Week three notes

Week three topics

MVC

* HTML fundamentals-elements, attributes, doctype, accessibility, HTML history, HTML 5, audio, video, canvas
* Common elements-, attrs, forms, inputs, validation
* Css fundamentals-rule,property, cascade, inheritance, box model, positioning, media query, responsive design, external/internal/inline
* Selectors-\*,class, tag,ID, parent-child, ancestor, descendant,attr, combinators, pseudo-classes, pseudo-elements
* MVC concepts-model, view, controller
* Http request lifecycle, DNS, request, response
* Controller-actions, HTTP methods
* Model, DataAnnotations, viewmodel,
* View,-layout, strongly/weakly typed, viewdata/viewbag/tempdata, partial, section
* Razor-syntax, expression, block, @model, html helpers, tag helpers
* Model binding
* Routing-conventional, attribute, router parameter, query string
* Validation-server, client, CSFR, token
* Testing-mock, Moq
* Filters, middleware
* Entity framework code-first
* Dependency injection- singleton, scoped, transient, Fromservices

Hypertext markup language

* Describe structure, meaning, and textual content of a document/page on the web

Started out lack of conformity, etc

HTML4🡪XHTML(nobody used it)-------------------🡪HTML5(added better defined error handling behavior)

^^🡪make HTML a strict subset of XML ^^WHATWG ^^-->”living standard”

W3C ^^add a bunch of element attrs, audio, video, canvas

attribute

Tag name name value

<div class=”header”> (opening tag)

Element🡪 Text (context)

</div> (closingtag)

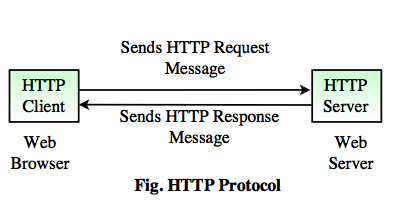
Check html basics.html in vs code inside trainer code for element, tag examples and definitions

Block element claims vertical space exclusively

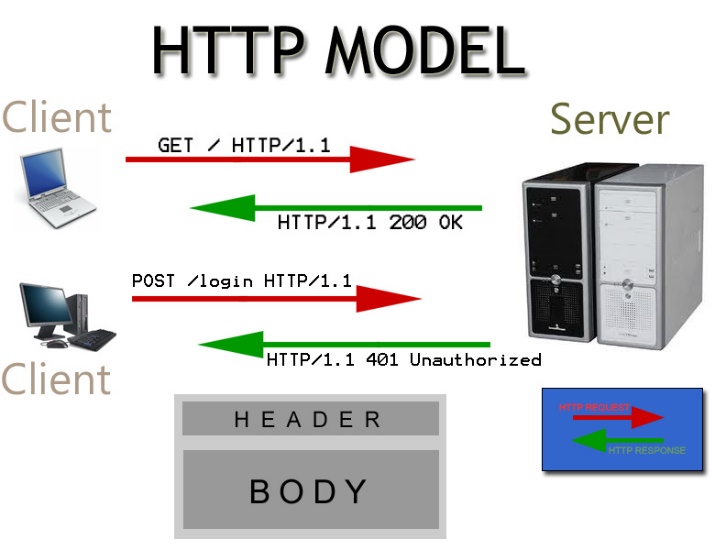
Inline element-don’t claim vertical space ex:ol,ul,li,p,img, stornge, em, u

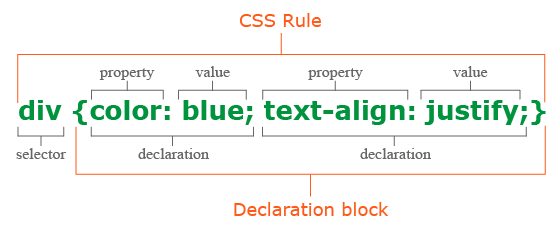
Div is the generic block element

Span is the generic inline



Browser google.com





Selectors

\*{ … } omni selector (all)

p { … } tag selector

.sidebar { … }class selector (many different things)

#login-form { …} id selector (one thing)

[attr=value] img[src=apple.jpg] exact

[attr~=value] if it contains value,not exact match

Combination

Concatenate🡪AND

,(comma)🡪OR

(Space)🡪ancestor descendant

> 🡪 parent child

~ 🡪any sibling (same parent)

+ 🡪 Immediate next sibling

LOOK UP THIS STUFF LATER

Pseudo-classes

(selector) : (pseudo-class)

--------------------------------------

Link (unvisited links)

Visited (visited link)

Focus (focused with keyboard mouse)

Valid form controls

Invalid based on validation attrs.

Hover

First-child

Nth-child(An+B)

Pseudo-element

:: (pseudo-element)

---------------------------

Before

After

Selection

Ways to apply css to file

Inline CSS (bad practice)

In .html

<p style=”color:blue”>

Internal CSS

In .html

In <style> (usually in head)

………

………

</style>

External

In .css file

Know code to include css file (<link rel…… )

cascade origins-

Author CSS (beats user-agent css in cascade)

User-agent CSS

Browser css

User css

{ !important (bad practice,gives yours priority)

Specifity rule:

1. !importance wins,
2. Inline wins
3. The most #id selector wins
4. The most class selectors wins
5. The most tag selectors wins
6. Source order

Know box model

inheritance

Margin is not inherited by default(like many things)

Color is inherited

“inherit” turns it on if not default

Know basic font size, font color properties, or at least be able to look up

Background-color

Font-family

Know sizes: px em %

Position

(default) static (inline/blockquote?)

Relative

Offset the element in those directions from the normal static position

absolute

offset element from the boundaries of its point

fixed

offset element from browser viewport

display

inline

block

none (removes the element entirely visibily)

.inline-list {

Display: inline

}

Visibility :hidden

Invisible, while taking up space

Media query

@

Device width, height, aspect ratio, touchscreen vs mouse pointer, print

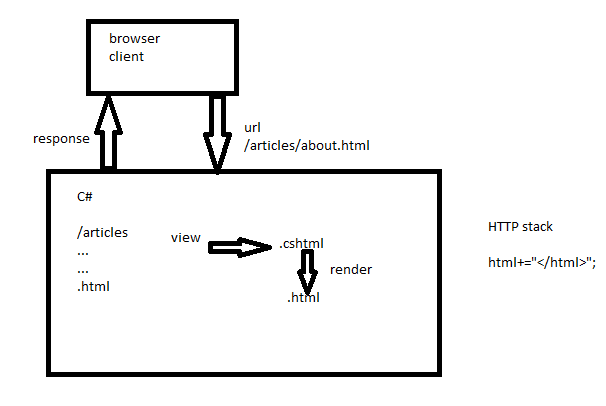
|  |  |
| --- | --- |
| Adaptive design | Responsive design (ex.Bootstrap.css) |
| Design for several screen sizes | All screen sizes at once |

Img srcset

Css reset

Normalize.css

ASP.NET



Controller

Receive and process the client’s request

Select a view to render and give it any data it needs (it’s model)

Writing any changes to db

View

Encapsulate one page and all its structure, content, + css/js as needed

Template for html +css+JS

Displays the model’s data

Model

The data

-MVC design pattern

Asp.net mvc

A framework in .net

For making web apps

With mvc design pattern

Check hello.mvc

Which controller should be instantiated and which action method should be called

Routing decides based on the requests url

When we make a class specifically to be the model for a view, we call it a view model

The purpose of a view model is to organize data in the way that one or more particular views needs. (probably different) from either (1) how C# thinks about the data in your business logic.

Check movie app

Two ways to connect using entity framework?

C# classes

-DbContext SQL

-classes inside the DbSets(empty classes)

Db first approach

Make sql db

“reverse engineering”

Scaffold-dbcontext

Sql is the source of truth

Code first

Migrations

C# classes are the source of truth

Ways to configure

* Conventions
* DataAnnotations
* Fluent API (OnModelCreating) (recommended)

Migration:a record of a reversible change to a DB schema

“Add-Migration” input command into VS package manager console

Default project should be data access project

Install-Package Microsoft.EntityFrameworkCore.SqlServer

Update-Database

Cross-side request forgery

How to prevent

* Add a random token to the form
* As a hidden field
* When we receive the form, we validate the token
* <form> tag helper auto adds the token
* Use [ValidateAntiForgeryToken] on action method(only action methods receving data, most httppost methods)

Recommend adding mvc controller with read/write actions

Ways to get data from the controller onto the page(view)…or from one page to the next

* The model
* Viewdata/Viewbag
  + ViewData

Dictionary<string,object>

Controller:

ViewData[“username”]=”Fred”;

View:

@ViewData[“username”]

(emptied between each request)

* ViewBag

“Dynamic” object

Controller:

ViewBag.Username=”fred”

View:

@ViewBag.Username

* TempData

Accessed like ViewData

Values stay set across HTTP Requests, until read

When you read a value, it marks that value to be deleted after the current request.

TempData.Peek(“username”)

Returns value without marking for deletion

TempData.Keep(“username”)

Unmarks from deletion

* Route parameter
  + (/Details/3)
* Query string (get forms)
  + /details/3?something=asdf
  + Int id^ ^string something
* Hidden fields (post forms)
* Database
  + Using partial row data, mark with bit column that tells if partial order

Partial view

@HTML.Partial (“(name)”, model)

[FromQuery] FromRoute FromBody ?

Testing

https://docs.microsoft.com/en-us/ef/core/miscellaneous/testing/in-memory

html, css, mvc quiz Monday, no coding assessment, qc Tuesday, project presentation