# HTML/CSS

핀테크 인턴십 코스

Hyo-Jeong Kim, Ph.D.

# Structure of Web Pages 웹페이지 구조

Web Design

Web Programming

| HTML | CSS (Cascading Style Sheets) | JavaScript & Python |
| --- | --- | --- |
| • Structural layer<br>• Organize content like pictures and words in a meaningful way. **그림이나 글 같은 내용을 의미있는 방식으로 구성** | • Presentational layer<br>• Make the content in the HTML look good. **HTML 안에 있는 내용을 보기 좋게 만듬** | • Behavioral layer<br>• Bring a web page to life so it interacts with web visitors. **웹 방문자들과 상호작용 할 수 있음** |

# Code Editors for HTML and CSS
# 코드편집기

*Free Program*

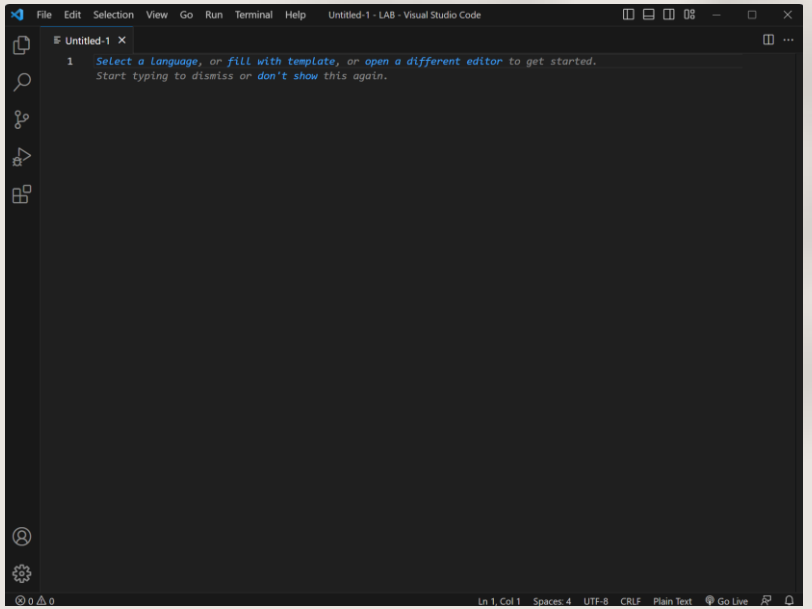
- Visual Studio Code
- Atom
- Sublime
- Brackets
- Notepad++
- EditPlus

*Commercial Software*

- IntelliJ
- WebStorm
- DreamWeaver

# Visual Studio Code **비주얼 스튜디오 코드**

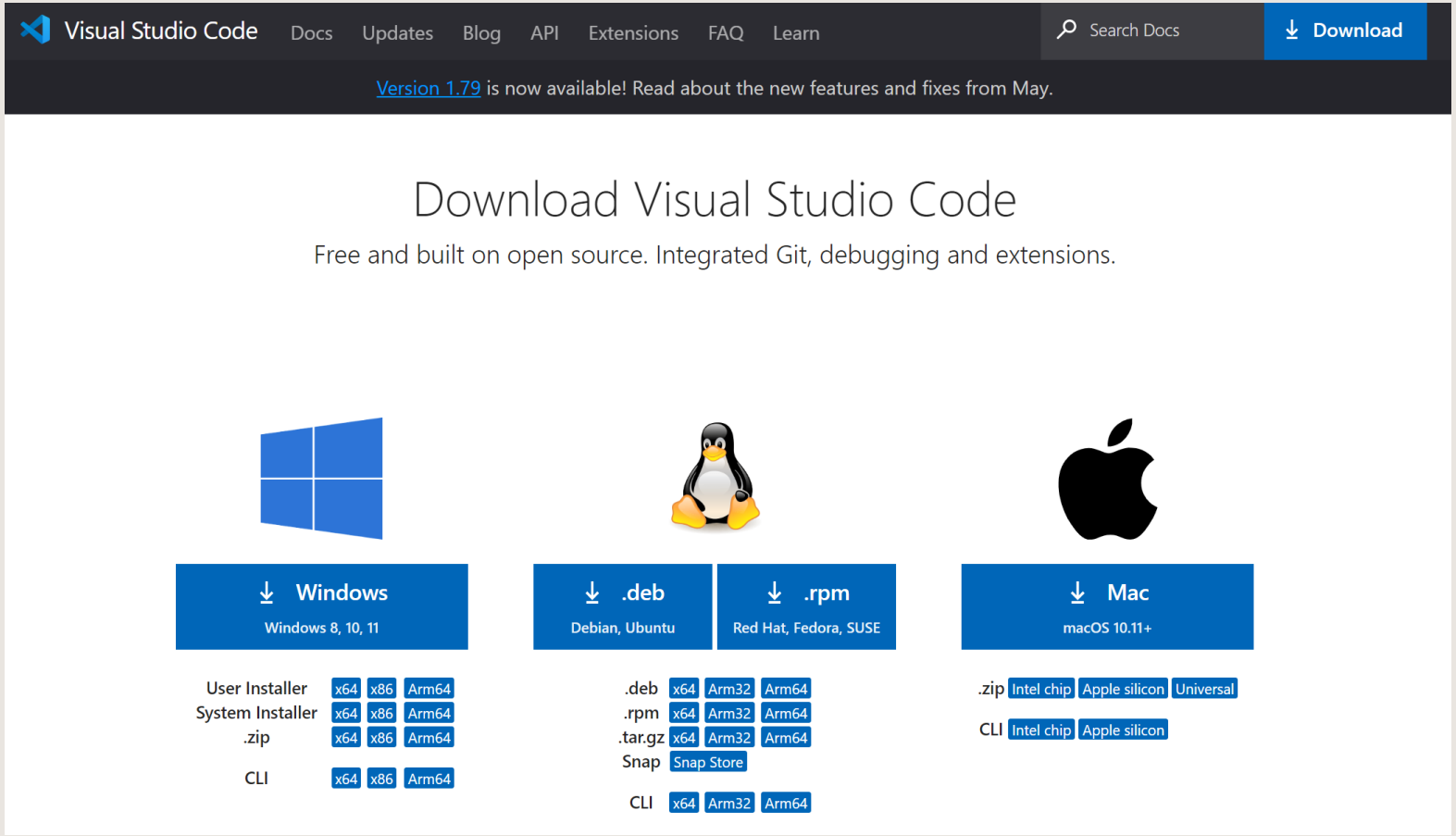
- Commonly referred to as VS Code. **VS Code라고도 함**
- A source-code editor made by Microsoft for Windows, Linux and macOS. **마이크로소프트에서 만든 윈도우, 리눅스, 맥OS용 소스코드 편집기**
- Supports for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. **디버깅, 문법 하이라이트, 지능형 코드 완성, 스니펫(재사용코드), 코드 리팩토링(기존코드 재구성), 임베디드 Git 지원**



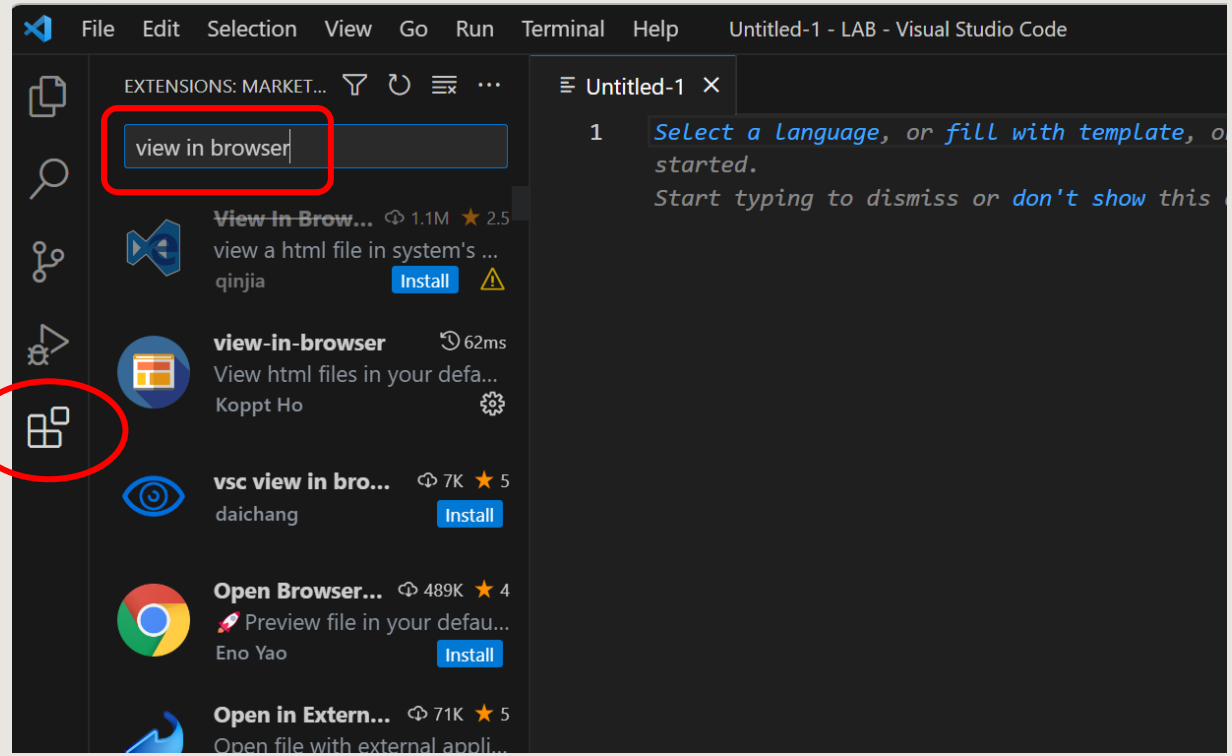

개발언어별로 다양한 확장 툴을 제공

# Visual Studio Code 다운로드 및 설치

- https://code.visualstudio.com/download



# VS Code Extensions 설치




- Activity Bar에서 Extensions 메뉴를 클릭
- “view-in-browser” 확장툴 설치 (HTML 페이지를 브라우저에서 바로 확인할 수 있음)
- 설치가 완료된 확장툴은 확장툴바에 표시

# HTML (Hypertext Markup Language)

- A language that describes the content and structure of a document by identifying, or tagging, different elements in the document. **다큐먼트안에 다른 요소들을 테깅해서 다큐먼트의 내용과 구조를 묘사하는 언어**

# HTML Tags **테그**

- HTML documents are consists of a lot of tags that tell a web browser how to display the web page. **HTML 문서는 웹 브라우저에 웹 페이지를 어떻게 표시하는 지 알려주는 많은 태그로 구성되어 있음**
- Opening tag tells the browser where the instruction begins e.g., , and closing tag tells it where the instruction ends. e.g.,  **오프닝 태그는 브라우저에 명령어가 시작되는 위치를 알려줌 (예: ). 클로징 태그는 명령어가 끝나는 위치를 알려줌 (예: )**
- Some tags don’t have closing tags, like <img>, <input>, and <br> tags. **일부 태그에는 크로징 태그가 없음**

# Structure of HTML Document

- Two main sections of an HTML document are the head and the body. **Html 다큐먼트의 2가지 주요부문은 헤드와 바디임**

```
<!DOCTYPE html>
<html>
    <head>
        head content
    </head>
    <body>
        body content
    </body>
</html>
```

# Document Type Declaration (doctype)

- Prior to the opening <html> tag, you must declare your document type at the top of your document. **다큐먼트의 맨위 <html> 테그 시작 전에 다큐먼트의 종류를 선언**
- States what type of document the page is and which standards it conforms to. **그 페이지가 무슨 다큐먼트 종류인지와 무슨 표준을 따르는 지 명시**

# Types of HTML

| Version | Date | Description |
|---|---|---|
| HTML1.0 | 1989 | The first public version of HTML. |
| HTML 2.0 | 1995 | Added interactive elements including Web forms. |
| HTML 3.0 | 1996 | A proposed replacement for HTML 2.0 that was never widely adopted. |
| HTML 3.2 | 1997 | Included additional support for Web tables and expanded the options for interactive form elements and a scripting language. |
| HTML 4.01 | 1999 | Added support for style sheets to give Web designers greater control over page layout and appearance, and provided support for multimedia elements such as audio and video. Current browsers support almost all of HTML 4.01. |
| XHTML 1.0 | 2001 | A reformulation of HTML 4.01 in the XML language in order to provide enforceable standards for HTML content and to allow HTML to interact with other XML languages. |
| XHTML 1.1 | 2002 | A minor update to XHTML 1.0 that allows for modularity and simplifies writing extensions to the language. |
| XHTML 2.0 | discontinued | The follow-up version to XHTML 1.1 designed to fix some of the problems inherent in HTML 4.01 syntax. Work on this version was discontinued in 2009 due to lack of browser support. |
| HTML 5.0 | In development | An update to HTML 4.01 that provides support for a variety of new features including semantic page elements, column layout, form validation, offline storage, and enhanced multimedia. |
| XHTML 5.0 | In development | A version of HTML 5.0 written under the XML language; unlike XHTML 2.0, XHTML 5.0 will be backward compatible with XHTML 1.1. |

- For example, HTML 4.01 Transitional, HTML 4.01 Strict, XHTML 1.0 Transitional, XHTML 1.0 Strict, etc. **예들**
- HTML5's doctype is short and simple! **HTML5가 가장 짧고 간단함**

```
<!DOCTYPE html>
```

# <head>

- Contains information about the document, for example the document title or the keywords. **다큐먼트에 대한 정보를 포함. 예를 들어 다큐먼트제목 또는 키워드**
- Includes Cascading Style Sheets, JavaScript programming, and links to JavaScript files. **CSS, 자바스크립트, 자바스크립트 파일에 대한 링크를 포함**
- The content of the head element is not displayed within the Web page. **헤드의 내용은 웹페이지에 나타나지 않음**

# Character Sets 글자세트

| ASCII | Latin-1 or ISO 8859-1 | Unicode | UTF-8 |
| --- | --- | --- | --- |
| • American Standard Code for Information Interchange<br>• Used for the alphabet of English characters. **영어 알파벳을 위해 사용** | • More extended character sets supporting 255 characters. **255 글자를 지원**<br>• Used by most languages that employ the Latin alphabet (e.g., English, French, Spanish, and Italian. **라틴알파벳에 사용** | • The most extended character set supporting up to 65,536 symbols. **6만개의 심볼을 지원**<br>• Used for any of the world's languages. **세계 모든 언어에 사용** | • Most commonly used languages on the Web. **웹에서 가장 자주 사용되는 글자세트**<br>• A compressed version of Unicode. **유니코드의 압축버젼**<br>• Default character set assumed by the browser. **브라우저에서 기본으로 사용** |

- Before adding special characters to web pages, you need to specify the chara cter set you are using. <meta charset="UTF-8">

# Comment Tag 주석테그

- The comment tag adds notes to your HTML code. **HTML 코드 사이에 노트를 쓸 수 있음**

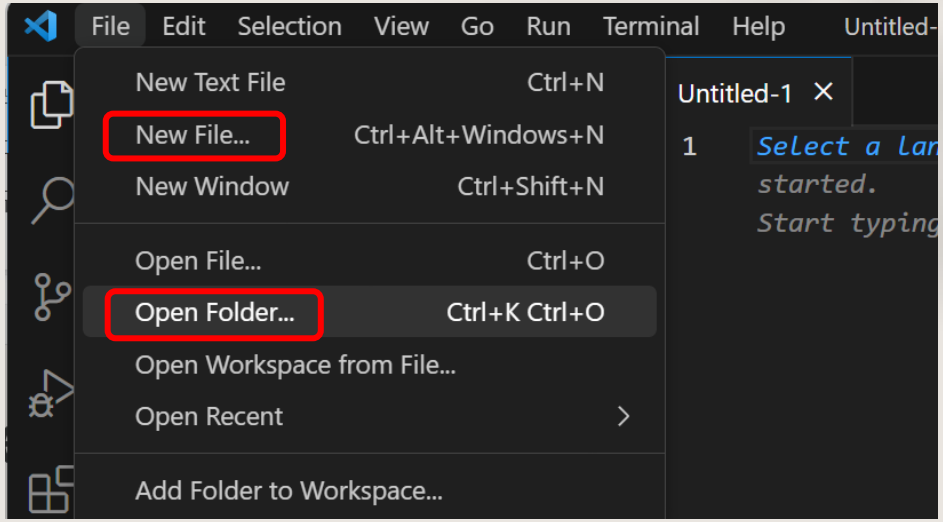
```
<!-- comment -->
```

- Comments are useful in documenting your HTML code for yourself and others. **자신이나 남을 위해 코드에 다큐먼트하는 것은 유용함**
- Comments can be spread over several lines. **여러 줄의 주석도 가능**

# Exercise #1

- File>Open Folder으로 작업폴더를 선택한 후 New File로 index.html 입력한 후 다음 코드를 작성

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>Tax Calculator</title>
</head>
<body>
</body>
</html>
```



# <body>

- Contains all the information that appears inside a browser window: headlines, text, pictures, and so on. **브라우저 안에 보이는 모든 정보를 포함. 헤드라인, 텍스트, 그림 등**
- Can contain code that tells the browser how to render the content. **내용을 브라우저에 어떻게 보여줄지를 말해주는 코드를 포함**

# Page Headings 페이지 헤딩

- Tags like <h1> and <h2> denote headlines and assign them relative importance. **헤드라인을 표시하며 상대적 중요도를 보여줌**

```
<h1>Heading 1 <small>Sub Heading 1</small></h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

# Paragraphs 문단

- The  tag indicates a basic paragraph of information. **기본 문단을 표시**
- You tell a web browser where a paragraph of text begins with a  (opening paragraph tag), and where it ends with a  (closing paragraph tag). **오프닝 테크와 클로징 테그로 단락의 시작과 끝을 보여줌**

```
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
```

# Inserting Line Breaks 줄바꿈 삽입

- The <br> tag marks line break! **<br>테그로 줄바꿈**

```
<p>
To break lines<br>in a text,<br>use the br
element.
</p>
```

# Tables 테이블

- Each table row is defined with the <tr> tag. **테이블의 줄**
- A table data/cell is defined with the <td> tag. **테이블의 셀**
- A table header is defined with the <th> tag. By default, table headings are bold and centered. **테이블 헤더. 기본으로 중앙에 볼드**

```
<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

# Images 이미지

- The <img> tag is used to insert images into the web page. **이미지를 웹페이지에 넣을 때**

```
<img src="img_girl.jpg" alt="Girl in a jacket"
width="500" height="600>
```

# Unordered Lists 순서없는 리스트

- A <ul> tag identifies a bulleted list. **글머리기호 목록**
- A <li> tag marks an item in the list. **리스트의 아이템**
- For example, a list of recipe ingredients. **요리재료의 리스트 등**

```
<ul>
    <li>Coffee</li>
    <li>Tea</li>
    <li>Milk</li>
</ul>
```

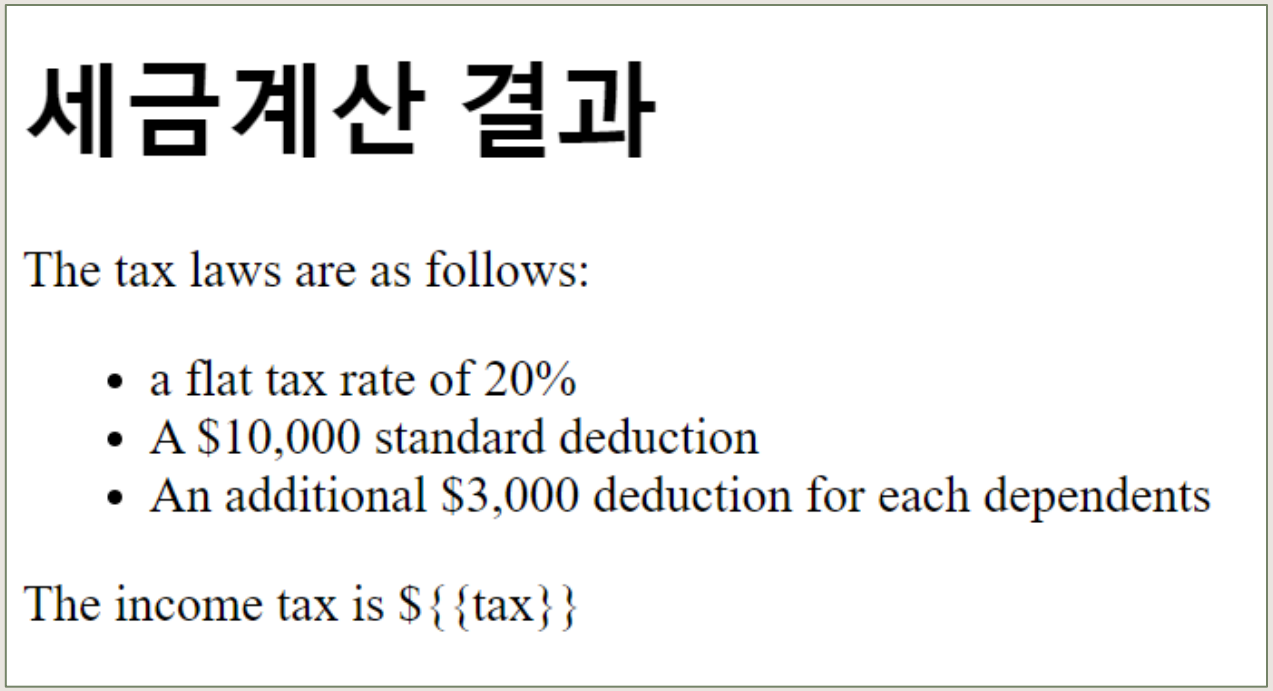
# Ordered Lists 순서있는 리스트

- The list items will be marked with numbers by default. **숫자로 표시되는 것이 기본**

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

# Exercise #1

- Create a html called result.html to display the result of tax calculations. **세금 계산 결과를 보여주는 html 페이지를 만드시오**




result.html

# Result Page Code 결과 페이지 코드

```
    <h1>세금계산 결과</h1>
    <p>The tax laws are as follows:</p>
    <ul>
        <li>a flat tax rate of 20%</li>
        <li>A $10,000 standard deduction</li>
        <li>An additional $3,000 deduction for each
dependents</li>
    </ul>

    <p>The income tax is ${{tax}}</p>
```

# Python Delimiters 파이썬 구분기호

- You can escape from HTML and execute the Python code using following tags. **다음 테그를 이용해서 html 에서 벗어나서 파이썬 코드를 실행 할 수 있음**
  - {% ... %} for Statements **코드**
  - {{ ... }} for Expressions to print to the output. **결과물 출력**
  - {# ... #} for Comments not included in the output. **결과에 포함되지 않는 주석**
  - # ... ## for Line Statements **여러 줄 코드**

# Forms 폼

- The HTML <form> element defines a form that is used to collect user input. **사용자 입력을 수집하기 위해 사용**
- The action attribute defines the action to be performed when the form is submitted. **폼을 보내질 때 무슨 액션을 취할지 정의**
- The method attribute specifies the HTTP method (GET or POST) to be used when submitting the form data. **폼을 보낼 때 http 방법을 표시**

```
<form action="/action_page.php" method="post"> </form>
```

# GET vs. POST

/action_page.php?firstname=Mickey&lastname=Mouse

## GET

- visible in the page address field. **페이지 주소창에 내용이 보임**

## POST

- does not display the submitted form data in the page address field. **주소창에 보내지는 폼데이터가 보이지 않음**

# Inputs 입력

| | |
|---|---|
| `<input type="text">` | Defines a one-line text input field. **한줄 텍스트** |
| `<input type="radio">` | Defines a radio button (for selecting one of many choices). **많은 초이스 중에 하나를 선택하는 라디오버튼** |
| `<input type="submit">` | Defines a submit button (for submitting the form). **폼을 보낼 때 제출버튼** |

```
First name:<br>
<input type="text" name="firstname" value="Mickey"><br>
Last name:<br>
<input type="text" name="lastname" value="Mouse"><br>
<input type="submit" value="Submit">
```

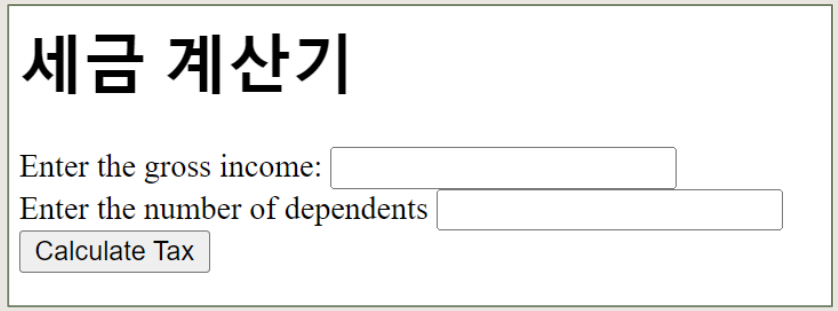
# Exercise #1

- <body> 테그 안에 다음 코드를 넣으세요

```
<form action="result.html" method="post">
<p>Enter the gross income:
<input type="text" name="income" /><br>
Enter the number of dependents
<input type="text" name="nd" /><br>
<input type="submit" value="Calculate Tax" />
</p>
</form>
```

- <H1> 테그로 세금계산기 제목을 넣으세요

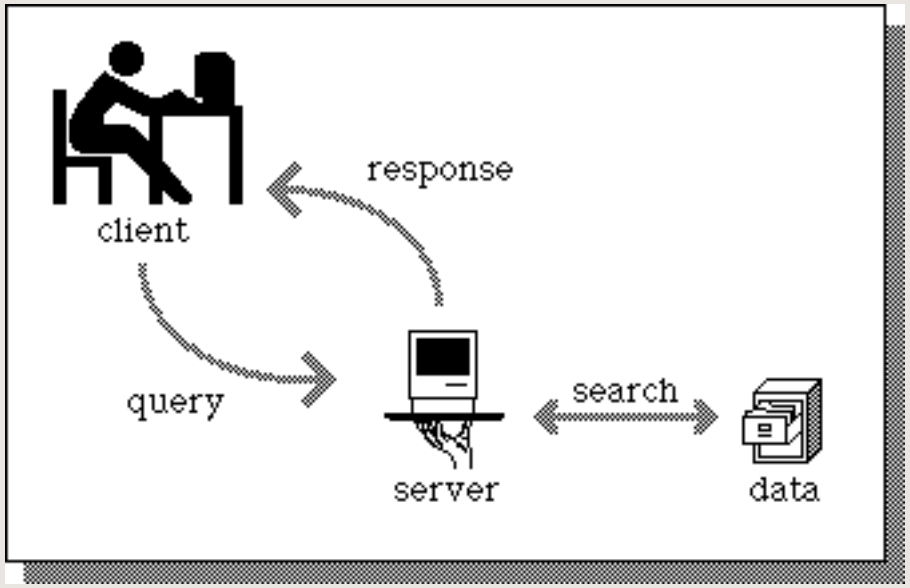
```
<h1>세금  계산기</h1>
```




Index.html

# Client Server Architecture
# 클라이언트 서버 구조

Client-Side Programming




Server-Side Programming

- Client makes a request to connect to server for providing some service. **클라이언트가 서버에 리퀘스트**
- The services running on the server run on known ports and the client needs to know the address of the server machine and this port in order to connect to the server (http://127.0.0.1:5000)
  **서버의 주소와 포트를 알아야 서비스를 리퀘스트 할 수 있음**

# Client-Side vs Server-Side Programming
# 클라이언트 대 서버 프로그래밍

## Client-Side Programming

- Execute in the browser.
  **브라우저에서 실행**
- JavaScript, Jscript, VBScript, PyScript, React

## Server-Side Programming

- Execute on the server.
  **서버에서 실행**
- Python, CGI/Perl, ASP, JSP, PHP, ColdFusion, Node.js

# Python Libraries (or Modules)
# 파이썬 라이브러리

| | |
|---|---|
| Data science and data analysis | • NumPy, SciPy, Scikit-Learn, Matplotlib |
| Web development | • Flask, Django |
| Web scrapping | • BeautifulSoup, urllib |

# Flask

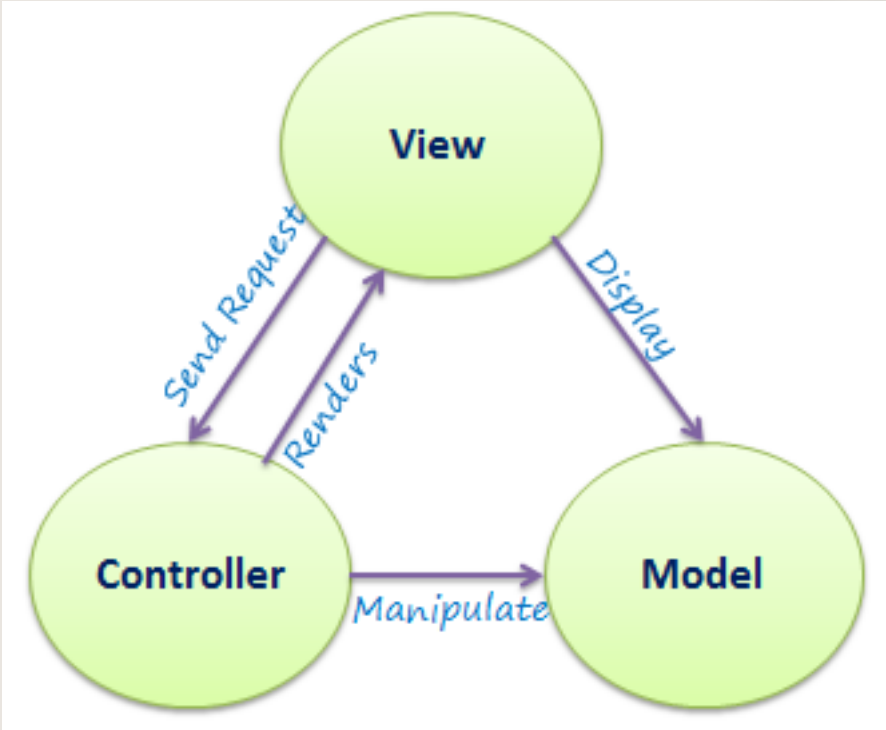
- A newer python web framework built to be easy and simple. **쉽고 간단하게 개발할 수 있는 새로운 파이썬 웹프레임워크**
- Recommended for smaller projects. **작은 프로젝트인 경우 플라스크 사용**
- Easy to learn and simple. **배우기 쉬움**

# Django

- A high-level Python web framework built for rapid development. **빠른 개발을 위한  고급 파이썬 프레임워크**
- Recommended for Larger projects. **큰 프로젝트에 적당함**
- Steeper learning curve. **처음에 배우기 어려움**
- Provides a full-featured MVC Framework. **MVC 프레임워크 사용**

# MVC (Model, View and Controller)

- Model represents shape of the data and business logic. It maintains the data of the application. Model objects retrieve and store model state in a database. **모델은 데이터를 관리**
- View is a user interface. **뷰는 유저인터페이스**
- Controller handles the user request. **컨트롤러는 사용자 요청을 처리**




**어플리케이션을 3개의 요소로 나눔**

# Organizing Project

```
app.py
config.py
requirements.txt
static/
templates/
```

Starts up a development server and includes route definitions. **서버시작 및 라우트 정의**

Contains most of the configuration variables that your app needs.**설정변수**

Lists all of the Python packages that your app depends on. **사용된 모든 파이썬 패키지(라이브러리)들**

CSS, JavaScript, images

Jinja2 templates and html files

# app.py

테그 내용을 바로
출력시키는 방법

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

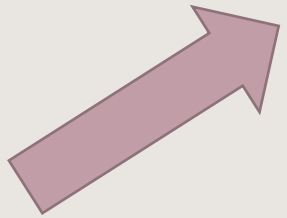
The route() function is a decorator, which tells the application which URL should call the associated function. **라우트 함수를 이용하여 url과 함수를 연결**

Class name is equal to the name of the scope in which top-level code executes. **클라스의 이름이 상위코드를 실행시키는 범위의 이름과 같을 때 app을 실행**

# Rendering Template

## html로 연결시키는 방법

```
from flask import Flask,
render_template
app1 = Flask(__name__)

@app1.route('/')
def index():
    return
render_template('hello.html')

if __name__ == '__main__':
    app.run(debug = True)
```



**hello.html**

```
<!doctype html>
<html>

<body>
    <h1>Hello!</h1>
</body>

</html>
```

Don't need to restart manually for each change in the code. **디버그옵션을 넣으면 코드가 바뀔 때마다 refresh할 필요없음**

# Passing Single Parameters

단일변수를
받아서 html로
보내는 방법

```
from flask import Flask, render_template
app = Flask(__name__)

@app.route('/hello/<user>')
def hello_name(user):
   return render_template('hello.html', name = user)

if __name__ == '__main__':
   app.run(debug = True)
```

# Displaying Parameters

```
<!doctype html>
<html>
    <body>
        <h1>Hello {{ name }}!</h1>
    </body>
</html>
```

보낸 변수를 받아서 출력

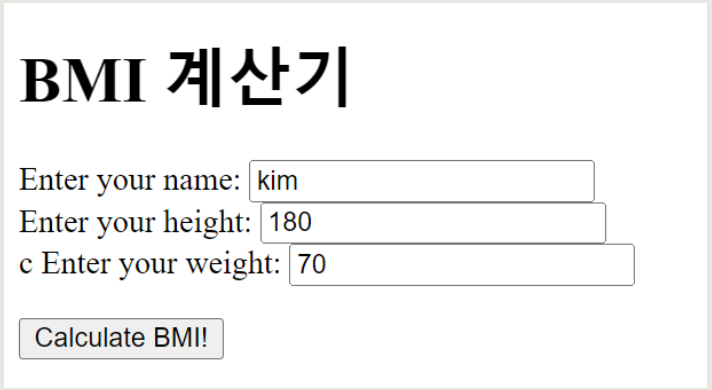
# Passing Multiple Parameters

여러 변수를 받아서 하나씩 html로 보내는 방법

```
from flask import Flask, render_template, request
app = Flask(__name__)
@app.route('/bmi_input')
def input():
    return render_template('bmi_input.html')
@app.route('/bmi_output', methods=['POST', 'GET'])
def bmi():
    name = request.form.get('name')
    height = int(request.form.get('height'))
    weight = int(request.form.get('weight'))
    bmi = round(weight / ((height/100)**2),2)
    return render_template("bmi_output.html", name = name, height=height, weight=weight, bmi=bmi)
if __name__ == '__main__':
    app.run(debug = True)
```

# Input Screen 입력화면

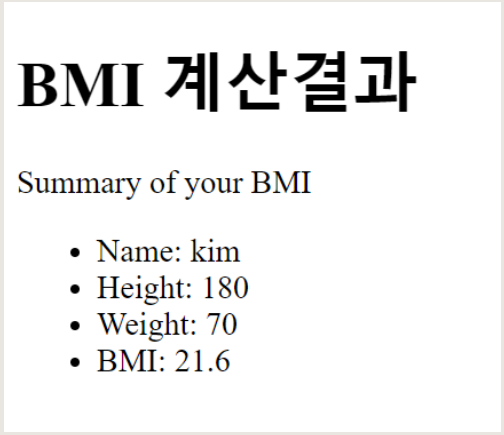
폼을 이용해서
입력값을 받음




```
<h1>BMI 계산기</h1>

  <form action="bmi_output" method="POST">
      <p>Enter your name:
          <input type="text" name="name"><br>
          Enter your height:
          <input type="text" name="height"><br>
          Enter your weight:
          <input type="text" name="weight">
      </p>
      <p><input type="submit" value="Calculate BMI!"></p>
  </form>
```

# Output Screen 출력화면

변수를 하나씩 받아서 출력




```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <title>BMI Calculator</title>
</head>

<body>
    <h1>BMI 계산결과</h1>

    <p>Summary of your BMI</p>
    <ul>
        <li>Name: {{name}}</li>
        <li>Height: {{height}}</li>
        <li>Weight: {{weight}}</li>
        <li>BMI: {{bmi}}</li>
    </ul>
</body>

</html>
```

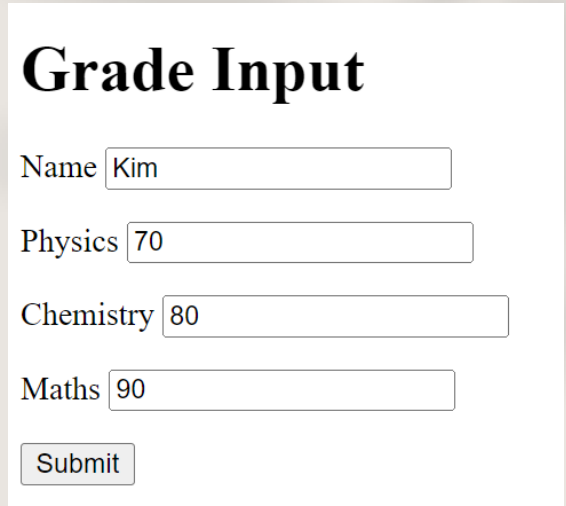
# Passing Multiple Parameters

여러변수를 받아서 한꺼번에 html로 보내는 방법

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
def input():
   return render_template('input.html')

@app.route('/output', methods = ['POST', 'GET'])
def result():
   if request.method == 'POST':
      result = request.form
      return render_template("result.html", result = result)

if __name__ == '__main__':
   app.run(debug = True)
```

# Input Screen 입력화면

```
<h1>Grade Input</h1>

<form action="output" method="POST">
    <p>Name <input type="text" name="Name"></p>
    <p>Physics <input type="text" name="physics"></p>
    <p>Chemistry <input type="text" name="chemistry"></p>
    <p>Maths <input type="text" name="mathematics"></p>
    <p><input type="submit" value="Submit"></p>
</form>
```

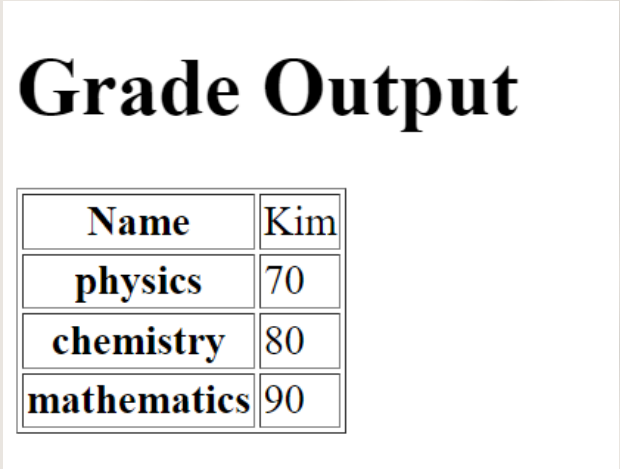



input.html

폼을 이용해서 입력값을 받음

# Output Screen 출력화면

```
<h1>Grade Output</h1>

<table border=1>
    {% for key, value in result.items() %}
    <tr>
        <th> {{ key }} </th>
        <td> {{ value }} </td>
    </tr>
    {% endfor %}
</table>
```

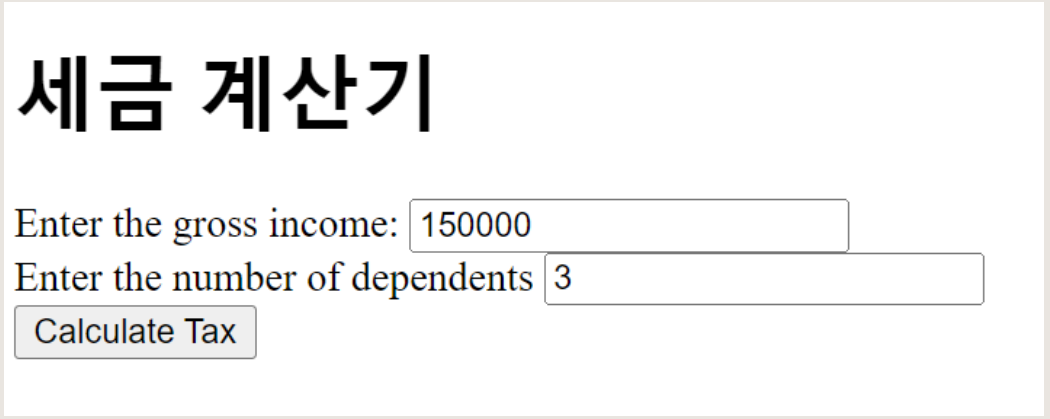


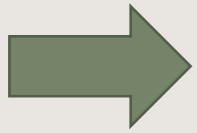

output.html

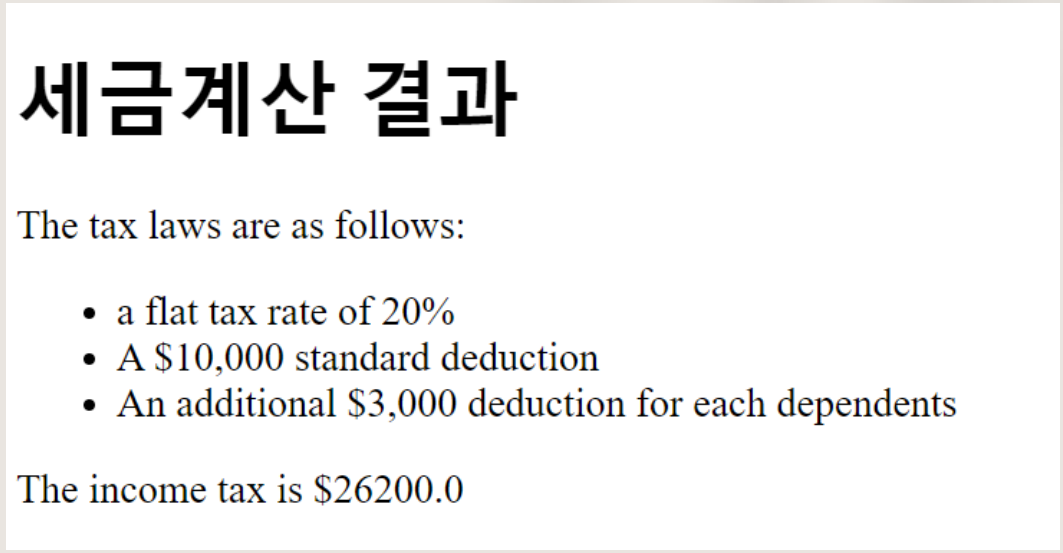
변수를 한꺼번에 받아서 루프로 출력

# Exercise #1

- Create a python file named app.py to requesting objects from the input page, calculate tax, and passing parameters to the result page. **입력화면에서 수입, 부양가족 수를 받아서 세금을 계산하는 app.py 생성**



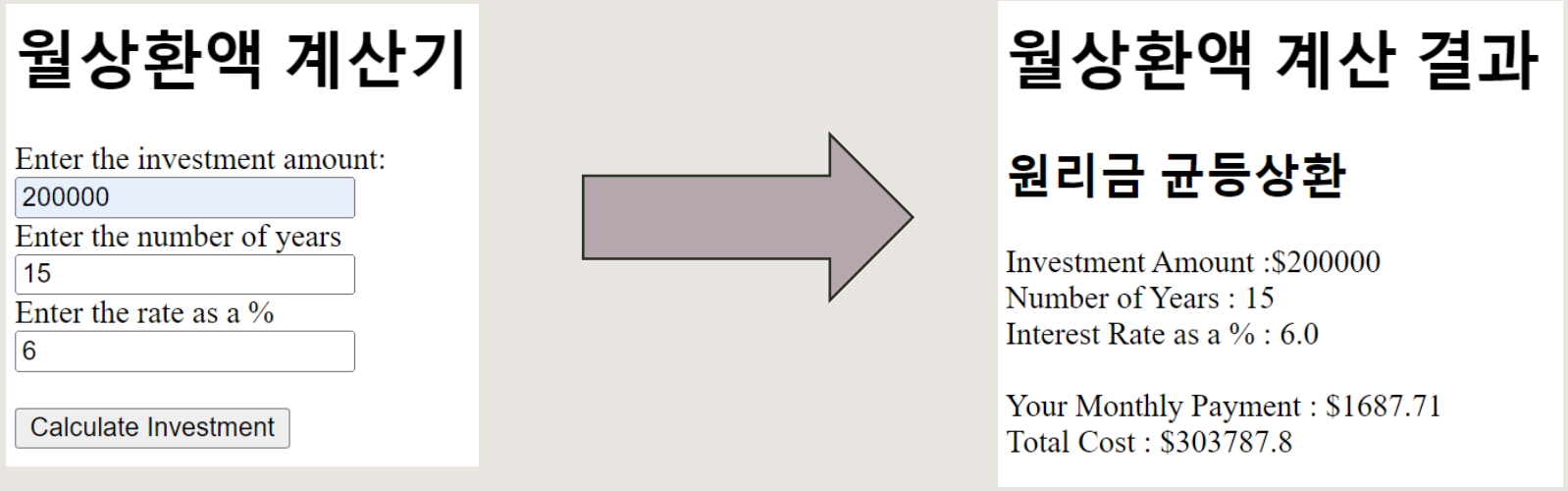


# app.py

```
from flask import Flask, render_template, request

app5= Flask(__name__)

@app5.route('/')
def input():
    return render_template('input.html')

@app5.route('/result', methods=['POST', 'GET'])
def tax():
    income = int(request.form.get('income'))
    nd = int(request.form.get('nd'))
    tax_rate = .2
    tax = round((income - (10000 + nd * 3000))*tax_rate, 2)
    return render_template("result.html", tax=tax)

if __name__ == '__main__':
    app5.run(debug=True)
```

# Exercise #2 Mortgage Program

- Write a program that computes a monthly payment. **월상환액와 총비용을 계산하는 프로그램을 짜시오.**
- The inputs to this program are the following: **입력값은 다음과 같습니다**
  - An initial amount to be invested (a floating-point number) **초기비용 (소수)**
  - A period of years (an integer) **투자기간 (정수)**
  - An interest rate (a percentage expressed as an integer) **이자율 (퍼센트가 정수로 표현)**



# 원리금 균등상환 월상환액 공식

- 매월상환액 $= \frac{Ab(1+b)^n}{(1+b)^n - 1}$
- 테스트 데이터

- A = 대출원금
- b = 대출이자율 (연이자율 / 12)
- n = 상환기간 (실제 상환 개월수)

| Principal | Years | Rate | Monthly Payment | Total Cost |
|---|---|---|---|---|
| 200,000 | 15 | 6 | $1687.71 | $303,787.8 |
| 200,000 | 30 | 6 | $1199.1 | $431,676.0 |
| 100,000 | 30 | 5 | $536.82 | $193,255.2 |
| 100,000 | 30 | 4 | $477.42 | $171,871.2 |
| 300,000 | 30 | 5 | $1610.46 | $579,765.6 |

# Bootstrap 부트스트랩

- A free and open-source CSS framework directed at responsive, mobile-first front-end web development. **반응형 모바일 우선 프런트 엔드 웹 개발을 지향하는 공짜 오픈 소스 CSS 프레임워크**
- Contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. **글씨형식, 폼, 버튼, 네비게이션, 및 기타 인터페이스 구성요소를 위한 HTML, CSS 및 JavaScript 기반 디자인 템플릿을 포함**

# Setup for Mobile

- Set the width of the page and the initial zoom level. **페이지의 넓이와 초기 줌레벨**

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

# Bootstrap CDN (Content Delivery Network)

- If you don’t want to download the bootstrap files, use the code below. 파일을 다운받지 않고 연결하는 경우 아래 코드를 이용

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```

# Container 컨테이너

- A responsive fixed width container. **고정된 넓이의 컨테이너**

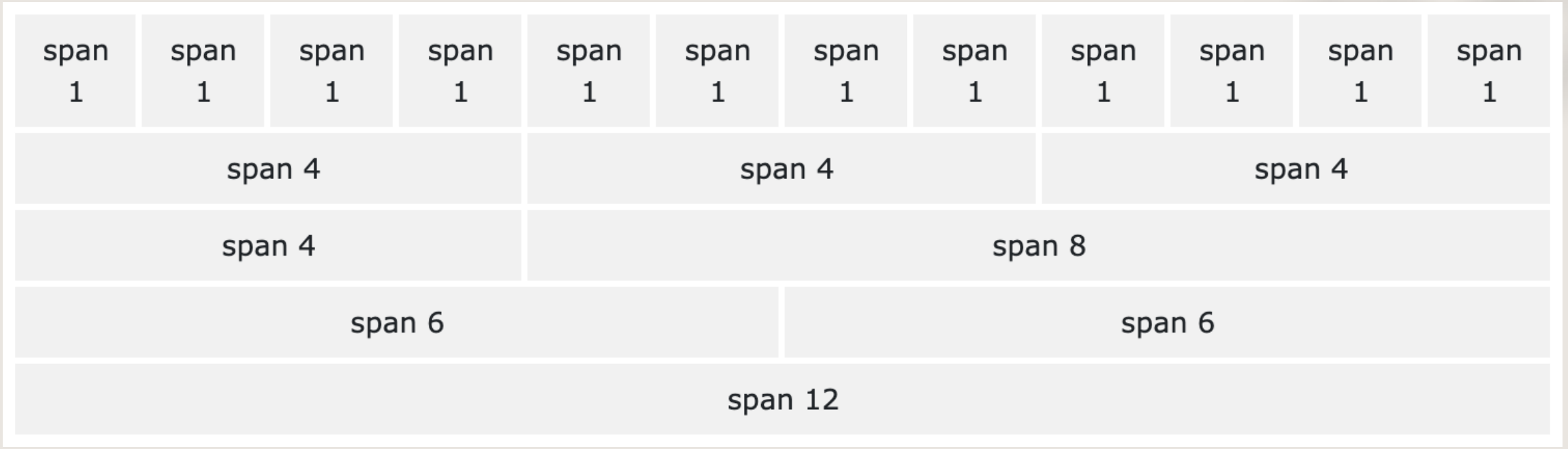
```
<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
```

- A full width container, spanning the entire width of the viewport. **화면의 전체 넓이를 사용하는 컨테이너**

```
<div class="container-fluid" background-color="red">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>
```

# Basic Structure of a Bootstrap Grid
# 부트스트랩 그리드의 기본구조

- Bootstrap's grid system allows up to 12 columns across the page.
  **각 페이지는 12개의 컬럼으로 구성**



# Basic Structure of a Bootstrap Grid (cont.)

- The Bootstrap grid system has four classes:
  - xs (for phones) **전화**
  - sm (for tablets) **테블렛**
  - md (for desktops) **데스크탑**
  - lg (for larger desktops) **큰 데스크탑**

```
<div class="row">
   <div class="col-md-12">Span 12
columns</div>
</div>


<div class="row">
  <div class="col-md-4">Span 4</div>
  <div class="col-md-4">Span 4</div>
  <div class="col-md-4">Span 4</div>
</div>
<div class="row">
  ...
</div>
```

# Tables 테이블

```
<table class="table table-bordered" style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

- table-striped

| Firstname | Lastname | Email |
|---|---|---|
| John | Doe | john@example.com |
| Mary | Moe | mary@example.com |
| July | Dooley | july@example.com |

- table-bordered

| Firstname | Lastname | Email |
|---|---|---|
| John | Doe | john@example.com |
| Mary | Moe | mary@example.com |
| July | Dooley | july@example.com |

- table-hover

| Firstname | Lastname | Email |
|---|---|---|
| John | Doe | john@example.com |
| Mary | Moe | mary@example.com |
| July | Dooley | july@example.com |

- table-condense

| Firstname | Lastname | Email |
|---|---|---|
| John | Doe | john@example.com |
| Mary | Moe | mary@example.com |
| July | Dooley | july@example.com |

# Images 이미지

```
<img src="img_girl.jpg" alt="Girl in a jacket" width="500"
height="600“ class=“img-rounded">
```

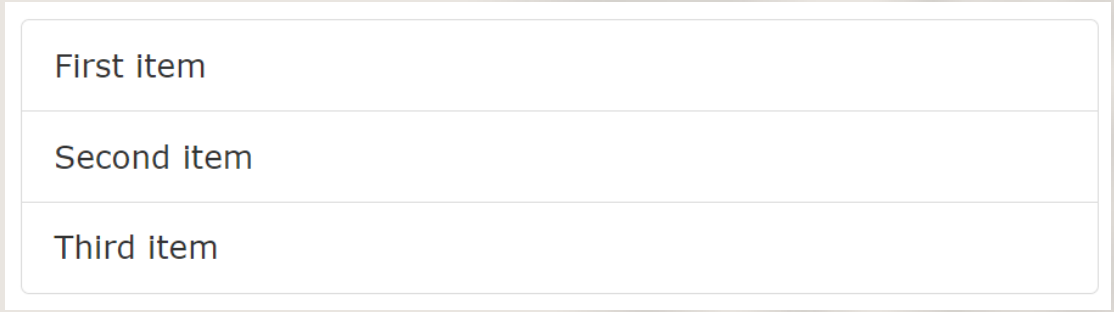
- img-rounded

- Img-circle

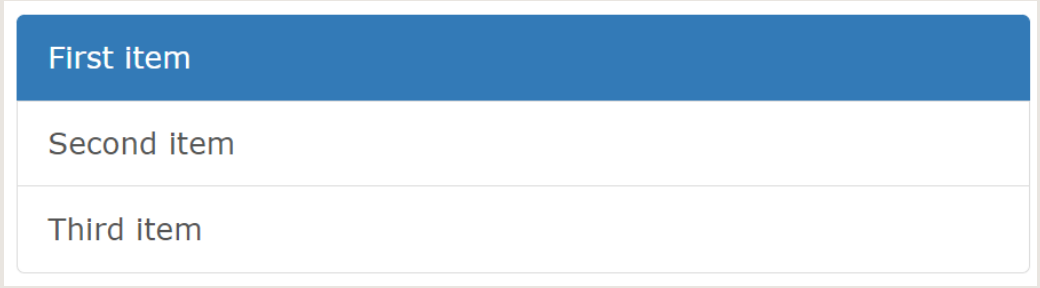
- Img-thumbnail

# Unordered Lists 순서없는 리스트

```
<ul class="list-group">
  <li class="list-group-item">Coffee</li>
  <li class="list-group-item">Tea</li>
  <li class="list-group-item">Milk</li>
</ul>
```

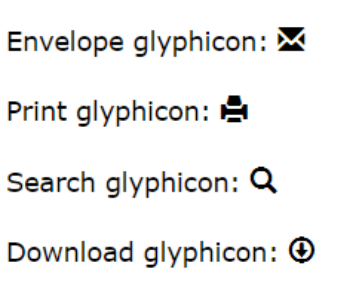



```
<div class="list-group">
  <a href="#" class="list-group-item">First item</a>
  <a href="#" class="list-group-item">Second item</a>
  <a href="#" class="list-group-item">Third item</a>
</div>
```

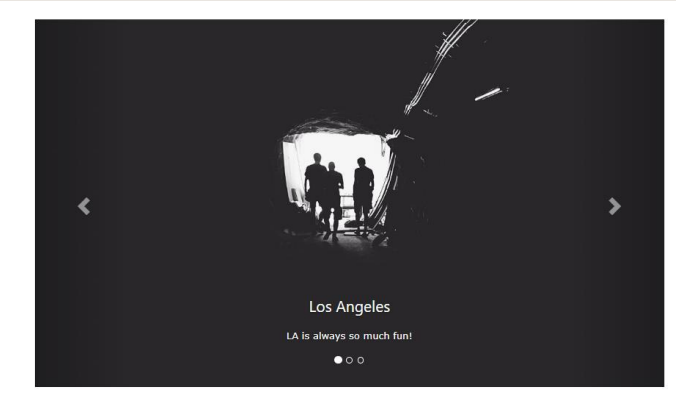


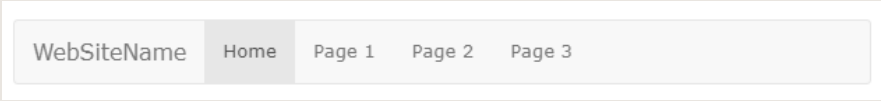
# BS Components 컴포넌트

- 부트스트랩에서 정의한 UI 요소들 (버튼, 경고창, 네비게이션바)과 같이 화면 구성에 필요한 요소들을 정의해둔 클래스 집합



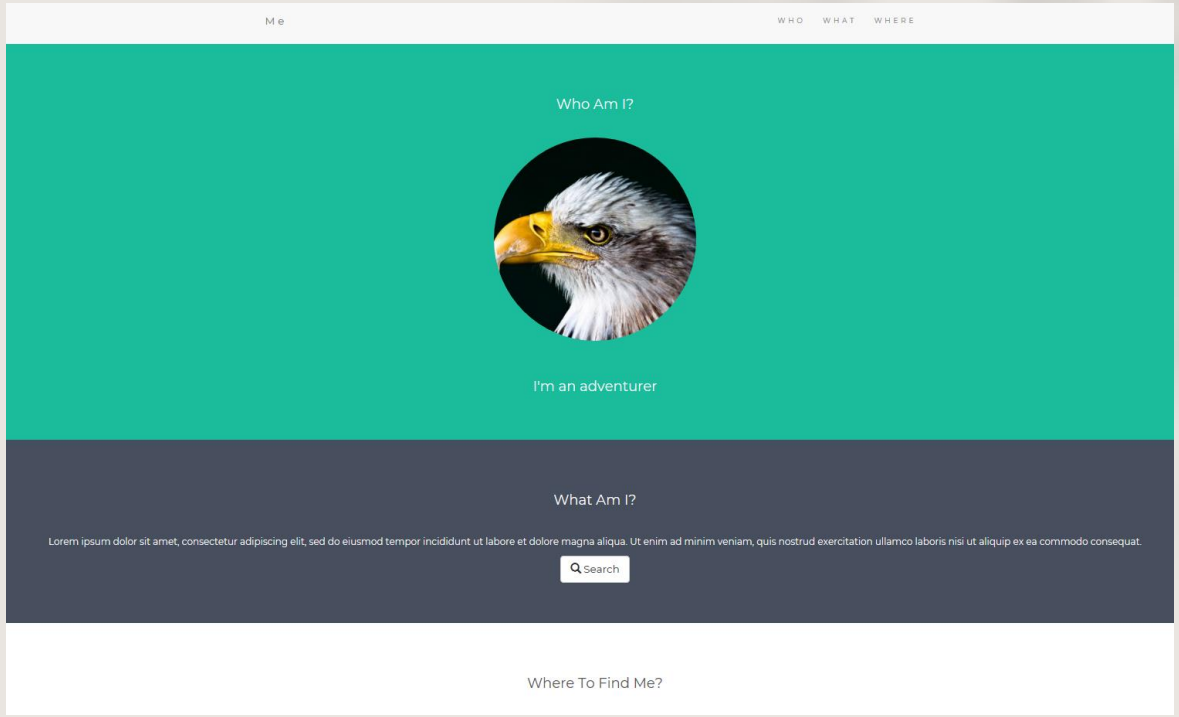




- BS Jumbotron
- BS Wells
- BS Alerts
- BS Buttons
- BS Button Groups
- BS Glyphicons
- BS Badges/Labels
- BS Progress Bars
- BS Pagination
- BS Pager
- BS List Groups
- BS Panels
- BS Dropdowns
- BS Collapse
- BS Tabs/Pills
- BS Navbar
- BS Forms
- BS Inputs
- BS Inputs 2
- BS Input Sizing
- BS Media Objects
- BS Carousel
- BS Modal
- BS Tooltip
- BS Popover
- BS Scrollspy
- BS Affix
- BS Filters

# Exercise #3

- Complete Bootstrap Theme "Simply Me"  on https://www.w3schools.com/bootstrap/bootstrap_theme_me.asp



# Collapsible Navbar 접을 수 있는 네비게이션바

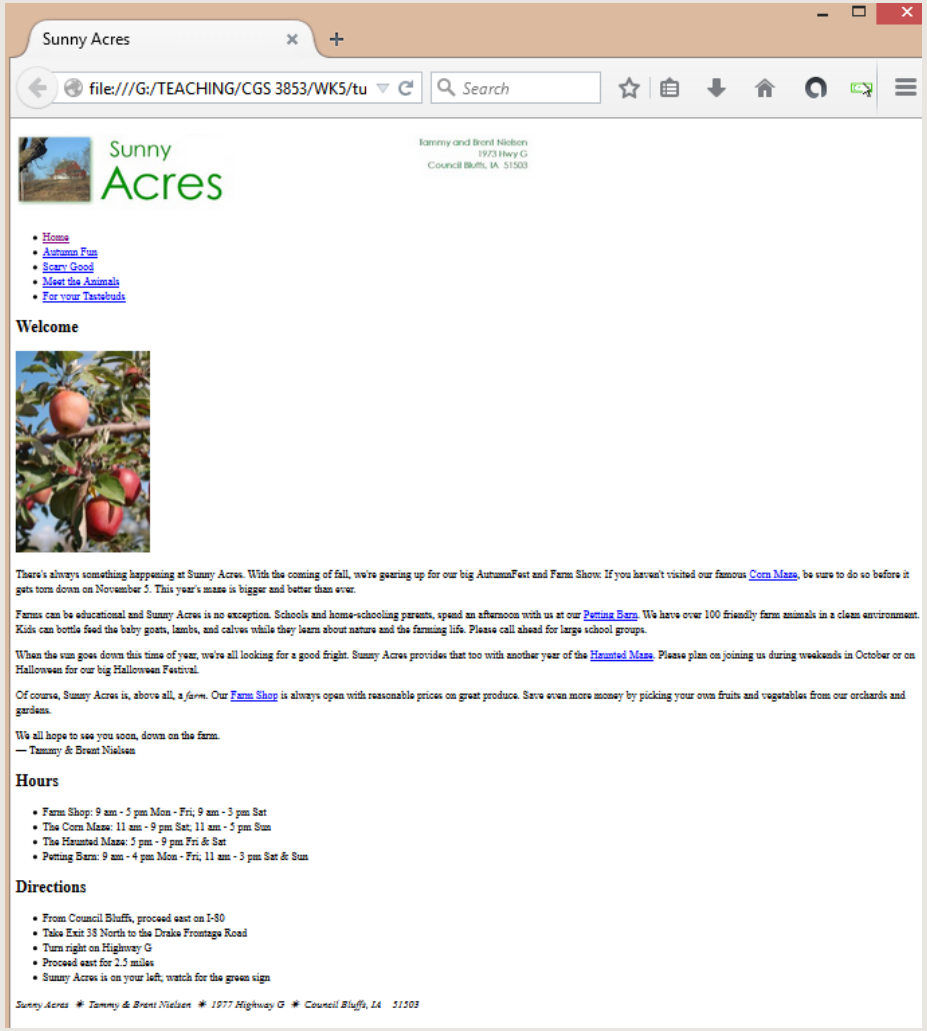
```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target="#myNavbar">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">WebSiteName</a>
    </div>
    <div class="collapse navbar-collapse" id="myNavbar">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Home</a></li>
        <li><a href="#">Page 1</a></li>
        <li><a href="#">Page 2</a></li>
        <li><a href="#">Page 3</a></li>
      </ul>
     </div>
  </div>
</nav>
```

# CSS (Cascading Style Sheets)

- A formatting language that tells a web browser how to display a particular element on a page. **웹브라우저가 웹페이지에 있는 특정 요소를 어떻게 보여줄지를 정해주는 포맷팅 언어**
- CSS allows us to apply formatting and styling to the HTML that builds our web pages. **웹페이지를 만들기 위해 HTML에 포맷팅, 스타일을 적용할 수 있음**

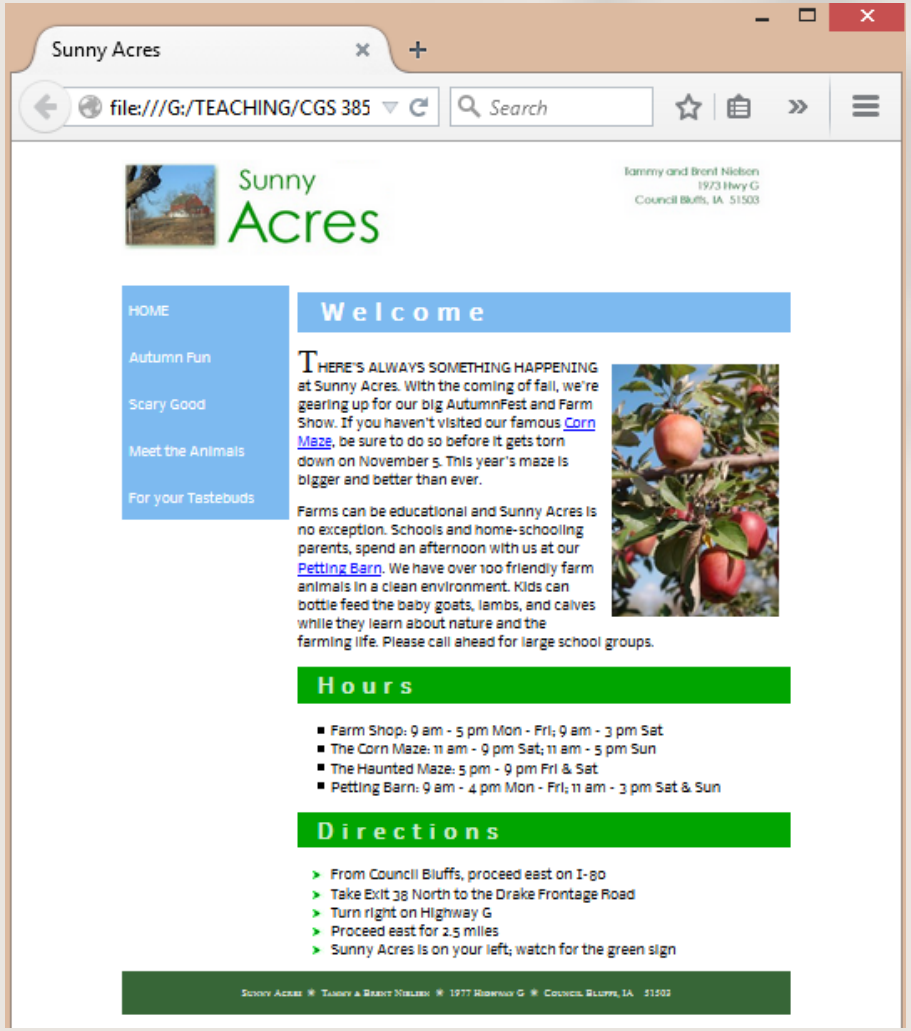
```css
h1 {
    font-family: courier, courier-new, serif;
    font-size: 20pt;
    color: blue;
    border-bottom: 2px solid blue;
}
p {
    font-family: arial, verdana, sans-serif;
    font-size: 12pt;
    color: #6B6BD7;
}
.red_txt {
    color: red;
}
```

# Initial page




CSS File

# Final page



# Syntax of CSS Style Rule 문법

```
selector {
        property1: value1;
        property2: value2;
        property3: value3;
...
}
```

- Example

```
h1 {
        color: blue;
        text-align: center;
}
h2, h3, h4, h5, h6 {
        color: red;
        text-align: right;
```

# Selectors 선택자

- Before you change the styles, you need to select the elements. **스타일을 적용하기 전에 요소를 선택해야 함**
  - Any elements on your document. **문서위에 한 요소를 선택할 때**

    ```
    body {
    }
    ```
  - Elements nested within elements. **요소 안에 있는 요소를 선택할 때**

    ```
    section h1{
    }
    ```
  - A certain id (#) or class ( .) **아이디나 클라스를 선택할 때**

    ```
    p.closing {
    }
    ```

# Applying Style Sheet 스타일시트 적용

- You can apply the style sheets to your pages using three different ways.
  **세가지 다른 방법으로 스타일 시트를 적용할수 있음**

External Style Sheets 외부시트

Embedded Style Sheets 포함된 시트

Inline Styles 줄안에 스타일

# External Style Sheets 외부 스타일시트

- If you'd like to use external style sheets, simply use the link in your <head> tag. **외부 스타일 시트를 사용하고 싶으면 헤드테그에 링크를 사용**
- Example

```
<link href="sa_layout.css" rel="stylesheet"
type="text/css" />
<link href="{{ url_for('static',
filename='css/main.css') }} " rel="stylesheet" >
```

rel specifies the relationship between the current document and the linked document/resource. **현재 문서와 연결된 문서/자원 사이의 관계를 보여줌**

# Embedded Style Sheets
# 포함된 시타일시트

- The styles are inserted directly within the head element of an HTML document using the style element. **HTML 헤드테그 안에 직접 코드를 집어넣음**

- Example

```
<style type="text/css">
  h1 {
  color: red;
  text-align: center;
  }
</style>
```

# Inline Styles 인라인 스타일

- Can directly apply to specific elements using the style attribute. **스타일 속성을 사용해서 특정 요소에 직접 적용할 수 있음**
- Example

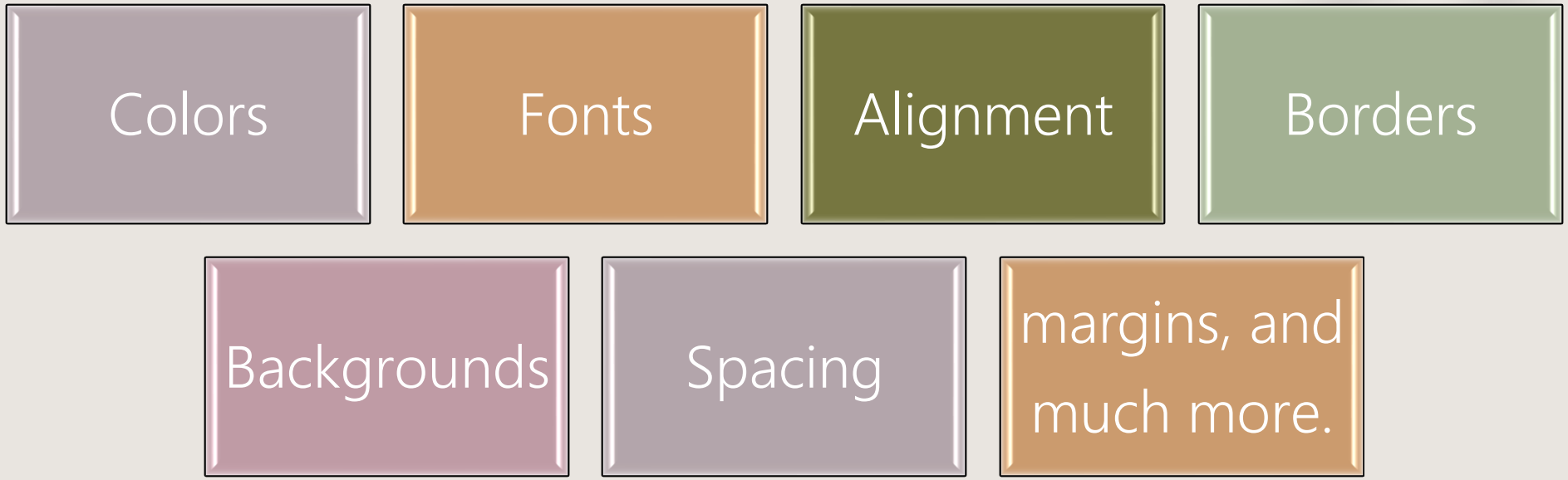
```
<h1 style="color: green; text-align: center;">
 Sunny Acres
</h1>
```

# Writing Style Comments 주석

- You can write comments on your css file using /* ... */. 다음 심볼을 이용해서 주석을 달 수 있음

# Elements of Web Pages 웹페이지 요소들

- CSS can control many elements of web pages. **CSS로 조절할 수 있는 요소들**



# Color in CSS 색상

- To express colors, you can use either color name, RGB triplet, or Hexadecimal numbers. **색상의 이름, RGB 트리플렛, 헥사 데시멀 넘버를 사용할 수 있음**
- RGB triplet
  - rgb(red, green, blue). For example, rgb(255, 255, 0) for yellow
- Hexadecimal numbers
  - #redgreenblue. For example, #FFFF00 for yellow

# 16 Basic Colors

| Color Name | RGB Triplet | Hexadecimal | Color Name | RGB Triplet | Hexadecimal |
|---|---|---|---|---|---|
| Aqua | (0, 255, 255) | 00FFFF | Navy | (0, 0, 128) | 000080 |
| Black | (0, 0, 0) | 000000 | Olive | (128, 128, 0) | 808000 |
| Blue | (0, 0, 255) | 0000FF | Purple | (128, 0, 128) | FF0000 |
| Fuchsia | (255, 0, 255) | FF00FF | Red | (255, 0, 0) | C0C0C0 |
| Gray | (128, 128, 128) | 808080 | Silver | (192, 192, 192) | 008080 |
| Green | (0, 128, 0) | 008000 | Teal | (0, 128, 128) | FFFFFF |
| Lime | (0, 255, 0) | 00FF00 | White | (255, 255, 255) | FFFF00 |
| Maroon | (128, 0, 0) | 800000 | Yellow | (255, 255, 0) | |

- Semi-transparent color **반투명 색상**
  - rgba(red, green, blue, opacity)
  - For example, rgba(255, 255, 255, 0.8) for white color with 80% opacity

# Color and Background Options
# 색상 및 배경옵션

- Background color **배경칼라**

```
Body {
  background-color: white;
}
```

- Foreground or Text Color **전경색이나 글자색상**

```
h2 {
  background-color: green;
  color: white;
  color: rgba(255, 255, 255, 0.8);
}
```

# Font Options 글자옵션

- font-family

```
body {
 font-family: Verdana,
Geneva, sans-serif;
}
```

- The default font is Times New Roman. **기본 글씨체는 타임즈 뉴로먼**

- font-size

```
section h1{
 font-size: 1.7em;
}
```

- Generally, 1em = 12pt = 16px = 100%.

# Font Property 폰트속성

- You can combine most of the text and font styles into a font property. **모든 텍스트와 폰트 스타일들을 하나의 폰트 속성에 합칠 수 있음**
- Syntax **문법**
  font: font-style font-variant font-weight font-size/line-height font-family;
- Example

```
footer address{
 font: normal small-caps .9em/4em   'Time New Roman', Times,
serif;
}
```

# Spacing Options 공간옵션

- line-height (space between lines of text) **텍스트의 줄사이의 공간**

```
body {
 line-height: 1.4em;
}
```

- text-indent (the indentation for the first line of a block of text) **텍스트 블락 첫줄의 들여쓰기**

```
section h1{
 text-indent: 1em;
}
```

# Spacing Options (cont.)

- letter-spacing (space between letters) **글자사이의 공간**

```
section h1{
 letter-spacing:
 .4em;
}
h2 {
 letter-spacing:
 .4em;
}
```

- word-spacing (space between words) **단어사이의 공간**

```
h3{
 word-spacing:
 .8em;
}
```

# Alignment Options 정렬옵션

- Text-align

```
section p.closing{
 text-align: right;
}
```

# Border Options 테두리옵션

- Syntax

```
border: width color style;
border-width: top right bottom left;
border-color: top right bottom left;
```

- style is none, solid, dashed, dotted, double, outset, inset, groove, or ridge

```
h1 {
    border: 5px solid red;
}

div {
    border-width: 10px 1px;
}

div {
    border-color: #ff0000 #00ff00 #0000ff rgb(250,0,255)
}
```

# Margin and Padding 마진과 패딩

- Margin (margin space around an element) **요소 주위의 공간**
- Syntax

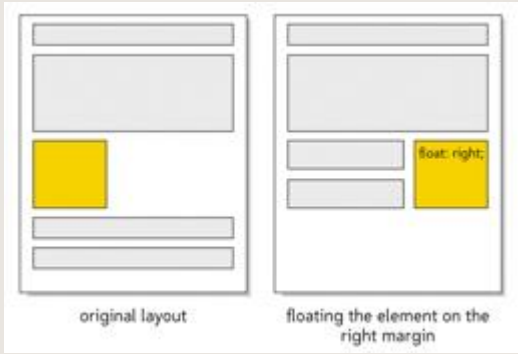
```
margin: length;
margin: top right bottom left;
margin: vertical horizontal;
```

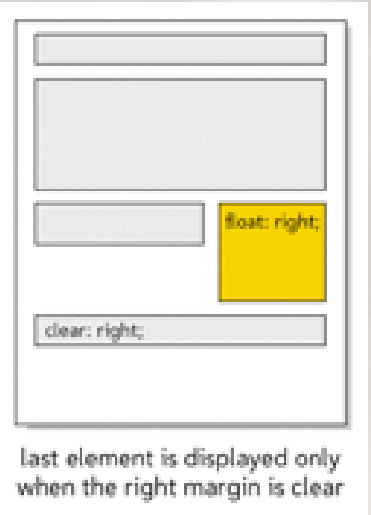
- Padding (padding space within an element) **요소 안의 공간**
- Syntax

```
padding: length;
```

# Floating Elements 떠다니는 요소들

- Takes that element out of the normal flow of the document and positions it along the left or right edge of its containing element.
  **문서의 일반적인 흐름과 상관없이 특정요소를 문서의 왼쪽이나 오른쪽으로 위치시킬 수 있음**




```
img {
    float: left;
}

p.clear {
    clear: both;
}
```

# Exercise #4

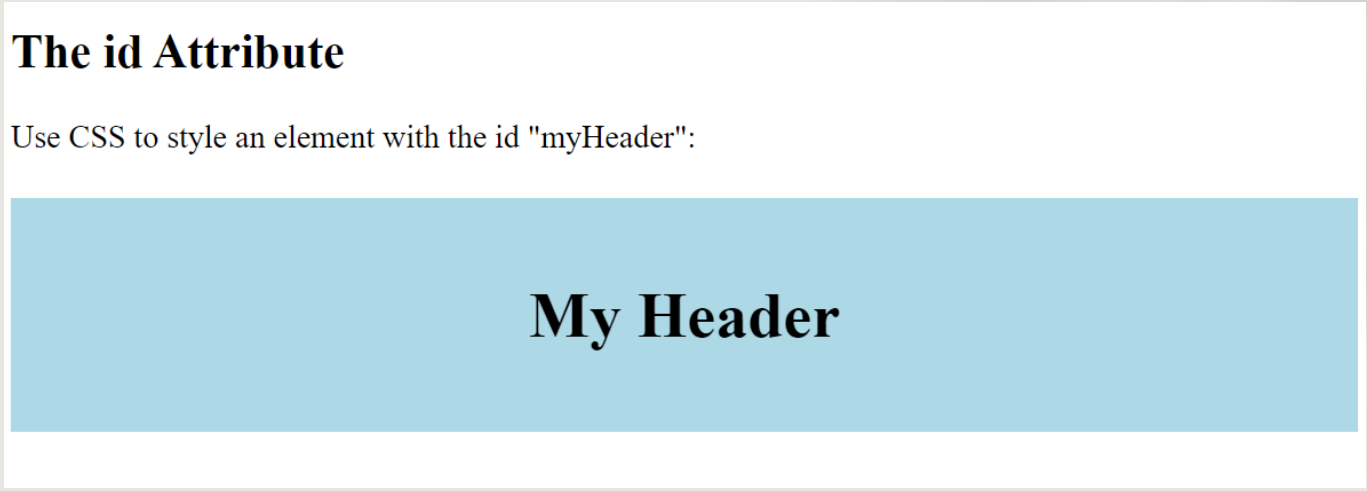
- Complete the CSS tutorial on https://medium.freecodecamp.org/get-started-with-css-in-5-minutes-e0804813fc3e

# class Attribute 클래스 속성

```
<!DOCTYPE html>
<html>
<head>
<style>
.note {
  font-size: 120%;
  color: red;
}
</style>
</head>
<body>

<h1>My <span class="note">Important</span> Heading</h1>
<p>This is some <span class="note">important</span> text.</p>

</body>
</html>
```

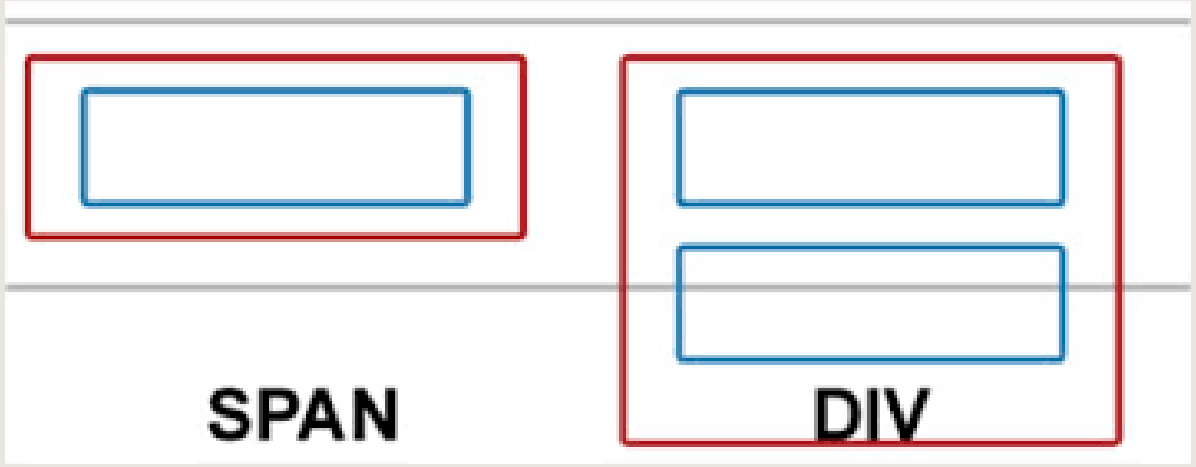
# id Attribute 아이디속성

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>
</head>
<body>

<h2>The id Attribute</h2>
<p>Use CSS to style an element with the id "myHeader":</p>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

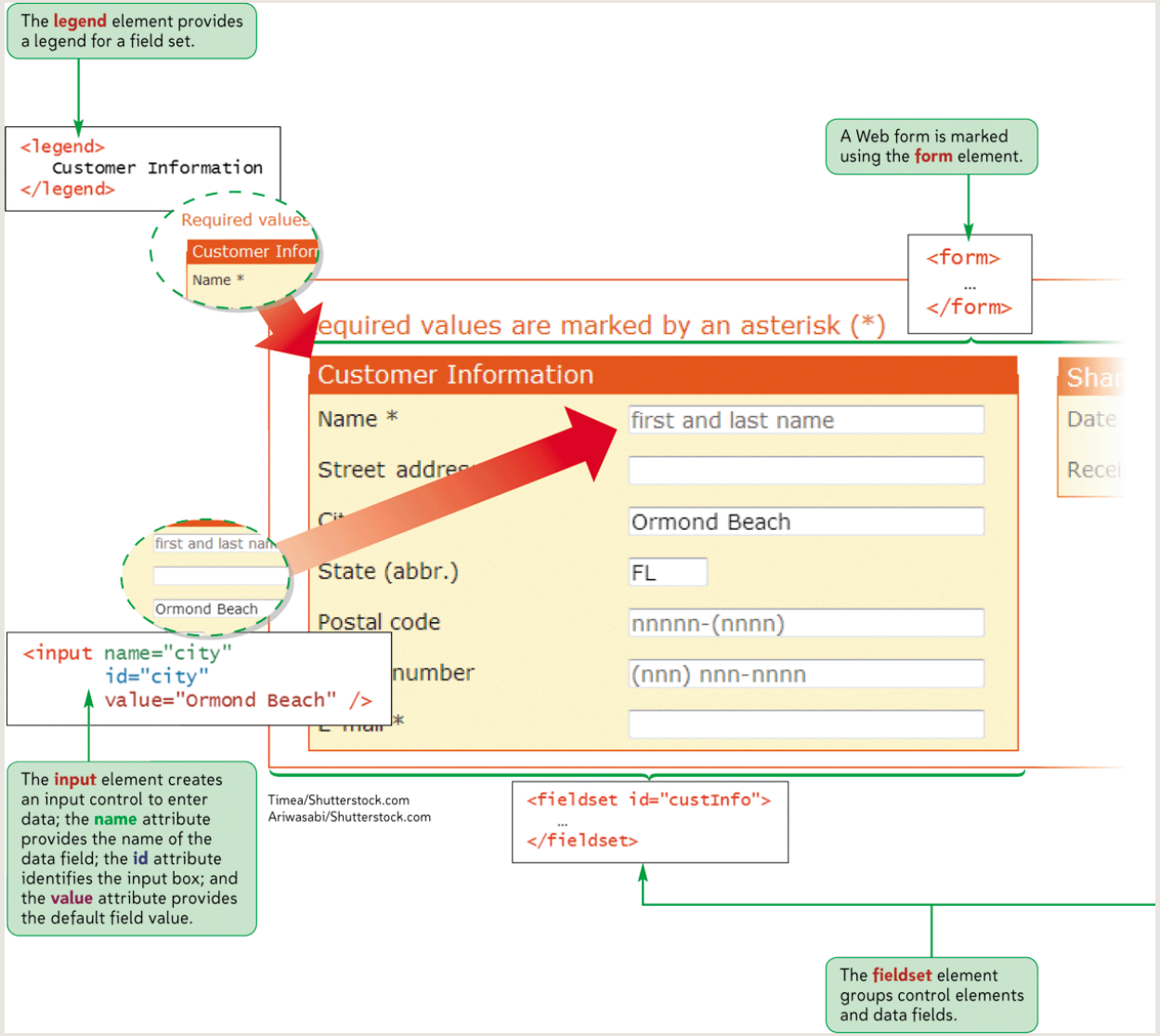


# DIV and SPAN

- Tags used as content wrapper. **내용을 감싸주는 테그**
- DIV is a block element while SPAN is an inline element. **DIV는 블락요소이고, SPAN은 인라인요소**
- DIV takes up the entire width of the screen whereas SPAN conforms to the width of whatever element it contains. **DIV는 화면의 전체 너비를 차지하는 반면 SPAN은 포함하는 요소의 너비를 따름**

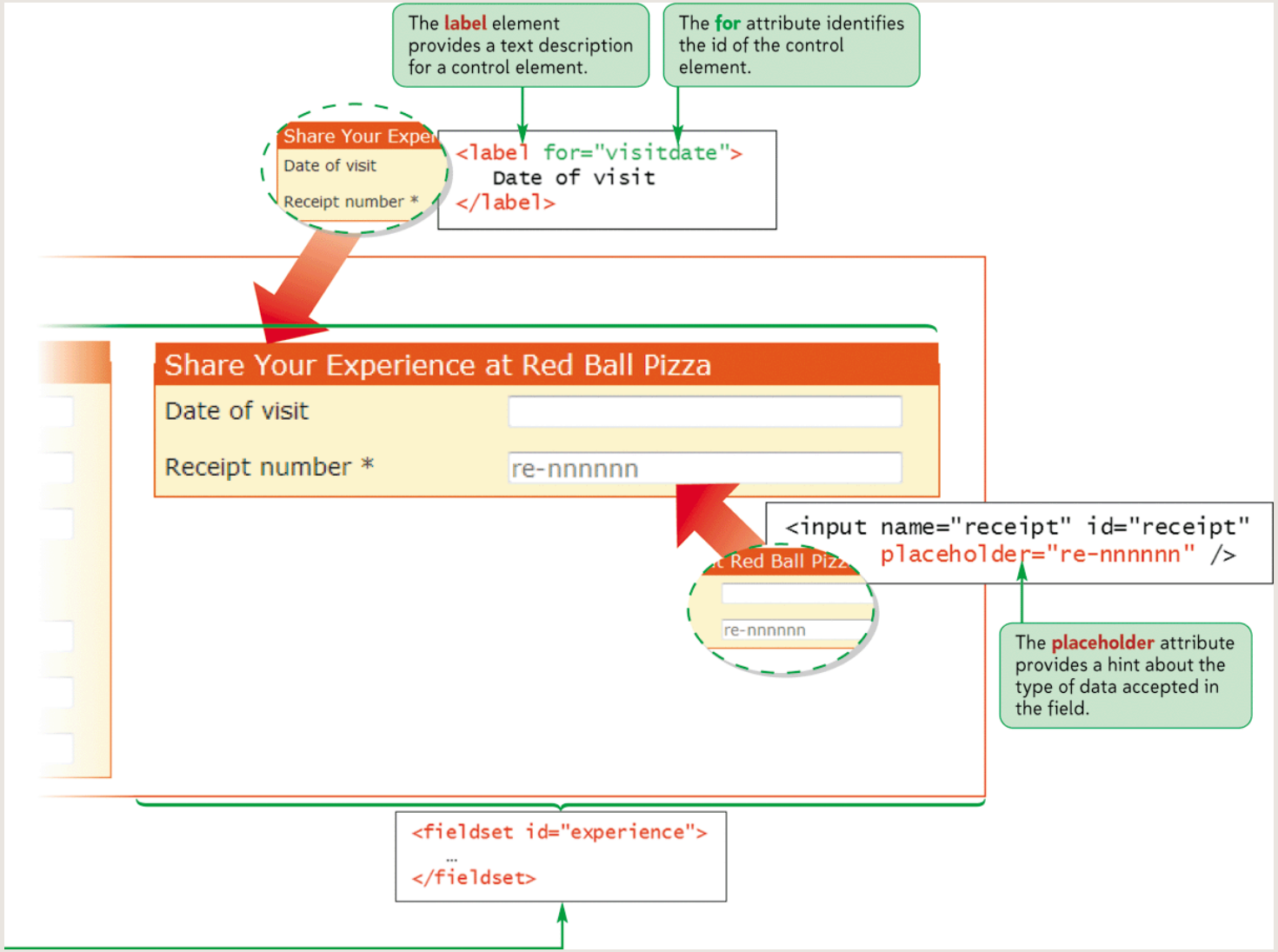


# Form Elements & Attributes
# 폼 요소 및 속성들



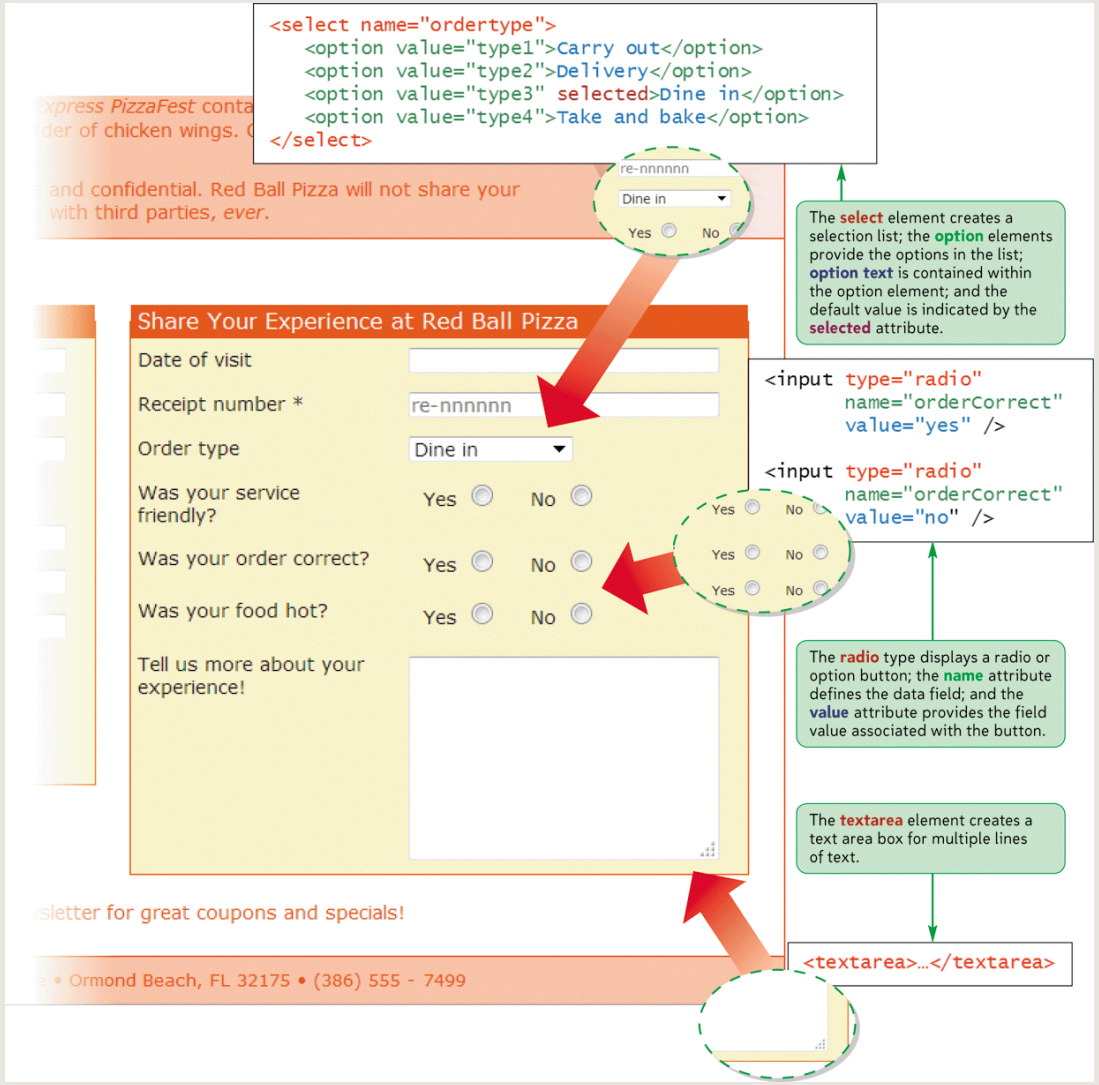

- Fieldset - 제어요소와 데이터 필드를 하나로 묶음
- Input - 데이터를 입력하기 위한 입력요소를 만듬
  - name 속성 – 데이터 필드의 이름을 제공
  - id 속성 – 입력박스를 식별
  - value 속성 – 기본 필드값을 제공
- Legend – 필드셋에 대한 범례를 제공

# Form Elements & Attributes
# 폼 요소 및 속성들



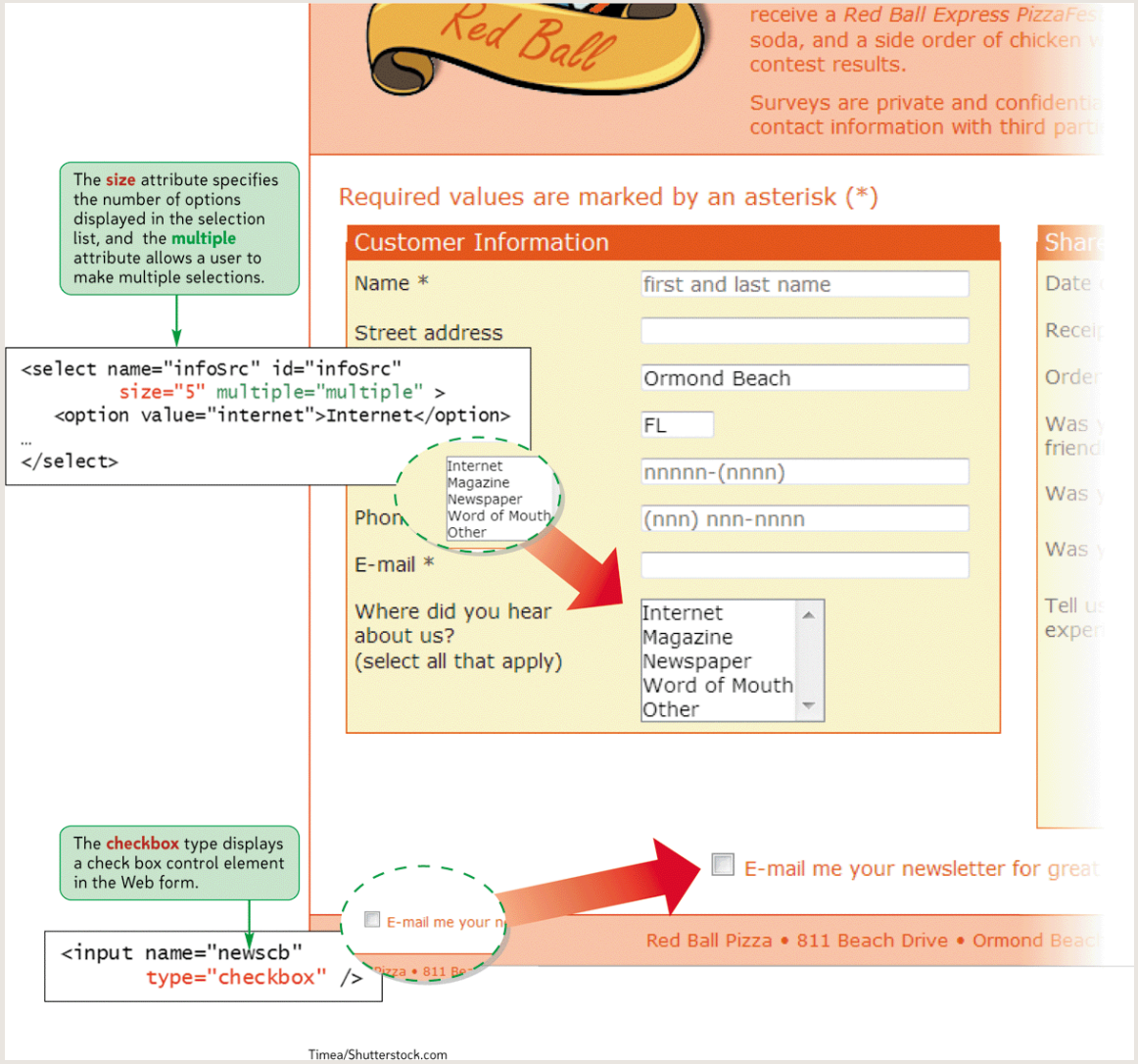

- Label – 제어요소에 대한 텍스트 설명을 제공
  - For 속성 – 제어요소의 id를 식별
- Input
  - Placeholder 속성 – 필드에 입력될 데이터의 종류에 대한 힌트를 제공

# Form Elements & Attributes
# 폼 요소 및 속성들



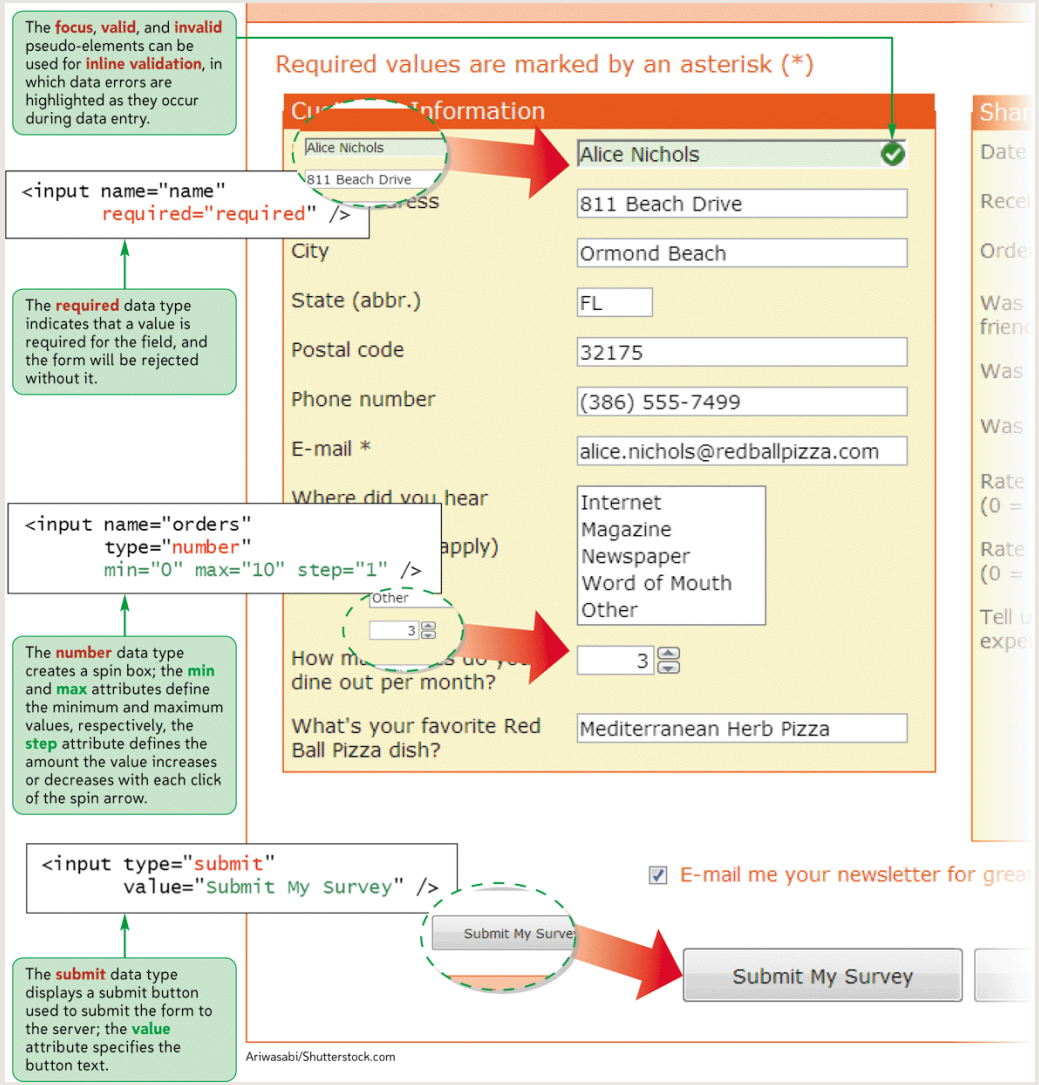

- Select – 선택 리스트를 생성
  - option 속성 – 리스트의 옵션을 제공, 옵션요소 안에 option text가 포함됨
  - selected 속성 – 기본값을 표시
- Radio – 라디오 또는 옵션 버튼을 보여줌
  - name 속성 - 데이터 필드를 정의
  - value 속성 – 버튼과 관련된 필드값을 제공
- Textarea – 여러줄의 텍스트를 위한 텍스트 영역박스를 생성

# Form Elements & Attributes
# 폼 요소 및 속성들



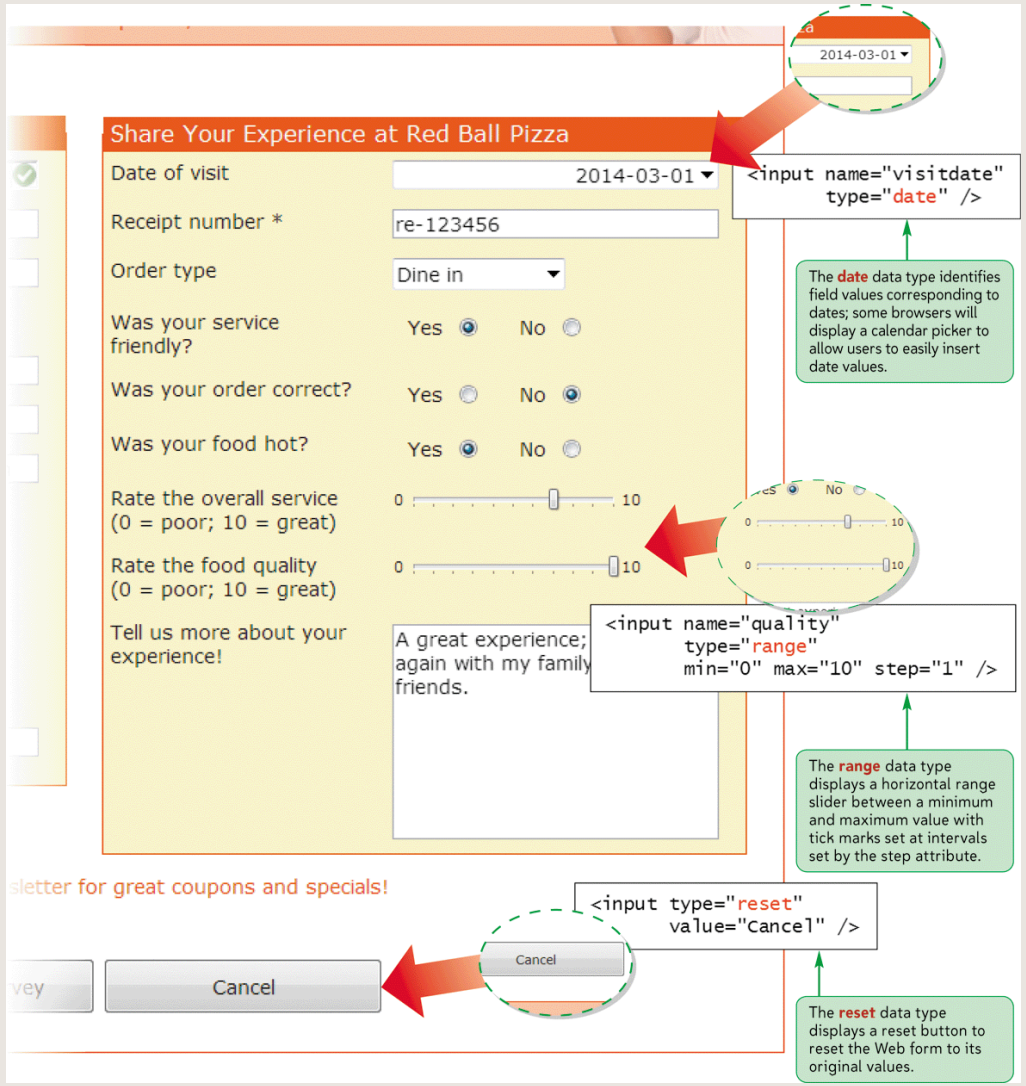

- Select
  - size 속성 – 선택 리스트에서 표시될 옵션의 숫자를 보여줌
  - multiple 속성 – 여러 개 선택
- Checkbox – 폼 안에 체크박스 제어요소를 보여줌

# Form Elements & Attributes
# 폼 요소 및 속성들




- Required – 필드의 필수사항 표시, 값이 없으면 에러출력
- Number – 스핀박스를 생성
  - min, max 속성 – 최소, 최대값을 정의
  - step 속성 – 스핀화살표를 클릭할 때 증가 또는 감소하는 값의 정도를 정의
- Submit – 서버로 폼을 보내기 위한 버튼을 표시
  - Value 속성 – 버튼의 텍스트를 표시
- Focus, valid, invalid 요소 – 입력시 데이터 에러가 발생되면 하이라이트 되는 인라인 검사에 사용됨

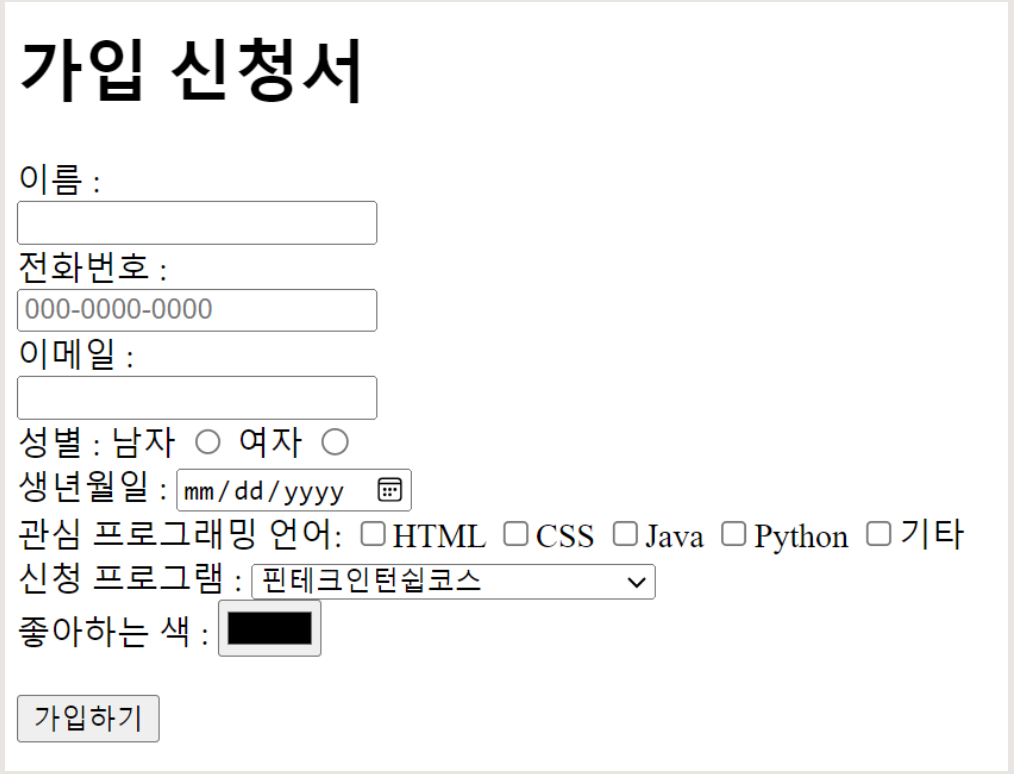
# Form Elements & Attributes
# 폼 요소 및 속성들




- Date – 날짜에 대응되는 필드값을 식별, 어떤 브라우저에서는 캘랜더 픽커를 보여줌
- Range – 수평 범위 스라이더를 보여줌
- Reset – 원래값으로 리셋하는 버튼을 보여줌

# Exercise #5

- 폼을 이용하여 다음 가입신청서를 만드시오





- 부트스트랩과 CSS로 폼을 향상 시키시오

# Web Font 웹폰트

```
@font-face {
            src: url(Yeongdeok.ttf);
            font-family: "yeongdeok";
        }

body {
            font-family: "yeongdeok";
        }
```

- @font-face
  웹브라우저에게 다운받을 위치와 서체 이름을 알려줌
- font-family
  다른 폰트와 사용방법은 같음

# Rounded Borders 둥근 테두리

- p.round2 {
- border: 2px solid red;
- border-radius: 8px;
- padding: 5px;
- }


- p.round3 {
- border: 2px solid red;
- border-radius: 12px;
- padding: 5px;
- }

# Exercise #6

- 테이블을 이용해서 상장법인목록 페이지를 완성하시오

| | 회사명 | 종목코드 | 업종 | 주요제품 | 상장일 | 결산월 | 대표자명 | 홈페이지 | 지역 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | AJ네트웍스 | 95570 | 산업용 기계 및 장비 임대업 | 렌탈(파렛트, OA장비, 건설장비) | 2015-08-21 | 12월 | 손삼달 | http://www.ajnet.co.kr | 서울특별시 |
| 1 | BNK금융지주 | 138930 | 기타 금융업 | 금융지주회사 | 2011-03-30 | 12월 | 빈대인 | http://www.bnkfg.com | 부산광역시 |
| 2 | DSR | 155660 | 1차 비철금속 제조업 | 합섬섬유로프 | 2013-05-15 | 12월 | 홍석빈 | http://www.dsr.com | 부산광역시 |
| 3 | GS | 78930 | 기타 금융업 | 지주회사/부동산 임대 | 2004-08-05 | 12월 | 허태수, 홍순기 (각자 대표이사) | NaN | 서울특별시 |
| 4 | HDC현대산업개발 | 294870 | 건물 건설업 | 외주주택, 자체공사, 일반건축, 토목 등 | 2018-06-12 | 12월 | 최익훈, 정익희, 김회언 (각자 대표이사) | http://www.hdc-dvp.com | 서울특별시 |

- 부트스트랩과 CSS로 테이블을 향상 시키시오

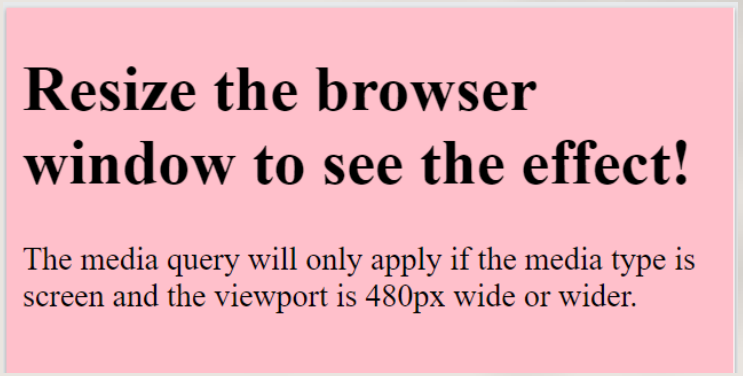
## 상장법인목록

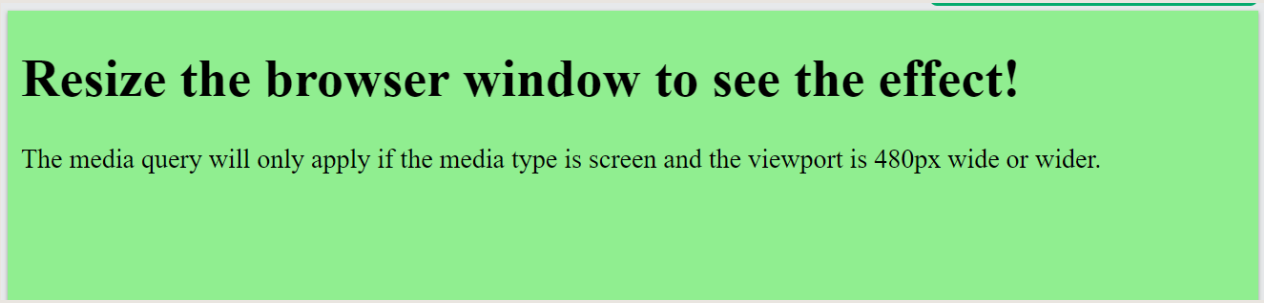
| 회사명 | 종목코드 | 업종 | 주요제품 |
|---|---|---|---|
| AJ네크웍스 | BNK금융지주 | DSR | GS |
| 95570 | 138930 | 155660 | 78930 |
| 산업용 기계 및 장비 임대업 | 기타 금융업 | 1차 비철금속 제조업 | 기타 금융업 |
| 렌탈 | 금융지주회사 | 합성섬유로프 | 지주회사/부동산 임대 |

# Media Queries 미디어 쿼리

- Define different style rules for different media types. **다른 미디어 타입에 따라 다른 스타일 룰을 정의**

```
body {
  background-color: pink;
}

@media screen and (min-width: 480px) {
  body {
    background-color: lightgreen;
  }
}
```



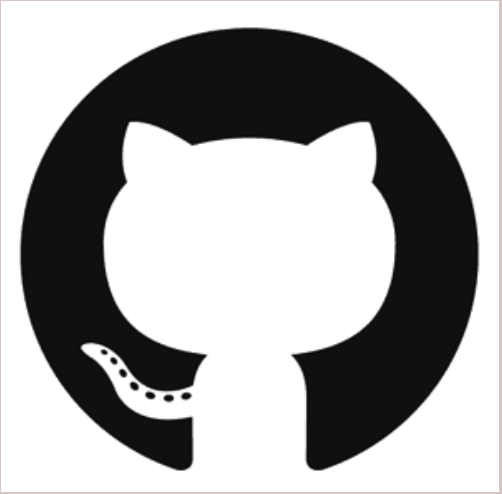
# CSS Prefixes 접두사

-webkit- : 구글, 사파리 브라우저
-moz- : 파이어팍스 브라우저
-ms- : 익스플로러, 보통 생략
-o- : 오페라 브라우저

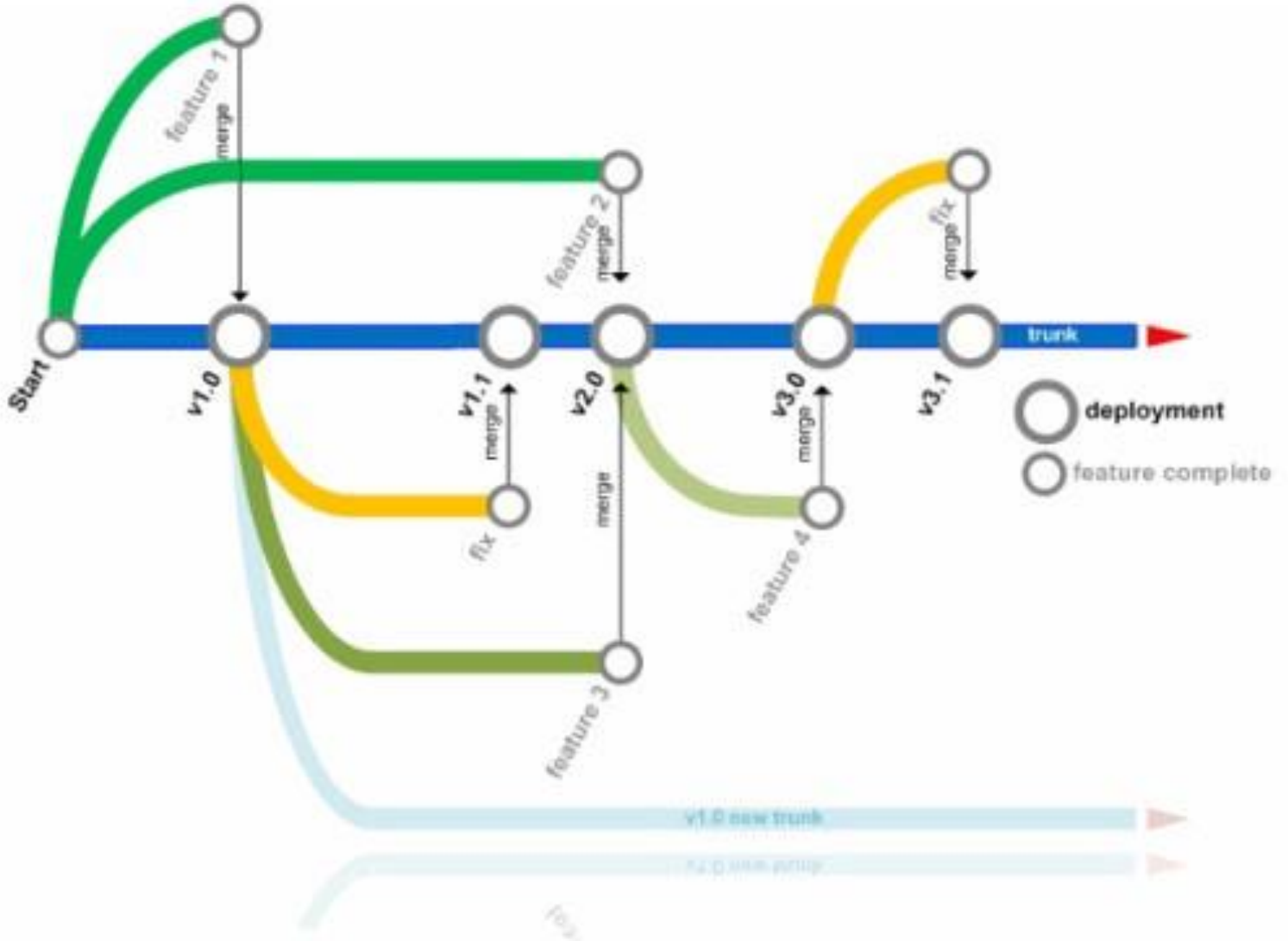
```
#grad1 {
height: 200px;
/* For Safari 5.1 to 6.0 */
background: -webkit-linear-gradient(left, red , blue);

/* For Opera 11.1 to 12.0 */
background: -o-linear-gradient(right, red, blue);

/* For Firefox 3.6 to 15 */
background: -moz-linear-gradient(right, red, blue);

/* Standard */
background: linear-gradient(to right, red , blue); }
```

# VCS (Version Control System)

- A system that records changes to a file or set of files over time so that you can recall specific versions later
  **시간에 따른 파일의 변화를 기록하는 시스템**
- Integrates work done simultaneously by different team members
  **다른 팀멤버들이 동시에 작업하는 것을 통합**
- Example: Bitbucket, Gitlab, Github, etc.

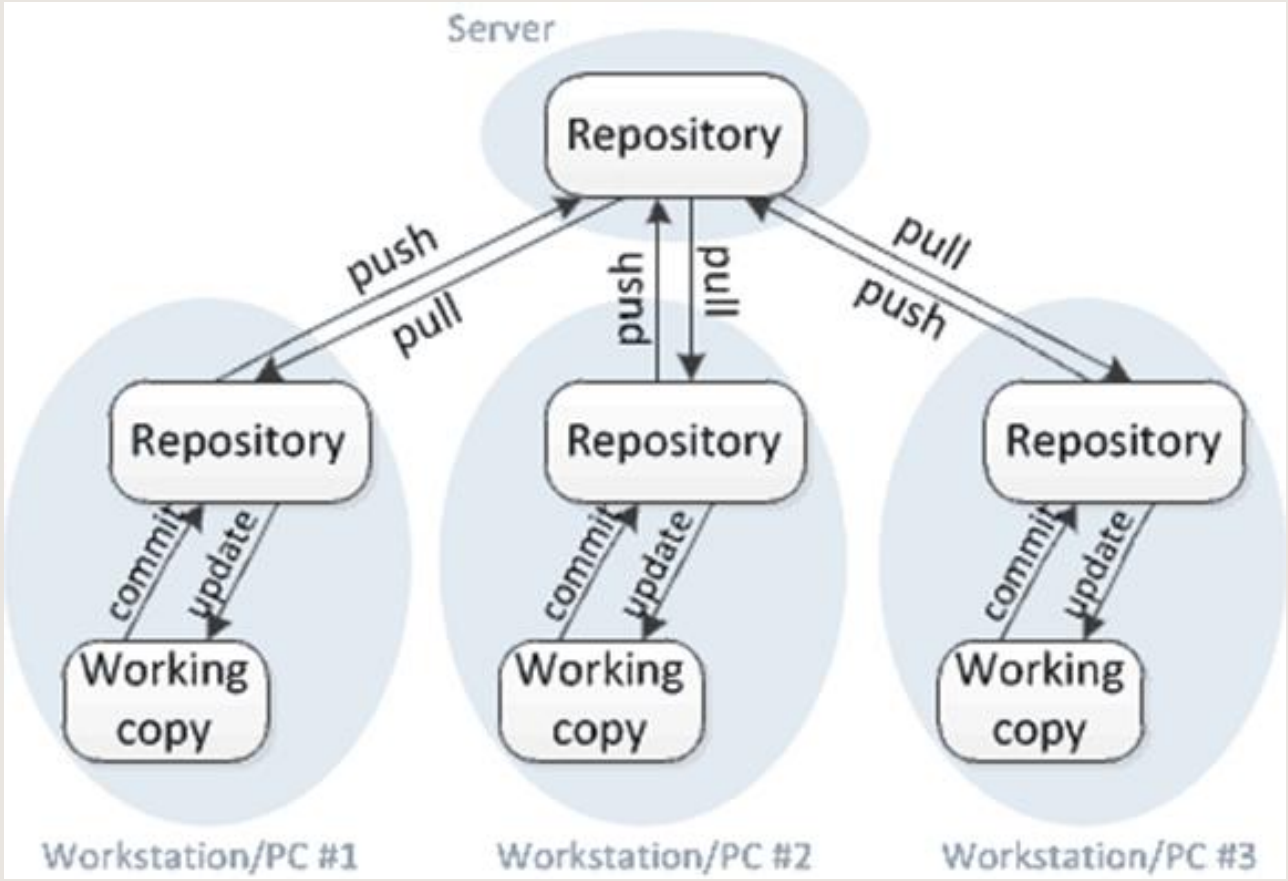


# Branch and Merge




브랜치에서 작업하다가 마스터파일에 합침

# Distributed Version Control



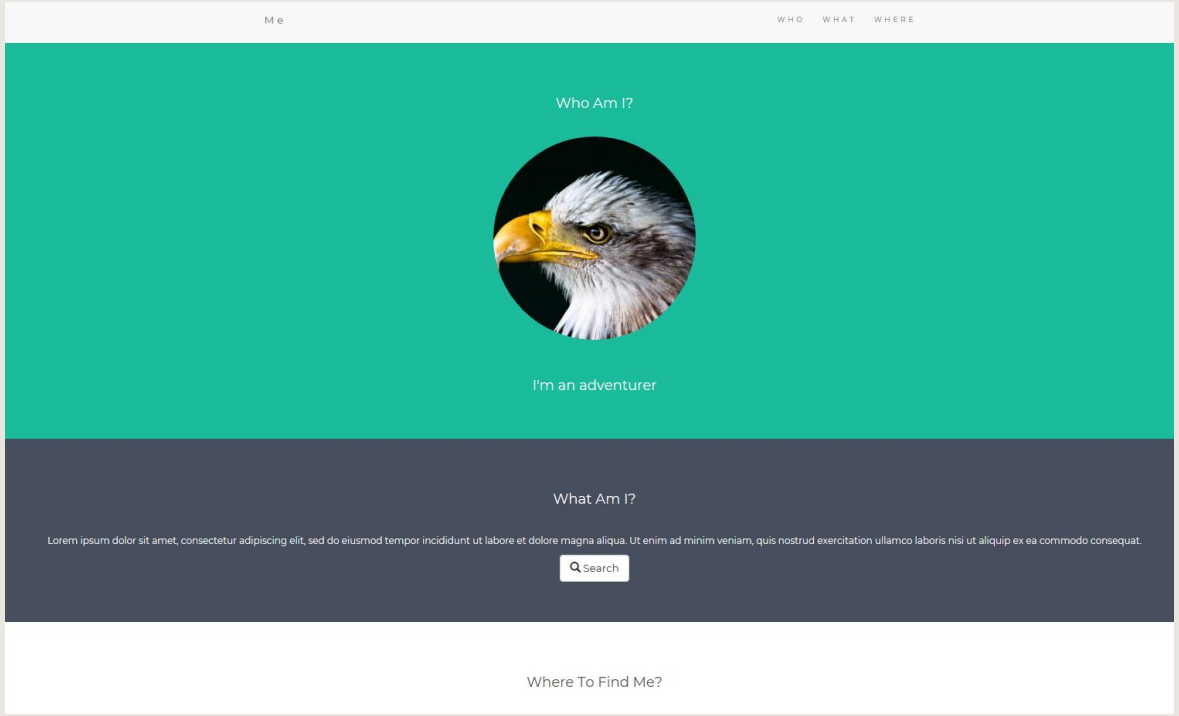

- 로컬컴퓨터에서 작업한 다음 로컬 repo에 커밋하고 서버에 있는 repo에 푸쉬

# Git Bash Commands

```
~ $ git clone https://github.com/username/username.github.io
~ $ cd username.github.io
~ $ echo "Hello World" > index.html
~ $ git add --all
~ $ git commit -m "Initial commit"
~ $ git push -u origin master
```

# Exercise #7

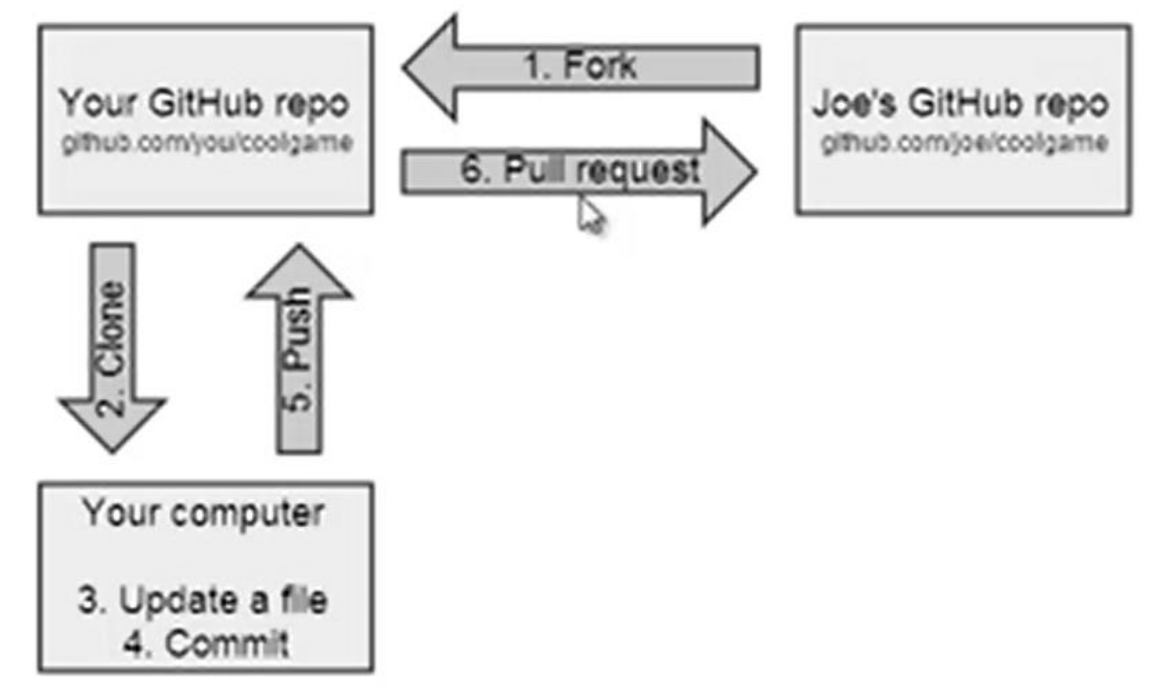
- Upload your file to Github. **부트스트랩 파일을 깃허브에 올리시오**
  - Instructions - https://pages.github.com/



# Fork 포크

- A copy of a repository. **저장소의 카피**
- Most commonly, forks are used to either propose changes to someone else's project or to use someone else's project as a starting point for your own idea without affecting the original project. **포크를 이용해서 다른 사람의 프로젝트를 바꾸는 것을 제안하거나 다른 사람 프로젝트를 가져와서 자기가 작업할 수 있음**

# Fork a Repository 저장소 포크




- Fork the repository.
- Make the fix.
- Submit a pull request to the project owner.
  **포크하고 고친다음 풀 리퀘스트 보냄**

# Exercise #8

- Create and use a repository, start and manage a new branch, make changes to a file and push them to GitHub as commits, and open and merge a pull request. **저장소를 만들어 작업하고, 새로운 브랜치를 만들고, 파일을 바꾼 후 커밋을 이용하여 푸쉬하고, 풀 리퀘스트를 이용하여 합치시오**
  - Instructions - https://guides.github.com/activities/hello-world/