



# 그래픽을 위한 웹 기술 기본 I

Web technology basics for graphics

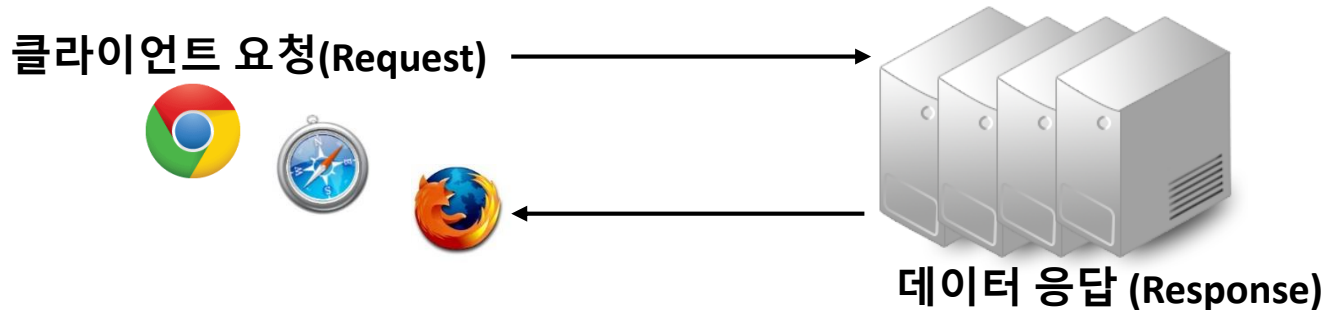
# 웹? Web?



- 웹 브라우저를 통해 전 세계를 대상으로 빠르게 전시
- 운영체제(Windows, Linux, OS X) 및 장비 종류(노트북, PC, 스마트폰, 태블릿)에 관계 없이 브라우저만으로 열람 및 경험
- 특정 업체 소프트웨어 / 플러그인을 사용하지 않아도 결과물을 어디서든지 확인 가능

# 웹? Web?

- 웹 기본 요청 개념



- URL(Uniform Resource Locator) 과 URI(Uniform Resource Identifier) 구분

URL : 인터넷 상 올려진 자료들이 있는 주소 구성

URI : 인터넷에 있는 자원을 나타내는 유일한 주소

URI



https://search.naver.com/search.naver → URL  
?where=nexearch&query=한성대학교&sm=top\_h ty&fbm=0&ie=utf8

## 1. 로컬 서버 환경 구축 후 테스트

### - 필요성

간혹 외부 데이터 파일을 불러오기 시도하면 이유 없이 실패

왜? 신뢰할 수 없는 제 3의 사이트(클라이언트)에서 스크립트나 별도의 파일을 불러오는 일을 방지하기 위한 브라우저 보안  
(크로스 도메인; cross-domain)

APACHE  
HTTP SERVER



NGINX

LAMP

MAMP

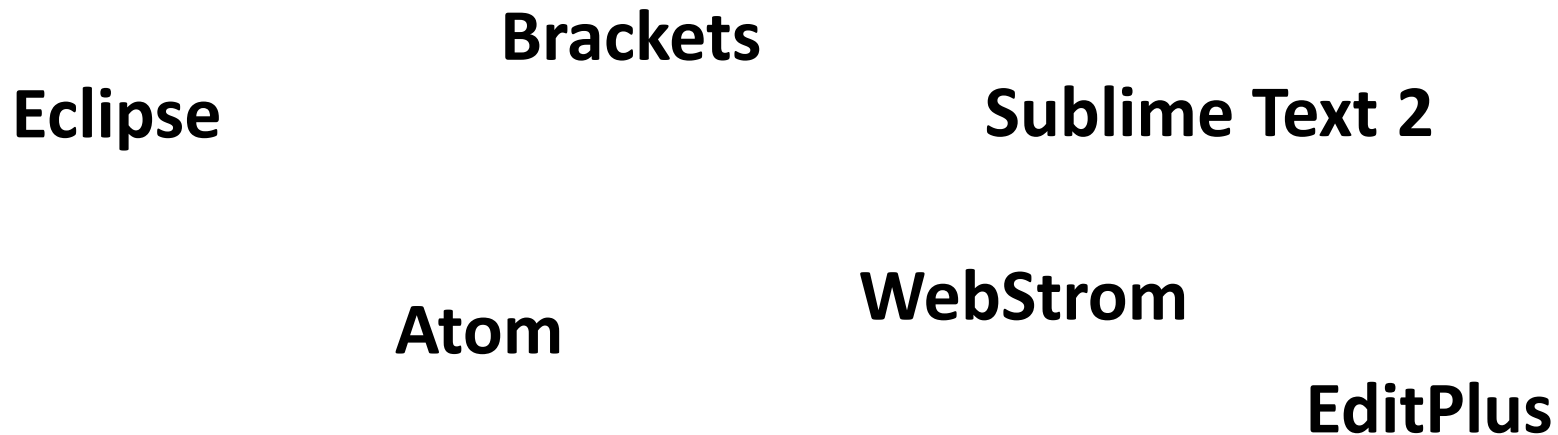
WAMP

# 개발 환경 설정

---

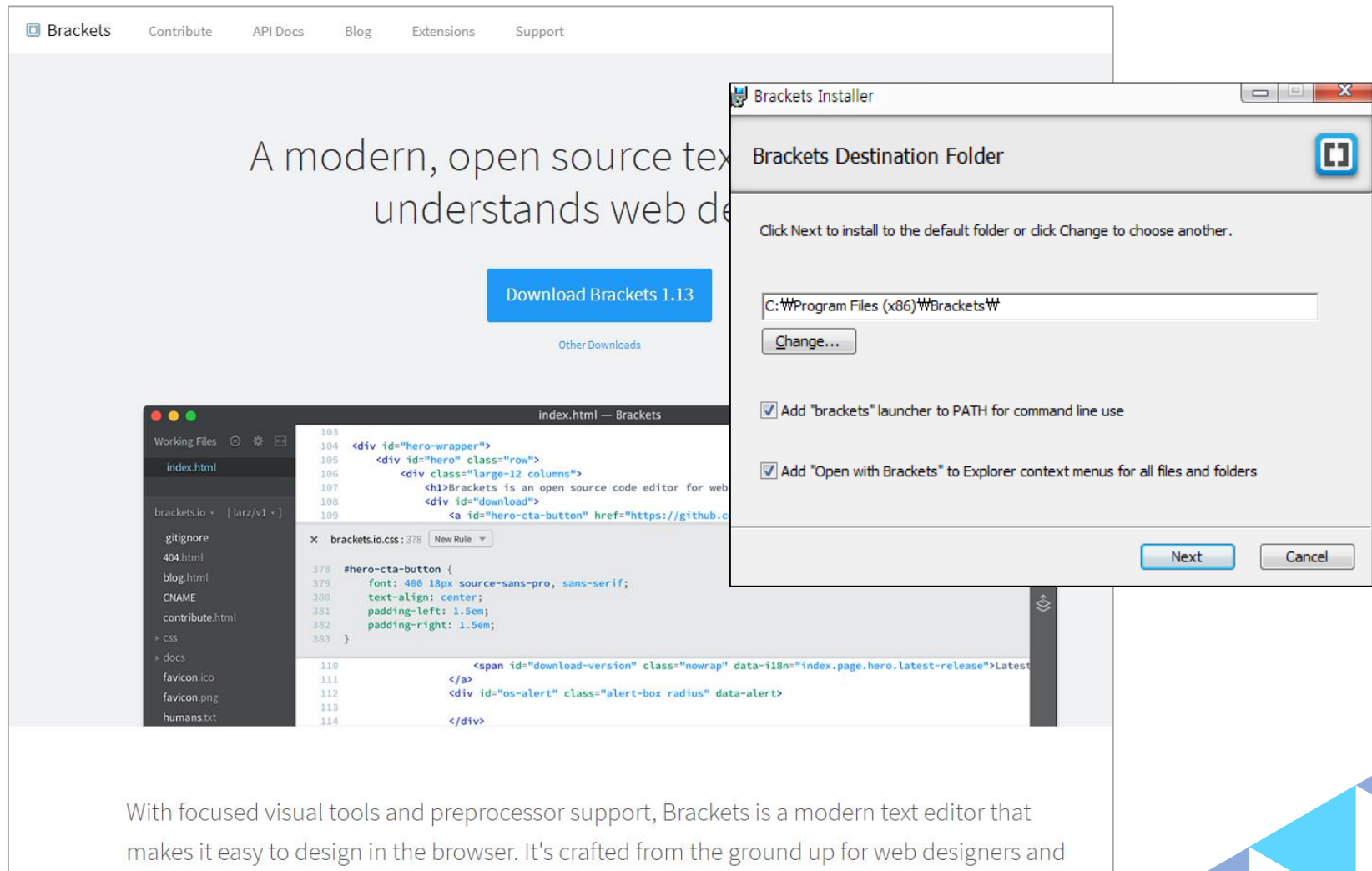
## 2. 클라이언트 도구만을 활용한 테스트

- 내부 데이터만을 가지고 테스트를 할 경우 권장
- 편집 에디터만 있으면 가능하므로 설정 편리
- 하지만 웹 개발이 편리한 통합 개발 환경을 사용하기 권장



본 수업에 예제는 Brackets 사용

- 다운로드 / 설치

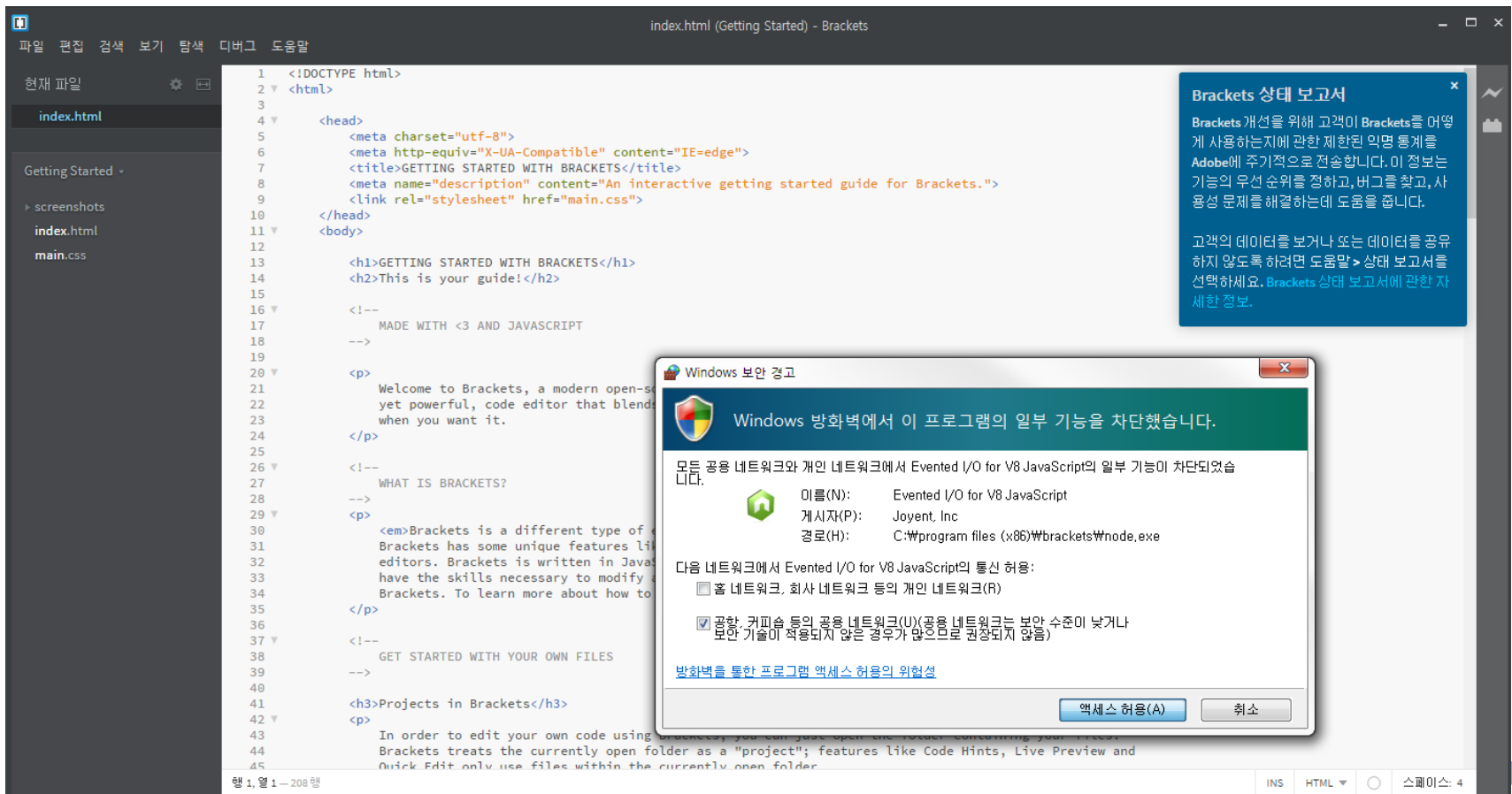


The image displays the Brackets website and the Brackets Installer window. The website features the text "A modern, open source text editor that understands web development" and a "Download Brackets 1.13" button. The Brackets Installer window is titled "Brackets Destination Folder" and shows the installation path "C:\Program Files (x86)\Brackets\". It includes checkboxes for "Add 'brackets' launcher to PATH for command line use" and "Add 'Open with Brackets' to Explorer context menus for all files and folders".

With focused visual tools and preprocessor support, Brackets is a modern text editor that makes it easy to design in the browser. It's crafted from the ground up for web designers and

- Brackets 처음 실행

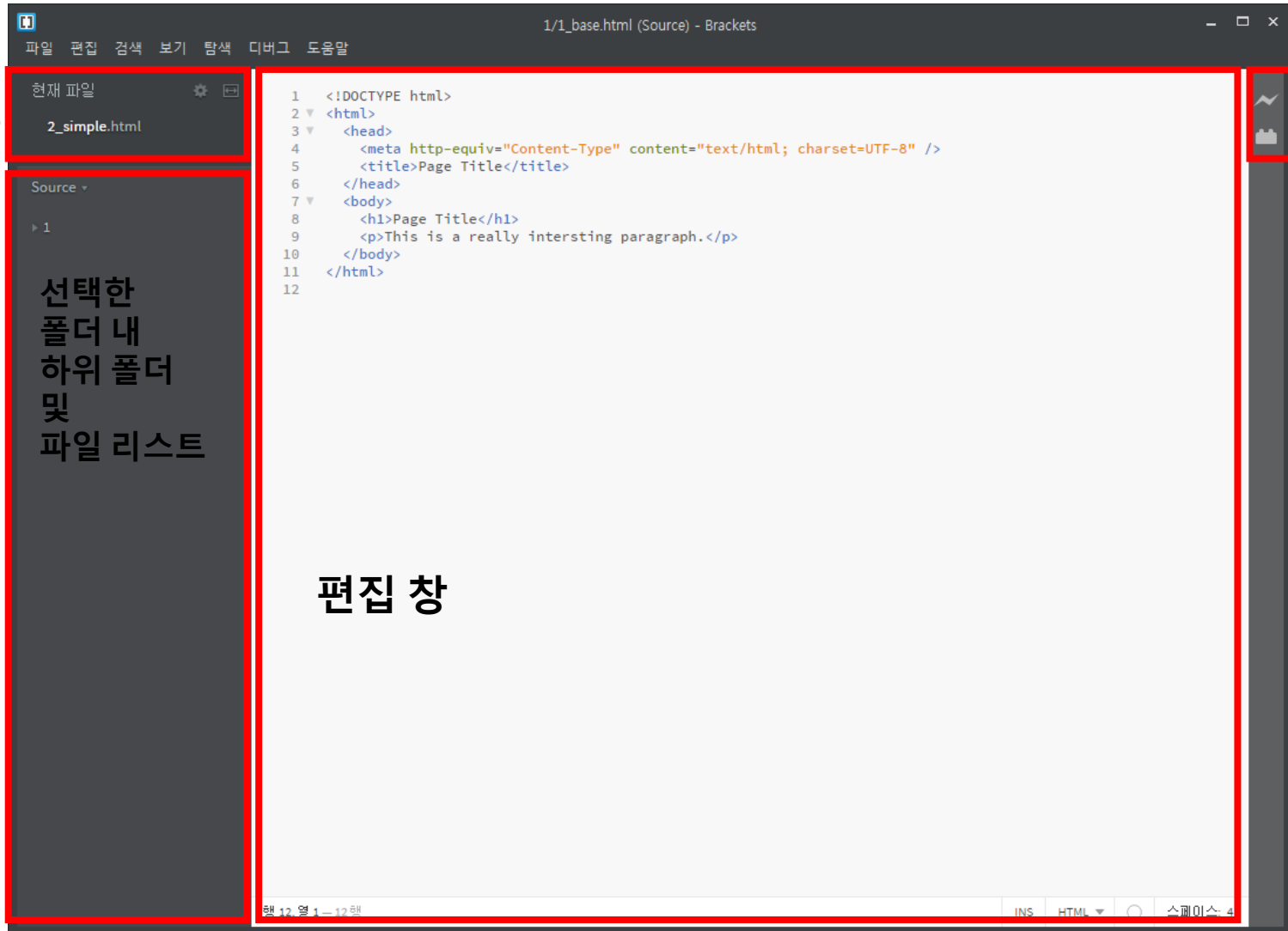
## 처음 실행 시 반드시 Windows 방화벽 액세스 허용



# 개발 환경 설정

- Brackets 인터페이스

열기 한 파일  
리스트  
및  
현재 파일

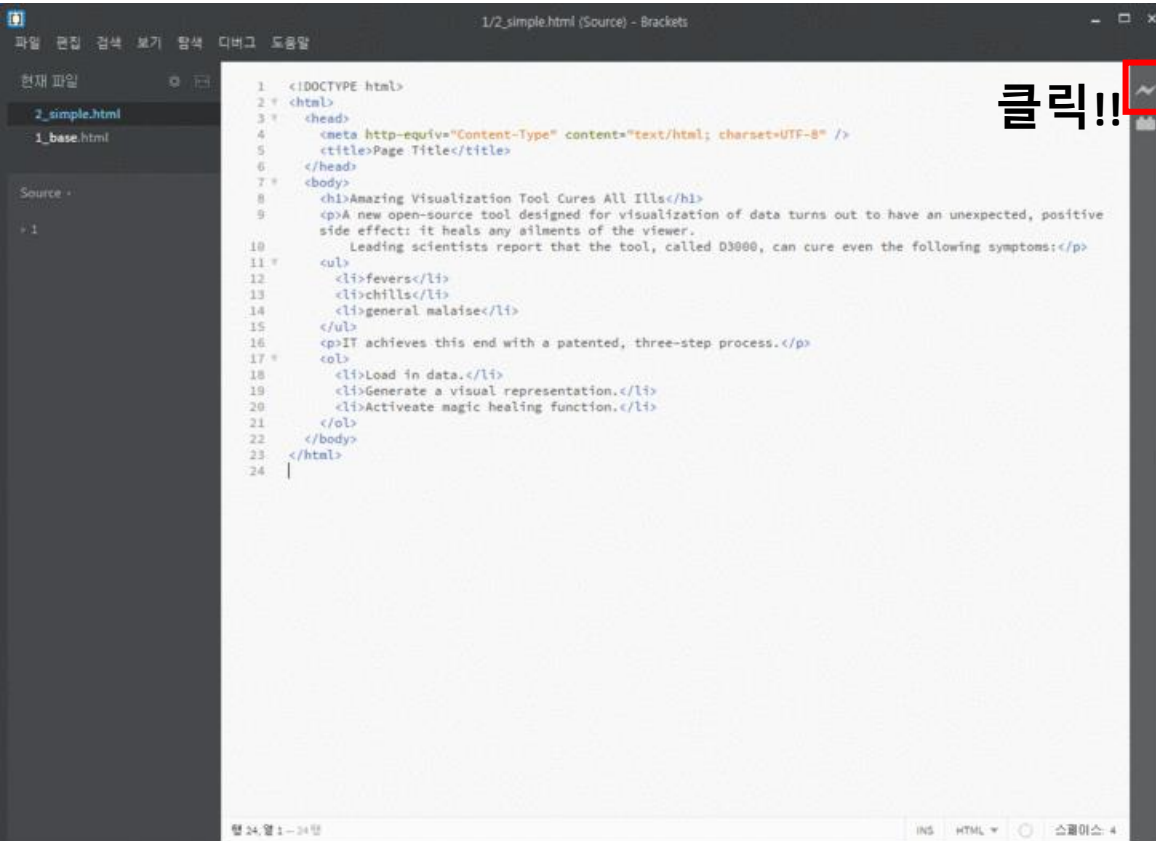


실시간  
브라우저  
확인  
및  
플러그인  
설치



# 개발 환경 설정

- 사용법 : Brackets 소스코드 실시간 편집 (움직이는 그림)



움직이는 그림

## ※ 주의사항

자바스크립트는 실시간 편집 불가능  
개발자 콘솔창(F12)를 열 경우 실시간 편집 종료

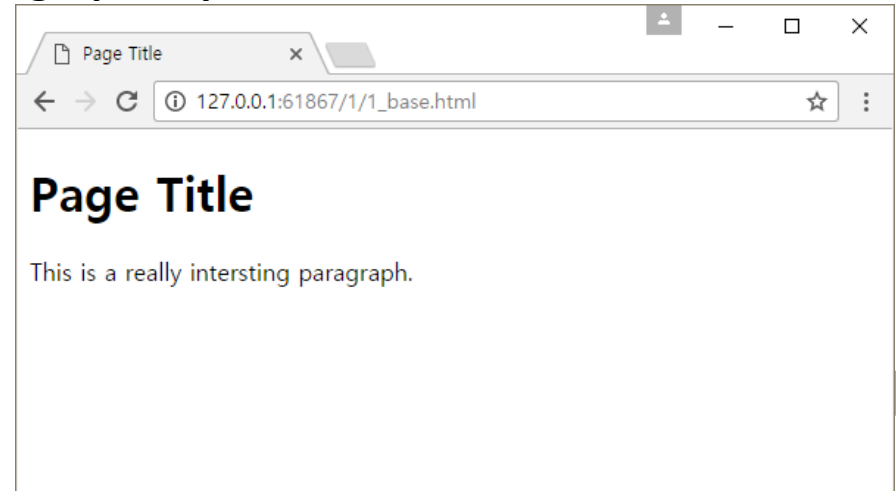
테스트 용도로만 사용할 것

- Brackets 유용 플러그인
  - 테마 설정 : stripper
  - 아이콘 : Brackets File Icons
  - 함수 리스트 뷰 : Brackets Outline List
  - 함수 폴딩 : Code Folding
  - 키보드로 파일간 이동 : Number Tabs
  - HTML5 템플릿 : HTML5 Template

# HTML

하이퍼텍스트 마크업 언어(Hypertext Markup Language)  
웹 브라우저를 위해 콘텐츠를 구조화하는데 사용

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>Page Title</h1>
    <p>This is a really intersting paragraph.</p>
  </body>
</html>
```



# HTML (태그)

---

- 기본 문서 요소

**<!DOCTYPE html>**

표준 문서 타입 선언. 문서 최 상단 존재

---

**<html>**

문서의 전체 HTML 콘텐츠에 대한 부모

---

**<head>**

문서에 대한 모든 부가 정보 작성

---

**<title>**

문서 제목으로 브라우저 창 상단 표시 (head 내부 작성)

---

**<body>**

페이지에 시각적으로 표현되는 주 콘텐츠

# HTML (태그)

---

- 많이 사용되는 문서 요소

**<h1>, <h2>, <h3>, <h4>**

단계별 제목 크기

---

**<p>**

문서 글 단락

---

**<ul>, <ol>, <li>**

목록 정의 (ul : 번호가 없는 목록, ol: 번호가 있는 목록)

---

**<em>, <strong>**

내용을 강조하기 위한 태그 (em: 강조, strong: 매우 강조)

---

**<a>**

링크 사용하지만 링크를 지정하지 않으면 밑줄과 파란색 텍스트로 렌더링

# HTML (태그)

---

- 많이 사용되는 문서 요소

**<div>**

문서 안 임의로 분할한 영역으로 문서 요소를 묶고 담는 데 사용

---

**<span>**

p 태그와 비슷한 문서 글 단락, 더 좁은 문단

# HTML (속성)

- 시작하는 태그 안 속성 삽입 가능

```
<tagname property="value"></tagname>
```

- class & id

- 가장 중요한 속성으로 콘텐츠에서 구체적인 한 부분 찾을 때 사용
- 특히, 자바스크립트 코드로 문서 요소를 알아낼 때 사용
- class의 경우 미리 정해놓은 다양한 스타일 적용 가능

## Class

- 한 문서에 다중 적용 가능
- 한 태그에 여러 개 class 지정 가능

```
<p class="uplifting">brilliant paragraph</p>  
<p class="uplifting awesome">  
  Awe-inspring paragraph  
</p>
```

## id

- 한 문서에 무조건 하나의 id
- 한 태그에 하나 지정

```
<div id="content">  
<div id="visualization"> </div>  
<div id="button"> </div>
```

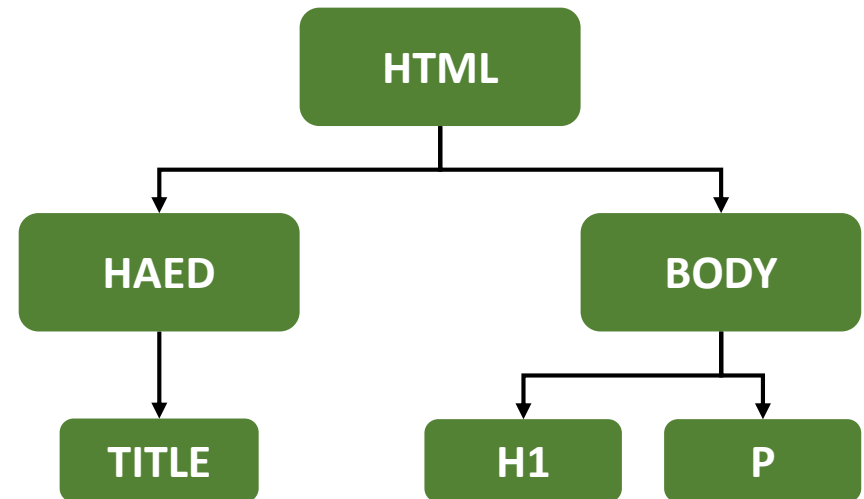
# HTML (DOM)

- 문서 객체 모델 (DOM : document Object Model)

HTML과 XML 과 같이 구조화된 문서를 표현하는 형식

```
<html>
  <head>
    <title>HOME</title>
  </head>
  <body>
    <h1>Breaking News</h1>
    <p></p>
  </body>
</html>
```

## 계층적 구조



- H1과 p1은 서로 형제 문서 요소
- body는 h1, p의 부모 문서 요소
- 이 문서의 모든 문서 요소는 html의 자손



## 문서 객체 모델을 시각적 표현으로 꾸미는데 사용

```
body{  
  background-color : white;  
  color : black;  
}
```

- 선택자 (selector) / 프로퍼티 (property)

```
selector{  
  property : value;  
  property : value;  
  property : value;  
  ...  
}
```

같은 프로퍼티 묶음은 콤마로 구분한 선택자에 한 번에 적용 가능

```
selectorA,  
selectorB,  
selectorC{  
  property : value;  
  ...  
}
```

# CSS

- 문서 요소를 찾을 때 , css 방식의 선택자를 자주 사용
- 선택자 종류
  - 타입 선택자: 가장 간단한 형태로 동명의 DOM 문서 요소를 찾음
  - 자손 선택자: 어떤 문서 요소를 감싸인 문서 요소들을 찾음
  - 클래스 선택자: 문서 요소 종류에 관계없이 지정한 클래스가 할당된 문서 요소를 모두 찾음
  - 아이디(id) 선택자: 주어진 id가 있는 문서요소 하나를 선택

**h1**  
**p1**  
**strong**  
**div**  
...

**.caption**  
**.label**  
...

**#header**  
**#nav**  
**#export**

- 여러 클래스를 가진 문서 요소를 지정할 수 있음

`.bar.highlight` /\* “bar”클래스와 “highlight”클래스를 모두 가지고 있는 경우

- 좀 더 세밀하게 선택하기 위해 서로 이을 수 있음

`div.sidebar` /\* “sidebar” 클래스를 가진 div를 모두 선택 \*/

`#button.on` /\* id가 “button”인 문서요소를 선택, 이 때 “on” 클래스가 적용되어 있을 때만 선택 \*/

- 스타일 참조하기

HTML 문서 내 삽입

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      p{
        font-size: 24px;
        font-weight: bold;
        background-color: red;
        color: white;
      }
    </style>
  </head>
  <body>
    <p>If I were to ask you, as a mere paragraph, would you say that I have style?</p>
  </body>
</html>
```



- 스타일 참조하기

CSS 파일을 삽입

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="4_cssReference_atFile.css">
  </head>
  <body>
    <p>If I were to ask you, as a mere paragraph, would you say that I have style?</p>
  </body>
</html>
```

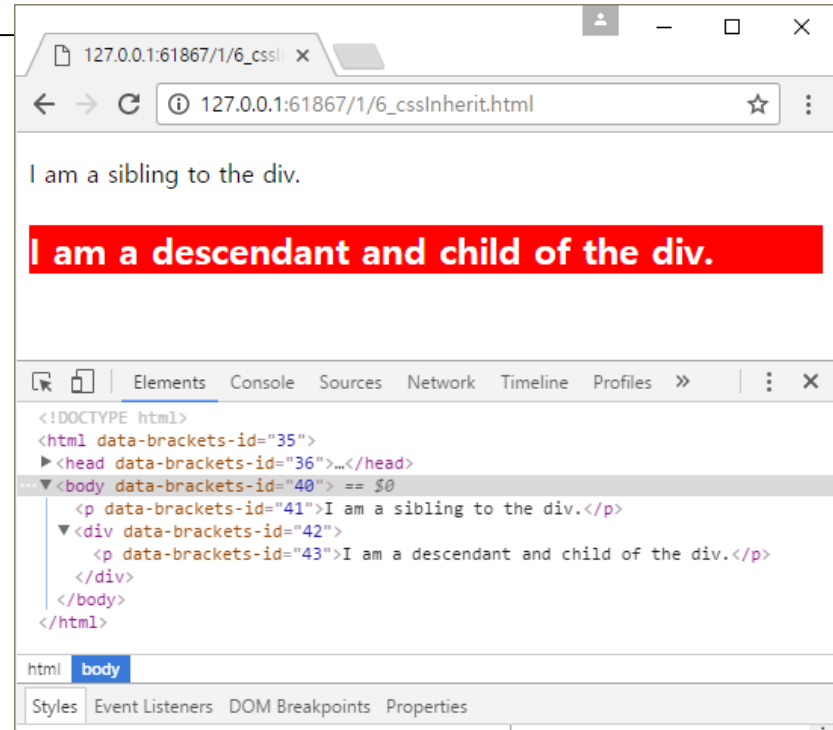
문서에 인라인으로 삽입

```
<!DOCTYPE html>
<html>
  <head>
  </head>
  <body>
    <p style="color: blue; font-size:48; font-style: italic;">If I were to ask you, as a mere
paragraph, would you say that I have style?</p>
  </body>
</html>
```

# CSS(상속, 연속성, 구체성)

- 대부분 스타일 프로퍼티는 별 다른 설정 없이 자식 문서 요소에 상속

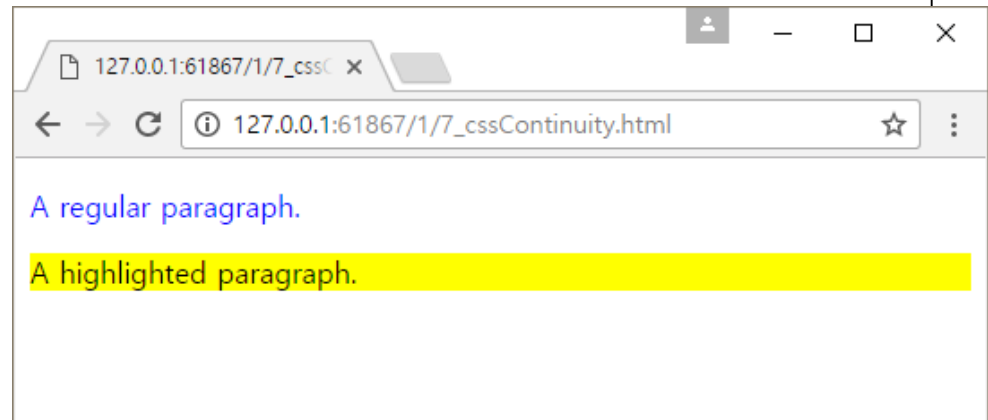
```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <style type="text/css">
      div {
        background-color: red;
        font-size: 24px;
        font-weight: bold;
        color: white;
      }
    </style>
  </head>
  <body>
    <p>I am a sibling to the div.</p>
    <div>
      <p>I am a descendant and child of the div.</p>
    </div>
  </body>
</html>
```



# CSS(상속, 연속성, 구체성)

- 선택자는 위에서 아래로 쏟아지며 문서 요소에 연결
- 여러 선택자를 한 문서 요소에 적용할 때, 일반적으로 나중에 지정한 스타일 규칙이 이전 규칙을 덮어씀

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title> </title>
    <style type="text/css">
      p {
        color: blue;
      }
      p.highlight {
        color: black;
        background-color: yellow;
      }
    </style>
  </head>
  <body>
    <p>A regular paragraph.</p>
    <p class="highlight">A highlighted paragraph.</p>
  </body>
</html>
```



# 자바스크립트

브라우저에 이미 떠 있는 페이지의 DOM을 조작해서 페이지를 동적으로 만들 수 있는 스크립트 언어

- 특징

- 브라우저에 의해 실행되는 클라이언트 사이드 스크립트 언어
- 명령어를 사용하는(imperative) 절차적(procedural) 언어로 구조적 프로그래밍(structured programming) 가능
- 객체(object)의 정의와 사용을 지원하는 객체 기반(object-based) 언어
- 실행 시간에 자료형을 검사하는 동적 타이핑(dynamic typing) 언어
- 동적인 웹 페이지와 향상된 사용자 인터페이스를 쉽게 구현 가능



# 자바스크립트

---

- 기본 구조

형식1 : `<script type = "text/javascript">`  
    JavaScript\_코드  
    `</script>`

형식2 : `<script language = "JavaScript">`  
    JavaScript\_코드  
    `</script>`

형식3 : `<script type = "text/javascript" src = "JavaScript 파일의 url">`  
    JavaScript\_코드  
    `</script>`

형식4 : `<script language = "JavaScript" src = "JavaScript 파일의 url">`  
    JavaScript\_코드  
    `</script>`

# 자바스크립트

- 자바스크립트는 **웹 스크립트 언어**
- 스크립트 언어는 **가벼운 프로그래밍 언어**
- 자바스크립트 프로그래밍은 HTML 페이지에 **삽입** 할 수 있음
- **모든 모던 웹 브라우저에서** 실행 할 수 있음
- 배우기 **쉬움**



## Writing Into HTML Output

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>
```

JavaScript can write directly into the HTML output stream:

```
</p>
```

```
<script>
```

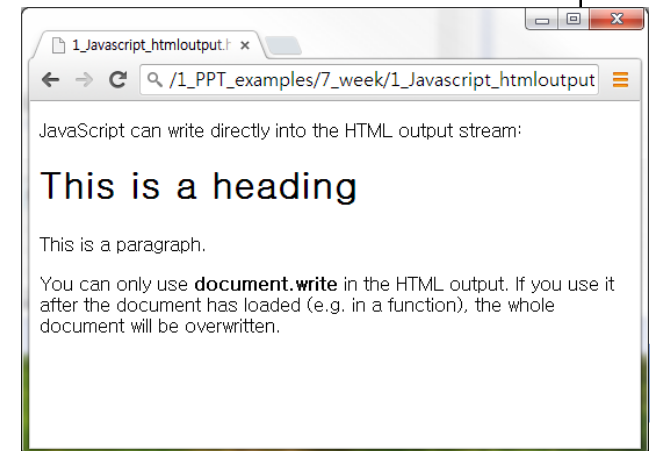
```
document.write("<h1>This is a heading</h1>");
```

```
document.write("<p>This is a paragraph.</p>");
```

```
</script>
```

```
</body>
```

```
</html>
```



# 자바스크립트

Alert Event [http://www.w3schools.com/tags/ref\\_eventattributes.asp](http://www.w3schools.com/tags/ref_eventattributes.asp)

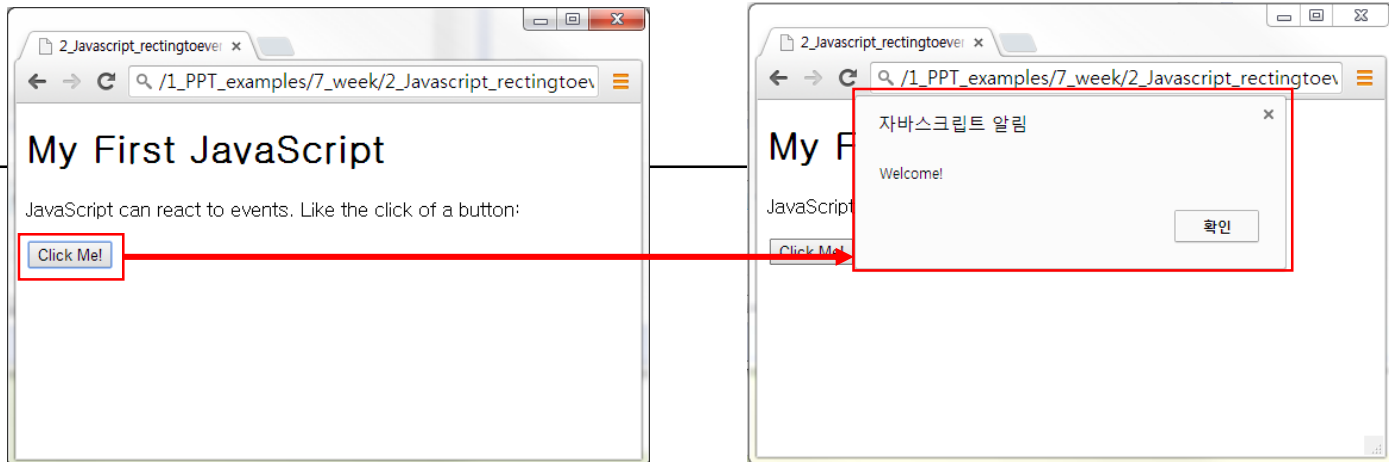
```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>

<p>
JavaScript can react to events. Like the click of a button:
</p>

<button type="button" onclick="alert('Welcome!')">Click Me!</button>

</body>
</html>
```



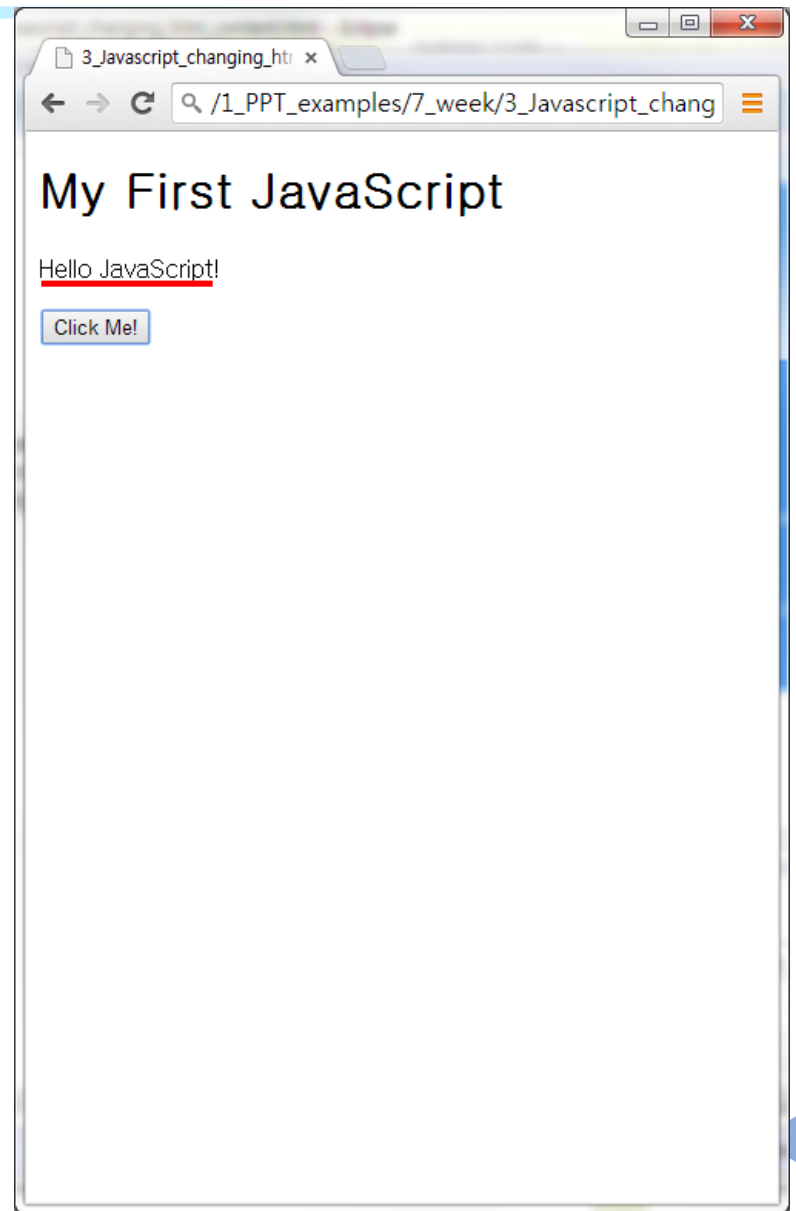
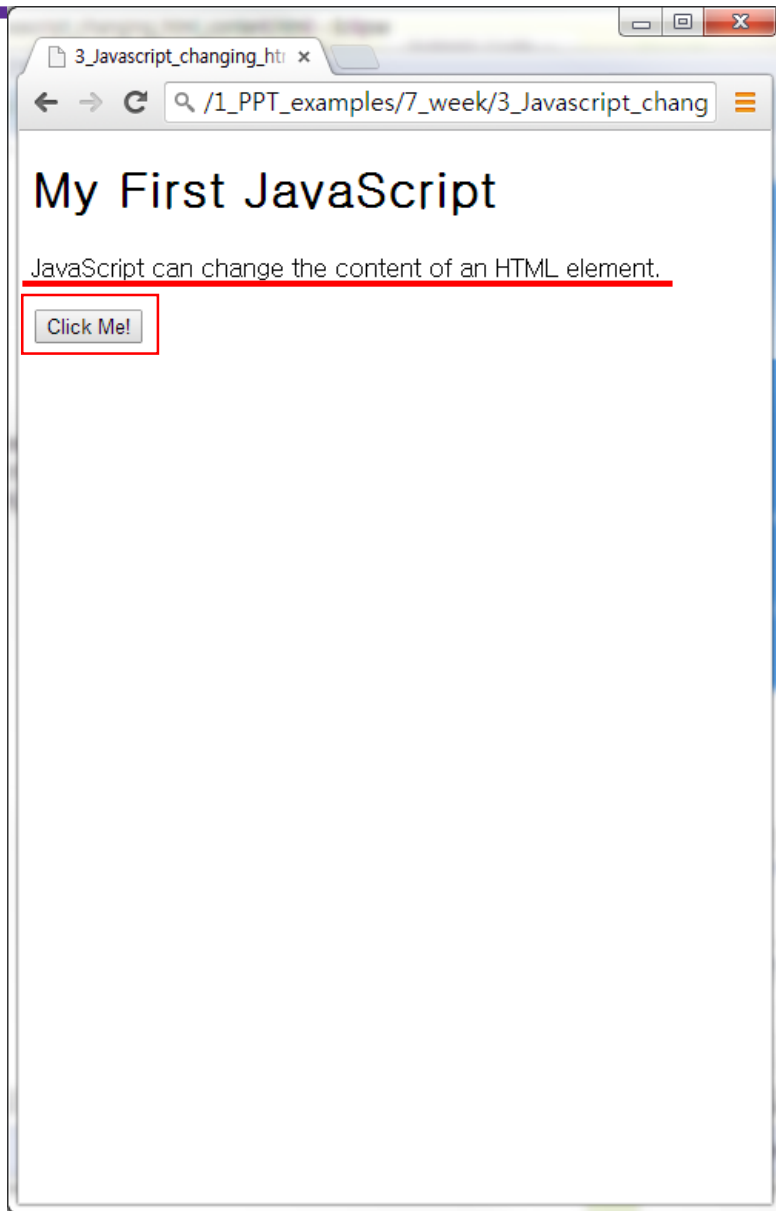
# 자바스크립트

## HTML 요소 내용 변경

자바스크립트를 사용하여 HTML요소의 내용을 조작 하는 것은 매우 일반적

```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>
<p id="demo">
JavaScript can change the content of an HTML element.
</p>
<script>
function myFunction(){
x=document.getElementById("demo"); // Find the element (DOM)
x.innerHTML="Hello JavaScript!"; // Change the content
}
</script>
<button type="button" onclick="myFunction()">Click Me!</button>
</body>
</html>
```

# 자바스크립트



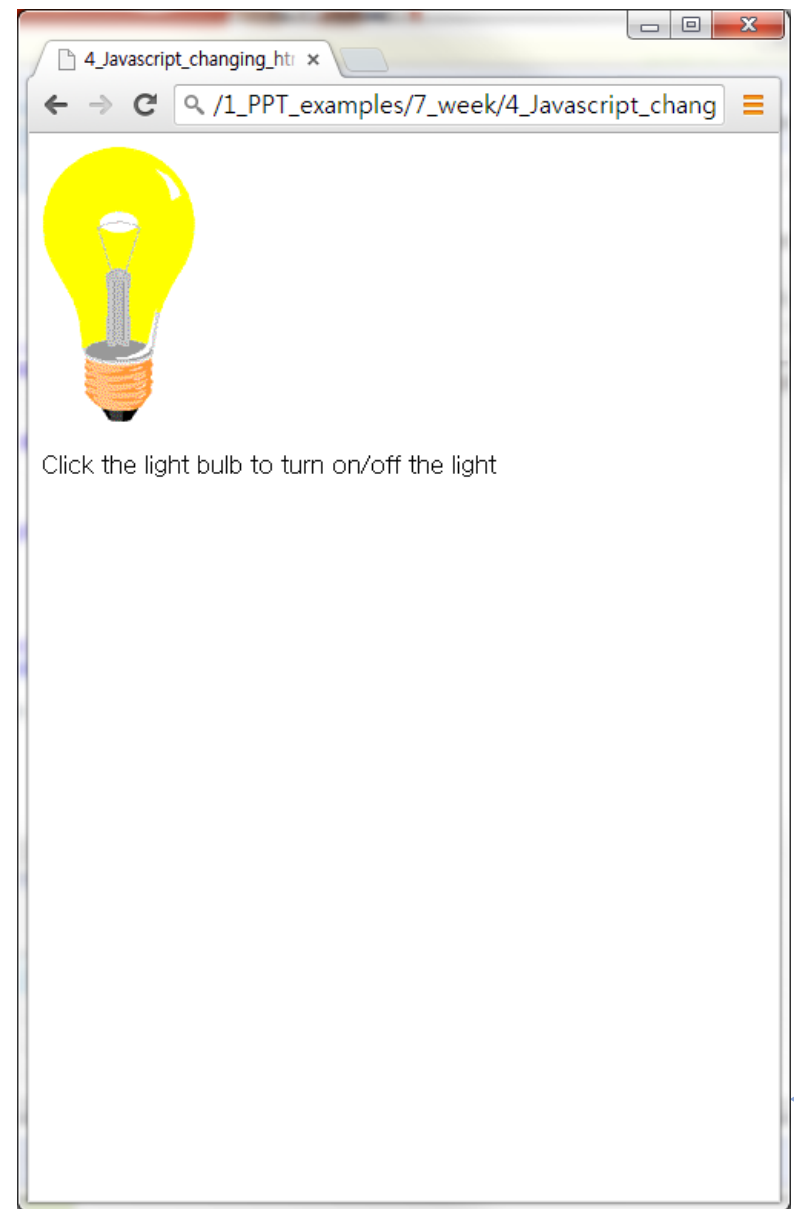
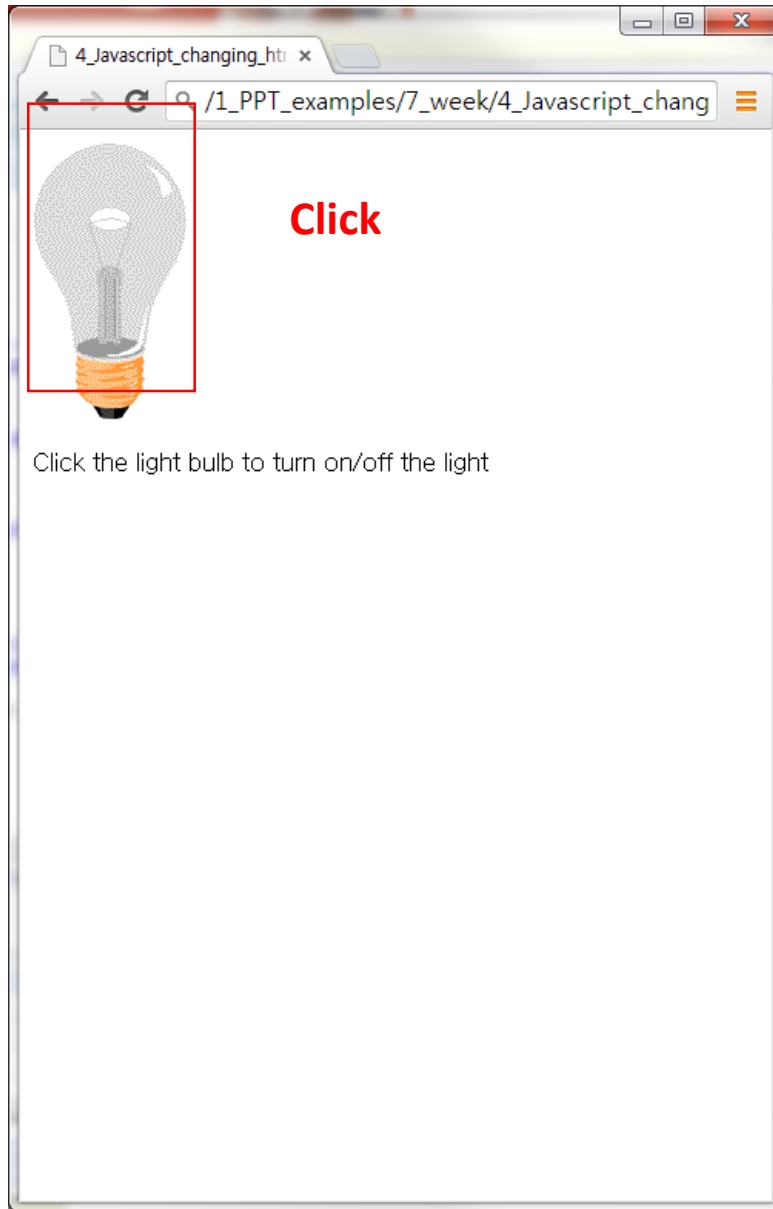
# 자바스크립트

## HTML 이미지 변경

```
<!DOCTYPE html>
<html>
<body>
<script>
function changeImage(){
element=document.getElementById('myimage');
if (element.src.match("bulbon")){
    element.src="http://113.198.80.189/web/image/pic_bulboff.gif";
}
else{
    element.src="http://113.198.80.189/web/image/pic_bulbon.gif";
}
}
</script>

<p>Click the light bulb to turn on/off the light</p>
</body>
</html>
```

# 자바스크립트



# 자바스크립트

## HTML 스타일 변경

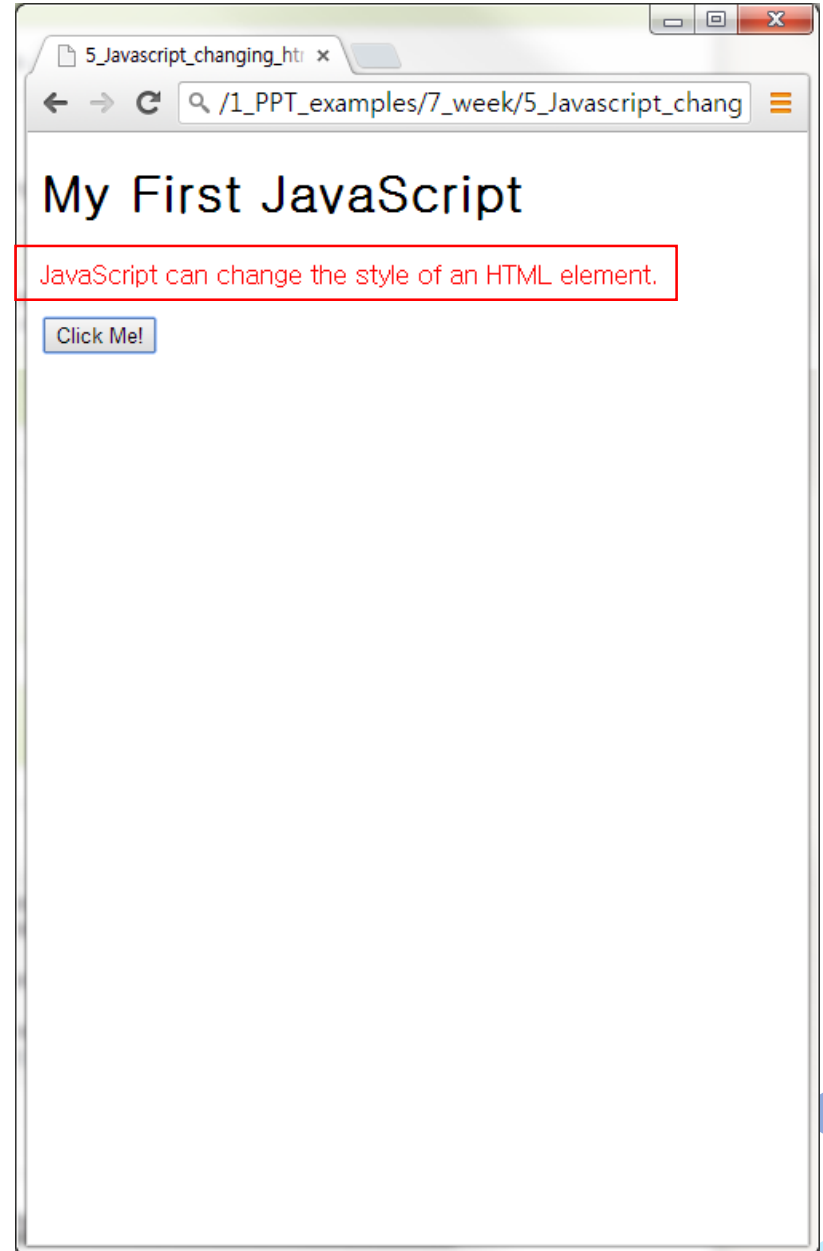
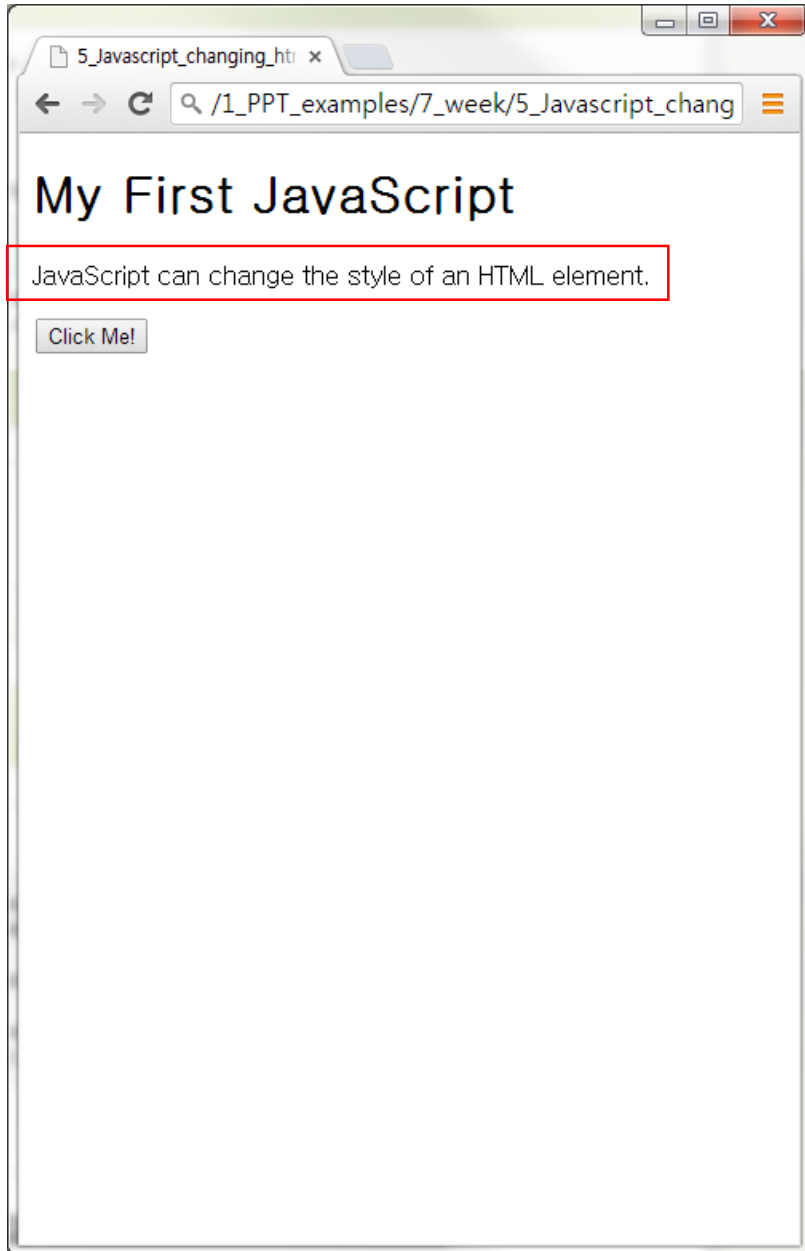
```
<!DOCTYPE html>
<html>
<body>
<h1>My First JavaScript</h1>
<p id="demo">
JavaScript can change the style of an HTML element.
</p>

<script>
function myFunction(){
x=document.getElementById("demo") // Find the element
x.style.color="#ff0000";          // Change the style
}
</script>

<button type="button" onclick="myFunction()">Click Me!</button>
</body>
</html>
```



# 자바스크립트



# 자바스크립트

1. HTML 태그 안에 `<script>` 태그와 `</script>`를 삽입하여 작성  
(`<body>`과 `<head>` 부분 모두 삽입 가능)

2. 외부 파일로 작성

Body 태그 안에 작성 하는 경우

```
<!DOCTYPE html>
<html>
<body>
...
<script>
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph</p>");
</script>
...
</body>
</html>
```

# 자바스크립트

## Head 태그 안에 작성 하는 경우

```
<!DOCTYPE html>
<html>
<head>
<script>
    document.write("<h1>This is a heading</h1>");
    document.write("<p>This is a paragraph</p>");
</script>
</head>
<body>
...
</body>
</html>
```

## 외부 파일을 이용하여 작성 하는 경우

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript" src="hello.js"></script>
</head>
...
```

# 자바스크립트

## 함수와 이벤트(head and body)

```
...
<head>
<script>
    function myFunction(){
        alert("hello");
    }
</script>
</head>
<body>
    <button type="button" onclick="myfunction()">Try it</button>
</body>
...
```

```
...
<body>
    <button type="button" onclick="myfunction()">Try it</button>
<script>
    function myFunction(){
        alert("hello");
    }
</script>
</body>
```

# 자바스크립트

## JavaScript Keywords

<b>break</b>	<b>else</b>	<b>instanceof</b>	<b>true</b>	<b>case</b>	<b>false</b>	<b>new</b>	<b>try</b>
<b>catch</b>	<b>finally</b>	<b>null</b>	<b>typeof</b>	<b>continue</b>		<b>for</b>	<b>return</b>
<b>var</b>	<b>default</b>	<b>function</b>	<b>switch</b>	<b>void</b>	<b>delete</b>	<b>if</b>	<b>this</b>
<b>while</b>	<b>do</b>	<b>in</b>	<b>throw</b>	<b>with</b>			

## JavaScript reservation Keywords

<b>abstract</b>	<b>enum</b>	<b>int</b>	<b>short</b>	<b>Boolean</b>	<b>export</b>
<b>interface</b>	<b>static</b>	<b>byte</b>	<b>extends</b>	<b>long</b>	<b>super</b>
<b>char</b>	<b>final</b>	<b>native</b>	<b>synchronized</b>	<b>class</b>	<b>float</b>
<b>package</b>	<b>throws</b>	<b>const</b>	<b>goto</b>	<b>private</b>	<b>transient</b>
<b>debugger implements</b>		<b>protected</b>	<b>volatile</b>	<b>double</b>	
<b>import</b>	<b>public</b>				

문장 끝 세미콜론(;)이 없어도 행을 바꾸면 문장으로 취급하기 때문에 실행에 문제는 없음.  
하지만 대부분 프로그래밍 언어에서 세미콜론을 입력하므로 관례상 입력

# 자바스크립트

자바스크립트는 일반적으로 HTML 요소를 조작하는데 사용

## HTML 요소 유지보수

자바스크립트를 이용하여 HTML 요소를 접근 하기 위해서는 요소의  
아이디로 접근 가능 : `document.getElementById(id)`

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p id="demo"> </p>
<script>
    document.getElementById("demo").innerHTML="My First JavaScript";
</script>
</body>
</html>
```



`document.getElementById("demo").innerHTML`

→ `<p id="demo"> </p>`

# 자바스크립트

## HTML 문서에 자바스크립트로 문서 작성

직접 Body 안에 write 함수를 사용하여 출력

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>

<script>
  document.write("<p>My First JavaScript</p>");
</script>

</body>
</html>
```



# 자바스크립트

## 대소문자 구분

- 자바스크립트는 HTML과 다르게 대소문자를 구분

ex)

함수 : `getElementByld` ≠ `getElementbyID`

변수 : `myVariable` ≠ `MyVariable`

## 여백 무시

- 자바스크립트는 여백을 무시

ex)

```
var person="Hege";
```

```
var person = "Hege";
```

## 코드 라인 변경 방법

- 백슬러시를 이용하여 문자열 코드라인을 변경 하여 작성 할 수 있음

ex)

```
document.write("Hello World");
```

```
document.write \  
("Hello World");
```



# 자바스크립트

주석은 자바 스크립트가 실행되지 못하게 하거나 소스 코드 설명 시 사용

## 싱글 라인 주석

```
// Write to a heading:
```

```
document.getElementById("myH1").innerHTML="Welcome";
```

```
//document.getElementById("myH1").innerHTML="Welcome";
```

```
var x=5;      // declare x and assign 5 to it
```

## 멀티 라인 주석

```
/*
```

```
The code below will write to a heading and to a paragraph,  
and will represent the start of my homepage:
```

```
*/
```

# 자바스크립트

String, Number, Boolean, Array, Object, Null, Undefined.

## Number

```
5
6
5+6
3.14;
```

## String

```
"Hello"
" JS"
"Hello JS"
"a"
```

## Boolean

```
true;
false;
```

## Array

```
[1,2,3]
["a","b","c"]
[1,"a",2,"b"]
```

## Object

```
{
  f_name:"john",
  l_name:"doe",
  id      :5566
}
```

## Undefined and Null

```
Undefined
: no value
Null
: emptied
```

# 자바스크립트

자바스크립트 변수들은 정보를 저장하기 위한 “containers”

수학과 마찬가지로 자바스크립트 변수는 변수 값( $x=5$ ) 또는 식( $z=x+y$ )을 표현 할 수 있음

변수 값은 짧은 알파벳 또는 설명하는 단어로 가능

- 변수 이름은 **문자**로 시작해야 함 ( name, age ) – O, ( 1name ) – X
- 문자 외에 시작 가능한 것으로는 **\$**와 **\_**가 있음 (**하지만 사용하지 x**)
- **대소문자**를 구분함 (y 와 Y는 다른 변수)

```
var x=5;  
var y=6;  
var z=x+y;  
var textVar="hi"
```

# 자바스크립트

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
  var carname="Volvo";
  document.getElementById("demo").innerHTML=carname;
}
</script>
</head>
<body>

<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

</body>
</html>
```