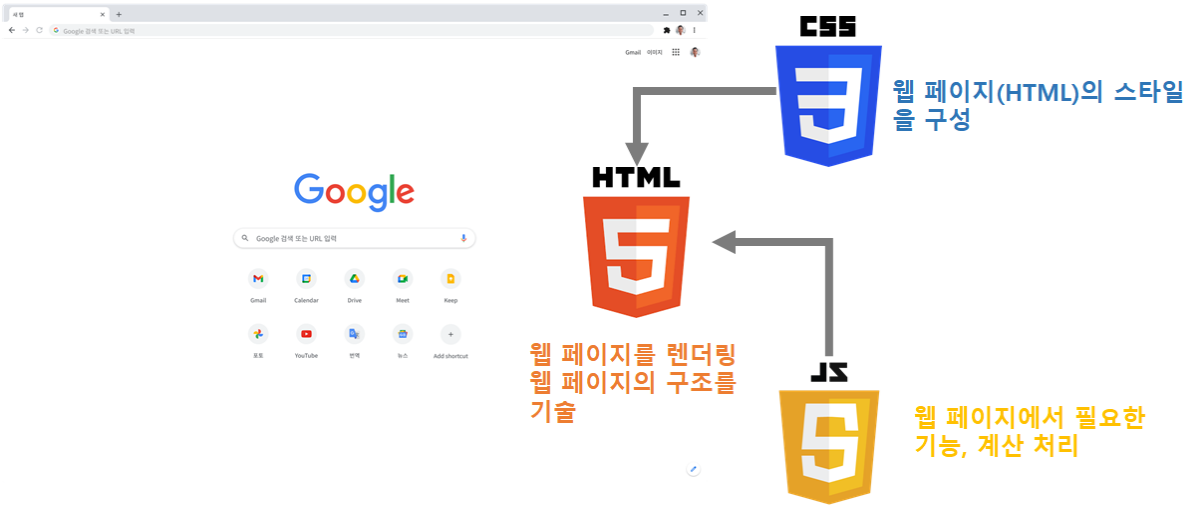
- API는 개발자가 직접 구현하기는 어렵거나 불가능한 기능들을 미리 만들어서 제공하는 것입니다.
- **브라우저 API** 는 웹 브라우저에 내장된 API로, 현재 컴퓨터 환경에 관한 데이터를 제공하거나 여러가지 유용하고 복잡한 일을 수행합니다.
    - DOM (Document Object Model) API로 HTML 콘텐츠를 추가, 제거, 변경하고, 동적으로 페이지에 스타일을 추가하는 등 HTML/CSS를 조작할 수 있습니다. 페이지 위에 뜨는 팝업창이나, (위쪽의 간단한 예제처럼) 새로운 콘텐츠가 나타나는 것을 본 적이 있으면 DOM이 동작한 겁니다.
    - Geolocation API로 지리정보를 가져올 수 있습니다. Google 지도에서 여러분의 위치를 찾아 지도에 그릴 수 있는 이유가 바로 이 API입니다.

    - Canvas와 WebGL API로 애니메이션을 적용한 2D와 3D 그래픽을 만들 수 있습니다. 두 웹 기술을 사용해서 만들 수 있는 놀라운 결과를 엿보려면 Chrome Experiments와 webglsamples를 방문하세요.
    - `HTMLMediaElement`와 `WebRTC`를 포함하는 오디오와 비디오 API로는 멀티미디어를 사용한 흥미로운 일을 할 수 있습니다. 예를 들어 오디오나 비디오를 웹 페이지에서 바로 재생하거나, 여러분의 웹캠으로 비디오를 찍어 다른 사람의 화면에 보여줄 수 있습니다. (간단한 스냅샷 데모를 방문해서 감을 잡아보세요)
- **서드파티 API**는 브라우저에 탑재되지 않은 API로, 웹의 어딘가에서 직접 코드와 정보를 찾아야 합니다.
    - Twitter API로 여러분의 최신 트윗을 웹 사이트가 보여주도록 구현할 수 있습니다.
    - Google 지도 API와 OpenStreetMap API로 웹 사이트에 지도를 삽입하고, 지도 관련 기능을 추가할 수 있습니다.

Javascript와 HTML

Javascript, HTML, CSS





1. HTML
    - HTML은 웹 콘텐츠의 구조를 짜고 의미를 부여하는 마크업 언어입니다. 예를 들어 페이지의 어디가 문단이고, 헤딩이고, 데이터 표와 외부 이미지/비디오인지 정의합니다.
    - 웹의 구조를 작성
2. CSS

- CSS는 HTML 콘텐츠에 스타일을 적용할 수 있는 스타일 규칙 언어입니다. 배경색을 추가하고, 글꼴을 바꾸고, 콘텐츠를 신문처럼 다열 레이아웃으로 배치할 수 있습니다.
- 웹 요소들에 대한 디테일한 스타일을 작성

3. JavaScript
    - JavaScript는 동적으로 콘텐츠를 바꾸고, 멀티미디어를 제어하고, 애니메이션을 추가하는 등 거의 모든 것을 만들 수 있는 스크립팅 언어입니다. (정말 모든게 가능하지는 않겠지만, JavaScript 코드 몇 줄만으로도 놀라운 결과를 이룰 수 있습니다)
    - 웹에서 수행되는 기능을 구현하기 위한 계산 수행

DOM(Document Object Model)

- **문서 객체 모델(DOM, Document Object Model)은 XML이나 HTML 문서에 접근하기 위한 일종의 인터페이스입니다.**
- 이 객체 모델은 문서 내의 모든 요소를 정의하고, 각각의 요소에 접근하는 방법을 제공합니다.
- 이러한 DOM은 W3C의 표준 객체 모델이며, 다음과 같이 계층 구조로 표현됩니다.
- DOM(Document Object Model)은 웹 페이지에 대한 **인터페이스**입니다. 기본적으로 여러 프로그램들이 페이지의 콘텐츠 및 구조, 그리고 스타일을 읽고 조작할 수 있도록 API를 제공합니다.
- 문서 객체 모델(The Document Object Model, 이하 DOM) 은 HTML, XML 문서의 프로그래밍 interface 이다. DOM은 문서의 구조화된 표현(structured representation)을 제공하며 프로그래밍 언어가 DOM 구조에 접근할 수 있는 방법을 제공하여 그들이 문서 구조, 스타일, 내용 등을 변경할 수 있게 돕는다. DOM 은 nodes와 objects로 문서를 표현한다. 이들은 웹 페이지를 스크립트 또는 프로그래밍 언어들에서 사용될 수 있게 연결시켜주는 역할을 담당한다.
- 웹 페이지는 일종의 문서(document)다. 이 문서는 웹 브라우저를 통해 그 내용이 해석되어 웹 브라우저 화면에 나타나거나 HTML 소스 자체로 나타나기도 한다. 동일한 문서를 사용하여 이처럼 다른 형태로 나타날 수 있다는 점에 주목할 필요가 있다. DOM 은 동일한 문서를 표현하고, 저장하고, 조작하는 방법을 제공한다. DOM 은 웹 페이지의 객체 지향 표현이며, 자바스크립트와 같은 스크립팅 언어를 이용해 DOM 을 수정할 수 있다.
- W3C DOM, WHATWG DOM 표준은 대부분의 브라우저에서 DOM 을 구현하는 기준이다. 많은 브라우저들이 표준 규약에서 제공하는 기능 외에도 추가적인 기능들을 제공

하기 때문에 사용자가 작성한 문서들이 각기 다른 DOM 이 적용된 다양한 브라우저 환경에서 동작할 수 있다는 사실을 항상 인지하고 있어야 한다.

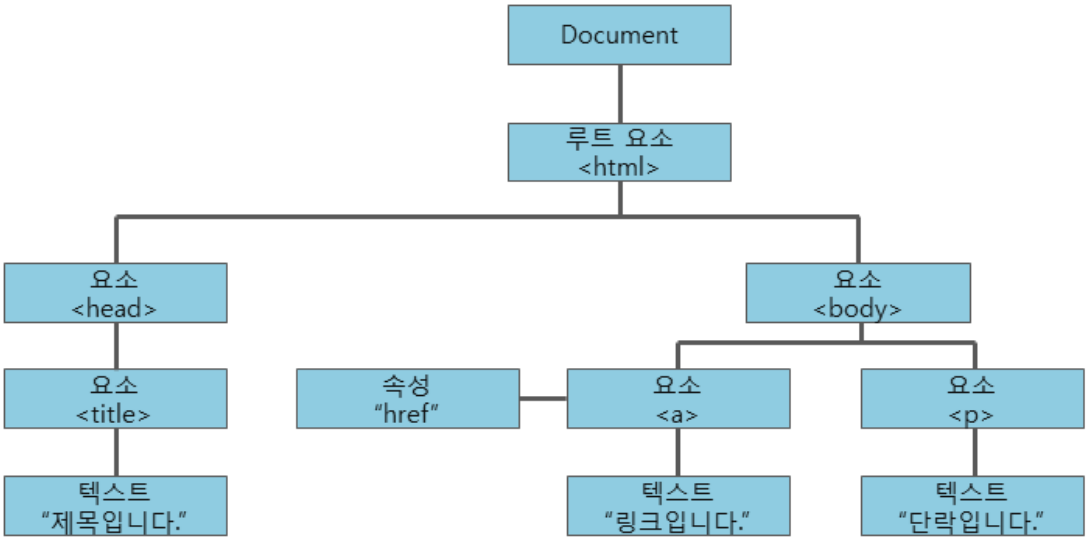
- 예제 코드
  - HTML에 출력

```
document.write("안녕하세요!")
```

  - 지정된 elements 선택

```
var paragraphs = document.getElementsByTagName("P");
// paragraphs[0] is the first <p> element
// paragraphs[1] is the second <p> element, etc.
alert(paragraphs[0].nodeName);
```

    - 표준 DOM 에서는 문서 안에서 모든 `<P>` elements 에 대한 list 를 리턴하는 `getElementsByTagName` method 를 정의하고 있다.
- 웹 페이지를 수정하거나 생성하는데 사용되는 모든 property, method, event 들은 objects 로 구성된다. 예를 들어 document object 는 document 자체를 의미하며, table object 는 HTML table 에 접근하기 위한 `HTMLTableElement` DOM 인터페이스를 구현한 것이다. 이 문서는 Gecko 기반의 브라우저에서 구현된 DOM 에 대한 object-by-object reference 를 제공한다.



- 자바스크립트는 이러한 객체 모델을 이용하여 다음과 같은 작업을 할 수 있습니다.

    - 자바스크립트는 새로운 HTML 요소나 속성을 추가할 수 있습니다.
    - 자바스크립트는 존재하는 HTML 요소나 속성을 제거할 수 있습니다.
    - 자바스크립트는 HTML 문서의 모든 HTML 요소를 변경할 수 있습니다.
    - 자바스크립트는 HTML 문서의 모든 HTML 속성을 변경할 수 있습니다.
    - 자바스크립트는 HTML 문서의 모든 CSS 스타일을 변경할 수 있습니다.
    - 자바스크립트는 HTML 문서에 새로운 HTML 이벤트를 추가할 수 있습니다.
    - 자바스크립트는 HTML 문서의 모든 HTML 이벤트에 반응할 수 있습니다.
- DOM 종류
    - W3C DOM 표준은 세 가지 모델로 구분됩니다.
        1. Core DOM : 모든 문서 타입을 위한 DOM 모델
        2. HTML DOM : HTML 문서를 위한 DOM 모델
        3. XML DOM : XML 문서를 위한 DOM 모델

Datatype 및 변수

기본 데이터타입

Type	`typeof` return value	Object wrapper
Null	`"object"`	N/A
Undefined	`"undefined"`	N/A
Boolean	`"boolean"`	`Boolean`
Number	`"number"`	`Number`
String	`"string"`	`String`
Symbol	`"symbol"`	`Symbol`

객체(Object)

- JavaScript에서의 객체는 속성의 컬렉션으로 볼 수 있습니다. 객체 리터럴 구문을 사용해 제한적으로 속성을 초기화할 수의 있고, 그 후에 속성을 추가하거나 제거할 수도 있습니다. 속성 값으로는 다른 객체를 포함해 모든 타입을 사용할 수 있으므로 복잡한 자료구조의 구축이 가능합니다. 속성은 '키' 값으로 식별하며, 키 값으로는 문자열 값이나 심볼을 사용할 수 있습니다.

- 객체란 이름(name)과 값(value)으로 구성된 프로퍼티(property)의 정렬되지 않은 집합입니다.
- 객체 예제

```
//object 선언
const person = {
  name: ['Bob', 'Smith'],
  age: 32,
  gender: 'male',
  interests: ['music', 'skiing'],
  bio: function() {
    alert(this.name[0] + ' ' + this.name[1] + ' is ' + this.age + ' years old. He
 likes ' + this.interests[0] + ' and ' + this.interests[1] + '.');
  },
  greeting: function() {
    alert('Hi! I\'m ' + this.name[0] + '.');
  }
};

//object 접근
//1. 점 표기법
person.name
person.name[0]
person.age
person.interests[1]
person.bio()
person.greeting()
//2. 괄호 표기법
person['age']
person['name']['first']
```

변수 선언과 할당

- 변수의 선언

```
var country;
let age;
```

- 변수의 할당

```
const name = hanamon;
//(const는 상수 선언 키워드이기 때문에 선언과 할당을 동시에 하지 않으면 에러가 난다. 재할당 불가능)
country = korea;
age = 16;
```

- 변수의 선언과 할당

```
let age = 16;
```

변수 종류

1. `var` : 전역 변수 선언할 때 사용, Function-scope

```
var i;  // 선언, "undefined"가 저장됨
var sum = 0;  // 선언과 초기화
var i, sum; // 한 번에 여러 개의 변수를 함께 선언할 수 있음
var i=0, sum=10, message=”Hello”;  // 선언과 초기화를 동시에 해줄 수 있음
name = "javascript";  // 선언되지 않은 변수는 전역 변수가 됨
```

- `var`문에서 변수에 초기 값을 지정하지 않는다면, 변수는 값이 설정될 때까지 `undefined` 값을 갖게 된다.
- **함수 유효범위**
  - 함수 유효범위는 어떤 함수 안에서 선언된 모든 변수는 그 함수 전체에 걸쳐 유효하다는 의미다. 이는 변수가 미처 선언되기 전에도 유효하다는 뜻이기도 하다.
- **Hoisting**
  - 자바스크립트는 변수 선언이 어느 위치에 있든 맨 처음 선언을 미리 해둡니다. 이것을 호이스팅이라고 부릅니다. 그래서 변수를 아직 선언하지 않고 사용하면 에러가 나지 않고 undefined가 출력됩니다.
  - var 선언자만 되는 것이 아닌 let, const 선언자도 호이스팅이 됩니다. 다만 let, const 선언자는 호이스팅이 되지만 값이 undefined로 초기화되지 않아 미리 변수를 사용하면 에러가 발생합니다.
  - 자바스크립트 코드는 함수 안에 있는 모든 변수를 함수 맨 꼭대기로 끌어올린 것처럼 동작한다.

```
var scope = “global”;
function f() {
 console.log(scope);  //"undefined"를 출력한다.(global이 아니다)
 var scope = “local”;
 console.log(scope);  //"local"을 출력한다.
}
=> 실제로 실행되는 코드는 이렇다.
function f() {
 var scope;  //끌어올림(hoisting)이라고 한다.(대신, 정의만 끌어올려진다)
 console.log(scope);  // "undefined"가 저장되어 있다.
 scope = "local";  // 초기화는 끌어올려지지 않는다.
```

```
 console.log(scope);
}
```

2. `const` : 상수, 변치 않는 값을 갖는 변수, 값 재할당 불가, Block-scope
   - 블록 범위의 상수를 선언한다. (let과 같은 블록 유효범위)
   - 상수의 값은 재할당할 수 없으며 다시 선언할 수도 없다.
   - 그래서 처음 선언할 때, 반드시 초기화를 해야한다.
   - 보통 대문자를 사용해서 선언한다. (강제는 아니지만 관습!)

```
const MY_NUM = 7;
```

   - **함수 hoisting**
     - 함수에서는 단지 함수 선언만 상단으로 끌어올려집니다. 함수 표현식은 그렇지 않습니다.

```
/* 함수 선언 */
foo(); // "bar"
function foo() {
  console.log('bar');
}
/* 함수 표현식 */
baz(); // TypeError: baz is not a function
var baz = function() {
  console.log('bar2');
};
```

   -

3. `let` : 지역 변수, Block-scope

```
let i;  // 선언, "undefined"가 저장됨
let sum = 0;  // 선언과 초기화
let i, sum; // 한 번에 여러 개의 변수를 함께 선언할 수 있음
let i=0, sum=10, message="Hello";  // 선언과 초기화를 동시에 해줄 수 있음
```

   - **블록 유효 범위**
     - **let**은 변수가 선언된 **블록**, **구문** 또는 **표현식** 내에서만 유효한 변수를 선언한다. 이는 **var** 키워드가 블록 범위를 무시하고 전역 변수나 함수 지역 변수로 선언되는 것과 다른 점이다.

- **프로그램이나 함수의 최상위에서는 let과 var는 서로 다르게 행동한다.**
  - var는 전역 객체의 프로퍼티를 생성하지만, let은 전역 객체의 속성 값을 생성하지 않는다.
- **hoisting되지 않습니다.**
  - 변수가 초기화(선언)되기 전에 참조할 경우 ReferenceError가 발생한다.

4. 자바스크립트에서 `var`, `let`, `const` 차이점

`var` 선언은 전역 범위 또는 함수 범위이며, `let` 및 `const`는 블록 범위이다.

`var` 변수는 범위 내에서 업데이트하고 다시 선언 할 수 있다. (재선언 O, 재할당 O)

`let` 변수는 업데이트 할 수 있지만 다시 선언 할 수는 없다. (재선언 X, 재할당 O)

`const` 변수는 업데이트하거나 다시 선언 할 수 없다. (재선언 X, 재할당 X)

두개의 공통점은 `var`와 다르게 **변수 재선언 불가능**하다.

즉, 자바스크립트에서 상수 선언은 `const`로 한다.