

HTML

Front end
HTML structure CSS style

JS
Interactivity

Backend
Lang Python framework Django

Database SQLite

FORMAT →

<DOCTYPE html> → signal to use standard HTML

<html>

<head> → describe main item on page

<meta charset = "utf-8" >

shows at tab <title> This is title! </title> → required!!

</head>

<body>

<!-- Anything in is comment --> → or write text
ctrl + /

Big heading <h1> This is </h1> → h₁ <p> </p> → para tags in body
on page

</body>

h₂ ↓ size decreases
h₃

</html>

Tags in body →

- ① `<h1></h1>` ② `<p></p>` ③ ``
Heading para bold
 \downarrow new
- ④ `<i></i>` $\xrightarrow{\text{new}}$ ``
italics
- ⑤ `<header></header>` → company logos, tagline, header info
- ⑥ `<nav></nav>` → links to diff website
- ⑦ `<section>` $\xrightarrow{\text{header}}$
`<article></article>` $\xrightarrow{\text{body}}$
- ⑧ `<aside></aside>` → info related to main topic
- ⑨ `<footer></footer>` $\xrightarrow{\text{links}}$
- ⑩ ~~`<div>`~~
`<div class="groupone">` $\xrightarrow{\text{new}}$ $\xrightarrow{\text{style}}$ $\xrightarrow{\text{font}}$
`<h1></h1>` $\xrightarrow{\text{style}}$ $\xrightarrow{\text{font}}$
`<p> I want to be, styled</p>` $\xrightarrow{\text{style}}$ $\xrightarrow{\text{font}}$
`</div>` apply effects for only 'be'

→ list < ordered
unordered

<body>

<h1> Lists </h1>

 first

 second

 first

- second

 first

 first subpoint

{

ordered

unordered

Lists

1. First

2. Second

• First

• second

1. First = bi

• First subpoint

#

freeze up copy
list tag taken

<" "=bi>

 1. noH92

 2. noH92

nested list<noH92>

→ Attributes → allow to add more info to html tags
- adding link, referencing image

① Images →
self ending tag

<image src = "wl.jpg" width = "400" height = "235"
alt = "This is the picture">

If image in same directory

shows if image is missing

Only ↓

② linking page →

 click me

* to open in new tab
the link

 click

If picture show

[]

If doesn't

☒ This is the picture

click me → will direct
to page

③ Linking on same page

< href on same page

< section >

< a href = "#section1" > section 1

< a href = "#top" > back to top

< h1 id = "top" > links to same page </h1>

< section >

< ul >

< li > < a href = "#section1" > section1

</section>

< section id = "section1" >

< h3 > section 1 < h3 >

< p > --- </p>

</section>

→ TABLES →

< body >

< table border = "1" >

space b/w inner & outer margin

<thead >

< th > Number </th >

< th > colour </th >

table Heads

creating
row of headings
for each column

<thead >

< td > 150 </td >

< td > yellow </td >

< /td >

< /td >

< /td >

</thead >

Number	Colour
150	yellow
200	Red

FORMS →

<body>

<form>

<h1> Log in </h1>

<h2> Please input email & pass </h2>

<input type = "text" name = " " value = " " >

</form>

</body>

type of input

name of input

value of input

(assigns value to this name)

(to be prefilled)

Submit → <input type = "submit" value = "submit" >

text → <input type = "text" value = " " >

password → <input type = "password" value = " " >

email → <input type = "email" value = " " >

date → <input type = "date" value = " " >

checkbox → <input type = "checkbox" value = " " >

radio → <input type = "radio" value = " " >

file → <input type = "file" value = " " >

Action →

<form action = "url" method = "get" >

destination page

on pressing

submit leads to

url

label →

① <label>

Individual

Enter text:

<input type = "text" name = " " value = " " >

</label>

id → uniquely identifies
the type.

②

for all

<label for = "user" > Enter Input: </label>

→ Enter text: _____

<input id = "user" type = " " >

* Instead of value, use placeholder.

→ need to
erase

<input type = "text" name = " " value = "Help" > → Help
placeholder = "Help" > Help
→ type over

* If you want to make a field mandatory to be filled before
submit, use required.

<input type = "password" name = " " value = "password" >
required

if submit
without
field → []
Please
fill this
field
popup

→ Input Methods

① Radio button ⚡

If multiple radio types
are named same, only one of those
can be selected.

```
{ <input type="radio" name="loc" value="inside">
```

```
<h2> How service? </h2>
```

Shows in list as

name
name = value

```
<select> name="stars" >
```

```
<option value="Great">3</option>
```

```
<option value="Bad">2</option>
```

```
</select> <br/>
```

Dropbox
(multiple)

```
<h2> any other feedback (rect box to write) </h2>
```

```
<textarea name="mytext" rows="8" cols="80"></textarea>
```

CSS

linking to HTML →

(In Head)

<Head>

<link rel = "stylesheet" href = "Parti-master.css" >

</head>

→ comment → /* * / → Ctrl + / ↪ repeat = 220

→ General form → selected tag { property: value; }

eg → h1 { color: blue; } changes colour of h1

↓ other way
color:rgb(8,84,1);
color:#0895a8;
color:rgba(8,84,1,0.5);

→ div, span, background → can also pass a pic by url(); instead of background: gray; changes background of body to gray

div { div { background: red; border-color: blue; border-width: 1px; border-style: dotted; } also instead of px, can choose thin, medium, thick }

Span { background: white; border-color: black; } for inside span

* to repeat background of body (image repeats on big page)

body { background: url(---); }

background-repeat: repeat;

not repeat

if don't want to repeat

<"223.html#H109" = part of text & style = see style>

→ Selectors →

id → targets single element

classes → targets groups of elements.

text-decoration
→ line-through
text-decoration

① html css
<div class="name"> ← → name { color: blue; }
</div> dot

② <p id="name"> </p> ← → #name { color: green; }
name

③ ~~color: black;~~ → will colour black all text
{ } except id

④ h3 + ul { } → any ul immediately after h3
border: 4px dotted purple; } can write all border
{} style together

⑤ li a { } → colour: red
{} anchor tag

li a[href="uv.php.or"] { } → specifically
{} colour: blue;
{} border: 5px solid orange;

→ Specificity →

<li class="n" ①

→ if

<li class="n" id="e" ②

.n {

color: blue } → ①, ②

* id has ↑ priority.

color: blue }

color: red }

bold text & coloured

red.

④ FONTS → cssfontstack.com → check availability of font

fonts.google.com → download fonts

① font-family: "Arial"; → font you know
font-family: cursive; → available in most

② font-size: 20px; → size of letters

Now, if we want size of 2nd para to be twice size of 1st para fonts-

body {

font-size: 20px;

}

two {

font-size: 2.0em;

}

1em = 20px

work acc
defined
to first
px

<body>

<p> - - - - - </p>

<p id="two"> - - - - </p>

</body>

③ font-style: italic; → style

④ font-weight: bold; → structure

⑤ text-align: center; → aligns center, left, right

BOX MODEL



can style left right
bottom top. is >
top = width is >

top {

border: 4px solid blue;

width: 28%; → border's length with respect to page size

text-align: center; → text is aligned at centre of border box

margin: auto; → sets border in centre of page

{ margin: 10px 40px 200px 7px;
top right bot left

bottom {

border: 4px solid red;

width: 50%;

padding: 200px;



→ 200px is the space b/w Content & border

</body>

Bootstrap

getbootstrap.com (site)

* to get bootstrap classes → getbootstrap.com / get started / copy download / CDN link.

- container class → centers everything (shifts right creates a 10px margin)

∴ various classes for various requirement

① button class (check various on [getbootstrap.com](#))

<button class=" " type="button" name="">
type of look needed (e.g. btn btn-success)
= green button

can add more class as per requirement

(i) like ~~btn~~ class="btn btn-success btn-lg"
green large

(ii) class " " disabled="disabled" ty
can't click the button

② jumbotron class → design a page with heading + para + button

<div class="container">

<div class="jumbotron" >

<h1> </h1>

<p> </p>

<p> </p>

</div> </div>

} From page link

③ form class →

① div <class = "form-group" >

provides spaces
b/w different inputs

② <input class = "form-control" >

provides design
• font highlighted
• box for email is stretched

puts email

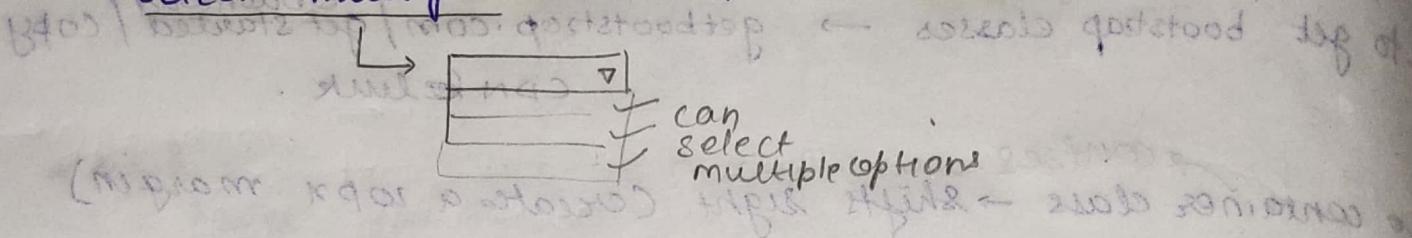
to email

③ <small id = " " class = "form-text form-muted" > </small>

email
we never share your info [para like muted text]

④ for multiple selection drop box →

<select multiple class = "form-control" ...>



⑤ text area →

* form-control also gives ability to click and stretch the area box

⑥ File upload input →

<input type="file" class = "form-control-file" id="" ...>

"choose file" NO FILE CHOSEN

click gets to open a folder

(selection - backend)

⑦ Radio button →

<legend> --- </legend>

& class = "form-check" used for radio button

* to disable any input just write <input type="radio" disabled>

Navbars → navigation bar even if scroll down, navbar is visible at top

① class = "navbar navbar-default navbar-fixed-top" in a grey tile

file = black

② class = "navbar-header"

→ main

"dropdown-menu" = sub > wh

Right click - inspect JAVASCRIPT ← use incognito tab
- console

- ① `alert("Hello World")` → string
- ② comment → // text
- ③ All no. are treated same
- ④ undefined, null → variable created but not defined
- ⑤ to clear console → clear()
- ⑥ can do calculation → 2+2, 9/2 etc → will output
• Power → $2^{**}4 = (2^4)$
- ⑦ concatenate string → "Django" + " here" → O/P = Djangohere
- ⑧ length of string → "Django".length → O/P = 6
- ⑨ newline → \n
- ⑩ \t → tab space
- ⑪ "Hello" [0] → O/P = H
- ⑫ Variables → var varName = value;
eg var account = 100; I/P
undefined
account
100 O/P.
- Eg - var greeting = "welcome back: ";
var name = "jose";
alert(greeting + name) → O/P = welcome back: jones
- ⑯ to see output in console → console.log("text") → text
- ⑰ prompt("Enter something") → O/P = Hello
- var age = prompt("Age")
undefined
age 90

link `js` and `html` → `<script src = " " ></script>`

→ Operators →

$3 > 2$ iff $4 \neq 4$ ~~$2 \neq 2$~~ "text" == "text" "2" == 2
true o/p true true true true
" " " " " " type check "2" == "2"
" " " " " " " " " " false

$= =$ → qs changes into simple datatype to check equality

$= = =$ → Check on datatype too

$5 != "5"$ → $5 != "5"$
false true false

• AND → $1 == 1 \& 2 == 2 \& 3 == 3 \rightarrow \text{true}$

• OR → $1 == 2 \mid 1 == 1 \rightarrow \text{true}$

• NOR → $!(1 == 1) \rightarrow \text{false}$ (opposite of and)

→ if else

`if () {
} else {
} else if () {
} else {
}`

→ while

a/ while condition

→ break

→ for (initialization; cond; incram)

`for (; ;) {
}`

leads out of while loop

→ functions →

`func_name (Parameter1, para2)`

`{
}`

→ In console, can call fn o/p - fn def.

console

function name → fn full

- ~~hello(name)~~ • function hello (name = "Frank") → default parameter
? console.log ("Hello" + name), in case no para is passed
- If on console →
 - ① hello ('jose'); O/P → Hello jose
 - ② hello (); O/P → Hello Frank
- function formal (name = "Sam", title = "Sir") ?
return title + " " + name;
- global variable assigned value doesn't change if assigned in a fn.
- function-variable inside fn remains to the scope of fn.

Arrays →

- var varName = ["1", "2", "3"].
- * string is immutable (can't be changed by changing index)
- arrayName.pop() → pops last element
eg. var lastElement = arr.pop();
 - arr.push ("item") → pushes "item" in the array
 - arr.length → gives length [1, 2, 3] [4, 5, 6] [7, 8, 9]
- 2D array → var matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
- for printing array elements → ① for (i=0, i < arr.length, i++) ② console.log(arr);
- ③ for (letter of arr) { console.log(letter); }
- ④ arr.splice(index, count, item) → arr.splice(1, 2, "new") → [1, "new", 3]
- index of element how many no. to delete starting from index
- ⑤ arr.forEach(function (fnName)) → arr.forEach(function (fnName)) over elements are passed as parameters to the function
- ⑥ arr.indexOf(name) → finds index of name

Objects

- Syn → { key1: "value one", key2: "value two", ... }
- ↓
values are accessed through keys

e.g. → var carInfo = { make: "Toyota", year: 1990 }

- To access any element → carInfo["make"] → o/p → "Toyota"
- when calling,
have to address
as string
- * When you make key don't have to address as string
But

- can nest objects

var my = { a: "Hello", b: [1, 2, 3], c: { inside: ["a", "b"] } };

→ my['b'] → o/p → [1, 2, 3]

→ my['b'][2] → o/p → 3

→ my → o/p → Object { a: "Hello", b: Array[3], c: Object }

→ my['c']['inside'][1] → o/p → 'b'

- can change values →

my['a'] = "Bye".

→ CarInfo['year'] + 10; → i.e. → i.e. of ①

object

- console.dir(carInfo) → displays objects

- for (K in carInfo) → object to which (K) is of type
name → console.log(K) → keys of K { make, year }
{ console.log(carInfo[K]) } → values in each key.

```
• var simple = {
    prop: "Hello",
    myMethod: function() {
        keyword
        console.log ("Hey" + this.prop)
    }
}

→ simple → o/p → object {prop: Hello}
use this → console.dir(simple) → o/p → object
→ simple.myMethod() → o/p → "Hey Hello"
```

Document Object Model

- In console,
→ document → HTML used in page
console.dir(document) → HTML properties
- Imp doc attributes →
 - document.URL → actual URL of website
 - document.body → inside body
 - document.head → return what is inside head
 - document.links → all the links on page
- ★ Parameters in injected commands →
 - "ClassName()" all items belonging to class
 - "TagName()"
 - "#id", ".class", "li" → returns 1st object matching the selector
- By CSS style → document.querySelector("selector") → returns 1st object matching the selector
- → document.querySelectorAll("selector") → returns all objects

Grabbing & changing style by JS. →

- var myheader = document.querySelector("h1");
myheader.style.color = "red" → change h1 to red colour
↳ edits style

• ★ Random colour changer →

```
→ function getRandomColor() {  
    var letters = "0123456789ABCDEF";  
    var color = "#";  
    for (var i=0; i<6; i++) {  
        color += letters[Math.floor(Math.random()*16)];  
    }  
    → return color;  
→ function changeHeaderColor() {  
    colorInput = getRandomColor();  
    header.style.color = colorInput;  
    ↳
```

→ setInterval ("changeColorHeader()", 500)

• myVariable.textContent → return text

→ myVariable.innerHTML → actual HTML

changes
color every
500 milisec

→ myVariable.getAttribute() → original attribute

→ myVariable.setAttribute() → sets an attribute

e.g. → var specialA = special.querySelector("a")
specialA.getAttribute("href") → get the anchor tag
specialA.setAttribute("href", "https://www.amazon.com") → only link

sets this

← adds this attribute

Events → trigger only on clicks or else.

Syn → myVariable.addEventListener(event, func);
action to trigger
what should happen

← action to trigger what should happen

(event) = function

J. Query

- to check if gg is loaded
inspect → console → `$()` displays function → loaded.
- `$(“h1”)` o/p all h1 items
- `var x = $('h1')` ~~property~~ two parameters → value
- multiple changes by object →
`var changevariable = {‘color’: ‘white’, ‘background-color’: ‘red’};`
executed when entered → `x.css(var_name)`
- Grabbing particular index →
`var listItems = $('li')`
listItems → o/p li, li, li, li #special
only 1st li changes → `listItem.eq(0).css('color', 'orange')`
last item changes → `listItem.eq(-1).css("color", "orange")`
- Grabbing text content → `<h1> Hello everyone</h1>`
→ `$('h1').text()` → o/p "Hello everyone"
→ `$('h1').text("Go Away")` → o/p Go Away
→ `($('h1').html())` → o/p Go Away
→ `($('h1').html(" New "))` → o/p New

- Attribu
→ `$(input)`
- addin
→ `$(C)`
- Even
→ `$(C)`
- \$(C)
- \$(C)

→ key
→

→ Or
\$

• Attributes → (another attribute) changes into ('type') submit
→ $\$(\text{input}) . eq(1) . attr(\text{'type'}, \text{'checkbox'})$ → after

→ $\$(\text{input}) . eq(0) . val(\text{'new'})$ → O/p new
 n now

• adding class →

→ $\$(\text{h1}) . addClass(\text{'turnRed'})$ → O/p h1 turns Red
→ $\$(\text{h1}) . removeClass(\text{'turnRed'})$ → O/p h1
→ $\$(\text{h1}) . toggleClass(\text{'turnBlue'})$ → Run 1 turns Blue
Run 2 turns Black
Run 3 turns Black

• Events →

→ $\$(\text{h1}) . click(function() \{ \})$
→ $\$(\text{li}) . click(function() \{ \})$ → works for all li
→ $\$(\text{h1}) . click(function() \{ \})$
 $\quad \downarrow$
 $\quad \$\text{(this)} . text(\text{"I was changed!"})$
 \quad refer to h1

• key Press →

→ $\$(\text{input}) . eq(0) . keyPress(function() \{ \})$
 $\quad \$\text{('h3') . toggleClass('turnBlue')} ;$ → as you type
 letter h3
 turns Blue then
 Black then
 Blue

→ $\$(\text{input}) . eq(0) . keyPress(function(event) \{ \})$
 $\quad \text{If } (\text{event} . which == 13)$ → which denotes
 ~~$\quad \$\text{('h3') . toggleClass('turnBlue')}$~~ → ASCII value
 ~~$\quad \text{console.log(h3)}$~~ → 13 → enter
 pressed so
 toggle happens

→ On →

→ $\$(\text{h1}) . on(\text{'dblclick'}, function() \{ \})$ → on double click
 $\quad \$\text{(this)} . toggleClass(\text{'turnBlue'})$ → turns Blue

→ `$(‘input’).eq(1).on(‘click’, function() {
 $(‘.container’).fadeout(3000);
})`

on submitting takes
eq(1)
3000 milisec to
fade out every
that's in container
class

(‘button’).submit(function() {
 (‘cancel’).val(function() {
 (‘submit’).prop(function() {
 return false;
 })
 })
})