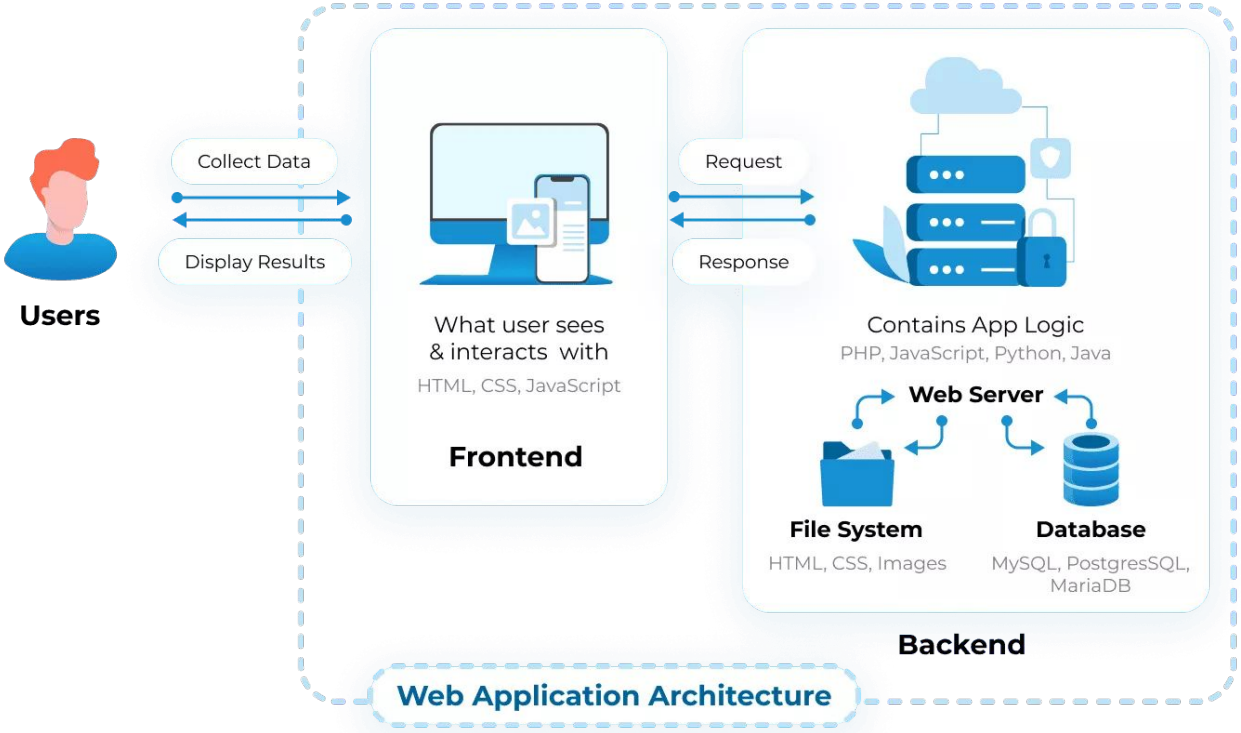
SWPP Practice Session #1
Basics of Python, Typescript, HTML

Sep. 7, 2022

Objectives

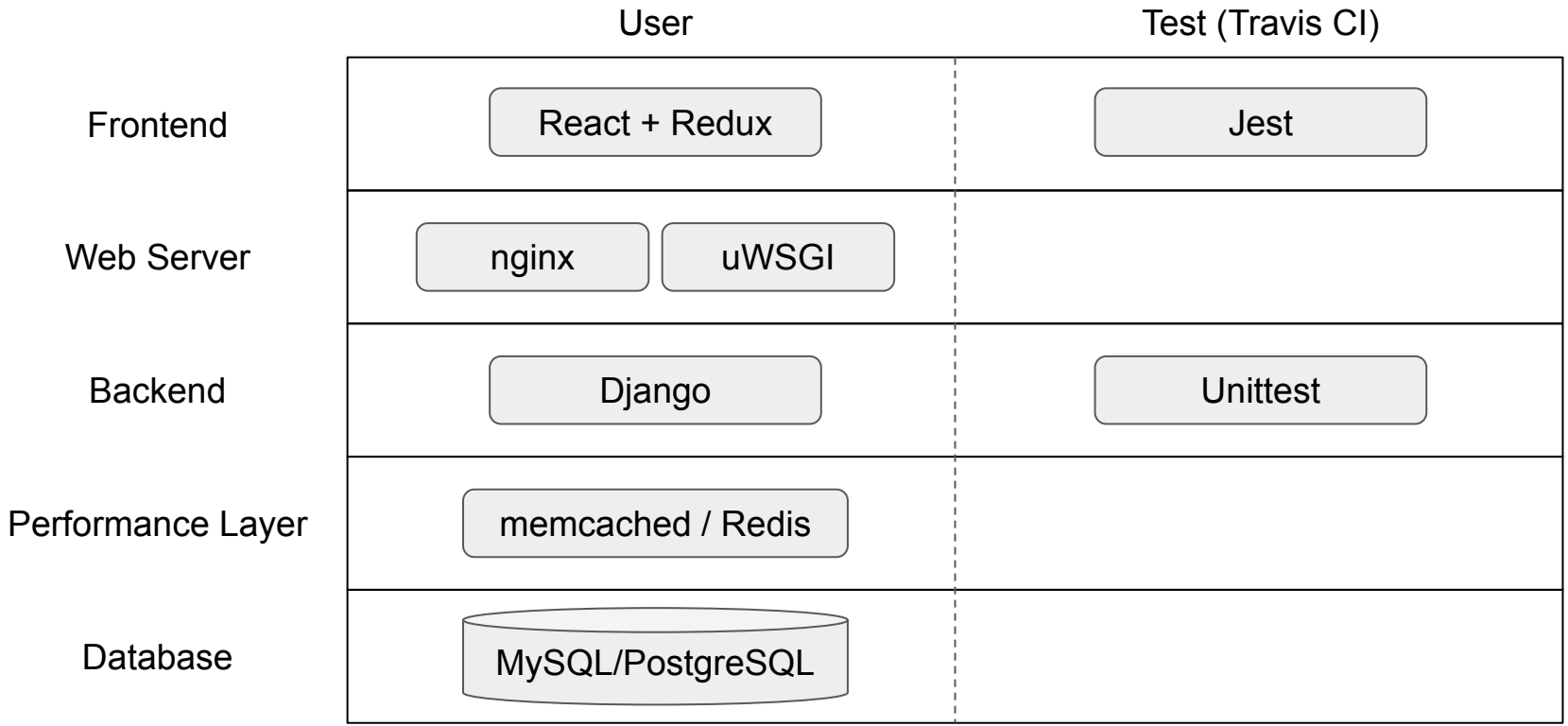
- Web Service Overview
- Environments
- Python Basics
- Typescript Basics
- HTML Basics

Web Service Overview



source:
https://litslink.com/blog/web-application-architecture

Web Service Overview



What We are Going to Learn

- https://github.com/swsnu/swppfall2022

Service Environment

- Frontend : React (17.0.2) + Redux (4.1.1)
- Backend : Django (3.2.6) (Python >= 3.7)
- Browser : Chrome, Firefox
- OS (highly recommended) : Ubuntu 18.04.6 LTS
  - especially for Windows users

IDE(Integrated Development Environment)

- Visual Studio Code (**Recommended**)
  - https://code.visualstudio.com/
  - Highly configurable IDE
- Intellij / Pycharm
  - https://www.jetbrains.com/idea/
  - Get student pack license for free with your *snu.ac.kr* email
- Vim :)

VSCode Setup

- https://docs.microsoft.com/en-us/learn/modules/develop-web-apps-with-vs-code/

Some Useful Extensions for SWPP in VSCode (1)

You can search and install extensions in VSCode app (`Code > Preferences > Extensions`).

- Version Control
  - GitLens, Git History
- Languages, Frameworks, and Tools
  - Python
  - Kite Autocomplete for Python and JavaScript
  - React Native Tools, ES7 React / Redux / GraphQL / React-Native Snippets, ESLint
  - Django, django-intellisense
  - npm intellisense
  - Docker
  - LiveServer

Some Useful Extensions for SWPP in VSCode (2)

You can search and install extensions in VSCode app (`Code > Preferences > Extensions`).

- IDE Decoration
  - Vim
  - vscode-icons, TODO Highlight, Date & Time
- Useful
  - Rainbow CSV, Resource Monitor, indent-switcher, markdownlint, Markdown Preview Enhanced

Python

Python 

- The most popular programming language nowadays (thanks to ML)

- You can download from official Python
- You can use famous data science platform, anaconda
- Or you can just use Linux's pre-installed Python!

Python 

- For today’s practice session, we will cover the followings for Python:
  - If-Else
  - For loop
  - Function and lambda function
  - Decorator
  - Class
  - Static method, Class method
  - Class inheritance

Basic Python

If-Else program example

```
#!/usr/bin/python

var = 100
if var == 200:
   print("1 - Got a true expression value")
   print(var)
elif var == 150:
   print("2 - Got a true expression value")
   print(var)
elif var == 100:
   print("3 - Got a true expression value")
   print(var)
else:
   print("4 - Got a false expression value")
   print(var)

print("Good bye!")
```

```
3 - Got a true expression value
100
Good bye!
```

Basic Python

For loop program example

```
#!/usr/bin/python

for letter in 'Python':     # First Example
   print('Current Letter :', letter)

fruits = ['banana', 'apple',  'mango']
for fruit in fruits:        # Second Example
   print('Current fruit :', fruit)

print("Good bye!")
```

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

```
#!/usr/bin/python

fruits = ['banana', 'apple',  'mango']
# range(len(fruits)) = [0, 1, 2]
for index in range(len(fruits)):
   print('Current fruit :', fruits[index])

print("Good bye!")
```

```
Current fruit : banana
Current fruit : apple
Current fruit : mango
Good bye!
```

Basic Python

For loop program example

```
#!/usr/bin/python

fruits = ['banana', 'apple',  'mango']
for i, fruit in enumerate(fruits):
  print('fruit number %d :' % i, fruit)

print("Good bye!")
```

```
#!/usr/bin/python

fruits = ['banana', 'apple',  'mango']
numbers = [0, 1, 2]
for i, fruit in zip(numbers, fruits):
  print('fruit number %d :' % i, fruit)

print("Good bye!")
```

```
fruit number 0 : banana
fruit number 1 : apple
fruit number 2 : mango
Good bye!
```

Basic Python

Function, lambda example

```
#!/usr/bin/python

# Define function
def is_even(n):
      return n % 2 == 0


print(is_even(5))
print(is_even(8))

# You can treat function like variable
is_not_odd = is_even

print(is_not_odd(6))
```

```
False
True
True
```

```
#!/usr/bin/python

is_even = lambda n: n % 2 == 0
# This is an anonymous function using ‘lambda’
# This is equivalent to left


is_odd = lambda n: not is_even(n)


print(is_even(8))
print(is_odd(8))
```

```
True
False
```

Basic Python

Decorator

```
#!/usr/bin/python

def logging(func):
      def wrapper():
            print("Logged")
            return func()
       return wrapper

def foo():
      print("Foo")
foo = logging(foo)
foo()
```

```
Logged
Foo
```

```
#!/usr/bin/python

def logging(func):
      def wrapper():
            print("Logged")
            return func()
       return wrapper

@logging # Equivalent to `foo = logging(foo)`
def foo():
      print("Foo")
foo()
```

```
Logged
Foo
```

Basic Python

Class example

```
#!/usr/bin/python

# Define class
class Person:
      def __init__(self, name):     # Class Constructor
            self.name = name        # The first argument ‘self’ means instance itself

      def yourName(self):
            print(self.name)

alice = Person("Alice")             # Create new instance
alice.yourName()                    # Equivalent to `Person.yourName(alice)`
```

```
Alice
```

Basic Python

Static method, Class method example

```
#!/usr/bin/python

# Define class
class Employee:
      salary = 1000
      @staticmethod
      def greeting():                 # Static methods don’t get argument ‘self’
            print("Hello!")

      @classmethod
      def get_salary(cls):            # Class method get first argument ‘cls’ instead of ‘self’
            return cls.salary

alice = Employee()                    # Create new instance
alice.greeting()                      # Call static method with instance
Employee.greeting()                   # Call static method with class
print(Employee.get_salary())          # Call class method
```

```
Hello!
Hello!
1000
```

Basic Python

Class inheritance example

```
#!/usr/bin/python

# Define class
class Person:
      def __init__(self, name):     # Class Constructor
            self.name = name        # The first argument ‘self’ means instance itself

      def yourName(self):
            print(self.name)

class Student(Person):
      def __init__(self, name):
            super().__init__(name)
      def isStudent(self):
            print(“I am a student!”)

alice = Student("Alice")            # Create new instance
alice.yourName()
alice.isStudent()
```

```
Alice
I am student!
```

Supplement

- You can follow simple python learn module of MS Learn Module (Not mandatory)
  https://docs.microsoft.com/ko-kr/learn/modules/intro-to-python/

Python Practice

Basic Python programming

Create file

```
$ vi prime5to50.py
```

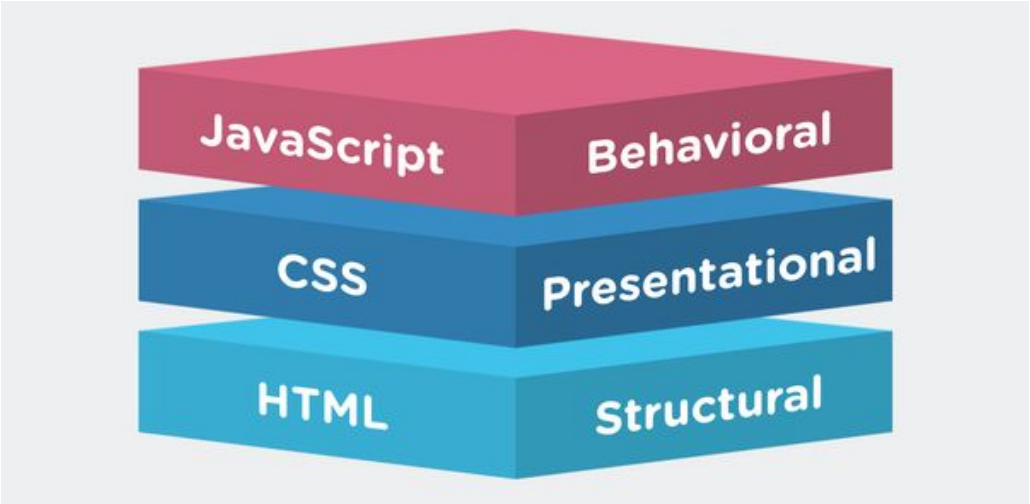
```
# Your task is to create a program that prints all prime numbers in [5, 50].

# References for If-else, for syntax is provided in the previous slides.

# It should print the outputs as below:
```

```
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
...
```

JavaScript

JavaScript

- The most popular scripting language for web pages



Node.js 

- Let’s install Node.js to run Javascript programs
  - It is a Javascript runtime built on Chrome’s V8 Javascript engine
  - npm(Node Package Manager) is a package manager for node.js
- How to Install
  - `$ sudo apt-get install npm # for ubuntu`
  - `$ brew install npm # for macOS`
  - or manually install node.js and npm in https://nodejs.org/en

-> We need node.js for React later on to build frontend systems

Basic JavaScript (Types)

Supported types

Data type	Keyword	Description
Number	number	Double precision 64-bit floating point values. It can be used to represent both, integers and fractions.
String	string	Represents a sequence of Unicode characters
Boolean	Boolean	Represents logical values, true and false
Void	void	Used on function return types to represent non-returning functions
Null	null	Represents an intentional absence of an object value.
Undefined	undefined	Denotes value given to all uninitialized variables

Note – There is no integer type in TypeScript and JavaScript.

Basic JavaScript (Variables)

Different ways to declare variables

1. var

```
var num = 12;
console.log(num);
var num = 15;
console.log(num);
```

let

```
let num = 12;
console.log(num);
let num = 15;
console.log(num);
```

const

```
const num = 12;
console.log(num);
const num = 15;
console.log(num);
```

no re-declaration allowed.

SyntaxError: Identifier 'num' has already been declared

Basic JavaScript (Variables)

Different ways to declare variables

1. var

```
var num = 12;
console.log(num);
var num = 15;
console.log(num);
```

2. let

```
let num = 12;
console.log(num);
num = 15;
console.log(num);
```

const

```
const num = 12;
console.log(num);
num = 15;
console.log(num);
```

const is immutable.

TypeError: Assignment to constant variable.

Basic JavaScript (If-Else)

If-Else statement

```
var num = 12; // It’s better to use const or let if possible
if (num % 2 === 0) {
   console.log("Even"); // console.log() is a function used to print to the browser console.
} else {
   console.log("Odd");
}
```

Difference between `==`(Equal Operator) and `===`(Strict Equal Operator)

- in javascript, === (triple equals) check strict equality, both type and value.
- Rather, == (double equals) check loose equality
- e.g.
  77 === 77 (true), 77 === ‘77’ (false)
  77 == 77 (true), 77 == ‘77’ (true)

Basic JavaScript (Loop)

For loop program example

```
var j;
var n = "hello world";

for(j in n) {  // keys
   console.log(n[j]);
}
```

```
for(j of n) {  // values
   console.log(j);
}
```

```
$ node loop.js
h
e
l
l
o

w
o
r
l
d
```

```
var num = 5;
var i;
var factorial = 1;

for(i = num; i>=1; i--) {
   factorial *= i;
}
console.log("The factorial is " + factorial)
```

```
The factorial is 120
```

Basic JavaScript (Function)

Functions

```
function add1(x, y) {
  // function declaration
  return x + y;
}
var add2 = function(x, y) {
  // function expression
  return x + y;
}
// var add2 = (x, y) => {
//  // an arrow function also works.
//  return x + y;
// }
console.log(add1(10, 20));
console.log(add2(10, 20));
```

```
$ node function.js
30
30
```

Basic JavaScript (Class)

class example

```
class Person {
  constructor(name) {
    this.name = name;
  }
  sayHello() {
    console.log(this.name + “ says Hello!”);
  }
}

person1 = new Person(“Alice”);
person1.sayHello();
```

```
 $ node class.js
Alice says Hello!
```

Basic JavaScript (Class)

static method example

```
class Person {
  constructor(name, number) {
    this.name = name;
    this.number = number;
  }
  static compare(person1, person2) {
    return person1.number - person2.number;
  }
}
var persons = [
  new Person(“Alice”, 0);
  new Person(“Bob”, 2);
  new Person(“Steve”, 1);
]
```

```
persons.sort(Person.compare)  // use static method

for (p in persons) {
  console.log(persons[p].name);
}
```

```
$ node static.js
Alice
Steve
Bob
```

Basic JavaScript (Class)

class inherit example

```
class Person {
  constructor(name) {
    this.name = name;
  }
  sayHello() {
    console.log(this.name + " says Hello!");
  }
}

class Student extends Person {
  goSchool() {
    console.log(this.name + " goes to school");
  }
  sayHello() {
    super.sayHello();
    console.log(this.name + " is a student!");
  }
}
```

```
s = new Student("Alice");
s.sayHello();
s.goSchool();
```

```
$ node inherit.js
Alice says Hello!
Alice is a student!
Alice goes to school
```

Basic JavaScript (Array)

Some Useful Functions for arrays

- forEach() : loop method for array
- map() : apply a given function to all the elements in an array
- filter() : filter elements that satisfy a specific condition

```
var arr1 = [1, 2, 3, 4];
arr1.forEach(n => {
      console.log(n);
}); // prints 1, 2, 3, 4
```

```
var arr2 = arr1.map(function(n) {
      return n + 1;
});
console.log(arr2); // [2, 3, 4, 5]
```

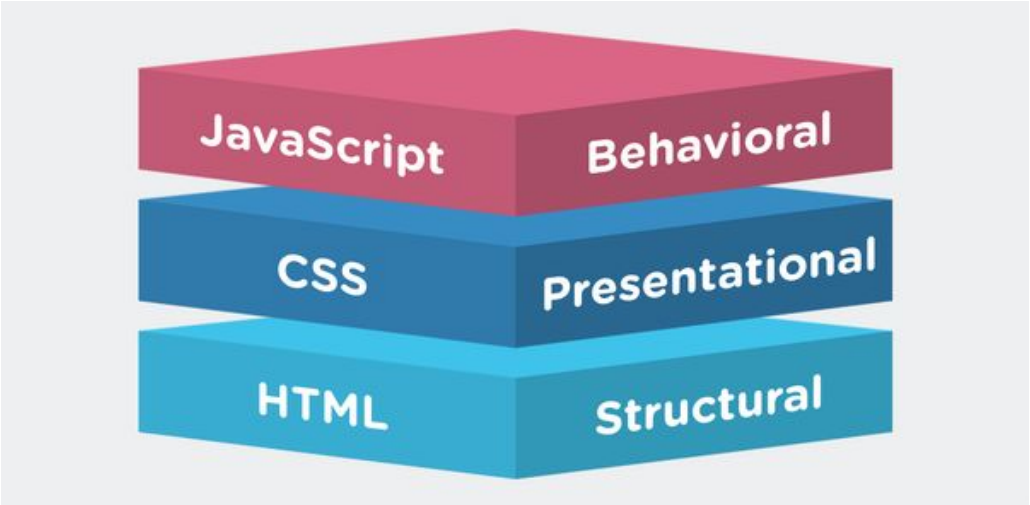
```
var arr3 = arr1.filter(n => {
      return n % 2 == 0;
});
console.log(arr3); // [2, 4]
```

You can also use arrow function that works like lambda function!

HTML

HTML

- Standard markup language for Web pages
  - HTML is not a programming language
  - HTML is used to format your Web page



Basic HTML

Structure overview

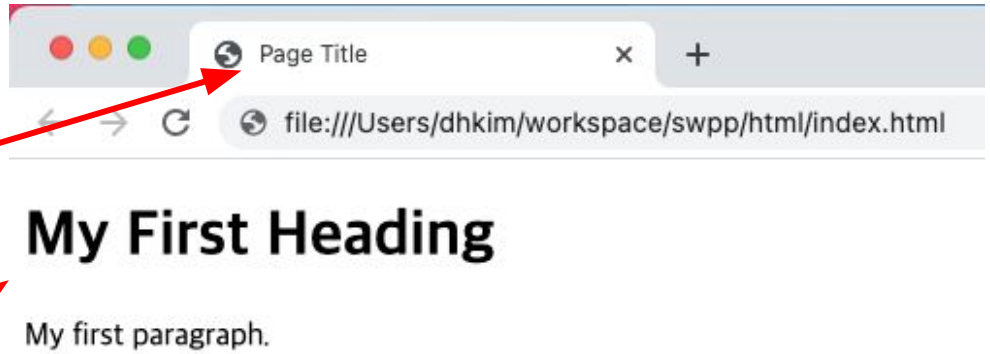
```
<!DOCTYPE html>
<html>
      <head>
            <title>Page Title</title>
      </head>
      <body>
            <h1>My First Heading</h1>
            <p>My first paragraph.</p>
      </body>
</html>
```

source:
https://www.w3schools.com/html/html_intro.asp

Basic HTML

Structure overview

```
<!DOCTYPE html>
<html>
      <head>
            <title>Page Title</title>
      </head>
      <body>
            <h1>My First Heading</h1>
            <p>My first paragraph.</p>
      </body>
</html>
```



source:
https://www.w3schools.com/html/html_intro.asp

Basic HTML

Structure overview

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>My First Heading</h1>
        <p>My first paragraph.</p>
    </body>
</html>
```

Element?

- defined by a start tag, content and an end tag
  - start tag <...>, end tag </...>
- Some empty elements (e.g.
) does not have an end tag!

Basic HTML

Structure overview

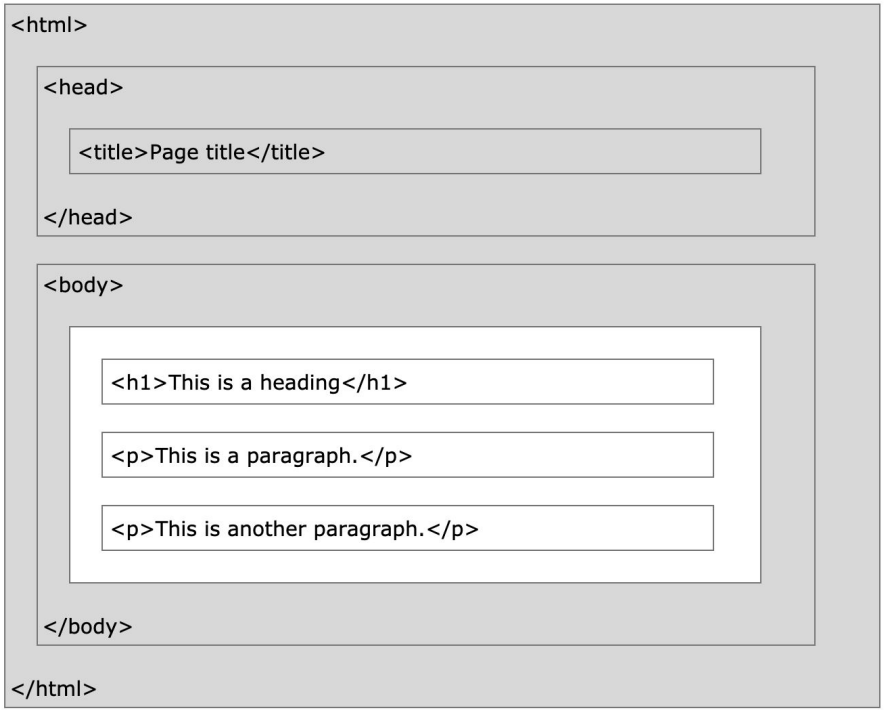
```
<!DOCTYPE html>
<html>
	<head>
		<title>Page Title</title>
	</head>
	<body>
		<h1>My First Heading</h1>
		<p>My first paragraph.</p>
	</body>
</html>
```

- `<!DOCTYPE html>` declares that this document is an HTML5 document
- `<html>` is the root element of an HTML page
- `<head>` contains meta information about the HTML page
- `<title>` specifies a title for the HTML page (shown in the browser's title bar)
- `<body>` defines the document's body, a container for all the visible contents (e.g. headings, paragraphs, images, hyperlinks, tables, lists, etc).
- `<h1>` defines a large heading
- `<p>` defines a paragraph
- ...

Basic HTML

Structure overview

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



Basic HTML Page

More examples

`<h1> ~ <h6>:` defines heading. h1 is the most important heading

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```

`<p>:` defines paragraph

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

`<a>:` defines links

```
<a href="a>This is a link</a>
```

`<img>:` defines images

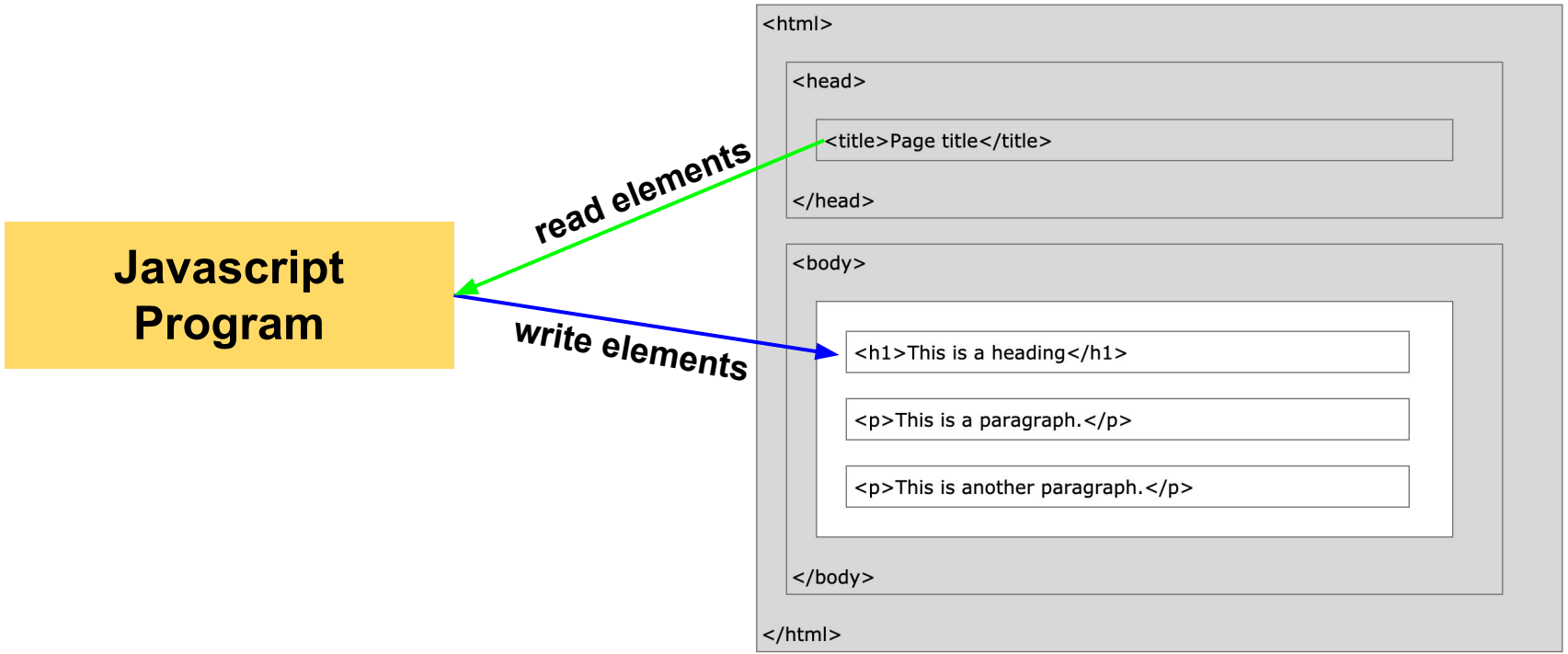
```
<img src="example.jpg" width="104" height="142">
```

(width, height in <img> are **attributes** of the element. Keep that in mind that there are attributes in the elements of html)

DOM (Document Object Model)

- Programming interface for HTML, XML ...
- Allows structured representation of a **document** (i.e. Web page)
- Expresses document using **nodes**, **property** and **objects** that have methods
  - Object-oriented representation of web pages
- Helps the programming language (e.g. Javascript) to access components of the DOM
  - change the structure, style, content, etc.

DOM (Document Object Model)



DOM (Document Object Model)

- Some examples of DOM APIs
  - `document.getElementById(id)`
  - `document.getElementsByTagName(name)`
  - `document.createElement(name)`
  - `element.innerHTML`
  - `element.setAttribute`
  - `element.getAttribute`
  - `element.addEventListener`
  - `window.content`
  - `window.onload`

HTML + Javascript

- First, create a file named **`index.html`**

index.html

```
<!DOCTYPE html>
<html>
      <head>
            <title>Page Title</title>
      </head>
      <body>
            <h1>This is a heading</h1>
            <p id=”p1”>This is a paragraph.</p>
            <p id=”p2”>This is another paragraph.</p>
            <script src="app.js"></script>
      </body>
</html>
```

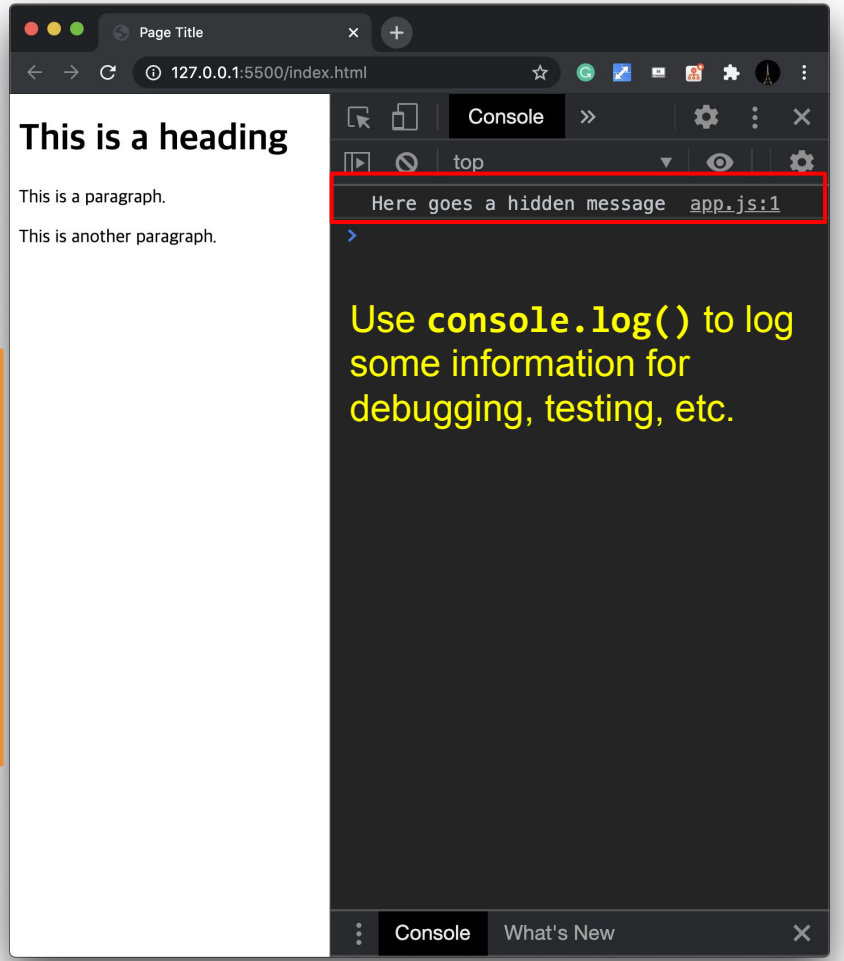
Same html content with the previous slide, but add a single line:
<script src="app.js"></script>

HTML + Javascript

- Next, create a file named **`app.js`**

app.js

```
console.log('Here goes a hidden message');
```

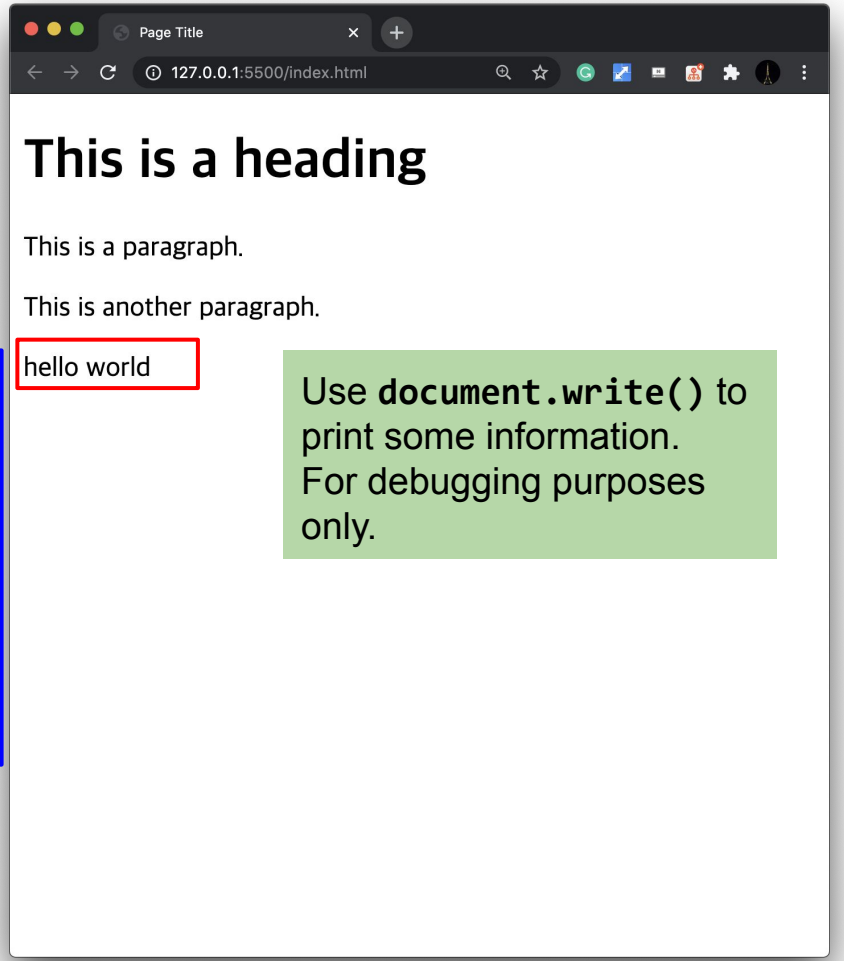


HTML + Javascript

- Next, create a file named **`app.js`**

app.js

```
console.log('Here goes a hidden message');

document.write("hello world");
```



Use **`document.write()`** to print some information. For debugging purposes only.

HTML + Javascript

- Next, create a file named **app.js**

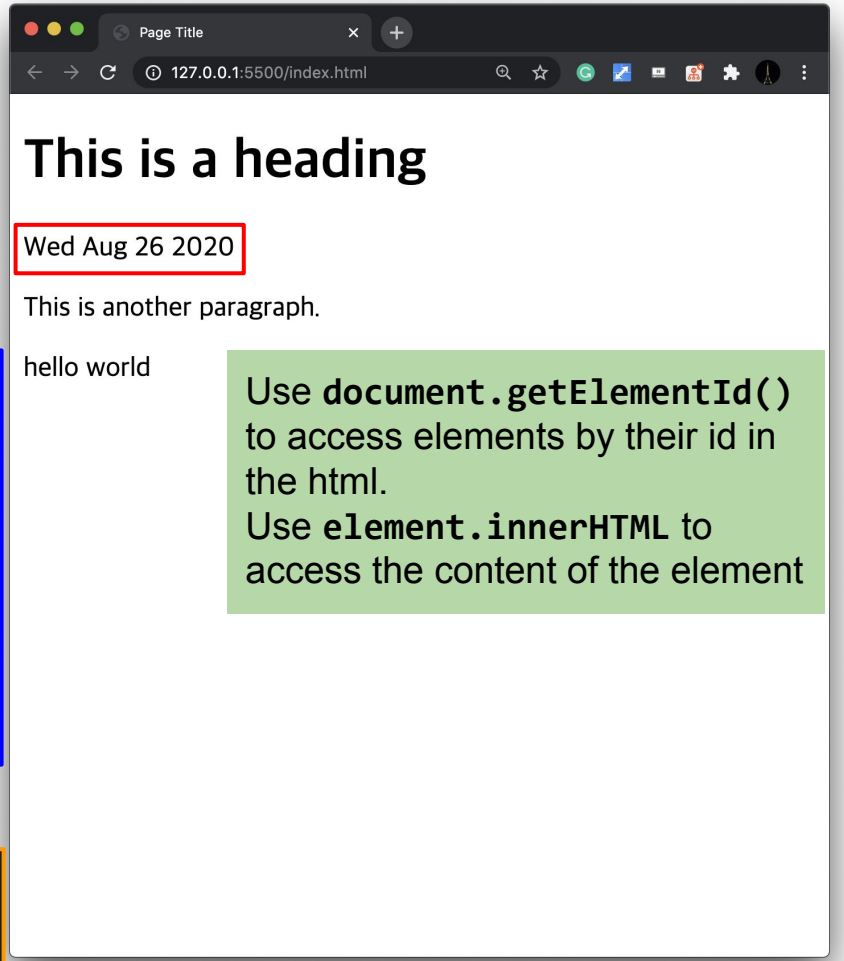
app.js

```
console.log('Here goes a hidden message');
document.write("hello world");

let today = new Date()
let formatDate = today.toDateString()
let selectElement = document.getElementById('p1')
selectElement.innerHTML = formatDate
```

index.html

specify id attribute for the elements

```
<p id='p1'>This is a paragraph.</p>
<p id='p2'>This is another paragraph.</p>
```



Use **document.getElementId()** to access elements by their id in the html.
Use **element.innerHTML** to access the content of the element

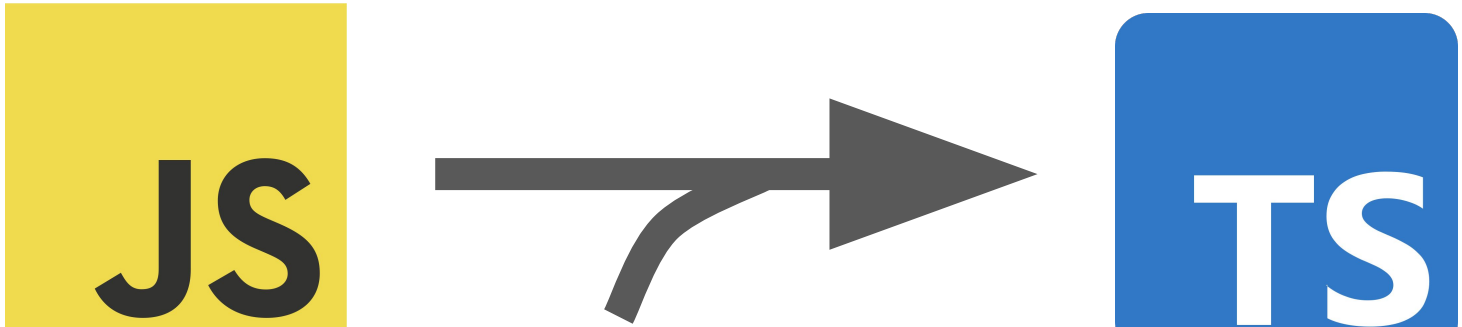
HTML + Javascript

For more details, follow this MS Learn Module

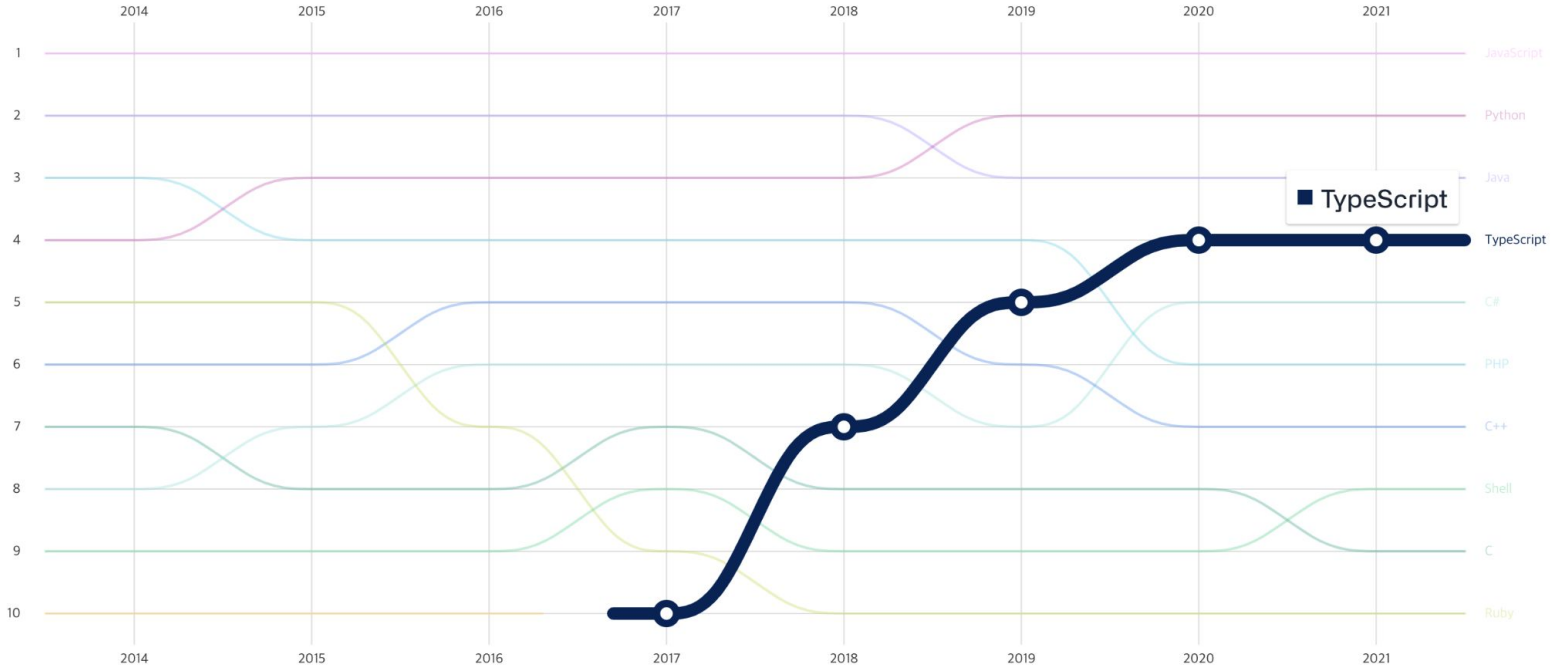
https://docs.microsoft.com/ko-kr/learn/modules/build-simple-website/

TypeScript

TypeScript = JavaScript + Types



- **Interface**
- **Strongly type**
- **Generic**



https://octoverse.github.com

Basic TypeScript (Type)

```
// printCoord.js
// $node printCoord.js
function printCoord(pt) {
 console.log("The coordinate's x value is " + pt.x);
 console.log("The coordinate's y value is " + pt.y);
}

printCoord({ x: 100, y: 100 });
```

```
// printCoord.ts
// $npx ts-node printCoord.ts
type Point = {
 x: number;
 y: number;
};


function printCoord(pt: Point) {
 console.log("The coordinate's x value is " + pt.x);
 console.log("The coordinate's y value is " + pt.y);
}

printCoord({ x: 100, y: 100 });
printCoord({ x: “100”, y: “100” });
```

Type 'string' is not assignable to type 'number'.ts(2322)
printCoord.ts(4, 3): The expected type comes from property 'x' which is declared here on type 'Point'

Basic TypeScript (Interface)

```
// printCoord.js
// $node printCoord.js
function printCoord(pt) {
 console.log("The coordinate's x value is " + pt.x);
 console.log("The coordinate's y value is " + pt.y);
}

printCoord({ x: 100, y: 100 });
```

```
// printCoord.ts
// $npx ts-node printCoord.ts
interface Point {
 x: number;
 y: number;
};

interface BoundingBox extends Point {
  width: number;
  height: number;
};

function printCoord(pt: Point) {
 console.log("The coordinate's x value is " + pt.x);
 console.log("The coordinate's y value is " + pt.y);
}

printCoord({ x: 100, y: 100 });
```

interface can be extended

Basic TypeScript (Generic)

```
// generic.js

class GenericNumber {
 zeroValue;
 add;
}

let myGenericNumber = new GenericNumber();
myGenericNumber.zeroValue = 0;
myGenericNumber.add = function (x, y) {
 return x + y;
};
console.log(myGenericNumber.add(5, 10));
```

```
// generic.ts
// with "strictPropertyInitialization": false
class GenericNumber<NumType> {
 zeroValue: NumType;
 add: (x: NumType, y: NumType) => NumType;
}

let myGenericNumber = new GenericNumber<number>();
myGenericNumber.zeroValue = 0;
myGenericNumber.add = function (x, y) {
 return x + y;
};
console.log(myGenericNumber.add(5, 10));
```

Ref:

- https://www.typescriptlang.org/docs/handbook/2/generics.html#handbook-content

Basic TypeScript (Transpiling)

To run typescript code on html, you need to transpiling to JS code.

1. Run `yarn add typescript @types/node` in shell
2. Prepare configuration file (tsconfig.json)
3. Run `npx tsc ([ts file])
4. JS code will be generated in <outDir>

```
// tsconfig.json
{
  "compilerOptions": {
    "outDir": "./built",
    "allowJs": true,
    "target": "es5",
    "lib": ["dom", "es2015"]
  },
  "include": ["./**/*"]
}
```

Ref:

- https://www.typescriptlang.org/docs/handbook/migrating-from-javascript.html
- https://www.typescriptlang.org/docs/handbook/babel-with-typescript.html

Practice

Basic TypeScript Program with HTML

```
$ vi prime5to50.ts, prime5to50.html
# Your task is to create and transpiling a typescript program that prints all prime numbers between 5 and 50.
# References for If-else, for syntax is provided in the previous slides.
# It should print the outputs as below when you open prim5to50.html:
```

```
5 is a prime number
7 is a prime number
11 is a prime number
13 is a prime number
17 is a prime number
...
```

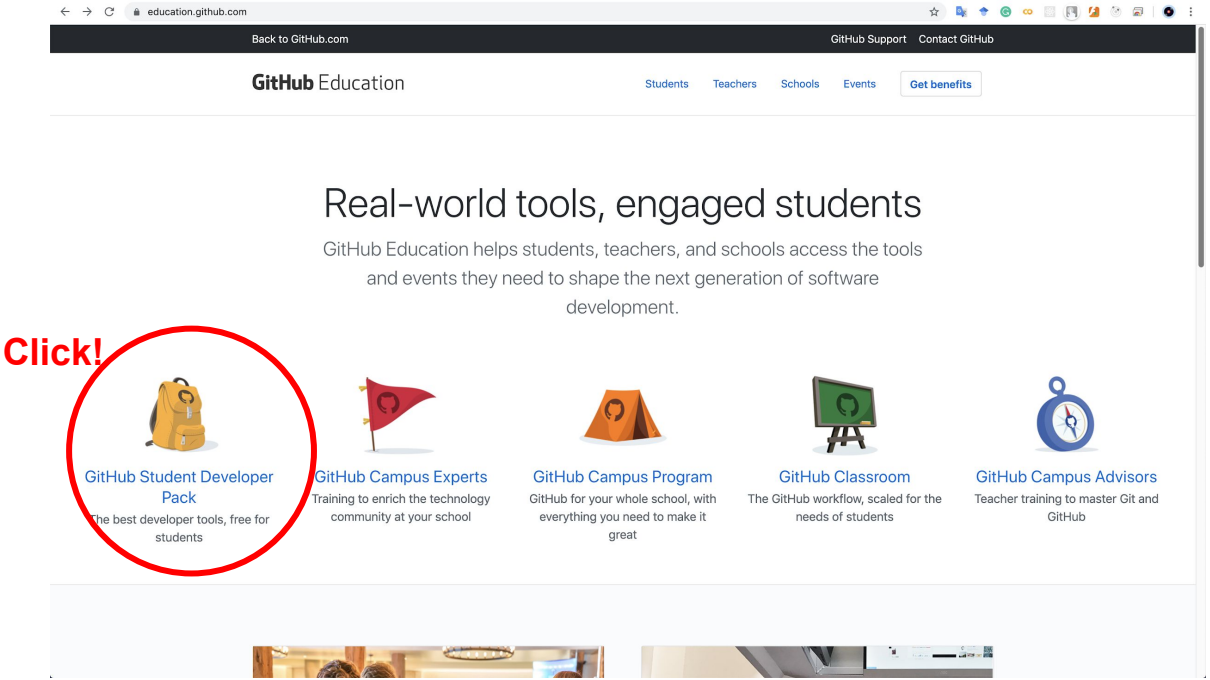
↓↓↓ Submit your codes ↓↓↓

https://forms.gle/feV4iAHUQdGj5BVN8

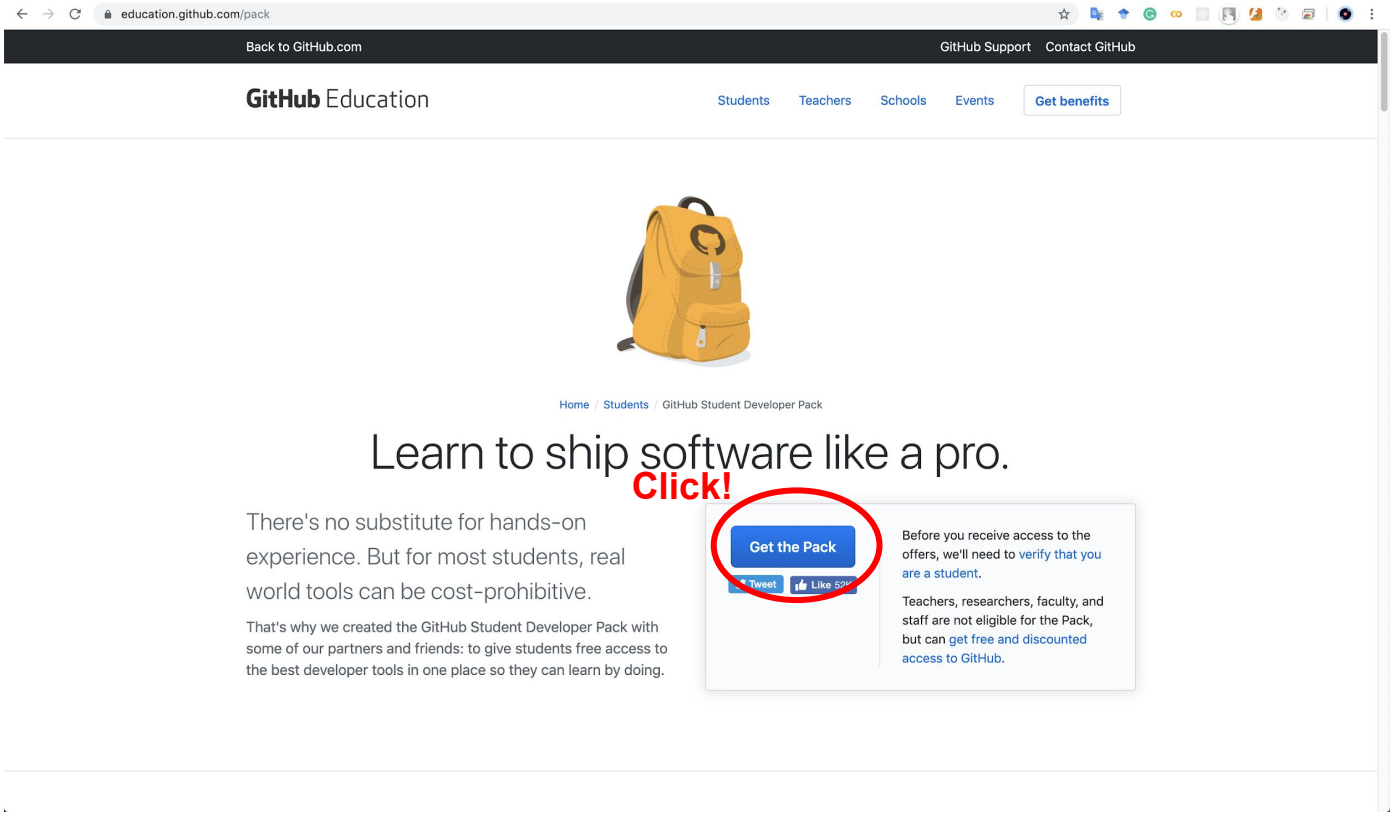
How to Setup HW1
& Git Basic

Git Student Developer Pack

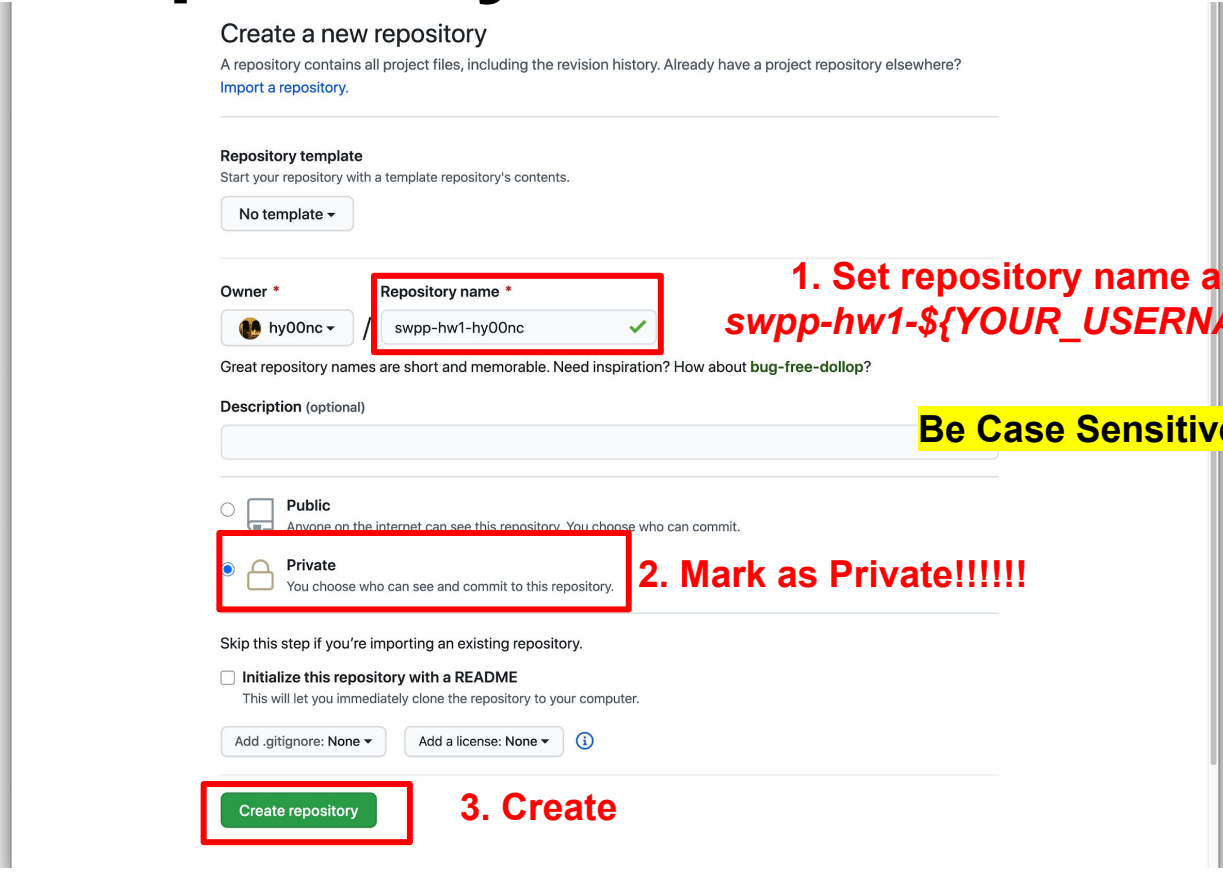
- First, sign up to github and goto https://education.github.com/



Git Student Developer Pack



Create Repository



Create Repository from skeleton

- First, clone from swppfall2022 repo
  - git clone https://github.com/swsnu/swppfall2022.git

```
(base) ~ ❯ cp -r swppfall2020/hw1 swpp-hw1-hy00nc
(base) ~ ❯ cd swpp-hw1-hy00nc
(base) swpp-hw1-hy00nc ❯ ls -la
total 88
drwxr-xr-x   10 haeyoon  staff    320  8 25 17:13 .
drwxr-xr-x+ 140 haeyoon  staff   4480  8 25 17:14 ..
-rwxr-xr-x    1 haeyoon  staff   2833  8 25 17:13 .gitignore
-rwxr-xr-x    1 haeyoon  staff  11358  8 25 17:13 LICENSE.txt
-rwxr-xr-x    1 haeyoon  staff    196  8 25 17:13 NOTICE.txt
-rwxr-xr-x    1 haeyoon  staff   9838  8 25 17:13 README.md
drwxr-xr-x   20 haeyoon  staff    640  8 25 17:13 babydata
-rwxr-xr-x    1 haeyoon  staff   3867  8 25 17:13 babyname_parser.py
-rw-r--r--    1 haeyoon  staff   1760  8 25 17:13 fetch.py
-rwxr-xr-x    1 haeyoon  staff   3713  8 25 17:13 run.py
(base) swpp-hw1-hy00nc ❯ git init
Initialized empty Git repository in /Users/haeyoon/swpp-hw1-hy00nc/.git/
(base) swpp-hw1-hy00nc ❯ git remote add origin https://github.com/hy00nc/swpp-hw1-hy00nc.git
(base) swpp-hw1-hy00nc ❯ git add .
```

Create Repository from skeleton

```
(base) swpp-hw1-hy00nc ❯ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitignore
        new file:   LICENSE.txt
        new file:   NOTICE.txt
        new file:   README.md
        new file:   babydata/2001.html
        new file:   babydata/2002.html
        new file:   babydata/2003.html
        new file:   babydata/2004.html
        new file:   babydata/2005.html
```

Create Repository from skeleton

```
(base) swpp-hw1-hy00nc ❯ git commit -m "initial commit"
[master (root-commit) 4798deb] initial commit
 25 files changed, 55897 insertions(+)
 create mode 100755 .gitignore
 create mode 100755 LICENSE.txt
 create mode 100755 NOTICE.txt
 create mode 100755 README.md
 create mode 100644 babydata/2001.html
 create mode 100644 babydata/2002.html
 create mode 100644 babydata/2003.html
 create mode 100644 babydata/2004.html
 create mode 100644 babydata/2005.html
 create mode 100644 babydata/2006.html
```

Create Repository from skeleton

1. `git branch -M main`
2. `git push -u origin main`

```
(base) swpp-hw1-hy00nc ❯ git push -u origin main
Counting objects: 28, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (28/28), done.
Writing objects: 100% (28/28), 217.00 KiB | 3.74 MiB/s, done.
Total 28 (delta 17), reused 0 (delta 0)
remote: Resolving deltas: 100% (17/17), done.
To https://github.com/hy00nc/swpp-hw1-hy00nc.git
```

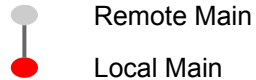
Push Your Work to Remote (status)

- When you modify some files in your git repository, you can check current statue by
- `$ git status`

Message may differ depending on Git version

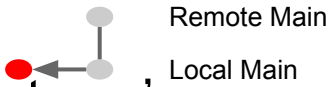
```
# On branch main
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#   new file:   helloworld.py
#
```

Push Your Work to Remote (add)



- You can add a file to track
- Track is like a shopping cart, where you put your stuff that you want to buy
- `$ git add ${target_files_to_track}`

Push Your Work to Remote (commit)



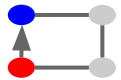
- Commit is like “wrapping box”, in which there are stuff that you’ve added to the track
- When you commit, you can tag “message” to identify stuff in certain wrapped box (commit)

※ If you want to edit the current commit instead of generating a new commit, run `git commit --amend`.

※ Recommendation : Generate commits **frequently, in small steps!**

Push Your Work to Remote (push)

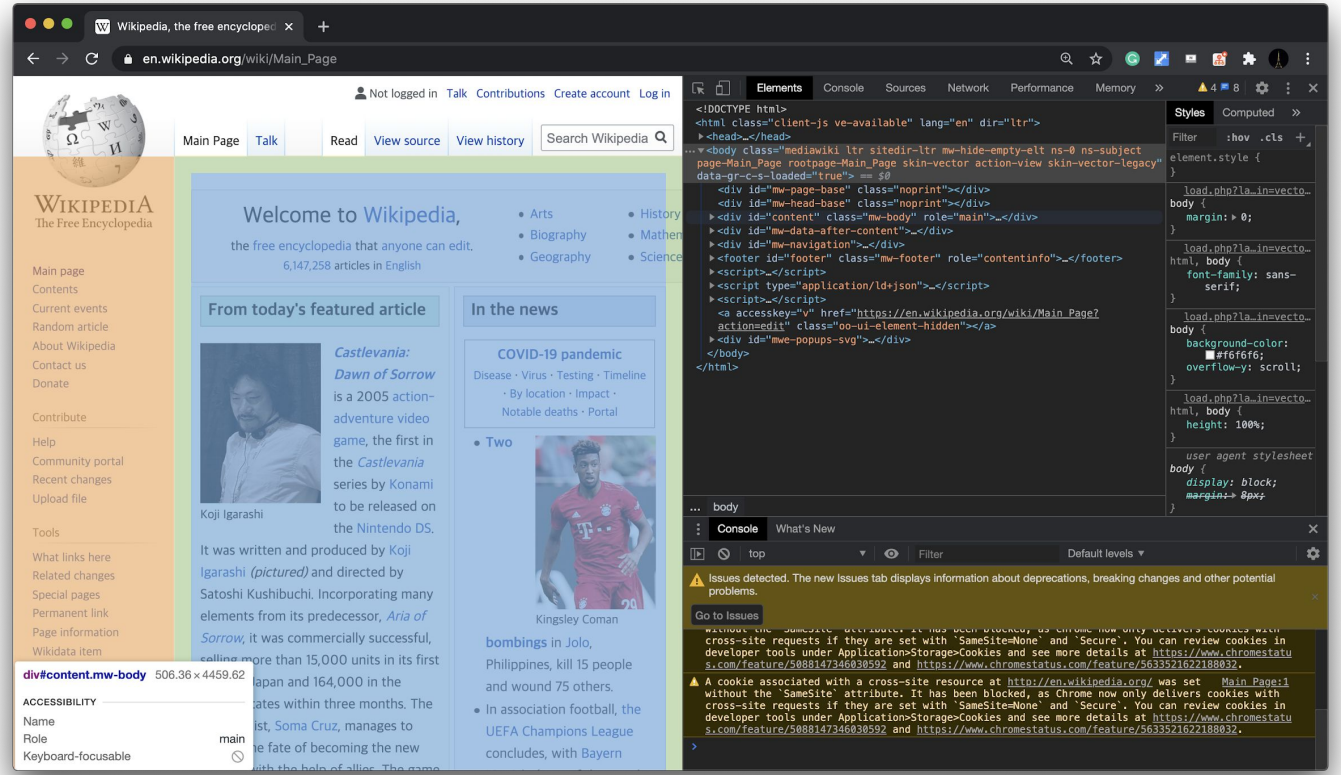
- After wrapping box, we must ship it!
- It synchronizes between your local repository and remote repository
- `$ git push origin main`



Bonus: Revisiting Python - Web Crawling

- Python provides many useful libraries
- Let’s try one of many libraries called **`urllib`**
- **`urllib`**
  - Provides functions and classes which help in opening URLs (mostly HTTP)
  - Can be used to fetch the source of the web page (html) by sending a request

Web pages can be inspected (1)

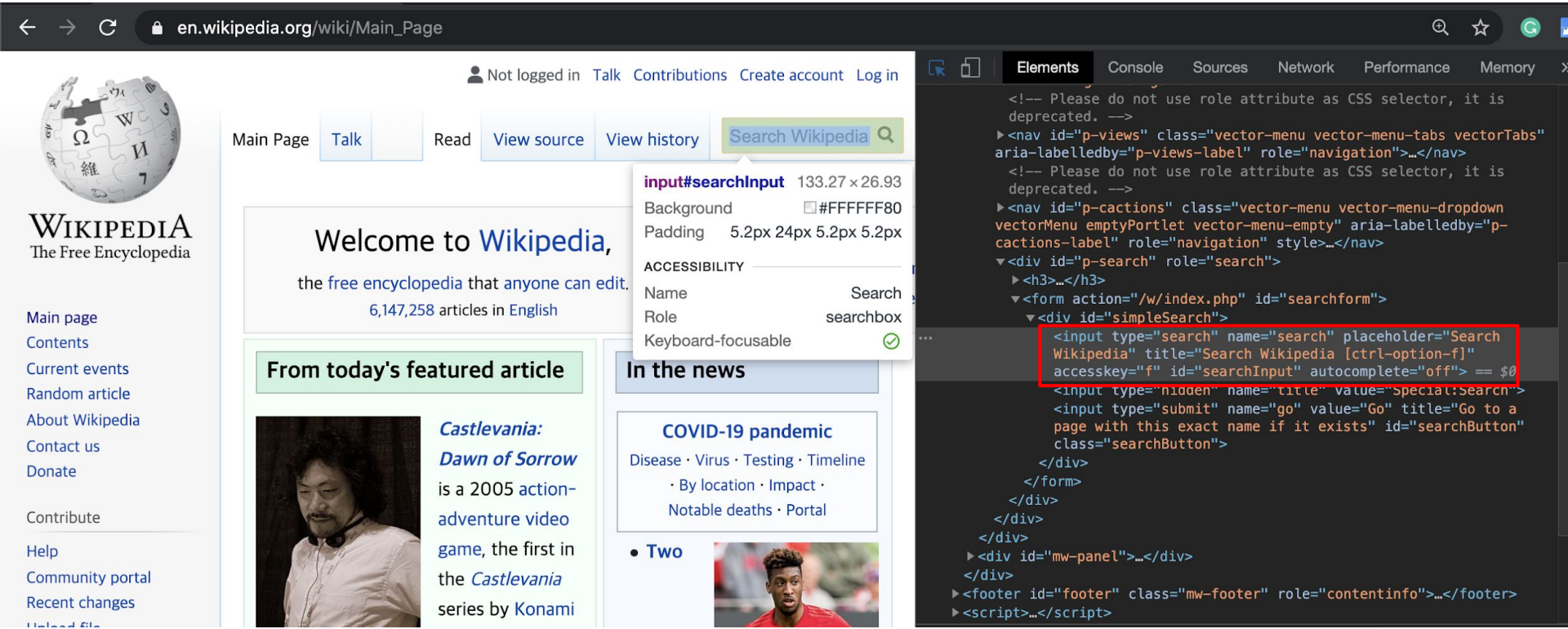


How to fetch a page using urllib?

```
#!/usr/bin/python

import urllib
import urllib.request

response = urllib.request.urlopen(
            "https://en.wikipedia.org/wiki" # open the web page by url
           )
text = response.read().decode() # decode the response so that it is human-readable

print(text)
```

Note that we do not give any data when requesting the page (GET)

Web pages can be inspected (2)



How to fetch a page with specific input?

```
import urllib
import urllib.request

response = urllib.request.urlopen(
            "https://en.wikipedia.org/wiki?search=Turing", # open the web page with search query
           )
```

You fetch a page by sending a data with a request (GET)

```
/* Send POST requests
response = urllib.request.urlopen(
            "https://en.wikipedia.org/wiki", # open the web page by url
            data = “search=Turing”.encode()            # string should be encoded to send
           )
*/
```

You fetch a page by sending a data with a request (POST)

```
text = response.read().decode() # decode the response so that it is human-readable
```

Alan Turing - Wikipedia

en.wikipedia.org/wiki/Alan_Turing

Not logged in Talk Contributions Create account Log in

Article Talk Read View source View history Search Wikipedia

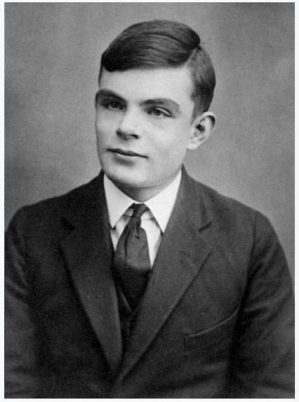
Alan Turing

From Wikipedia, the free encyclopedia
(Redirected from Turing)

"Turing" redirects here. For other uses, see Turing (disambiguation).

Alan Mathison Turing OBE FRS (/ˈtjʊərɪŋ/; 23 June 1912 - 7 June 1954) was an English mathematician, computer scientist, logician, cryptanalyst, philosopher, and theoretical biologist.[6][7] Turing was highly influential in the development of theoretical computer science, providing a formalisation of the concepts of algorithm and computation with the Turing machine, which can be considered a model of a general-purpose computer.[8][9][10] Turing is widely considered to be the father of theoretical computer science and artificial intelligence.[11] Despite these

Alan Turing
OBE FRS



Turing c. 1928 at age 16

Born	Alan Mathison Turing 23 June 1912 Maida Vale, London, England
Died	7 June 1954 (aged 41)

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

Alan Turing - Wikipedia

파일 /Users/haeyoon/Desktop/Experiment...

This is a good article. Click here for more information.
Page semi-protected

Alan Turing

From Wikipedia, the free encyclopedia
(Redirected from Turing)
Jump to navigation Jump to search

"Turing" redirects here. For other uses, see Turing (disambiguation).

Alan Turing
OBE FRS

Turing c. 1928 at age 16

Born	Alan Mathison Turing 23 June 1912 Maida Vale, London, England
Died	7 June 1954 (aged 41) Wilmslow, Cheshire, England
Cause of death	Suicide (disputed) by cyanide poisoning
Resting place	Ashes scattered in gardens of Woking Crematorium
Nationality	English
Education	Sherborne School
Alma mater	• University of Cambridge (BA, MA) • Princeton University (PhD)

- Cryptanalysis of the Enigma
- Turing's proof
- Turing machine
- Turing test

Don’t forget to watch

https://github.com/swsnu/swppfall2022

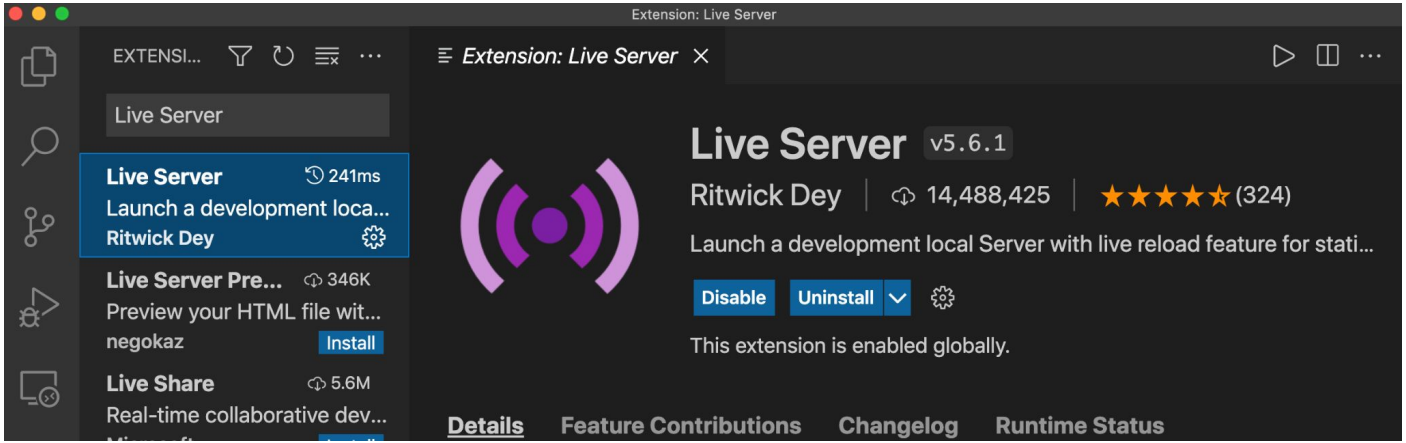
Reminder

- Team formation due: 9/13 6pm → Final announcement: 9/14
  - Send your team's info to swpp.22.staff@spl.snu.ac.kr
  - Add [SWPP] header at the mail title
  - List all team member's *name, student id, github id*
  - For those who does not form team until team formation due, TAs may form arbitrary teams among them
- Team project proposal due: 9/28 6pm
- HW2 due: 9/19 6pm
  - Add `swpp-tas` as collaborator
  - Push your codes

Tips for HW2

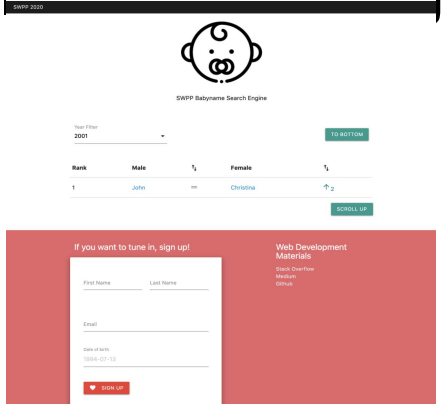
Liveserver on VSCode (1/2)

- For HW2, you should run an HTTP server.
- One easy way to run an HTTP server is to open with VSCode extension “Live Server”.

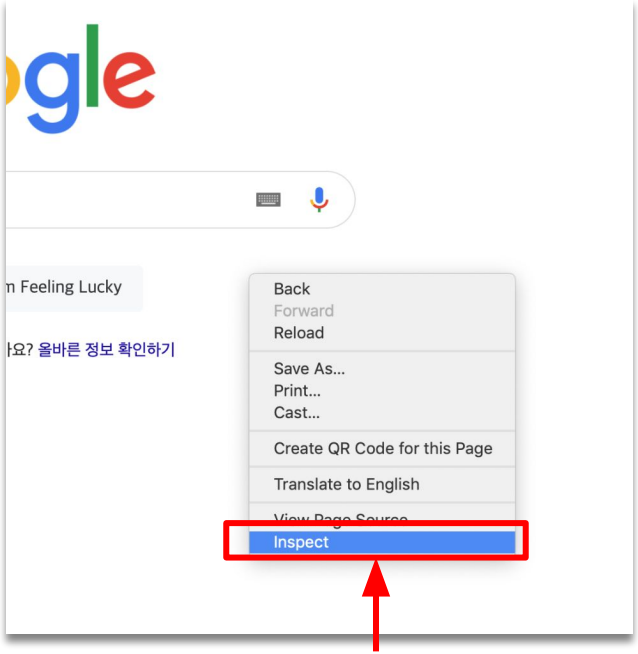


Liveserver on VSCode (2/2)

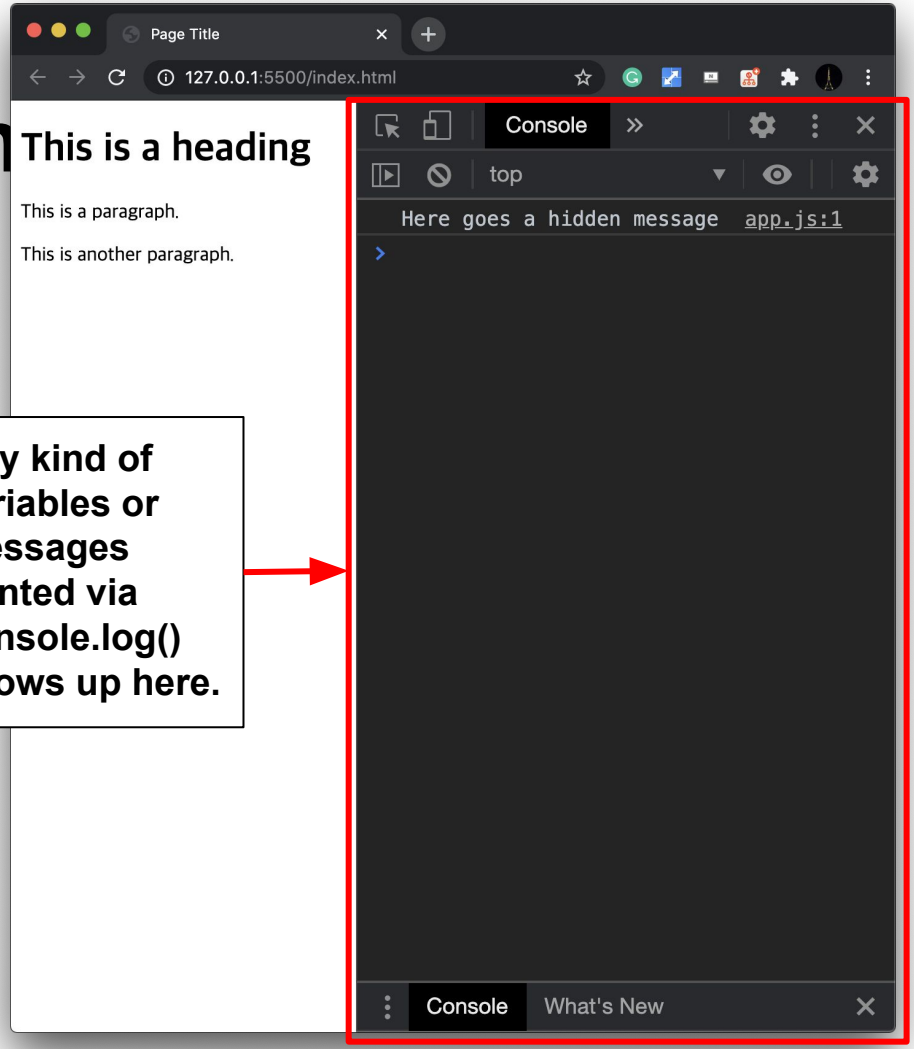
1. Go to VSCode extension tab (Code > Preferences > Extensions) and look for “Live Server”.
2. Install Liveserver.
3. Open the root folder of your HW2 directory on VSCode.
4. You will find a “Go Live” button at the status bar. Click the button and your default browser will open up. The page will look like below.



Debug with chrome in



Click Inspect



Any kind of variables or messages printed via console.log() shows up here.