

ZPR PWr – Zintegrowany Program Rozwoju Politechniki Wrocławskiej

Aplikacje webowe na platformie .NET

W01 – Wprowadzenie

Syllabus

- Wstęp przeglądowny
 - HTML 5
 - CSS3
 - JavaScript
 - Przeglądarki i zgodność z HTML5, CSS3 i JS
 - Walidatory
 - URL, URI
 - Protokół HTTP – żądania GET i POST
 - Narzędzie konsolowe `curl`
 - Narzędzia przeglądarek
 - Wielowarstwowa architektura aplikacji
 - Skrypty/programy po stronie klienta i serwera
- HTML5
 - Struktura prostego dokumentu – znaczniki `<html>`, `<head>`, `<title>`, `<body>`, `<p>`
 - Walidatory HTML
 - Nagłówki: znaczniki `<h1>` do `<h6>`
 - Linkowanie: znacznik `<a>`
 - Obrazki: znacznik ``

Wstęp

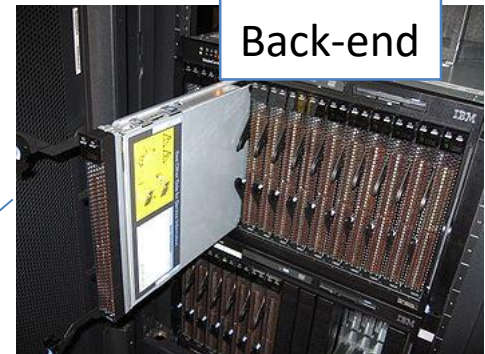
- Cechą programowania aplikacji **webowych** jest ich **przenośność**, szczególnie jeśli chodzi o **możliwość uruchamiania ich na wielu urządzeniach z dostępem do internetu**.
- Cechą **odróżniającą** aplikacji webowych od **stron** internetowych jest **interaktywność** oraz bardzo często **komunikacja** z serwerem w celu wykonania pewnej funkcjonalności
 - **Aplikacja webowa** może również **nie komunikować** się z serwerem w ogóle, albo cache'ować dane wcześniej pobrane.
- Aplikacja webowa dzieli się na dwie części
 - **Front-end**
 - Część aplikacji działająca po stronie **klienta**, najczęściej przeglądarki
 - **Back-end**
 - Część aplikacji działająca po stronie **serwera**

Klient
Front-end



Internet

Serwer
Back-end



[To zdjęcie](#), autor: Nieznany autor, licencja: [CC BY-SA](#)

Techniki **client-side/server-side**

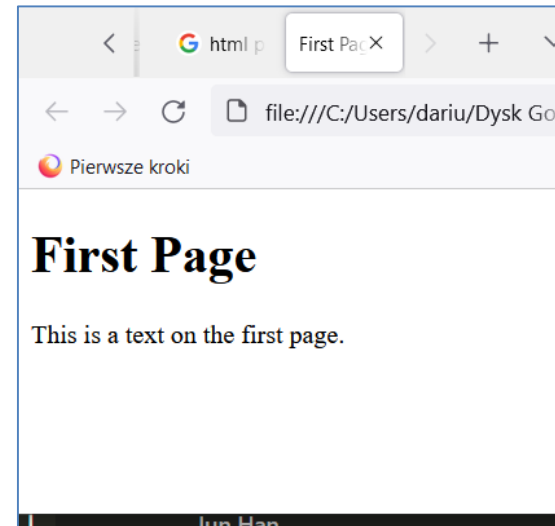
- Techniki programowania **po stronie klienta (client-side)** są wykorzystywane **do budowania stron** internetowych i aplikacji, które są uruchamiane na **kliencie**:
 - HTML5, CSS3, JavaScript
 - Ajax
 - SVG – do dwuwymiarowej grafiki
 - Frameworki JS, np. Bootstrap, jQuery itp. do obsługi interfejsu użytkownika.
 - Istnieje całe mnóstwo przydatnych frameworków i ciągle pojawiają się nowe, które stają się standardami.
 - i in.
- Techniki programowania **po stronie serwera (server-side)** - aplikacje, które **odpowiadają na żądania** przeglądarek (po stronie **klienta**).
 - Język PHP
 - Język Java
 - Framework Spring
 - inne
 - JavaScript: Node.js
 - C#: (ten kurs)
 - ASP .Net (ten kurs)
 - Python, Ruby i in.

- **HTML** (HyperText Markup Language) to ***język znaczników***. Powstał, by określić ***treść*** i ***strukturę*** stron (dokumentów) internetowych w przenośny sposób.

- **Prezentacja treści może zależeć** od urządzenia (komputer, smartfonach, tabletach, wielkie ekrany) i oprogramowania np. przeglądarki.
- Jednak większość oprogramowania (**przeglądarek**) **stara się trzymać zaleceń standardu**.

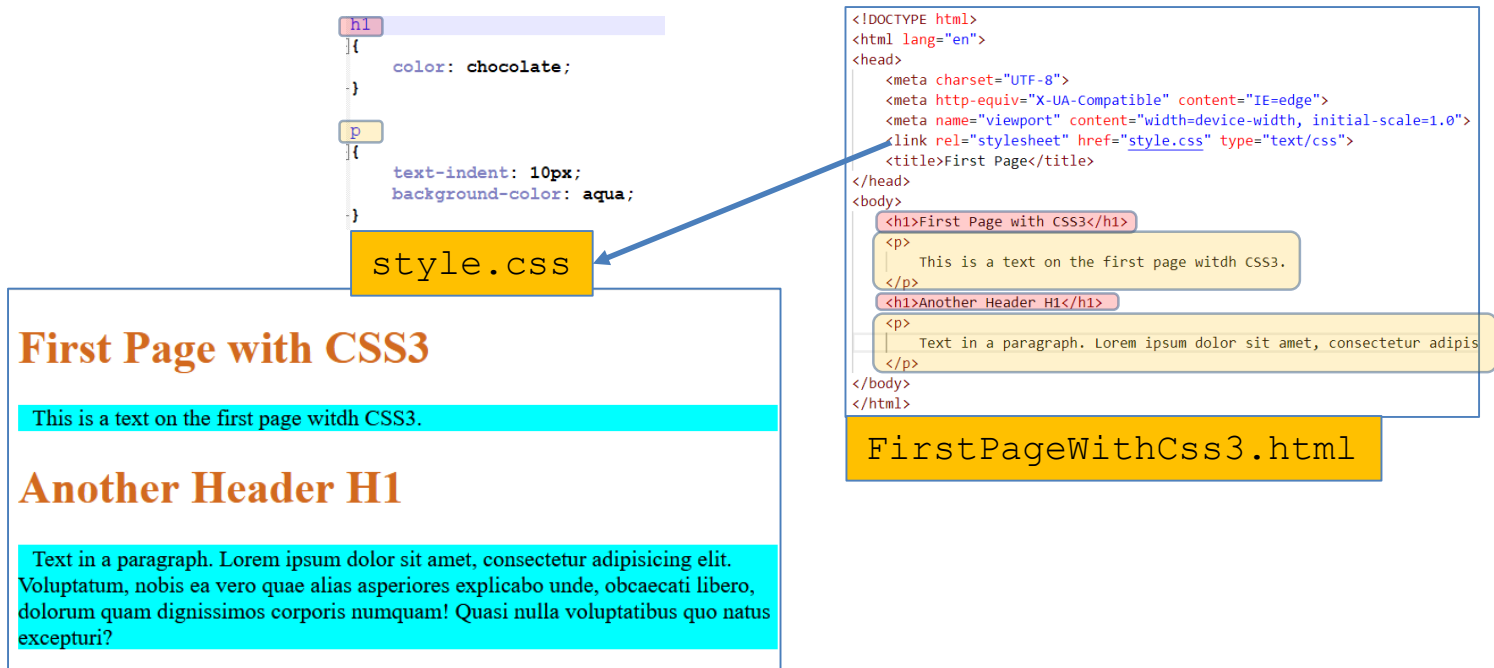
- HTML5, to **obecny standard**, ale **cały czas w rozwoju**.
 - Nie przewiduje się HTML6 itd.
- Istnieje bardziej restrykcyjny standard ***XHTML*** (*Extensible HyperText Markup Language*), który bazuje na **XML** (eXtensible Markup Language)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>First Page</title>
</head>
<body>
  <h1>First Page</h1>
  <p>
    This is a text on the first page.
  </p>
</body>
</html>
```



CSS3 – Kaskadowe arkusze stylów

- Teoretycznie w ramach HTML5 treść i strukturę dokumentu można *mieszać* z informacjami o jej prezentacji, ale **nie jest to wskazane**
 - Uniemożliwia łatwą zmianę szaty graficznej itp.
- **CSS3 - Cascading Style Sheets** wersja 3 pozwalają wydzielić informacje o prezentacji treści **w osobny fragment pliku** lub wręcz **osobny plik**.
 - Posiada swój własny **język opisu**



CSS3

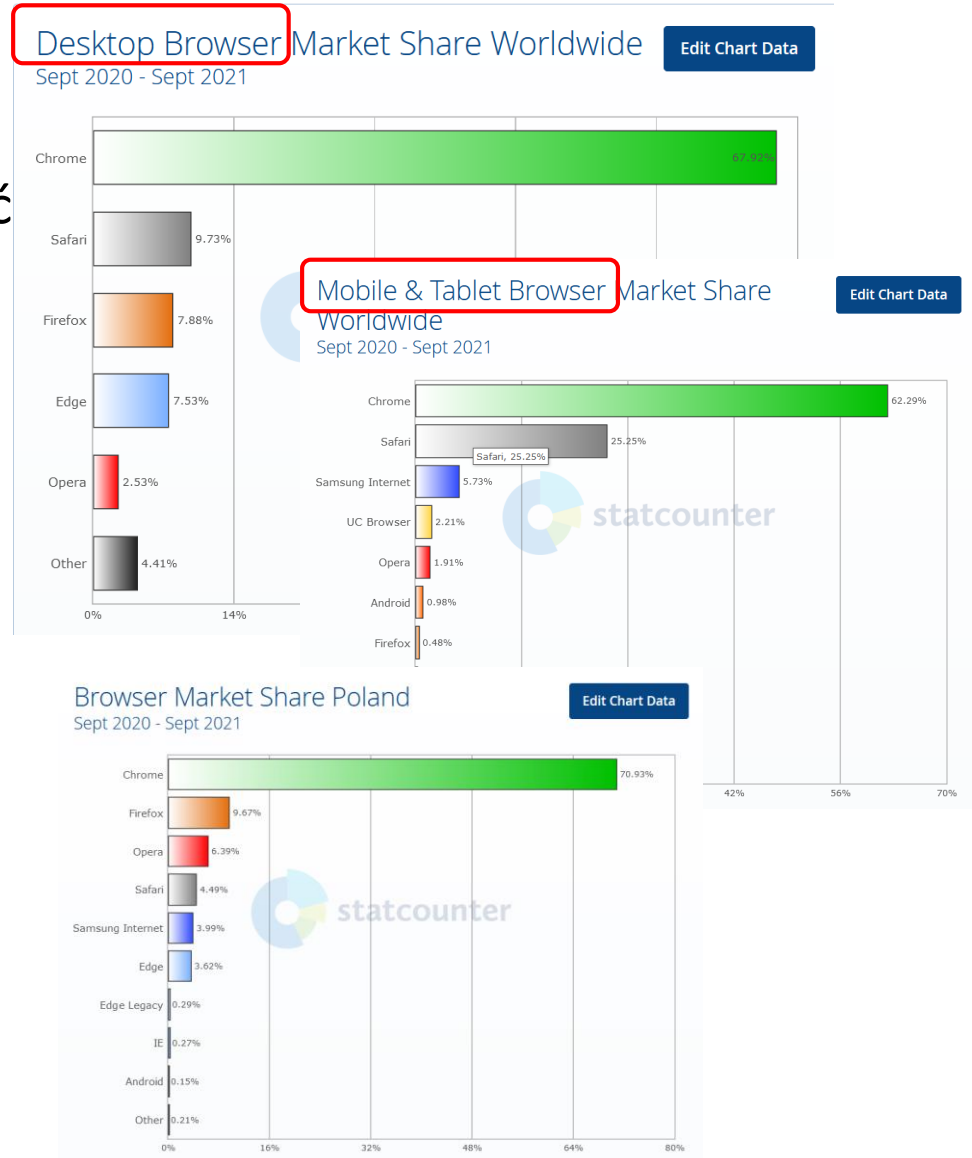
- **Cascading Style Sheets (CSS)** pozwala określić **prezentację** lub **stylizację** elementów na stronie (np. czcionki, odstępów, rozmiarów, kolorów, wyrównania itd.).
- Ponieważ CSS oddziela styl od treści i struktury można łatwo zmienić wygląd stron ***całego*** serwisu WWW lub jego części poprzez zmianę stylu w jednym pliku.
- CSS miał wersje od 1 do 3. CSS3 też może ulegać rozwojowi. Często jeśli używa się skrótu CSS w domyśle chodzi o CSS3.

Język JavaScript

- JavaScript – język skryptowy (kiedyś tylko do stron webowych)
 - do **manipulowania** zawartością **stron** WWW
 - pozwala tworzyć **dynamiczne strony**, które np. **reagują na zdarzenia** typu ruch myszką, zmianę czasu itp.
 - Pozwala też na programowanie server-side: **Node.js**
- Twórcy: najpierw Netscape (JavaScript), potem z Microsoft-em (JScript)
- Standaryzacji JavaScript w ECMA International (dawniej European Computer Manufacturers Association) jako ECMAScript.
- ECMAScript 5 (ES5) – bardzo popularna wersja
- ECMAScript 6 (ES6) – wersja standardu z klasami i obiektami.
 - ES6 == ES2015
 - Aktualna wersja ES2021, programowanie asynchroniczne itd.
- Programy napisane w JavaScript mogą działać w przeglądarkach internetowych na wielu urządzeniach.
 - W przeglądarce można też **wyłączyć** działanie **JavaScript**.
- Teoretycznie mogą istnieć inne języki skryptowe po stronie klienta, ale tylko teoretycznie:
 - VBScript dla Internet Explorer-a do wersji 10, potem już nieobsługiwany.

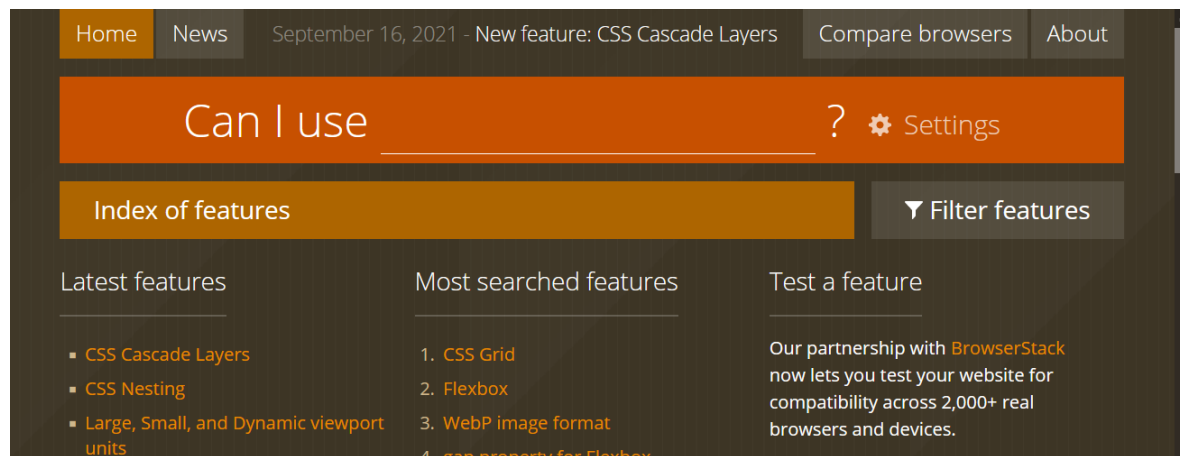
Przeglądarki stron internetowych

- Przeglądarki posiadają **podobne możliwości**, jednak każda z nich może interpretować i wyświetlać strony w na swój sposób
 - **Nie istnieje** norma, do której muszą się stosować producenci oprogramowania przy tworzeniu przeglądarek internetowych.
 - **Wsparcie** dla możliwości HTML5, CSS3 i JavaScript itd. **jest zależne** od przeglądarki.
 - Preferencje użytkowników mogą zależeć od innych cech przeglądarki niż „zgodność” z HTML5
- Dane z gs.statcounter.com



Dostępność funkcji HTML5 i CSS3: caniuse.com

- Sprawdzenie dostępności danej funkcji HTML5 i CSS3:
 - <http://caniuse.com/>

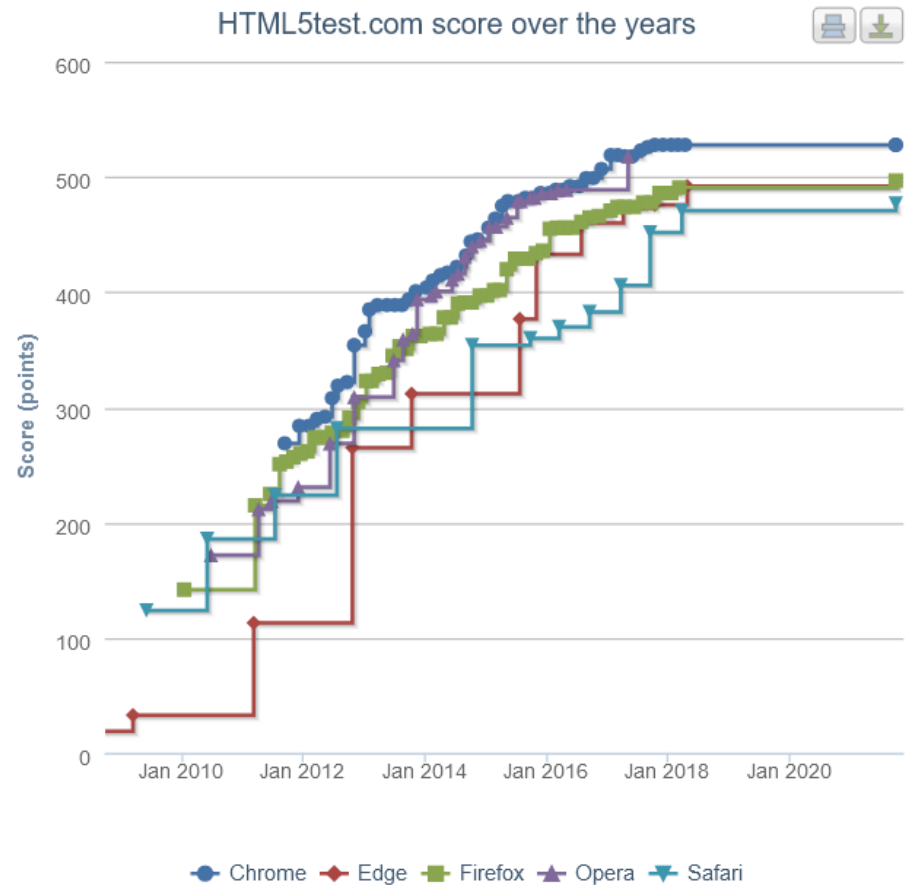
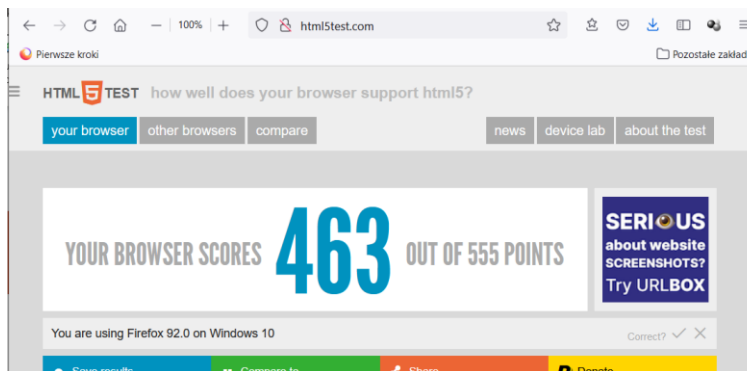


Ocena zgodności przeglądarki: html5test.com

- Ocena zgodności przeglądarki:

– <http://html5test.com/>

TIMELINE



Walidatory kodu HTML5, CSS3 i JavaScript

- Sprawdzają **składnię** kodu HTML5, CSS3 i JavaScript, aby przeglądarki poprawnie przetworzyły dokumenty.
 - Przeglądarki, w przypadku błędów, starają się **domyśleć** o co chodziło twórcy strony i poprawić go „w locie”, więc oglądając efekt w przeglądarce błąd **nie zawsze zostanie zauważony**.
- Poniżej najważniejsze walidatory.
- Inny ciekawy walidator i narzędzia dla web deweloperów:
 - <https://www.freeformatter.com/>

Język	Walidator
HTML5	http://validator.w3.org/ https://html5.validator.nu/
CSS3	http://jigsaw.w3.org/css-validator/ http://www.css-validator.org/
JavaScript	https://beautifytools.com/javascript-validator.php https://www.jshint.com/

WWW, HTML, HTTP

- Jak działa **przesyłanie pakietów** – przedmiot „Sieci komputerowe”
 - Komutacja pakietów
 - Protokół TCP/IP
 - Routing sieciowy
- Sieć WWW (World Wide Web) pozwala na korzystanie z aplikacji webowych i pobieranie multimedialnych.
- Tim Berners-Lee z CERN (European Organization for Nuclear Research) stworzył:
 - **Język HTML** - HyperText Markup Language.
 - **Protokół HTTP** (Hypertext Transfer Protocol) – do wysyłania informacji przez Internet
 - Były/są też inne protokoły np. gopher z większymi restrykcjami co do struktury dokumentu.
 - **Zaproponował URL** (Uniform Resource Locator) – unikalny lokalizator zasobów, który określa adres (lokalizację) strony (i nie tylko) internetowej wyświetlanej w oknie przeglądarki
- URL dla stron internetowych zaczyna się od `http://` lub `https://`

HTTPS

- **HTTPS** - Hypertext Transfer Protocol Secure.
- Jest standardem do przesyłania **zaszyfrowanych** danych w Internecie.
- Łączy on HTTP z **Secure Sockets Layer (SSL)** i systemem kryptograficznym — **Transport Layer Security (TLS)** — w celu zabezpieczenia komunikacji i informacji poufnych w Internecie.
- URL strony internetowej zaczynający się od `https://` jest obecnie obowiązującym standardem ze względu na niejawność zawartości żądań i odpowiedzi.

Podstawy WWW - hiperłącza

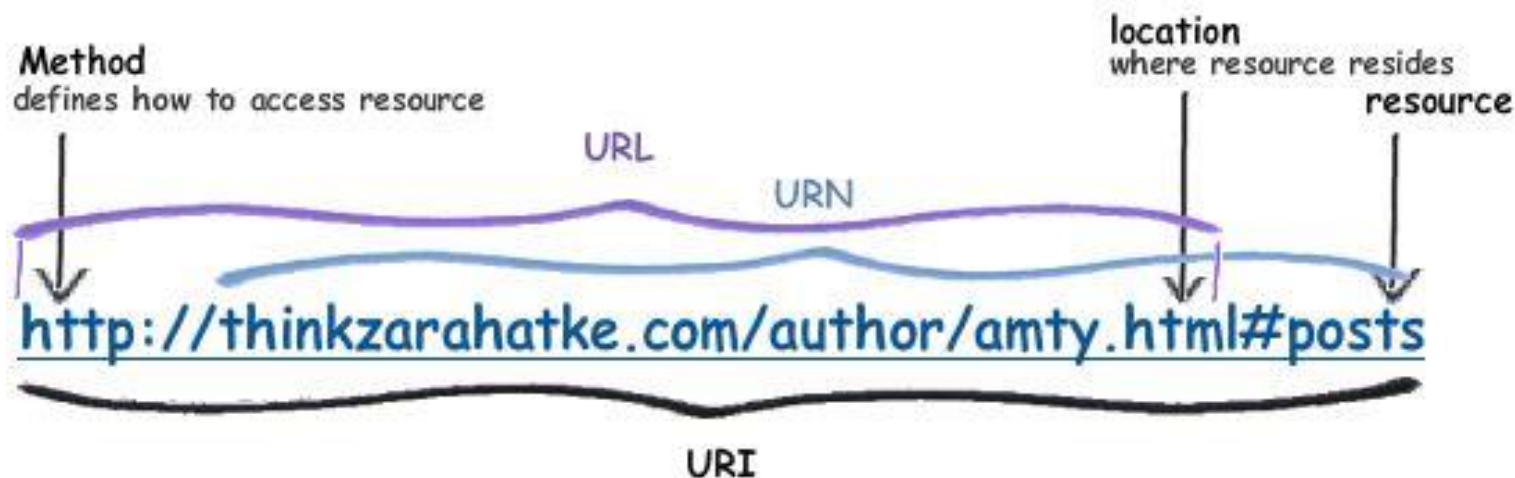
- W najprostszym przypadku strona internetowa to plik HTML (HyperText Markup Language) o rozszerzeniu `.html` lub `.htm`, który opisuje przeglądarce internetowej **treść i strukturę** dokumentu.
- Dokumenty HTML zazwyczaj zawierają **hiperłącza**, które kierują **do innego zasobu** (np. po kliknięciu ładują inny dokument WWW).
- Hiperłącza mogą prowadzić do tekstu/dokumentu, obrazu, filmu itp.
- Kiedy użytkownik **klika** hiperłącze, przeglądarka **komunikuje się** z docelowym serwerem WWW, który **lokalizuje** żądany zasób (np. stronę internetową) i **wysyła** ją do przeglądarki użytkownika.
- Użytkownik może również **wpisać adres** zasobu w **pasku adresu** przeglądarki i nacisnąć `Enter`, aby go **pobrać** (i **wyświetlić**, jeśli jest to strona WWW).

Hipertącza – c.d.

- W przypadku URL hipertącza w formie `mailto:adres_e-mail`, **kliknięcie** w link uruchamia **domyślnego klienta poczty** i otwiera okno wiadomości **zaadresowanej** do wskazanego odbiorcy.
- Istnieją **kilka** różnych rodzajów hipertączy (informacje pojawią się w **trakcie wykładu**)
- W przypadku hipertącza **do pliku**, który przeglądarka **nie jest w stanie wyświetlić**, przygotowuje się ona do **pobrania go** i pyta użytkownika, gdzie go **zapisać** (albo zapisuje do domyślnego folderu).

URL i URI

- URI (Uniform Resource Identifier) identyfikuje **zasób** w Internecie wraz z **metodą (protokołem)** dostępu.
- URL (Uniform Resource Locator) - URI, **bez** podania **konkretnego** miejsca w zasobie (**cały** zasób)
 - Strony WWW, to zasób, który zaczyna się od `http://` (lub `https://`)
- URN (Uniform Resource Name) - URI, który wskazuje nazwę zasobu (**bez protokołu**).
- W URI jest jeszcze miejsce na **login** i **hasło** (otwartym kodem, więc rzadko używane)

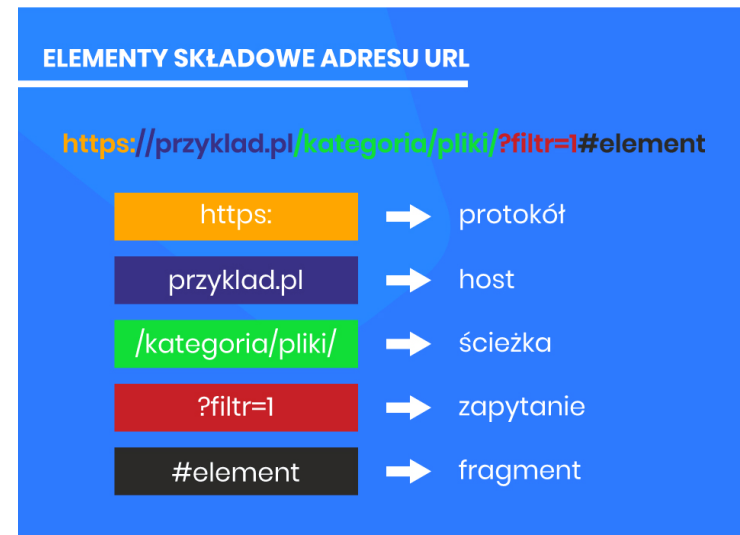
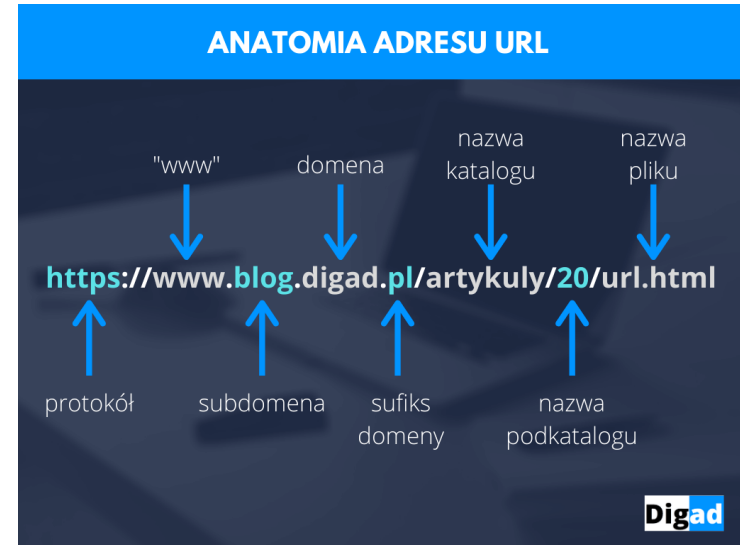


Źródło obrazka:

<https://stackoverflow.com/questions/42534419/examples-of-uri-url-and-urn>

URL - składowe

- Serwery WWW analizują przysłany URL/URI i udostępniają żądany zasób (jeśli istnieje i mamy do niego uprawnienia)
- Przykładowe serwery WWW (i nie tylko WWW):
 - Apache HTTP Server
 - Microsoft Internet Information Services
- Składowe adresu: górny rysunek.
- W URL mogą być też elementy zapytania.



Źródło obrazków: <https://digad.pl/adres-url-co-to-jest/>, <https://ks.pl/blog/adres-url>

Analiza działania

- Przykładowy adres URL: <https://www.dblog.digad.pl/artykuly/20/url.html>
- Fragment `https://` wskazuje, że HyperText Transfer **Protocol** Secure (HTTPS) powinien zostać zastosowany w celu uzyskania zasobu.
- Fragment www.dblog.digad.pl to **nazwa hosta** (hostuje zasób), czyli nazwa komputera z serwerem WWW, na którym znajduje się zasób.
 - Nazwa hosta www.dblog.digad.pl jest tłumaczona na adres IP (Internet Protocol) — 4-bajtową liczbę, która jednoznacznie identyfikuje serwer w Internecie. Zajmują się tym serwery DNS (Domain Name System) zarządzające bazą danych nazw hostów i odpowiadających im adresów IP.
- Fragment `/artykuly/20/` oznacza **ścieżkę** lokalizacji zasobu na hoście
- Fragment `url.html` **konkretny zasób** na tej ścieżce
- Ścieżka zasobu **nie musi odpowiadać** rzeczywistemu **drzewu folderów i plików** na serwerze.
 - Najprostsze rozwiązanie mapowania adresu URL na ścieżkę w drzewie folderów serwera:

`sun10.pwr.edu.pl/~koniecz/mac/macierz.html`

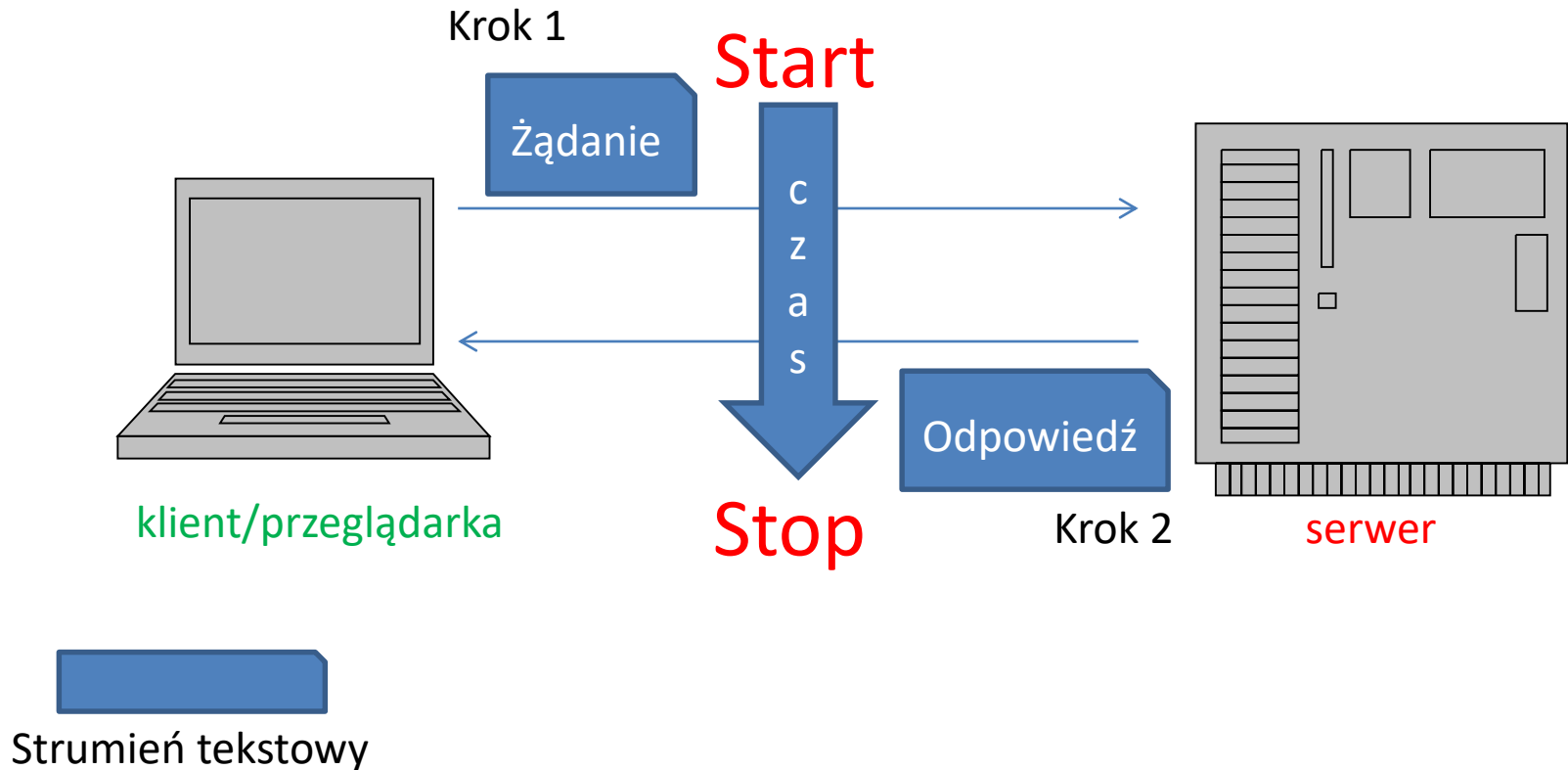
URL

`sun10.pwr.edu.pl
/users/staff/koniecz/wwwroot/mac/macierz.html`

Ścieżka
na sun10

Protokół bezstanowy HTTP

- Bezstanowy, brak historii, bez pamiętania sekwencji żądań
 - To cecha **protokołu**
 - W praktyce, jeśli trzeba z kolejnych żądań tworzyć sesję, przeglądarka i serwer muszą współpracować przysyłając odpowiednie informacje, najczęściej w nagłówkach (ciasteczka)



Budowa żądania/odpowiedzi w HTTP

- **Żądanie i odpowiedź to strumień tekstowy** zawierający linie rozdzielone znakiem przejścia do nowej linii
 - Można **traktować** jak przesyłane pliki tekstowe (.TXT)
- Żądanie (ang. request) jak i odpowiedź (ang. response) mają **taką samą budowę**:
 - **Jedna linia** żądania i odpowiedzi
 - Kolejne **linie** aż do **pierwszej pustej** – linie **nagłówków**
 - **Pusta linia** (rozdzielająca)
 - Cała reszta linii (mogą być też puste) – **ciało** pytania/odpowiedzi
- W celu **obserwacji** żądań i odpowiedzi istnieje wiele narzędzi. Najprostsze i najczęściej pod ręką to:
 - Komenda/program konsolowy `curl`
 - W każdej sensownej przeglądarce – „Narzędzia dla twórców witryn”

Komenda curl

- curl to narzędzie do przesyłania danych z lub na serwer w protokole HTTP(S).
- Bez opcji wypisuje treść otrzymanego strumienia HTML.
- Opcja -h wypisuje wszystkie opcje, ich skróty i jakie dodatkowe parametry są wymagane
- Poniżej efekt wywołania:
 - curl -h
 - curl <http://www.wp.pl/>
- Ostatnie wywołanie wskazuje, że pojawia się (teoretycznie) strona z błędem:
 - 303 Moved Permanently
- Jednak nie widać informacji pod jaki inny adres strona została przesunięta. Aby to sprawdzić należy wybrać opcję pokazującą również linię odpowiedzi i nagłówki odpowiedzi.

Wiersz polecenia

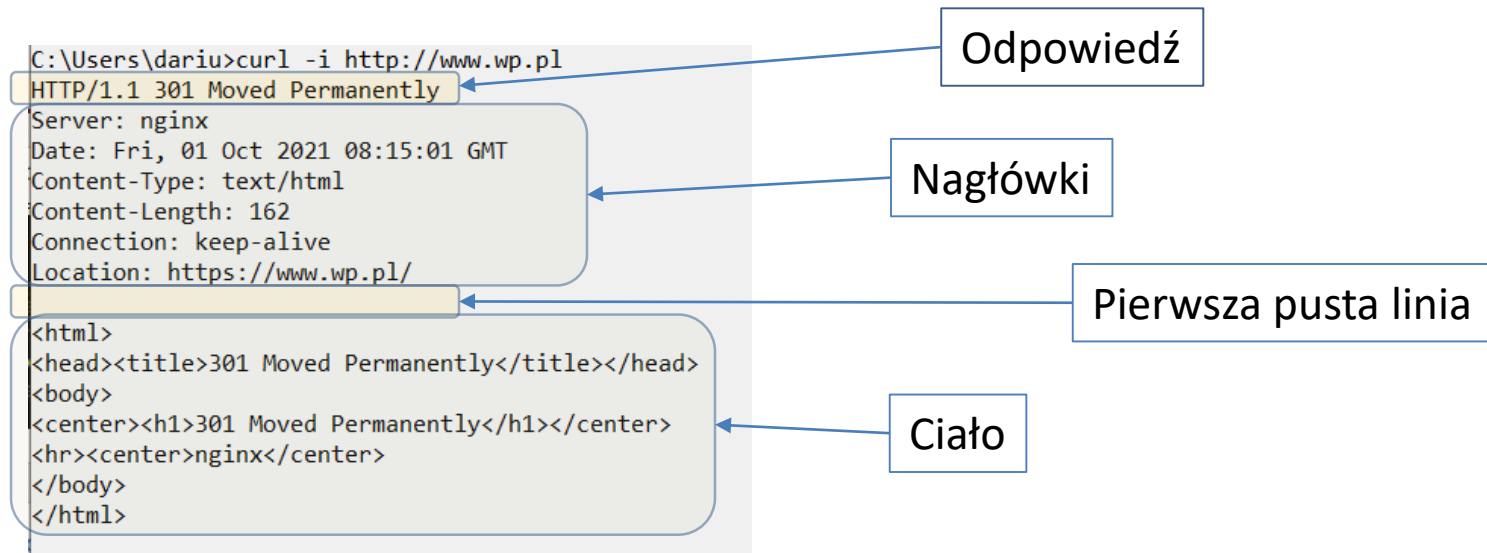
```
C:\Users\dariu>curl
curl: try 'curl --help' for more information

C:\Users\dariu>curl -h
Usage: curl [options...] <url>
--abstract-unix-socket <path> Connect via abstract Unix domain socket
--anyauth             Pick any authentication method
-a, --append          Append to target file when uploading
--basic              Use HTTP Basic Authentication
--cacert <CA certificate> CA certificate to verify peer against
--capath <dir>       CA directory to verify peer against
-E, --cert <certificate[:password]> Client certificate file and password
--cert-status        Verify the status of the server certificate
--cert-type <type>   Certificate file type (DER/PEM/ENG)
--ciphers <list of ciphers> SSL ciphers to use
--compressed        Request compressed response
-K, --config <file>  Read config from a file
--connect-timeout <seconds> Maximum time allowed for connection
--connect-to <HOST1:PORT1:HOST2:PORT2> Connect to host
-C, --continue-at <offset> Resumed transfer offset
```

```
C:\Users\dariu>curl http://www.wp.pl
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

Komenda curl – ciekawe opcje

- opcja `-i`, `--include` Include protocol **response headers** in the output
 - Pokazuje również **nagłówki** odpowiedzi
- Poniższy przykład:
 - `curl -i http://www.wp.pl/`
 - W odpowiedzi:
 - HTTP/1.1 301 Moved Permanently
 - W nagłówku znajduje się linia:
Location: <https://www.wp.pl/>
- Przykład na kolejnym slajdzie po żądaniu do zabezpieczonego serwera:
 - `curl -i https://www.wp.pl/`



Komenda `curl` – rozbudowany nagłówek

[illegible]

Komenda `curl` – rozbudowany nagłówek

- Tym razem otrzymujemy odpowiedź (w pierwszej linii):
 - `HTTP/1.1 200 OK`
- Następnie jest wiele linii nagłówków od
 - `Server: nginx`
- Aż do:
 - `X-Op-Id-All: 5e2s`
- Kolejna linia jest pusta, zatem tu kończą się nagłówki.
- W następnej linii zaczyna się ciało odpowiedzi, czyli treść strony w języku HTML, zaczynając od linii o treści:
 - `<!DOCTYPE html>`
- Kolejne ewentualne puste linie będą po prostu zawartością dokumentu HTML.

Komenda curl – inne opcje

- Z opcją `-v` (verbose – gaduła) zobaczyć można wszystkie kroki komunikacyjne)
 - Najlepiej wynik przekierować do pliku (b. dużo linii):
`curl -i -v https://www.wp.pl > output.txt`
 - W strumieniu będą również linie z poprzedniego slajdu
- Wybrane opcje:
 - `-H, --header <header/@file>` Pass custom header(s) to server
 - `-G, --get` Put the post data in the URL and use GET
 - `-d, --data <data>` HTTP POST data

```
C:\Users\darius\Documents>curl -i -v https://www.wp.pl > output.txt
* Rebuilt URL to: https://www.wp.pl/
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
0          0          0      0         0      0 --:--:-- --:--:-- --:--:--    0*   Trying 212.77.98.9...
* TCP_NODELAY set
0          0          0      0         0      0 --:--:-- --:--:-- --:--:--    0*   Connected to www.wp.pl (212.77.98.9) port
t 443 (#0)
* schannel: SSL/TLS connection with www.wp.pl port 443 (step 1/3)
* schannel: checking server certificate revocation
* schannel: sending initial handshake data: sending 180 bytes...
* schannel: sent initial handshake data: sent 180 bytes
* schannel: SSL/TLS connection with www.wp.pl port 443 (step 2/3)
* schannel: failed to receive handshake, need more data
* schannel: SSL/TLS connection with www.wp.pl port 443 (step 2/3)
* schannel: encrypted data got 3796
* schannel: encrypted data buffer: offset 3796 length 4096
* schannel: sending next handshake data: sending 93 bytes...
* schannel: SSL/TLS connection with www.wp.pl port 443 (step 2/3)
* schannel: encrypted data got 290
* schannel: encrypted data buffer: offset 290 length 4096
* schannel: SSL/TLS handshake complete
* schannel: SSL/TLS connection with www.wp.pl port 443 (step 3/3)
* schannel: stored credential handle in session cache
> GET / HTTP/1.1
< Host: www.wp.pl
```

Przebieg komunikacji - przeglądarka

- Serwer otrzymuje część adresu URI za **nazwą serwera** np. `artykuly/20/url.html`
- Otrzymuje najczęściej jako zapytanie GET protokołu HTTP(S) z podaną nazwą i wersją protokołu:
`GET artykuly/20/url.html HTTP/1.1`
- Inne częste zapytanie to POST (gdy wysyłane są dane z **formularza** na stronie WWW). Istnieją jeszcze inne rodzaje żądań, część z nich będzie w temacie programowania kontrolera API.
- Po tej linii mogą być linie nagłówek (tworzy je przeglądarka, zwykły użytkownik tego nie widzi)
- Dla części rodzajów zapytań może być linia pusta i ciało żądania (np. dla POST będą tam dane z formularza w odpowiednim formacie)

Przebieg komunikacji - serwer

- W odpowiedzi na żądanie serwer w pierwszej linii wskazuje wersję protokołu HTTP oraz opisuje stan transakcji za pomocą kodu liczbowego i frazy:
 - Np. dla sukcesu:
 - `HTTP/1.1 200 OK.`
 - A gdy nie może zlokalizować żadanego zasobu:
 - `HTTP/1.1 404 Not found`
- Następnie serwer wysyła jeden lub więcej nagłówków HTTP, z dodatkowych informacjami na temat przesyłanych danych.
 - Nazwa nagłówka jest przed dwukropkiem ,:’
 - Składowe wartości nagłówka rozdzielane są średnikami ,;’

Przebieg komunikacji - serwer

- Np. jeden z nagłówków HTTP dla przykładu www.wp.pl jest następujący:
 - `Content-type: text/html; charset=utf-8`
- Jest to informacja dla przeglądarki, że w ciele będzie plik tekstowy (kod HTML5) oraz użyte jest kodowanie znaków utf-8.
- Pierwsza część to typ określany przez Multipurpose Internet Mail Extensions (MIME) type, wymyślony dla przesyłania emaili, ale używany wszechobecnie:
 - <https://www.iana.org/assignments/media-types/media-types.xhtml>
- W tym przypadku MIME type `text/plain` oznacza, że wysyłane informacje są tekstem i mogą być bezpośrednio wyświetlane.
- Analogicznie MIME type `image/jpeg` oznacza, że zawartość jest obrazem JPEG.
 - Gdy przeglądarka otrzymuje ten MIME type, próbuje wyświetlić obraz.

Przebieg komunikacji – serwer/przeglądarka

- Po nagłówkach następuje jedna pusta linia
- Ostatecznie serwer wysyła zawartość żadanego dokumentu (jeśli nie ma nazwy dokumentu na najczęściej domyślnie wysyła `index.html`)
`index.html`.
- Następnie przeglądarka po stronie klienta renderuje (lub wyświetla) dokument, co może wiązać się z dodatkowymi żadaniami HTTP, aby uzyskać powiązane pliki CSS, obrazy i inne, których URI odkryte zostaną podczas renderowania.
 - **Renderowanie** ([ang. rendering](#)) – graficzne przedstawienie treści zapisanej cyfrowo w formie właściwej dla danego środowiska (np. wyświetlenie w oknie przeglądarki, strony WWW zapisanej w kodzie HTML) (źródło: wikipedia.pl)

Żądanie Get – parametry w URL, query string

- Żądanie GET może posiadać **parametry**, które znajdują się za znakiem zapytania '?',
`www.google.com/search?q=konieczny`
- W tym wypadku 'search' jest nazwą programu/strony obsługującego formularze po stronie serwera Google, 'q' jest nazwą zmiennej w wyszukiwaniu Google'a 'konieczny' jest szukaną frazą.
- Ciąg za znakiem zapytania to tzw. **query string**.
- **Para** 'nazwa=wartość' jest przekazywana do serwera i jest rozdzielona **znakiem równości** '='.
- Pary **rozdzielane** są znakiem **ampersand** '&'.
- Dane w ten sposób przekazywane często służą do odnalezienia odpowiedniego zasobu na serwerze.
- Żądanie GET ma **puste ciało** żądania.

Żądanie POST

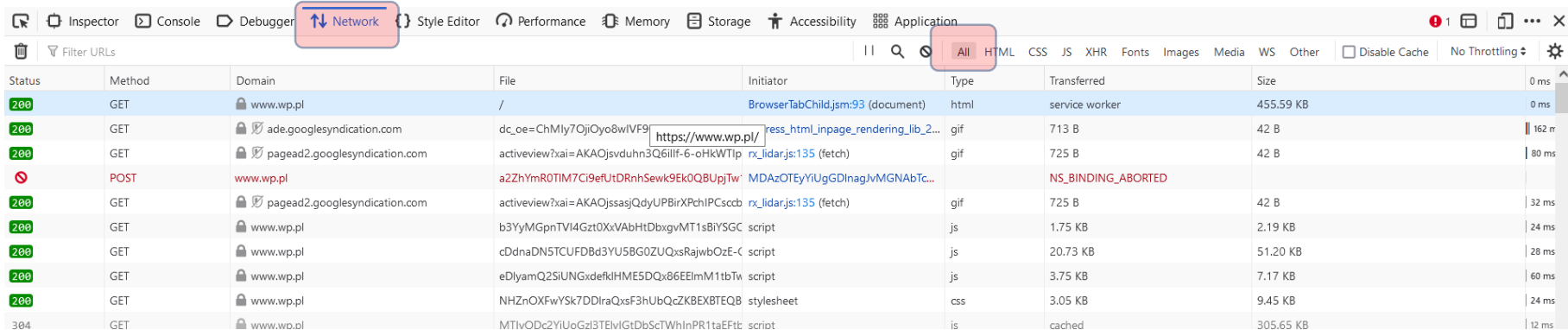
- Żądanie POST wysyła **dane z formularza** jako **ciało** wiadomości HTTP a **nie** jako **część adresu URL**
 - Standardowo format jest taki sam jak w query string
 - Może być inny format, np. często JSON
- Żądanie GET może mieć pewną **ograniczoną długość** co ogranicza długość query string, więc często niezbędne jest wysyłanie większych ilości informacji używając metody POST.
 - Minimalna ograniczenie długości URI: **255 znaków**
 - Limit na długość (zależna od przeglądarki): 2KB-8KB
 - Pośrednicy na drodze wiadomości do/z serwera mogą dodawać własne ograniczenia(ale nie poniżej 255)
- Metoda POST ukrywa przesyłane dane przed użytkownikiem, ale z tego powodu nie da się jej „podlinkować” w innym dokumencie WWW.
 - Linkować można tylko za pomocą URL, zatem nie ma możliwości wpisania czegokolwiek do **ciała** wysyłanego **zapytania**

Pamięć podręczna po stronie klienta (**cache**)

- Przeglądarki często **przechowują na dysku** (cache) ostatnio przeglądane strony internetowe w celu ich przyszłego szybkiego wczytania jak i wykonania operacji „cofnij do poprzedniej strony”.
- Jeśli nie ma **żadnych zmian** pomiędzy wersją przechowywaną w pamięci podręcznej i aktualną wersją w Internecie, **przyspiesza** to przeglądanie.
- Odpowiedź HTTP może wskazywać **czas** (w nagłówku odpowiedzi), przez który zawartość **pozostaje aktualna**.
- Przeglądarka może kierować się tymi danymi i wczytać dokument z pamięci podręcznej.
- Istnieje także odpowiedź HTTP “304 not modified” oznaczająca, że zawartość pliku nie zmieniła się od ostatniego żądania.
- Przeglądarki zazwyczaj nie zapamiętują odpowiedzi serwera na żądania POST (np. z danymi formularza), ponieważ następne takie żądanie może nie zwrócić tej samej odpowiedzi.
 - Dodatkowo mogłoby to generować powtórna operację (gdy np. cofniemy się do poprzedniej strony) w banku/sklepie, co raczej nie jest oczekiwane przez użytkownika.

Narzędzia programistyczne przeglądarki 1/3

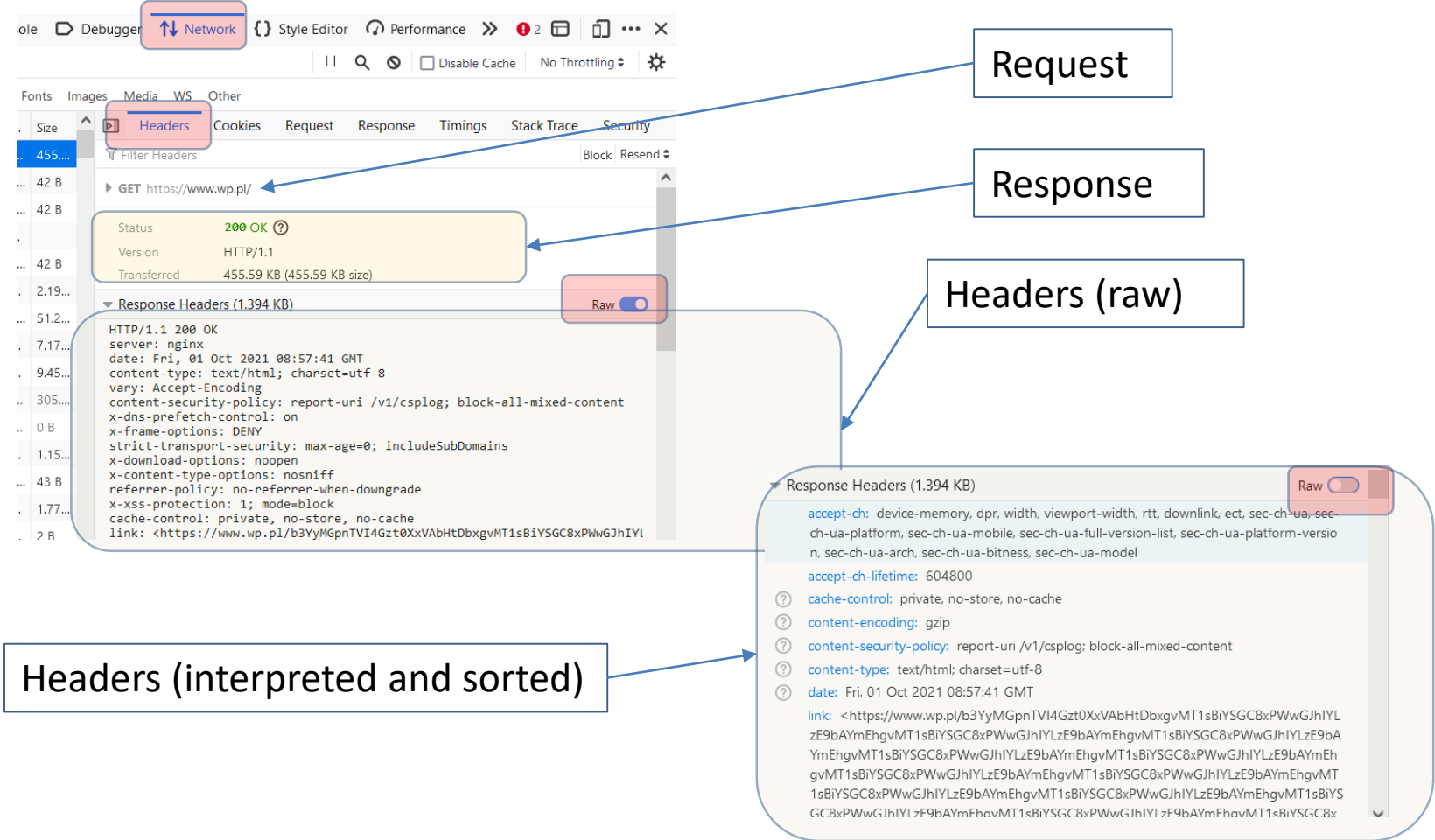
- Przełącznik w większości przeglądarek: klawisz <F12> , <Ctrl>+<Shift>+<i>
- Czasami należy wykonać **odświeżenia strony** (z powodu cache'owania zasobów)
- Pokazuje **wszystkie żądania** wykonane przez przeglądarkę (również dla zdjęć, plików CSS, JS itp.)
- Poniżej zrzut ekranu dla przeglądarki Firefox dla strony <https://www.wp.pl/>
- Okno narzędzia może być na bocznej stronie okienka, w innym języku narodowym itd., prze co wyglądać trochę inaczej.



Status	Method	Domain	File	Initiator	Type	Size	Time
200	GET	www.wp.pl	/	BrowserTabChild.jsm:93 (document)	html	455.59 KB	0 ms
200	GET	ade.googleadsyndication.com	dc_oe=ChMly7OjOyo8wIVF9...	https://www.wp.pl/ress_html_inpage_rendering_lib_2...	gif	42 B	162 ms
200	GET	pagead2.googleadsyndication.com	activeview?xai=AKAOjsvduhn3Q6illF-6-oHkWTip	rx_lidar.js:135 (fetch)	gif	42 B	80 ms
NS_BINDING_ABORTED	POST	www.wp.pl	a2ZhYmR0TIM7C9eUtdRnhSewk9Ek0Q8UpjTw...	MDAzoTEyYiUgGDlnagJvMGNAbTc...			
200	GET	pagead2.googleadsyndication.com	activeview?xai=AKAOjsasjQdyUPBirXPchIPCscdb	rx_lidar.js:135 (fetch)	gif	42 B	32 ms
200	GET	www.wp.pl	b3YyMGpnTVI4Gzt0XxVAbHtDbxgvMT1sBiYSGC	script	js	2.19 KB	24 ms
200	GET	www.wp.pl	cDnaDN5TCUFD8d3YU58G0ZUQxsRajwbOzE-C	script	js	51.20 KB	28 ms
200	GET	www.wp.pl	eDlyamQ25iUNGxdefklHME5DQx86EElmM1tbTw	script	js	7.17 KB	60 ms
200	GET	www.wp.pl	NH2OXFwYsk7DDlraQxsF3hUbQcZKBXBTEQB	stylesheet	css	9.45 KB	24 ms
304	GET	www.wp.pl	MTIvODc2YUoGzI3TElviGtdBScTWhlnPR1taEftb	script	is	305.65 KB	12 ms

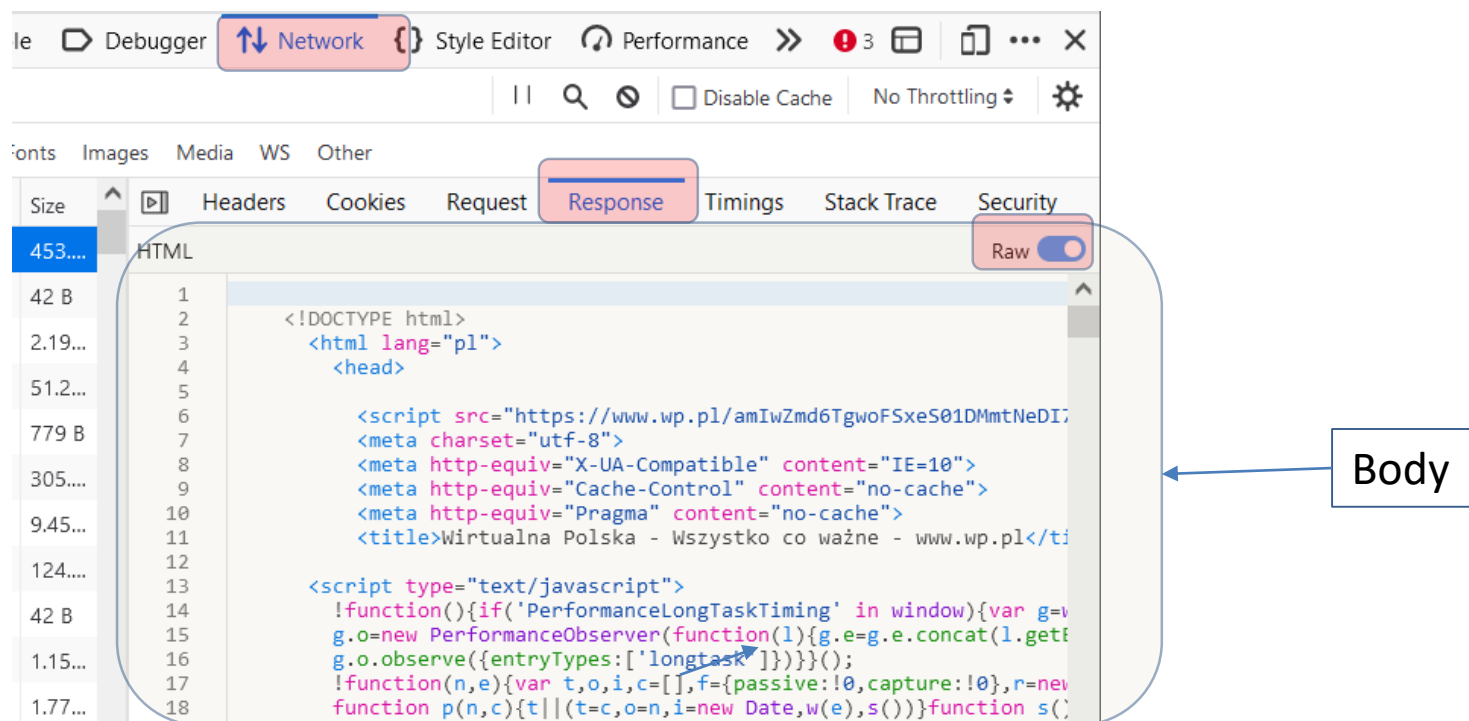
Narzędzia programistyczne przeglądarki – 2/3

- Poniżej zrzut po kliknięciu np. pierwsze żądanie GET www.wp.pl z listy
- W zakładce „Nagłówki” (Headers): Żądanie (Request), Odpowiedź (Response), Nagłówki (Headers)
- Ostatnie w wersji nieprzetworzonej (raw) i przetworzonej.



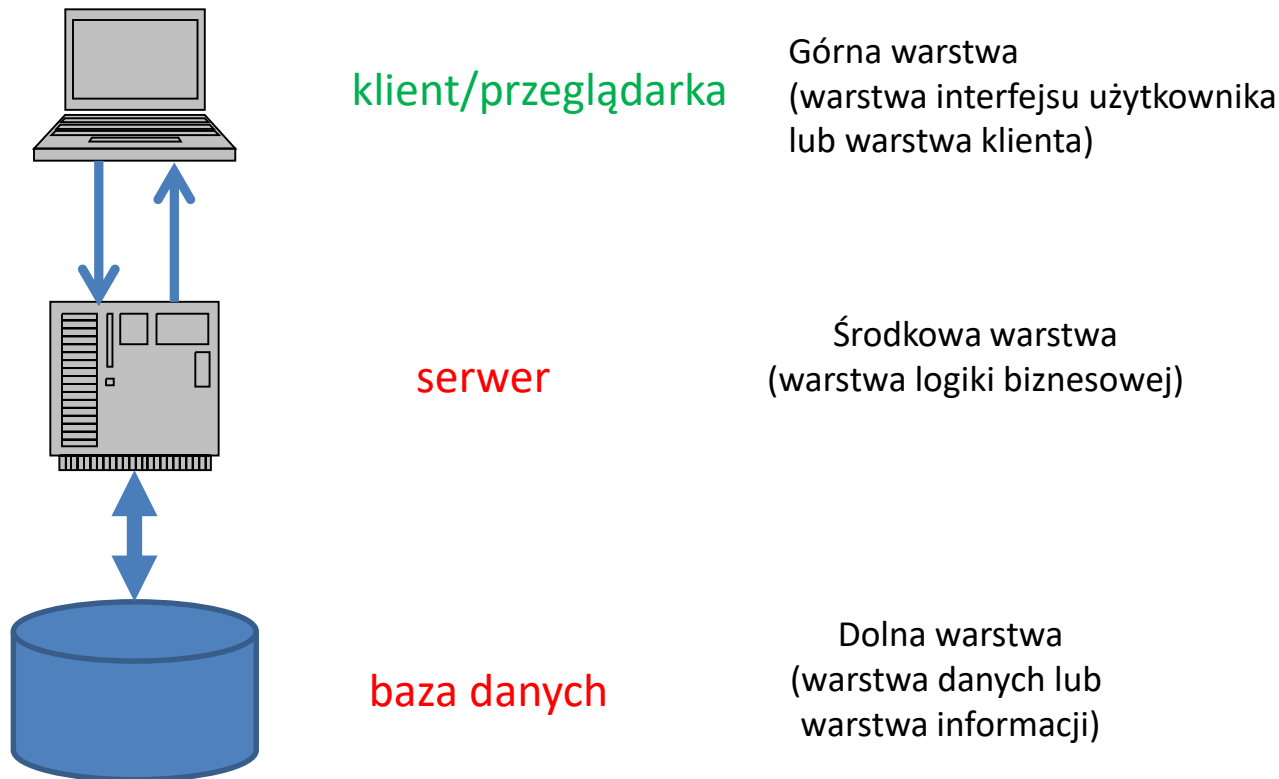
Narzędzia programistyczne przeglądarki – 3/3

- Poniżej zrzut po kliknięciu np. pierwsze żądanie z listy oraz w zakładce „Odpowiedź” (response): Treść odpowiedzi, czyli dokument HTML (też z przełącznikiem raw).



Wielowarstwowa architektura aplikacji 1/3

- Aplikacje webowe są często aplikacjami wielowarstwowymi
 - Warstwy odpowiadają za różne, logicznie powiązane, funkcjonalności.
- Trójwarstwowa aplikacja webowa:
 - Klient/przeglądarka – Górna warstwa (warstwa interfejsu użytkownika lub warstwa klienta)
 - WWW Serwer – Środkowa warstwa (warstwa logiki biznesowej)
 - Baza danych – Dolna warstwa (warstwa danych lub warstwa informacji)



Wielowarstwowa architektura aplikacji 2/3

- **Dolna warstwa** (warstwa **danych**, warstwa **informacji**) zarządza danymi aplikacji.
 - Przechowuje, najczęściej, dane w systemie zarządzania relacyjną bazą danych — relational database management system (RDBMS).
- **Środkowa warstwa** (warstwa **logiki biznesowej**) steruje i kontroluje dane potrzebne do interakcji pomiędzy klientami aplikacji i jej danymi.
 - Środkowa warstwa działa jako pośrednik pomiędzy danymi w warstwie informacyjnej i klientami aplikacji.
 - Logika sterowania środkowej warstwy przetwarza żądania klienta i odzyskuje dane z bazy danych.
 - Część środkowej warstwy odpowiedzialna za logikę prezentacji przetwarza dane z warstwy informacji i prezentuje zawartość klientowi.
- Aplikacje webowe zazwyczaj prezentują dane klientowi jako dokumenty HTML.

Wielowarstwowa architektura aplikacji 3/3

- Logika biznesowa w środkowej warstwie pilnuje zasad biznesowych i zapewnia, że dane są wiarygodne, zanim aplikacja aktualizuje bazę danych lub przedstawi dane użytkownikowi.
- Zasady biznesowe określają, jak klienci [aplikacje, a nie ludzie] uzyskują dostęp do danych i jak aplikacje przetwarzają dane.
- **Górna** warstwa (warstwa **klienta**) zawiera interfejs graficzny aplikacji, który zbiera dane wejściowe i wyświetla dane wyjściowe.
- Użytkownik steruje aplikacją poprzez interfejs użytkownika, którym zazwyczaj jest przeglądarka internetowa bądź urządzenie mobilne.
- Użytkownik podejmuje akcję (np. kliknięcie hiperłącza), w odpowiedzi warstwa klienta kontaktuje się ze środkową warstwą, której wysyła żądanie aby (poprzez środkową warstwę) odzyskać dane z warstwy informacyjnej. Po ich uzyskaniu wyświetla dane uzyskane dla użytkownika.

Skrypty po stronie klienta

- Skrypty po stronie klienta w języku JavaScript mogą być użyte:
 - do sprawdzania poprawności danych wejściowych wprowadzonych przez użytkownika,
 - do rozszerzenia funkcjonalności stron internetowych, ich graficznej atrakcyjności
 - do obsługi „samoistnej” komunikacji pomiędzy przeglądarką a serwerem WWW.
 - do naprawiania kolizji między frameworkami
- Ograniczenia skryptów po stronie klienta:
 - zależność od przeglądarki;
 - przeglądarka lub tzw. scripting host musi obsługiwać język i jego możliwości.
 - ze względów **bezpieczeństwa** mają **ograniczoną** możliwość **dostępu do lokalnego sprzętu i systemu plików**.
 - kod źródłowy skryptów po stronie klienta **może być wyświetlony** przez klienta (opcja „pokaż źródło strony” i narzędzia programistyczne).
 - Poufne informacje (hasła do bazy danych, hasła użytkownika, dane osobowe), **nie powinny się znajdować** po stronie klienta.
 - Mimo, że klient sprawdzi dane np. w formularzu, sprawdzanie to musi być dublowane również po stronie serwera.
 - Można użyć np. `curl`, aby wysłać zapytanie do środkowej warstwy z pominięciem warstwy klienta
 - umieszczenie pewnych operacji w JavaScript po stronie klienta może być również niebezpieczne dla całej aplikacji.
- W zasadzie jedyny język skryptów to JavaScript.

Skrypty po stronie serwera

- Możliwość używania dużej gamy języków programowania (teoretycznie każdego dojrzałego języka posiadającego możliwości strumieni i rejestracji w usługach danego systemu operacyjnego lub serwera usług sieciowych).
 - PHP
 - Java
 - C#
 - itd.
- Wiele gotowych serwerów z możliwością rozszerzania funkcjonalności:
 - MS IIS – rozszerzenia ASAPI
 - Apache – moduły
 - itd.
- Rozszerzenia mogą być pisane w różnych językach
- Skrypty/programy po stronie serwera mogą komunikować się z innymi serwerami, aby pobrać informacje, wykonać ich przetwarzanie i odesłać wynik do klienta (porównywarki)
 - Między innymi komunikują się z warstwą danych
- Skrypty muszą najczęściej działać dla wielu sesji
 - Rozwiązanie powinno być skalowalne (jeśli liczba żądań wzrasta, wystarczy dokupić sprzętu, zainstalować oprogramowanie i czas odpowiedzi znacząco nie wzrośnie).

Podstawy języka

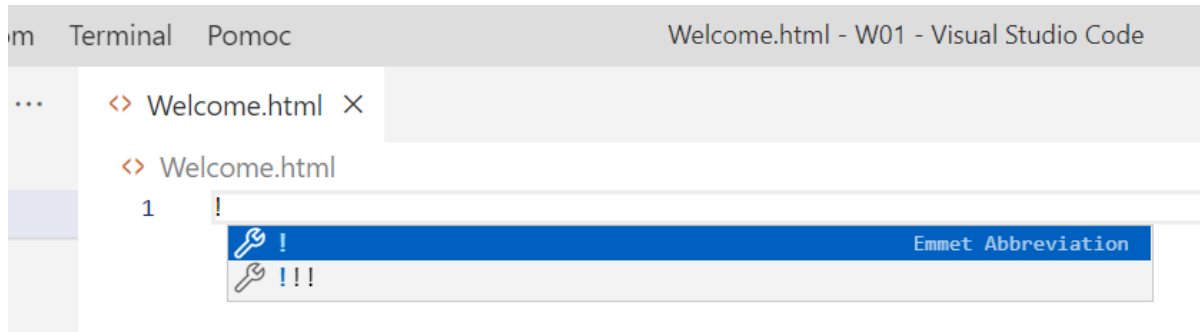
HTML5

Wprowadzenie

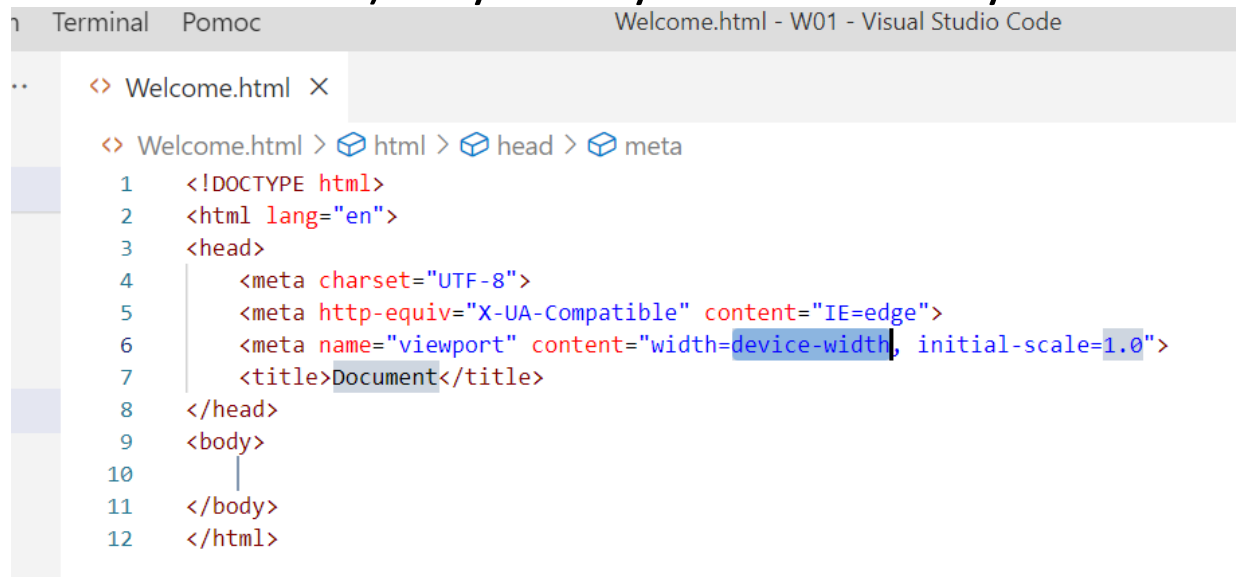
- **HTML5 (HyperText Markup Language 5)** - język znaczników opisujący **strukturę** i **zawartość** dokumentów wyświetlanych w przeglądarkach internetowych
- Najprostsze tworzenie – dowolny edytor tekstowy:
 - Notatnik
 - Notepad++
 - vi
 - Visual Studio Code z odpowiednimi wtyczkami (będzie głównie używany w przykładach na wykładzie)
- Plik zapisywać z rozszerzeniem `.html` (kiedyś również `.htm`)
- Następnie plik otworzyć w przeglądarce (np. poprzez podwójne kliknięcie w niego)

Tworzenie szkieletu HTML w VS Code

- Po stworzeniu i otwarciu pliku `Welcome.html` w VS Code wpisujemy znak wykrzyknika:



- A potem klawisz `<tab>`. Zadziała tzw. emmet (dla stron z rozszerzeniem `.html`) i wytworzy szkielet strony HTML



Przykładowy plik welcome.html i otwarcie w przeglądarce

<> Welcome.html X

<> Welcome.html > ...

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Welcome</title>
8 </head>
9 <body>
10  <!-- This is a comment. -->
11  <p>This is a paragraph in Html5.</p>
12 </body>
13 </html>
```

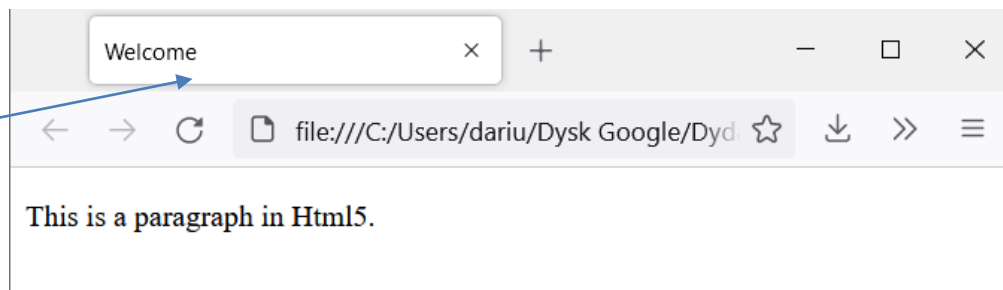
Element head

Element title
zawierający
tekst Welcome

Komentarz HTML, nie interpretowany
przez przeglądarke

Element p wewnątrz body

Karta pokazuje
zawartość elementu
title



Składowe przykładowego pliku

- Deklaracja Typu Dokumentu (DOCTYPE)
 - wymagana w dokumentach HTML5, aby przeglądarki renderowały stronę wg najnowszych standardów.
 - Gdy deklaracji brak - niektóre przeglądarki działają w specjalnym trybie („trybie dziwactw”).
- Komentarze:
 - zaczynają się od `<!--`
 - i kończą z `-->`.
 - mogą być wielolinijkowe

Składowe przykładowego pliku

- Dokumenty HTML5 ograniczają większość elementów znacznikiem początkowym i końcowym.
 - **Znacznik początkowy** składa się z nazwy elementu w nawiasach ostrych
Np. `<html>`
 - **Znacznik końcowy** składa się z nazwy elementu poprzedzonej ukośnikiem (/) w nawiasach ostrych
Np. `</html>`
- Istnieją tzw. „puste elementy”, które **nie mają** znaczników końcowych.
 - Dygresja: XHTML czerpiący z XML musi posiadać zamknięcie znacznika, stąd w HTML powinno się stosować `
`, a w XHTML `
 </br>` lub w jednym zapisie `
`.
 - Większość przeglądarek poprawnie naprawi zastosowanie składni XHTML w HTML.
- Wiele znaczników początkowych ma atrybuty przekazujące dodatkowe informacji o elemencie. Przeglądarki używają ich do ustalenia, jak przetworzyć dany element.
 - Atrybut ma nazwę i wartość oddzielone znakiem równości (=).
 - Wartości **muszą być** w cudzysłowach.
 - Atrybuty rozdzielone są białym znakiem (spacja, tabulacja, Enter)

Składowe przykładowego pliku

- Element głowy `<head>`
 - nazywany elementem zagnieżdżony, ponieważ jest ujęty pomiędzy znacznikiem początkowym a końcowym elementu `<html>`.
- Element tytułu `<title>`
 - również jest elementem zagnieżdżonym, ponieważ jest ujęty pomiędzy znacznikiem początkowym a końcowym elementu `<head>`.
 - opisuje stronę internetową.
 - tytuły zazwyczaj pojawiają się:
 - na pasku tytułowym na górze okna przeglądarki,
 - na karcie przeglądarki, na której strona jest wyświetlona,
 - jako tekst identyfikujący stronę, kiedy użytkownik dodaje ją do listy ulubionych.
 - Wyszukiwarki używają tytułu do indeksowania oraz podczas wyświetlania wyników.
- Element paragrafu `p`
 - Cały tekst umieszczony pomiędzy `<p>` i `</p>` stanowi jeden akapit.

Walidatory HTML5

- Dokumenty HTML5, które są składniowo poprawne, będą wyświetlane prawidłowo.
- Dokumenty HTML5, w których są błędy składni, mogą nie być wyświetlane prawidłowo.
 - Przeglądarki starają się domyślać, gdzie jest błąd (np. domykanie znacznika) i wyświetlić ostatecznie stronę, ale każda może inaczej naprawiać błędy
- Warto używać serwisów walidujących (np. `validator.w3.org/#validate-by-upload`) albo wtyczek i środowisk programistycznych, które posiadają wbudowane walidatory.

Nagłówki

- HTML5 oferuje sześć nagłówków (od `<h1>` do `<h6>`) do określenia względnej istotności informacji
- Element nagłówkowy `<h1>` jest uważany za **najistotniejszy** nagłówek i jest renderowany przy użyciu **największej** czcionki.
- Każdy kolejny element nagłówkowy (tj. `<h2>`, `<h3>`, itp.) jest renderowany **coraz mniejszą** czcionką
- Za pomocą atrybutów lub CSS **można to zmienić**, ale powyższa zależność powinna być zachowana.
- Podpowiedź do VS Code: wpisując ciąg „`lorem`”`<TAB>` otrzymujemy dłuższy tekst w łacinie, który służy za dobry wypełniacz te(k)stowy

Nagłówki <h1> do <h6> - przykład użycia

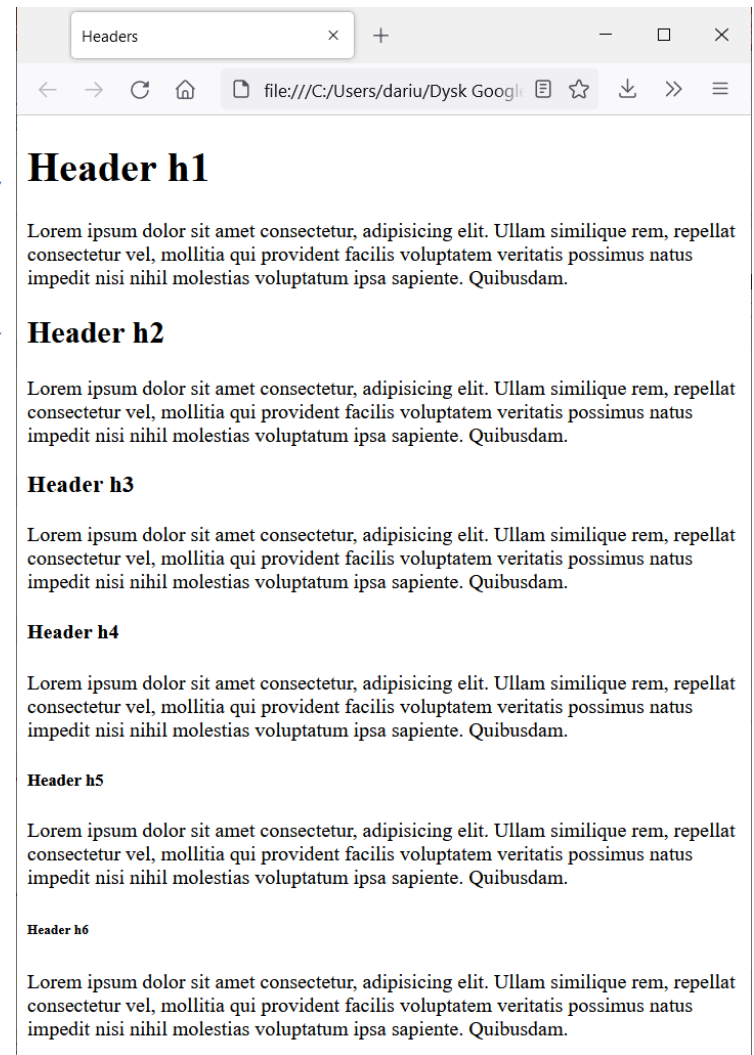
Zawieranie się znaczników dla pozycji kursora

<> Headers.html ✕

<> Headers.html > html > body > p

```
1  <!DOCTYPE html>
2  <!-- Header from h1 to h6-->
3  <html>
4  <head>
5    <meta charset="UTF-8">
6    <title>Headers</title>
7  </head>
8  <body>
9    <h1>Header h1</h1>
10   <p>Lorem ipsum dolor sit amet consectetur, adipisicing
11   <h2>Header h2</h2>
12   <p>Lorem ipsum dolor sit amet consectetur, adipisicing
13   <h3>Header h3</h3>
14   <p>Lorem ipsum dolor sit amet consectetur, adipisicing
15   <h4>Header h4</h4>
16   <p>Lorem ipsum dolor sit amet consectetur, adipisicing
17   <h5>Header h5</h5>
18   <p>Lorem ipsum dolor sit amet consectetur, adipisicing
19   <h6>Header h6</h6>
20   <p>Lorem ipsum dolor sit amet consectetur, adipisicing
21 </body>
22 </html>
```

Kursor



Znacznik <a> - Linkowanie

- Hipertłącza odsyłają bądź linkują do innych zasobów, takich jak dokumenty HTML5 lub obrazki.
 - Przeglądarki internetowe zazwyczaj podkreślają tekst hipertłącza i kolorują go na niebiesko (jeśli nie był jeszcze odwiedzany) lub fioletowo (gdy był odwiedzany).
- Znacznik <a> (ang. anchor — kotwica) zawiera najczęściej atrybut `href` (hypertext reference), który **określa lokalizację zasobu**, np.
 - stronę internetową lub lokalizację na stronie internetowej
 - plik
 - adres e-mail
- **Wnętrze** znacznika to **tekst**, który **pojawi się** w przeglądarce i **którego kliknięcie** spowoduje uruchomienie **link-u**.
- Kiedy URL w tym atrybucie nie wskazuje na żaden określony dokument na stronie, serwer WWW zwraca domyślną stronę. Zazwyczaj jest to strona `index.html` (lub `index.php`, `index.asp`), jednak serwer WWW może być tak skonfigurowany, aby zwracać inny plik/strumień jako domyślną stronę.
- Kotwice mogą linkować **do adresu** e-mail
 - za pomocą ` ... `
 - Kliknięcie takiego linku spowoduje uruchomienie domyślnego klienta poczty z uzupełnionym polem adresata maila.
- Jeśli chcemy, aby nowa strona otwierała się **w nowej zakładce** należy dodać atrybut `target="_blank"`.
- Inne możliwości kotwic pojawią się później.

Linkowanie – przykład użycia

The image illustrates the process of linking from a local HTML file to a website. It consists of four main components:

- Code Editor (Left):** Shows the HTML code for `Links.html`. The code includes a DOCTYPE declaration, a meta charset of "UTF-8", a title "Links", and a body with a heading "My Links" and a paragraph "Interesting links for garden:". The paragraph contains three links: "Zielony ogródek (Green garden)", "Fajny ogródek (Nice garden)", and "Poradnik ogrodnicy (A gardening guide)", followed by the author "Dariusz Konieczny".
- Browser Preview (Top Right):** Shows the rendered HTML page in a browser window. The page title is "My Links". The content includes the heading "My Links", the paragraph "Interesting links for garden:", and the three links: "Zielony ogródek (Green garden)", "Fajny ogródek (Nice garden)", and "Poradnik ogrodnicy (A gardening guide)", followed by the author "Dariusz Konieczny".
- Email Client (Bottom Left):** Shows an email being sent from the "Skrzynka odbiorcza — Wp" (Inbox) to "dariusz.konieczny@pwr.edu.pl". The email content is the same as the HTML code, including the heading "My Links" and the paragraph "Interesting links for garden:". The email is sent from the "Poczta" application for the Windows system.
- Target Website (Bottom Right):** Shows the website "Zielony Ogródek" (Green Garden). The website has a green header with "PORADNIK" and "SKLEP" buttons. The main content area shows a large image of a garden and a section titled "NAWÓŻENIE" (Fertilization) with the subtitle "GNOJÓWKA Z POKRZYWY: JAK PRZYGOTOWAĆ I STOSOWAĆ?" (Compost from weeds: how to prepare and use?).

Znacznik `` - obrazki

- Najpopularniejszym formatem obrazków używanym w dzisiejszych czasach przez projektantów stron WWW jest PNG (Portable Network Graphics), JPEG (Joint Photographic Experts Group), GIF (Graphics Interchange Format). W zależności od przeglądarki pozostałe formaty graficzne są lub nie są obsługiwane.
- Znacznik `img` – do wstawienia obrazka
 - Jest to „pusty element”, **nie ma** znacznika **zamykającego**
 - Atrybut `src` elementu `img` określa lokalizację obrazka
 - Atrybuty `width` i `height` są opcjonalne, określają szerokość i wysokość obrazka w pikselach
 - Jeśli zostaną pominięte, przeglądarka użyje rzeczywistą szerokość i wysokość obrazka
 - Atrybut `title` określa tekst, który pojawi się po najechaniu wskaźnikiem myszki nad obraz.

Przykład - kod pliku Images.html

<> Images.html X

<> Images.html > ...




```
1  <!DOCTYPE html>
2  <!-- Images-->
3  <html>
4  <head>
5      <meta charset="UTF-8">
6      <title>Images</title>
7  </head>
8  <body>
9      <h1>My images</h1>
10     <p> incorrect ratio:
11         
12         correct ratio:
13         
14         another image:
15         
16         non-existent image:
17         
18     </p>
19     <p>
20         outside source:
21         
22     </p>
23     <p>
24         Original size:
25     </p>
26     <p>
27         
28     </p>
29 </body>
30 </html>
```

Przykład - działanie

Images


file:///C:/Users/dariu/Dysk Google/Dydaktyka/2021-2022/semestr zimowy 2021_22/Aplikacje webowe na platformę .NET/Wykl

My images

incorrect ratio:  correct ratio:  another image:  non-existent image: rose's flowers

outside source: **allegro**

Original size:



Wskazówki do elementu ``

- Wskazane jest zawsze używać atrybutów `width` i `height` w elemencie ``, dzięki czemu przeglądarka, wczytując plik HTML5, będzie wiedziała, **ile miejsca** na ekranie przeznaczyć na obrazek. **Pozwoli** to przeglądarce **rozmieścić** pozostałe elementy na stronie poprawnie, nawet **zanim obrazek** zostanie **wczytany**. Umożliwi to **przyspieszenie renderowania** i **wyświetlania** strony.
- Wpisanie **rozmiarów** obrazka, które **zmieniają** jego **proporcje** (szerokości do wysokości) często **pogarsza jakość** obrazka. Najlepiej stosować te same (lub zbliżone) proporcje szerokości do wysokości w tych atrybutach w stosunku do oryginalnych wymiarów obrazka.
- Każdy element `` w dokumencie HTML5 powinien posiadać atrybut `alt`.
- Przeglądarka może nie móc wyrenderować obrazka (chwilowo niedostępny, niepoprawny format). Wówczas wyświetli ona wartość atrybutu `alt`.
- Atrybut `alt` jest także ważny dla dostępności — synteza mowy dla niewidomych czytają wartość atrybutu `alt`. Z tego powodu atrybut `alt` powinien opisywać zawartość obrazka z istotnymi dla zrozumienia treści szczegółami.

Obrazek jako link

- Wstawiając obrazek do wnętrza elementu `<a>` otrzymamy **klikowalny obrazek** linkujący do wybranego miejsca.
 - Oprócz obrazka może być użyty jednocześnie tekst
 - Jeśli jako link jest tylko obrazek, w celu zapewnienia dostępności dla osób niepełnosprawnych, powinien być ustawiony atrybut `alt`. **W przeciwnym wypadku** nie jest to potrzebne (a nawet raczej niewskazane).

Obrazek jako link – przykład użycia (ImagesAsLinks.html)

```
<> ImagesAsLinks.html X
<> ImagesAsLinks.html > ...
1  <!DOCTYPE html>
2  <!-- Images as links-->
3  <html>
4  <head>
5      <meta charset="UTF-8">
6      <title>Images as links</title>
7  </head>
8  <body>
9      <h1>Links with images</h1>
10     <p>Only image as link to allegro website:
11         <a href="https://allegro.pl/">
12             </a>
13             <!-- </a> in this line will create clickable SPACE character. -->
14             A text after anchor.
15         </p>
16     <p>A image and text as link to allegro website:
17         <a href="https://allegro.pl/">
18             
19             jump to Allegro.</a>
20             A text after anchor.
21         </p>
22     </body>
23     </html>
```

