

# **Programowanie Aplikacji Internetowych**

Podstawy

# Plan na dziś

- HTML
- CSS
- DNS
- HTTP
- REST

# HTML

- Co warto wiedzieć?

# HTML

Skorzystajmy Najpierw z Web Developer Tools w Firefoxie



# HTML

- Tim Berners-Lee
- HTML5
- [podstawowa struktura](#)

# HTML

- [tagi otwierające i zamykające](#)
- [atrybuty](#)
- `<br />`

# HTML

- `<a href="https://...">link do </a>`
- `<img src="" />`
- `<p>123</p>`
- `<h2></h2>`

# HTML

Listy:

- `<ul>` oraz `<ol>`
- `<li>`



# HTML

## Kontenery:

- `<div>` - kontener
- `<span>` - inline

# HTML

- `<script>`
- [forms](#)

# HTML - Praktyka

- [Responsive design](#),
- [Responsive typography](#),
- [media queries](#).

# HTML - Praktyka

Jeśli nie mamy możliwości budowy frontendu:

- prezentacje z [revealjs](#),
- blog na wordpress lub na alternatywnej platformie,
- Warto skorzystać z gotowych komponentów, np., [bootstrap](#).

# CSS

- CSS - Cascading Style Sheets
- [przykład](#)

# CSS - Reguły

```
selektor { cecha: wartość; }
```

- selektor - dowolny znacznik, np. p (akapit), h1 (nagłówek), li (lista)
- cecha - właściwość stylu dla znacznika, np. kolor (color)
- wartość - opis cechy np. kolor czcionki (color) czerwony (red)

```
p { color: red; }
```

# CSS - Reguły

```
h1 p { color: red; }
```

element `p` w `h1`.

# CSS - Reguły

```
p,li,h1 { color: red; }
```

wszystkie wspomniane elementy czerwone



# CSS - Reguły

Po id:

```
#intro { color: red; }
```

```
<p id="intro">czerwony</p>
```

# CSS - Reguły

Po klasach:

```
<h2 class="content">Nagłówek ma klasę content.</h2>  
<p class="content">Ten paragraf oznaczyłem za pomocą klasy content. </p>
```

```
.content {  
color: #00f;  
font-weight: bold;  
font-style: italic; }
```

# CSS - w dokumencie

```
<head>
<style>
h1 {
  color: blue;
}
</style>
</head>
```

# CSS - zewnętrzny

```
<head>  
<link rel="stylesheet" href="style.css" type="text/css" />  
</head>
```

# CSS

- kaskady reguł
- wszystkie + ostatnia nadpisuje poprzednie lub bardziej precyzyjne

# CSS

Zacznij od korzystania z gotowych komponentów:

1. [bootstrap](#)
2. ... ([more](#))
3. [materializecss](#)
4. [ant.design](#)
5. [tailwindcss](#)

# Selektory: xpath, css

Wróćmy do naszego Web Developer Tools.

- Copy -> CSS selector,
- Copy -> XPath,
- Potrzebne przy pracy z [cypress](#) czy [selenium](#).

# DNS

DNS - Domain Name Server

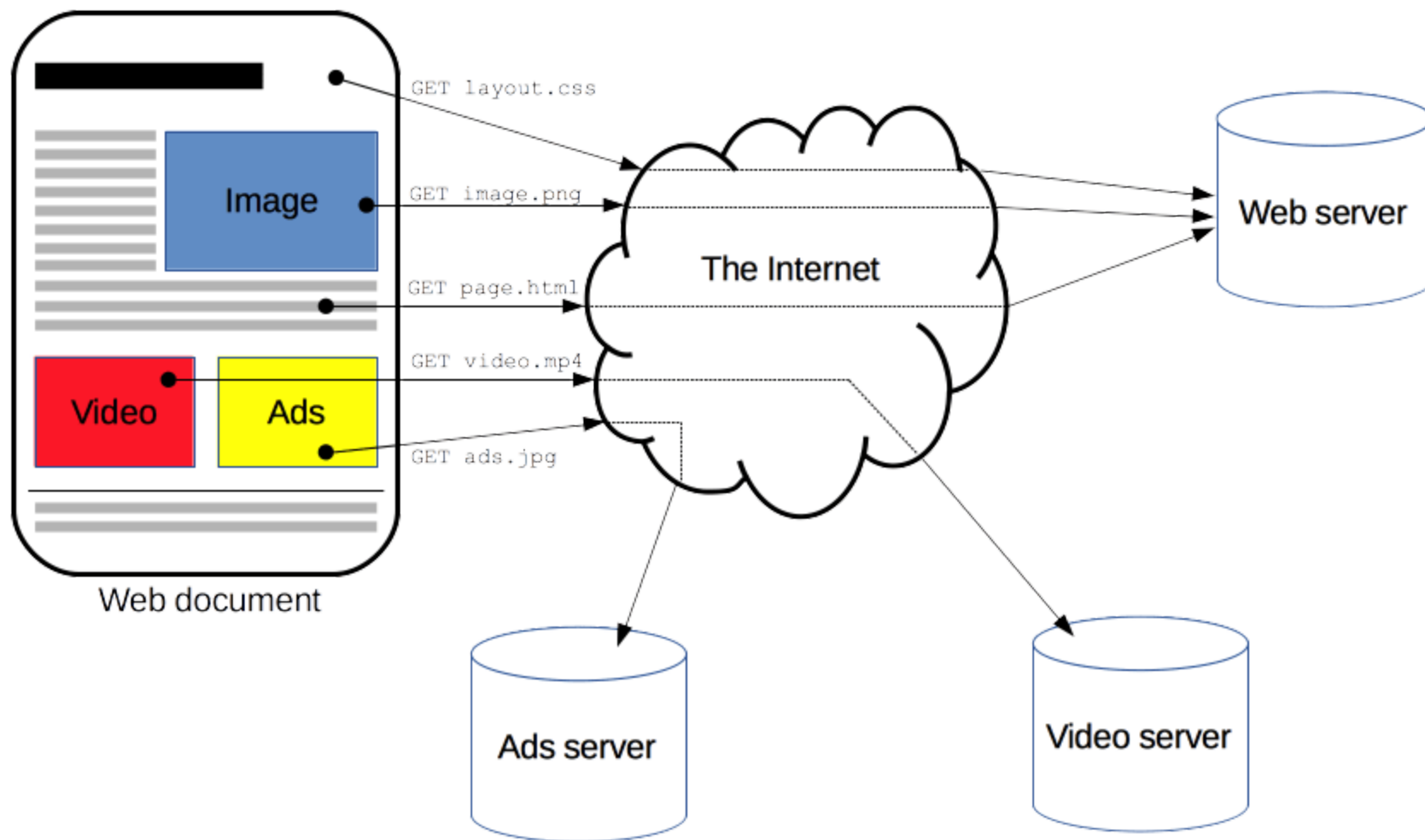
```
dig google.com
```



# DNS

Sprawdźmy kto posiada domenę  
`wsb.pl` na <https://www.dns.pl/whois>

# HTTP

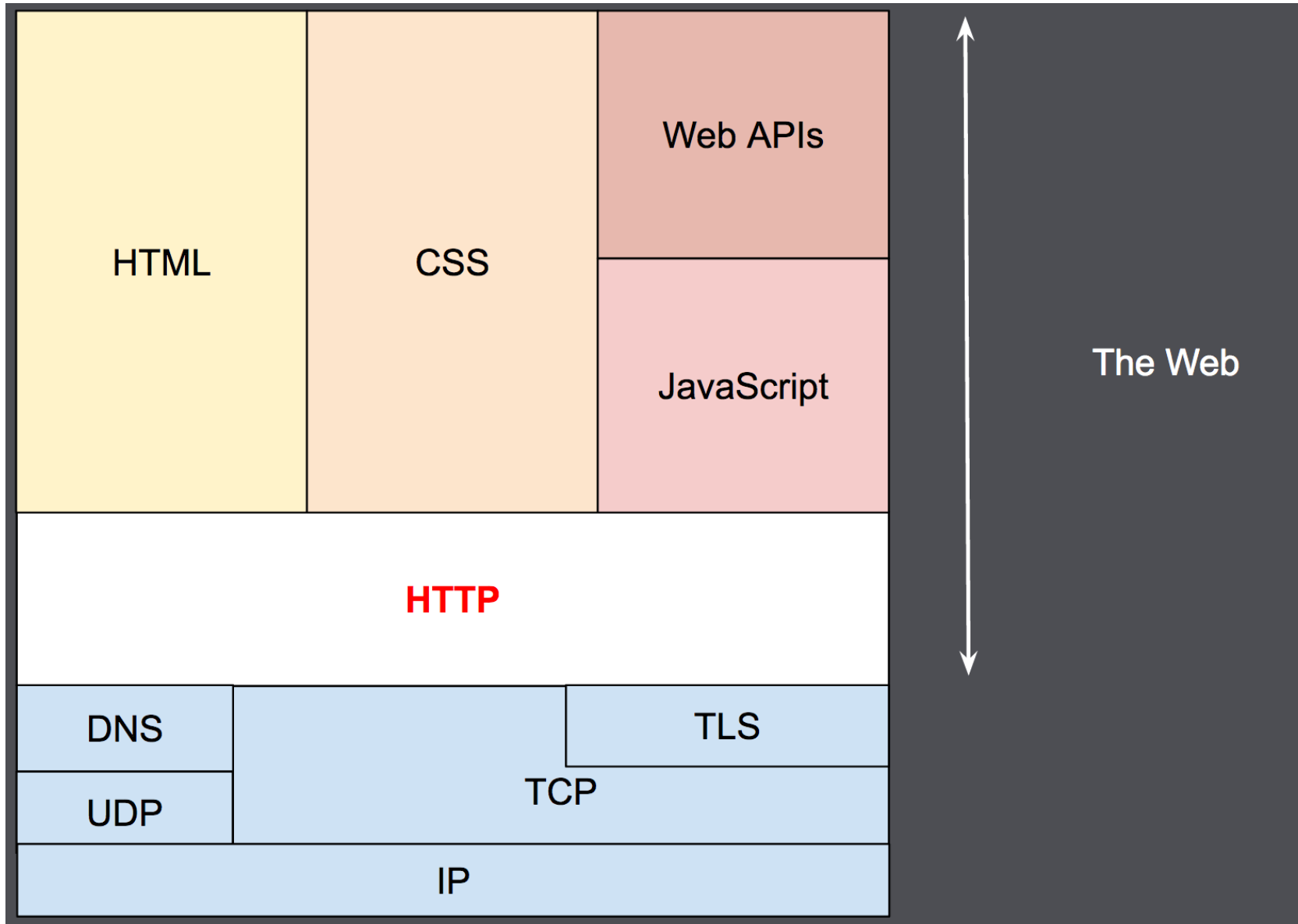


# HTTP

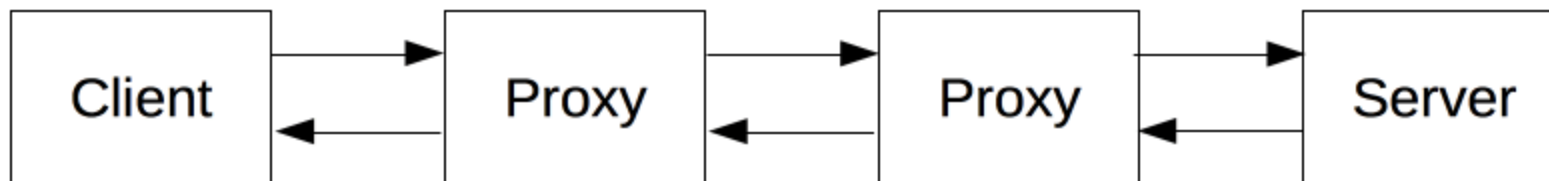
## Podstawy:

- HTTP is simple
- HTTP is extensible
- HTTP is stateless (but not sessionless)
- HTTP relies on TCP (connection based)

# HTTP



# HTTP



# HTTP

Demo:

```
curl -I www.google.com
```

```
curl -I -L google.com
```

# HTTP - methods

Methods:

- GET
- POST
- DELETE

# HTTP - methods

Demo:

```
http POST https://httpbin.org/post "name"="natalia"
```



# HTTP - status code

Status code:

- 5xx: 500, 502
- 4xx: 404, 400, 401
- 3xx: 301, 302
- 2xx: 200, 201, 02

# HTTP - status code

Demo:

```
curl -I -X GET https://httpbin.org/status/404 --fail
```

```
curl -I -X GET https://httpbin.org/status/200 --fail
```

# **A co z serwisami?**

- +/- Wiemy jak działają przeglądarki
- co z web API?

# **A co z serwisami?**

- +/- Wiemy jak działają przeglądarki
- co z web API?

# Protokoły

Najpopularniejsze:

- RPC
- REST / REST-like
- GraphQL

**Dziękuję za uwagę**