Aplikacje internetowe
HTML, CSS + rozszerzenia

dr inż. Piotr Grochowalski

Sieć WWW

- **Sieć WWW** (*World Wide Web*) jest systemem hipertekstowym, bedącym światowa „pajęczyna” odnośników łączących zasoby zgromadzone na serwerach rozlokowanych na całym świecie.
- **Hipertekst** to organizacja danych w postaci dokumentów powiązanych ze sobą hiperłączami (odnośnikami, odsyłaczami).
- Siec WWW działa w oparciu o protokół HTTP (*HyperText Transfer Protocol*). Za pomocą tego protokołu programy-klienci (przeglądarki WWW) komunikuja sie z serwerami WWW w celu pobrania do wyświetlenia dokumentów (stron) WWW - **model klient-serwer**.
- Dokumenty udostępniane w sieci WWW tworzone są przede wszystkim w języku **HTML** (*HyperText Markup Language*). Są to pliki tekstowe, zawierające specjalnie zdefiniowane grupy znaczników.

Model klient – serwer

- **Klient** jest systemem, który zleca określone zadanie systemowi zwanemu serwerem.
- **Serwer** na zadanie klienta wykonuje określone usługi.

Model klient – serwer

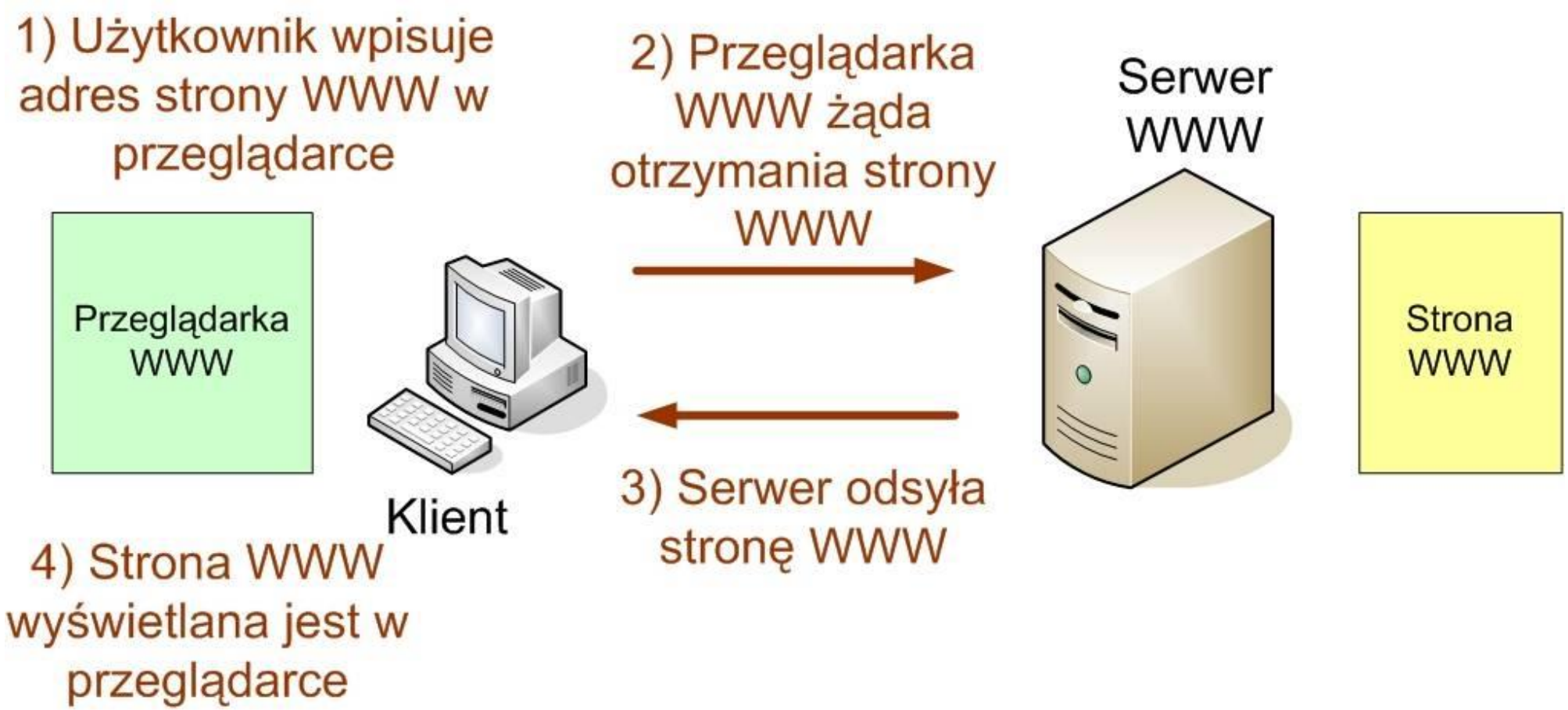
Serwer

- To komputer w sieci, z którego funkcji korzysta wielu użytkowników sieci.
- To oprogramowanie sieciowe realizujące określone zadania, np.: serwer WWW, serwer poczty elektronicznej.

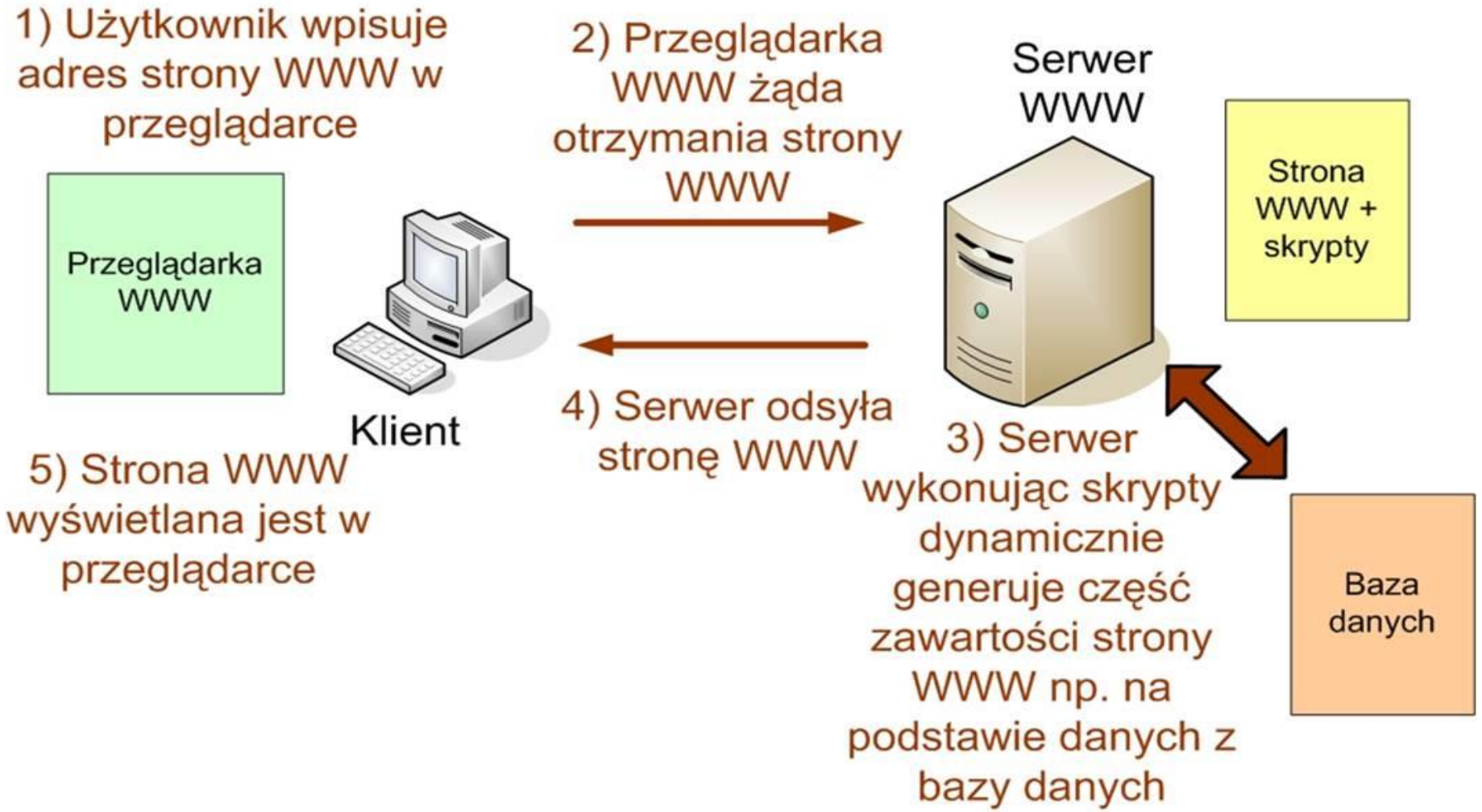
Klient

- *Cienki klient* – typ klienta, w przypadku którego zadania realizowane są głównie na serwerze, a rola użytkownika polega przede wszystkim na przekazywaniu zadań do serwera za pomocą dedykowanej aplikacji, np. przeglądarki internetowej.
- *Gruby klient* – typ klienta, w przypadku którego realizacja zadań odbywa się częściowo na komputerze użytkownika (klienta), a częściowo na serwerze.

Strony www - statyczne



Strony www - dynamiczne

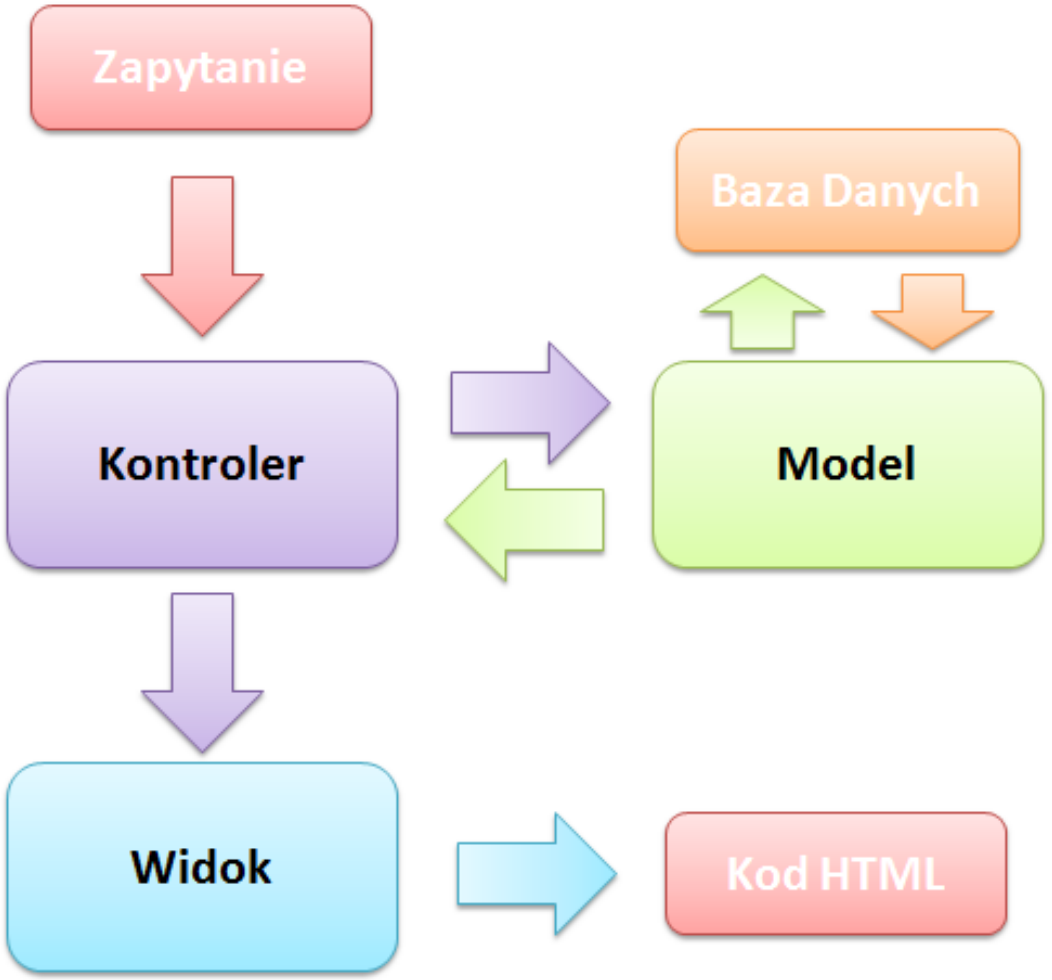


Architektura (wzorzec) MVC
(ang. Model View Controller)

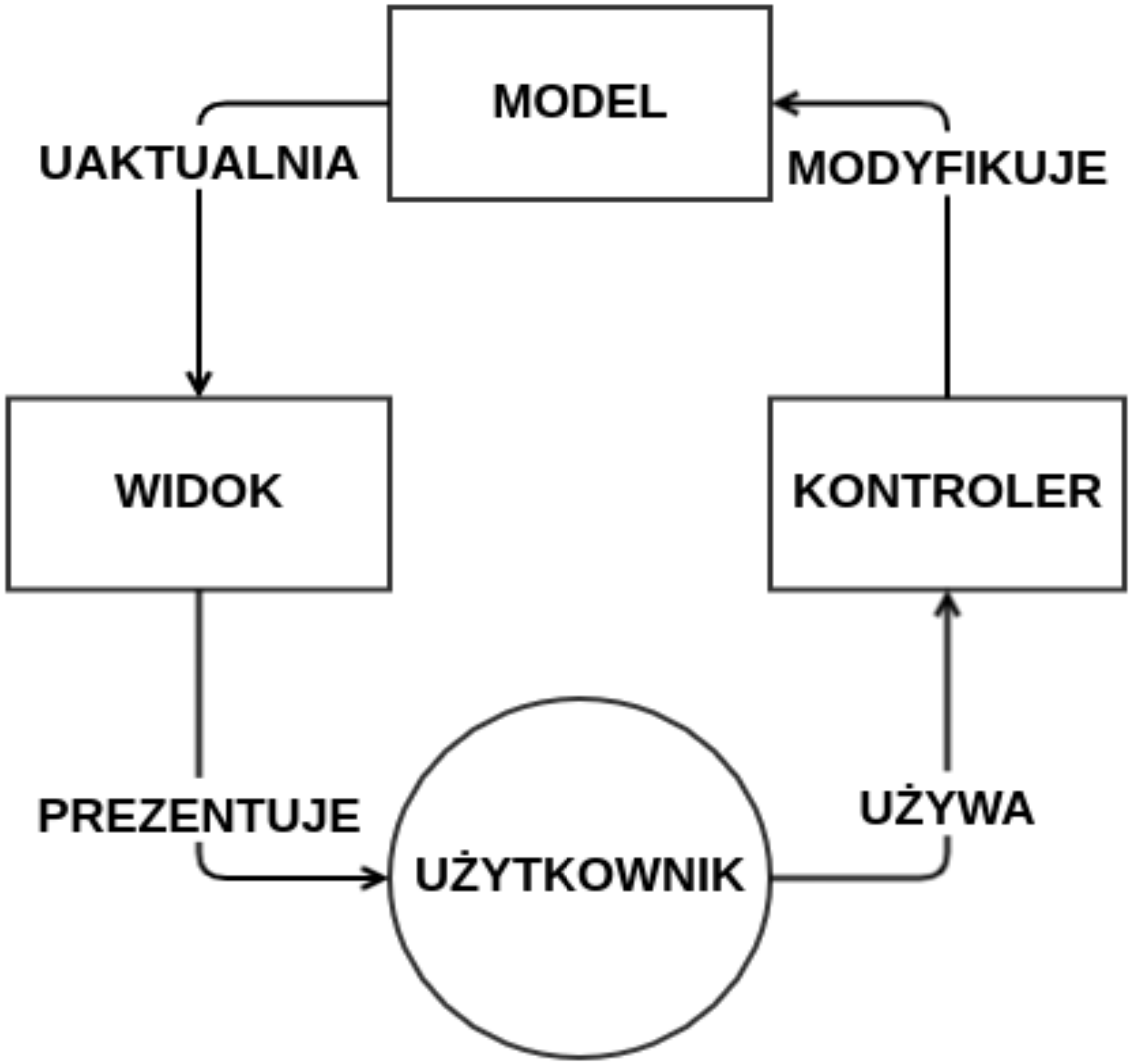
Główne zadanie to oddzielenie interfejsu użytkownika od warstwy logiki biznesowej. Dzięki temu możliwa jest całkowita wymiana interfejsu bez większego wpływu na zaimplementowany model biznesowy systemu.

- ***Model*** - zawiera logikę biznesowa aplikacji, realizuje wszystkie funkcjonalności systemu, zarządza zdarzeniami systemu, korzysta z bazy danych, dokonuje obliczeń.
- ***Widok*** - jest interfejsem łączącym aplikacje z jej klientem, wyświetla wygenerowane przez serwer informacje, a także pobiera dane od użytkownika.
- ***Kontroler*** - łączy ze sobą model i widok, pobiera dane przekazane przez klienta aplikacji, po czym na ich podstawie uruchamia odpowiedni kod modelu, a otrzymany wynik ponownie przekierowuje do klienta.

Architektura MVC
(źródło: mcwolf.net)



Architektura MVC
(źródło: Wikipedia.pl)



Architektura REST API

(ang. Representational state transfer application programming interface)

To aplikacyjny interfejs programistyczny używany jako architektura przesyłania wiadomości i danych dla architektur klient-serwer i mikrousług.

Koncepcje REST API zostały opracowane dla sieci WWW i stanowiły podstawę jej rozwoju w chmurze, Internecie mobilnym i Internecie rzeczy (IoT).

API to zestaw reguł opisujących, jak jeden program może się łączyć oraz komunikować z innym. API REST przekazuje na każde żądanie stan każdej transakcji, co daje korzyści związane z opracowaniem, wydajnością i zasobami w porównaniu do innych metod.

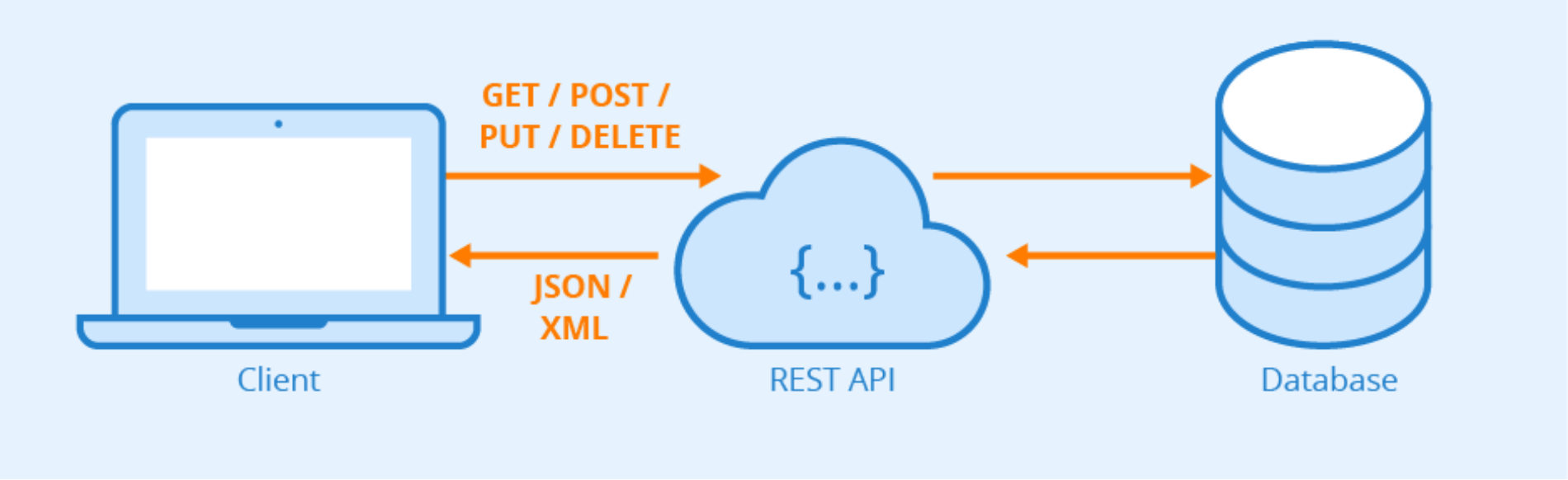
API REST

Zasób jest podstawowym pojęciem dla API REST. Zasób jest obiektem, który ma typ, powiązane dane, relacje z innymi zasobami i zestaw metod, które na nim działają.

Jest bardzo podobny do idei obiektów w programowaniu, chociaż zdefiniowanych jest tylko kilka standardowych metod, typowych dla HTTP: **GET, POST, PUT i DELETE**. Zasoby mogą istnieć same lub w zbiorach, które same są zasobami.

- **GET**
  - Pobiera określony zasobu według podanego identyfikatora.
- **POST**
  - Tworzy nowy zasób.
  - Jest wykorzystywany do wszystkich innych operacji, które nie wpisują się w ramy innych metod. Może służyć do pobierania danych w przypadku kiedy musimy w ramach body dostarczyć dodatkowe parametry.
- **PUT**
  - Aktualizuje dany zasób na podstawie podanego identyfikatora.
  - Może służyć do tworzenia nowego zasobu jeśli jego identyfikator jest znany.
- **DELETE**
  - Usuwa określony zasób według identyfikatora.

API REST
(źródło: seobility.net)



API REST

Reguły (wytyczne) do projektowania

Aby stać się API REST, API musi spełniać sześć reguł zwanych ograniczeniami architektonicznymi lub zasadami projektowania.

1. Jednolity interfejs

Wszystkie zapytania kierowane przez API REST **muszą spełniać reguły formatowania określone przez API**. Niezależnie od tego, jaki klient wysyła zapytanie, musi on przesłać każdą informację tam, gdzie zostawiłby ją każdy inny klient. Przykładem może być adres URL używany do identyfikacji zasobów za pośrednictwem protokołu HTTP.

2. Rozdzielenie klient-serwer

Interfejsy API REST wymagają, aby **aplikacje klienckie i serwerowe były całkowicie niezależne od siebie**. Klient powinien znać jedynie pełną nazwę zasobu, którego potrzebuje w przestrzeni wirtualnej dozwolonej przez API. W przeciwnym razie, jedyną wiedzą, jaką klient i serwer dysponują o sobie nawzajem są dane przesyłane przez transakcje API.

3. Bezstanowość

Każde żądanie klienta musi zawierać wszystkie informacje niezbędne do jego przetworzenia, a serwer nie musi przechowywać żadnych informacji o tym żądaniu po jego otrzymaniu. W projekcie API REST nie ma pojęcia sesji, a serwer jest bezstanowy w odniesieniu do konkretnego klienta.

API REST

Reguły (wytyczne) do projektowania

4. Buforowalność

W przeciwieństwie do bezstanowości serwera w odniesieniu do klienta, **zasoby powinny być dostępne w pamięci podręcznej w jednym lub kilku punktach wewnątrz lub między klientem a serwerem**. Jeśli dany zasób został obsłużony i istnieje prawdopodobieństwo, że po pewnym czasie zostanie ponownie uruchomiony, należy umieścić go w pamięci podręcznej w celu uzyskania szybszej odpowiedzi. Klient powinien podjąć podobną decyzję w sprawie otrzymanych zasobów. Serwer powinien poprzez API wskazać, czy zasób może być bezpiecznie umieszczony w pamięci podręcznej lokalnie u klienta.

5. Warstwowa architektura systemu

Konsekwencją rozdzielenia klient-serwer, bezstanowości i buforowalności jest to, **że klient nie może przyjmować żadnych założeń co do tego, czy komunikuje się bezpośrednio z serwerem posiadającym określony zasób, czy też jest on obsługiwany przez pośrednika** np.: broker usług, system dostarczania treści lub inny podsystem, znajdujący się bliżej klienta niż serwer. Zapewnia to projektantom systemów i infrastruktur dużą elastyczność, która pozwala maksymalizować wydajność i niezawodność odpowiedzi na zapytania w globalnej infrastrukturze przewodowej i bezprzewodowej.

6. Kod na żądanie

Chociaż API REST może udostępniać tylko dane do wykorzystania przez klienta i często tak robi, coraz częściej zdarza się, że **do klienta dostarczany jest kod**, na przykład obiekty Java lub aplikacje internetowe Javascript. W takim przypadku taki **kod może być uruchomiony wyłącznie na żądanie klienta**.

HTML
Kalendarium

- 1991r. - początek, pierwsza specyfikacja (Tim Berners-Lee)
- 1993 - 1995r. - HTML 2.0/3.0 (kwiecień) (IETF: HTML Working Group)
- 14.01.1997r. - HTML 3.2 (W3C: World Wide Web Consortium)
- 18.12.1997r. - HTML 4.0 (W3C: World Wide Web Consortium)
- 24.04.1998r. - HTML 4.0 z poprawkami (W3C: World Wide Web Consortium)
- 24.12.1999r. - HTML 4.01 (W3C: World Wide Web Consortium)
- 15.05.2000r. - międzynarodowy standard (ISO/IEC 15445:2000)
- 22.01.2008r. - HTML5 (W3C: World Wide Web Consortium) - szkic
- 28.10.2014r. - HTML5 oficjalne wydanie standardu (W3C)
- 1.11.2016r. - HTML 5.1, rozpoczęcie prac nad wersja 5.2
- 12.12.2017r. - HTML 5.2, rozpoczęcie prac nad wersja 5.3
- 18.10.2018r. - HTML 5.3 - szkic

HTML

Ważne wydarzenia - zwroty

- Wprowadzenie JavaScript (twórcy: Netscape) - brak uwzględnienia w standardzie.
- Rywalizacja przeglądarek: początkowo pod katem wzbogacania o unikalne funkcje Microsoft (Internet Expolorer) i Netscape (Netscape Navigator), teraz pod katem zgodności ze standardami.
- Dominacja wtyczek: głównie Adobe Flash.
- Semantyczny HTML.
- Standard HTML nie nadąża za sposobami użycia HTML.

HTML5, CSS3, JavaScript

HTML5 - to najnowsza wersja specyfikacji HTML, ale jest to również ogólne określenie zbioru powiązanych technologii, służących do tworzenia nowoczesnych zasobów internetowych.

Do których należą:

- **HTML5** (bazowa specyfikacja) (*ang.* HyperText Markup Language) - określa elementy, którymi oznacza się treść i określa jej znaczenie w dokumentach udostępnianych w sieci Internet.
- **CSS3** (*ang.* Cascading Style Sheets) - kaskadowe arkusze stylów pozwalają określać wygląd oznaczonych treści, gdy są one prezentowane użytkownikowi.
- **JavaScript** - jest to język skryptowy działający po stronie klienta (użytkownika) i np.: pozwala na modyfikowanie zawartości dokumentu HTML, reagowanie na działania użytkownika, itp.

HTML5

Nowe funkcjonalności

- **Natywna obsługa multimediów** - natywne tj. bez konieczności użycia wtyczek odtwarzanie plików audio i video w przeglądarce.
- **Zasoby programistyczne** - przykładowo element canvas (tworzy obszar graficzny, który można wykorzystywać do wykonywania zadań, które do tej pory wymagały Adobe Flash). Zwiększenie roli programowania poprzez większą integracje z JavaScript.
- **Sieć semantyczna** - wprowadzono szereg funkcji i zasad służących do oddzielenia znaczenia elementów od tego, jak treść jest prezentowana.

HTML5
Stan bieżący

- Bazowy standard został opracowywany, tworzone są kolejne jego modyfikacje poprawiające występujące usterki w pierwotnej specyfikacji.
- Wiele funkcji HTML5 już jest obsługiwanych przez przeglądarki internetowe, ale nadal nie jest to 100%. Dlatego warto wykorzystywać dodatkowe narzędzia np.:
  - biblioteki pozwalające na automatyczne sprawdzenie, czy przeglądarka uzytkownika obsługuje odpowiednie funkcje HTML5 = możliwość dopasowania aplikacji np.: Modernizr (*http://modernizr.com*),
  - witryny pozwalające zweryfikować jak aktualne wersje przeglądarek wspierają HTML5 i jak szybko następuje wdrażanie nowych funkcjonalności np.: *http://html5test.com*, *http://caniuse.com*.

HTML5

- HTML – język znaczników (tagów).
- Znaczniki przyjmują formę elementów, które odnoszą się do treści (na ogół tekstu).

Przykład

```
<p align=''center''>Przykładowy tekst</p>
```

- Element HTML: `<p align=''center''>Przykładowy tekst</p>`
- Znacznik: otwierający `<p>`, zamykający `</p>`
- Atrybut elementu: `align=''center''`, tj. nazwa atrybutu `align`, wartość atrybutu `center`
- Treść: `Przykładowy tekst`

Typy elementów

Specyfikacja HTML5 określa trzy kategorie elementów:

- **metadane** - dostarczają przeglądarce informacji i wskazówek jak dokument należy przetworzyć,
- **strukturalne** - służą do tworzenia podstawowej struktury dokumentu HTML,
- **treściowe** - służą do tworzenia zawartości dokumentu HTML.

Atrybuty w HTML5

Rodzaje atrybutów

- **Lokalne** - są to atrybuty właściwe dla każdego elementu. Każdy atrybut lokalny daje możliwość kontrolowania jakiegoś wyjątkowego aspektu funkcjonowania elementu np.: alt, href.
- **Globalne** - określają zachowanie wspólne dla wszystkich elementów. Każdemu elementowi można przypisać każdy atrybut globalny np.: accesskey, class, *contenteditable*, *contextmenu*, dir, *draggable*, *dropzone*, hidden, id, style, title.

{*kursywa*} - wprowadzone w HTML5.

Przykładowa struktura dokumentu HTML

examples/html1.htm

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>Przyklad</title>
5   </head>
6   <body>
7     <p align="center">Przykladowy tekst</p>
8   </body>
9 </html>
```

Rodzice, dzieci, potomkowie i bracia

Elementy HTML w dokumencie HTML są miedzy sobą w określonych relacjach.

- Element zawierający inny element jest jego **rodzicem**. Element body jest rodzicem dla elementu p.
- Analogicznie w druga stronę element p jest **dzieckiem** elementu body. Element może mieć wiele dzieci, ale tylko jednego rodzica.
- Elementy mogą zawierać elementy, które również zawierają elementy - element html zawiera element body, który z kolei zawiera element p. Elementy body i p są **potomkami** elementu html, ale jedynie element body jest dzieckiem elementu html. Dzieci są bezpośrednimi potomkami.
- Elementy o wspólnym rodzicu to **bracia**. Elementy head i body są braćmi, ponieważ obydwa są dziećmi elementu html.

Rodzice, dzieci, potomkowie i bracia

Relacje pomiędzy obiektami w HTML są bardzo istotne, ponieważ:

- zdefiniowane zostały ograniczenia określające, jakie elementy mogą być względem siebie rodzicami - wyrażone w kategoriach typów elementów,
- wykorzystywane w CSS, do wyboru elementów do których maja być zastosowane poszczególne style, często takie wskazywanie jest wykonywane za pomocą relacji rodzic-dziecko,
- w obiektowym modelu dokumentu (DOM) drzewo dokumentu, jest odzwierciedleniem relacji miedzy elementami.

HTML
Znaczniki

- Elementy dokumentu i metadanych np.: html, head, body, meta, style;
- Elementy tekstowe np.: a, br, b, i, span, sub, sup;
- Grupowanie treści np.: div, dl, figure, ul, ol, li, p;
- Dzielenie treści na sekcje np.: article, footer, h1-h6, header, nav, section;
- Tworzenie tabel np.: table, tr, td;
- Tworzenie formularzy np.: form, input, fieldset, option, select;
- Osadzanie zasobów np.: canvas, audio, img, object, param, video.

HTML5
Nowości

- Znaczniki do zarzadzania treścia np.: header, article, footer, nav - *examples/html11.htm*.
- Element canvas (płótno) - *examples/html12.htm*.
- Przeciągnij i upuść - *examples/html13.htm*.
- Formularze - *examples/html14.htm*.
- Wykrywanie lokalizacji użytkownika (Geolocation API).
- Przechowywanie danych po stronie klienta – localStorage (możliwe przechowanie 5MB, lub czasami 10MB), sessionStorage - *examples/html16.htm i html17.htm*.
- Wykorzystanie lokalnej bazy danych SQL - *examples/html18.htm*.

Kaskadowe arkusze styli CSS

- *Kaskadowe arkusze styli CSS* służą do określania sposobu prezentacji (wyglądu i formatowania) dokumentu HTML (oficjalny standard W3C).
- Według zaleceń tego konsorcjum znaczniki HTML powinny określać tylko strukturę dokumentów hipertekstowych, natomiast ich wygląd powinien być określany za pomocą CSS.

Kaskadowe arkusze styli CSS

Styl CSS składa się z jednej lub więcej deklaracji, oddzielonych średnikami. Każda deklaracja składa się z właściwości CSS oraz wartości, po której stoi średnik.

Przykład

```
background-color: blue; color: white
```

- Styl CSS: `background-color: blue; color: white`
- Deklaracja: `background-color: blue`
- Właściwość: `background-color`
- Wartość: `blue`

Kaskadowe arkusze styli CSS

Zalety:

- większy stopień kontroli nad wyglądem strony (m.in. możliwość określania wciec, marginesów i pozycji elementów),
- oddzielenie stylu od struktury dokumentów,
- mniejszy rozmiar dokumentów,
- łatwiejsze zarzadzanie wyglądem dokumentów.

Wady:

- zróżnicowana obsługa przez przeglądarki WWW.

Stosowanie styli CSS

Sposoby „nakładania” stylów

Po zdefiniowaniu stylu należy w dokumencie HTML ”nałożyć” go na odpowiednie elementy, aby zaczął on na nie oddziaływać.

- **Styl inline** - pojedyncze elementy.
- **Styl osadzony** - pojedyncze dokumenty, wykorzystanie selektorów CSS.
- **Zewnętrzny arkusz styli** - plik *.css, cała witryna, łatwe w utrzymaniu.
- *Style przeglądarek* - domyślne style, które przeglądarka nakłada na elementy, kiedy żaden inny styl nie jest podany.
- *Style użytkowników* - przeglądarki pozwalają użytkownikom na definiowanie własnych arkuszy stylów.

CSS

Styl inline

examples/html3.htm

```
1  <!doctype html>
2  <html>
3    <head>
4      <title>Przyklad</title>
5    </head>
6    <body>
7      <p style="background-color:gray;color:red">
8        Przykladowy tekst
9      </p>
10     <p align="center">Przykladowy tekst2</p>
11   </body>
12 </html>
```

CSS

Styl osadzony

examples/html4.htm

```
1  <!doctype html>
2  <html>
3    <head>
4      <title>Przyklad</title>
5      <style type="text/css">
6        p {
7          background-color:gray;
8          color:red;
9        }
10     </style>
11   </head>
12   <body>
13     <p>Przykladowy tekst</p>
14     <p align="center">Przykladowy tekst2</p>
15   </body>
16 </html>
```

CSS

Zewnętrzny arkusz styli

examples/styles5.css

```
1 p {
2   background-color:gray;
3   color:red;
4 }
```

examples/html5.htm

```
1  <!doctype html>
2  <html>
3    <head>
4      <title>Przyklad</title>
5      <link rel="stylesheet" type="text/css" href="styles5.css
           "></link>
6    </head>
7    <body>
8      <p>Przykladowy tekst</p>
9      <p align="center">Przykladowy tekst2</p>
10   </body>
11 </html>
```

Kaskadowość stylów

1. Style inline
2. Style osadzone
3. Zewnętrzne arkusze styli
4. Style użytkownika
5. Style przeglądarki

Zmiana kolejności ważności stylów

examples/styles6.css

```
1 p {
2   background-color:gray;
3   color:red !important;
4 }
```

Dziedziczenie stylów

Jeżeli przeglądarka nie może znaleźć wartości dla właściwości w którymś z dostępnych stylów, to posłuży się **zasada dziedziczenia** tj. użyje wartości, jaka ta właściwość ma w rodzicu danego elementu.

Co jest dziedziczone?

- Dziedziczone są: właściwości, które odnoszą się do wyglądu elementu np.: kolor tekstu, szczegółów fonta, itp.
- Nie dziedziczone są: właściwości, które odnoszą się do położenia elementu na stronie - wymuszenie dziedziczenia wartość inherit.

Dziedziczenie stylów

examples/html6.htm

```
<!doctype html>
<html>
  <head>
    <title>Przyklad</title>
    <style type="text/css">
      p {
        background:yellow;
        color:red;
        border: medium solid black;
      }
      span {
        border: inherit;
      }
    </style>
  </head>
  <body>
    <p><span>Przykladowy</span> tekst</p>
    <p align="center">Przykladowy tekst2</p>
  </body>
</html>
```

CSS

Określanie stylów dla elementów

Elementy

Element w dokumencie HTML: `<div> Tresc </div>`
Opis stylu w CSS:`div { opis stylu }`

Identyfikatory

Element w dokumencie HTML: `<div id=main> Tresc </div>`
Opis stylu w CSS:`#main { opis stylu }`

Klasy

Element w dokumencie HTML: `<div class=main> Tresc </div>`
Opis stylu w CSS:`.main { opis stylu }`

CSS

Określanie stylów dla elementów

Pseudo-klasy

Element w dokumencie HTML: `<div> Tresc </div>`
Opis stylu w CSS:`div:hover { opis stylu }`
CSS3 wprowadza nowe pseudoklasy: `:root` (korzeń), `:nth-child` (n-te dziecko), `:checked` (zaznaczony), `:invalid` (nieaktywny), `not()` (nie), itp.
Przykład: `p:nth-child(3) {color: red}`, `input:checked { opis stylu }`

Atrybuty

Element w dokumencie HTML: `<input type=text value=> Tresc`
Opis stylu w CSS:`input[type=text] { opis stylu }`
CSS3 wprowadza szersze możliwości dopasowywania atrybutów do reguły na podstawie wyrażeń: $\wedge =$ (początek łańcucha znaków). $+ (koniec łańcuch znaków), $* =$ (zawieranie w łańcuchu).
Selektory te pozwalają na uzależnienie stylu od fragmentu zawartości atrybutu, nie od zawartości tekstu dokumentu.

CSS

Określanie stylów dla elementów

Pseudo-elementy

Różnica pomiędzy pseudo-klasami a pseudo-elementami: Pseudo-klasa nie zmienia samego elementu, dostarcza tylko dodatkową właściwość (klasę), która pozwala na określenie stylu całego elementu (dokumentu). Pseudo-element dotyczy jedynie części elementu, a nie jego całości.

Element w dokumencie HTML: `<p> Tresc </p>`

Opis stylu w CSS:`p:first-letter { opis stylu }`

Frameworki CSS

Bootstrap

Darmowy, bardzo popularny framework CSS i Javascript, wspierający tworzenie responsywnych interfejsów webowych.

Bootstrap realizuje filozofie **najpierw mobilne**, czyli interfejs powstały przy jego użyciu ma się poprawnie wyświetlać i funkcjonować na najmniejszym dostępnym rozmiarze ekranu.

- Dostępność: https://getbootstrap.com/
- Twórcy: Mark Otto, Jacob Thorton (pracownicy Twiterra).
- Udostępniony: 2011r.
- Składniki: CSS (wykorzystanie preprocesora Sass), JavaScript.

Frameworki CSS

Bootstrap

Charakterystyka:

- Nadpisuje domyślne style elementów w różnych przeglądarkach, przez co zawsze te elementy wyglądają tak samo.
- Daje wsparcie dla responsywnych układów stron - system siatkowy oparty na 12 kolumnach.
- Udostępnia kolekcje komponentów (połączenie CSS i JavaScript) dla interfejsu użytkownika (UI) np.: zakładki, okna modalne, paski nawigacyjne, karuzele, karty, itp.
- Wspiera urządzenia mobilne – responsywność.
- Zawiera duży wybór pomocniczych klas CSS, upraszczających kod CSS.
- Wspiera aktualnie najnowsze, stabilne wersje przeglądarek desktopowych i mobilnych.

Preprocesory CSS

- **Charakterystyka**
  - Preprocesor CSS - metajęzyk, rozszerzenie CSS, nadzbiór CSS.
  - Nie jest to nowy język, oddzielny język - cały czas jest w korelacji z CSS. Jego wynikiem jest arkusz CSS.
  - Wymaga kompilacji dla przeglądarek - wynik: arkusz stylu w formacie CSS.
- **Możliwości**
  - rozszerzenie statycznego CSS o elementy języków programowania np.: zmienne, funkcje, pętle, warunki logiczne;
  - tworzenie modułowej struktury CSS, która lepiej się skaluje i pozwala na lepszą organizacje arkuszy stylów w dużych projektach;
  - unikniecie zbędnych powtórzeń i skrócenie kodu = tworzenie bardziej wydajnego, eleganckiego i łatwiejszego w utrzymaniu CSS;
  - przyspieszenie tworzenia kodu CSS.

Preprocesory CSS

Przykłady:

- Sass
- Less
- tailwindcss
- Stylus
- Babel
- CSS-Crush
- Myth
- Rework
- PostCSS
- …

Sass (ang. Syntactically Awesome StyleSheets)

- Twórcy: Hampton Catlin, Natalie Weizenbaum, Chris Eppstein
- Pierwsze wydanie: 2006r.
- Ugruntowana pozycja.
- Rozbudowana społeczność, szeroka dostępność dodatków, bibliotek, rozszerzeń, itp.
- Wykorzystywany m.in. na stronach Apple, Amazon, BBC, The Guardian, ...

Preprocesory CSS

Sass a CSS

- Technologie koegzystujące: kod Sass jest to kod CSS wzbogacony o elementy języka Sass - konsekwencja to, że każdy poprawny kod CSS jest zgodny z Sass.
- Format zapisu: SCSS, SASS, pliki: *.scss, *.sass.
- Format SCSS - przypomina CSS.
- Tryb pracy: nadzorowanie plików i bieżące kompilowanie (konwertowanie) do kodu CSS.

Preprocesory CSS

Biblioteki i narzędzia rozszerzające Sass

- *Zadanie:* ułatwienie i przyspieszenie pisania kodu, również wsparcie tworzenia poprzez używanie sprawdzonych rozwiązań, standardowych elementów wykorzystywanych na stronach www.

Przykłady:

- Framework: Compass (http://compass-style.org) - napisany przez jednego z autorów Sass, jest to zbiór narzędzi i gotowych domieszek.
- Biblioteka: Bourbon - bogata kolekcja domieszek.
- Dodatki: Susy i Bourbon Neat - wspomaganie tworzenia siatek (ang. grids), Animate.sass - do szybkiego tworzenia animacji, Breakpoints-Sass – ułatwiający pisanie zapytań medialnych (ang. media queries).

Preprocesory CSS
Możliwości

- Zmienne - *examples/scss/test1.scss*
- Zagnieżdżanie selektorów - *examples/scss/test2.scss*
- Importowanie plików - *examples/scss/test3.scss*
- Domieszki, wstawki (ang. mixin) - *examples/scss/test4.scss*
- Dziedziczenie - *examples/scss/test5.scss*
- SassScript (elementy programowania) - *examples/scss/test6.scss*

Aplikacje internetowe

JavaScript, AJAX, PHP

dr inż. Piotr Grochowalski

JavaScript

Jest to język skryptowy, pozwalający tworzyć skrypty uruchamiane w dokumentach HTML.
Skrypty te wykonywane są po stronie klienta (przeglądarki internetowej).

Sposoby dołączania skryptów JavaScript do dokumentu HTML

- Skrypt inline – to skrypt, będący częścią dokumentu HTML – znacznik script.
- Skrypt zewnętrzny - to skrypt znajdujący się w osobnym pliku, do którego dokument odnosi się poprzez adres URL – znacznik script z atrybutem src.

Przykład: *examples/html7.htm* i *examples/html7 1.htm*

Aktualnie JS znajduje wykorzystanie:

- **do tworzenia interfejsu użytkownika poprzez różnego rodzaju framework'i związane z front-end aplikacji internetowych,**
- **poprzez wsparcie środowiska Node.js wkracza w sferę back-end aplikacji, czyli części serwerowej.**

JavaScript
Możliwości

- **Typy danych, zmienne** - Przykład: *examples/html8.htm*

  Język JavaScript jest językiem o słabym typowaniu tzn. nie trzeba deklarować typów danych zmiennych, parametrów przekazywanych do funkcji.
- **Obiekty: właściwości, metody** - Przykład: *examples/html9.htm*
- **Tablice, obsługa błędów, konwersja typów** - Przykład: *examples/html10.htm*

JavaScript
Możliwości

- Programy napisane w języku JavaScript nie działają samodzielnie, tylko wewnątrz określonego środowiska np. przeglądarki, środowiska Node.js.

- **BOM** (*Browser Object Model*) – obiektowy model przeglądarki.
- **DOM** (*Document Object Model*) – obiektowy model dokumentu.

Kod JavyScript osadzony wewnątrz HTML ma dostęp do obiektów, które można podzielić na:

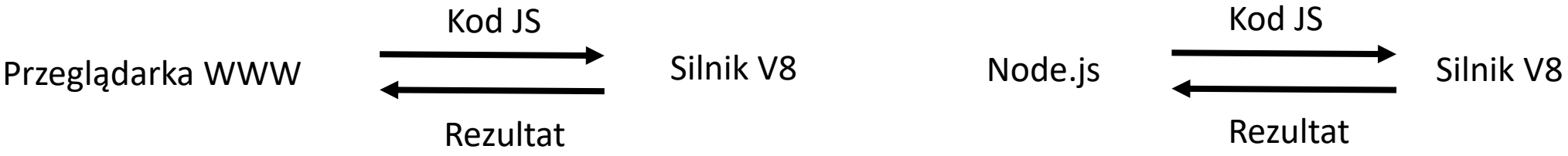
- Obiekty związane z obecnie załadownym dokumentem – DOM,
- Obiekty zewnętrzne tj. ekran, okno przeglądarki – BOM.

JavaScript vs Node.js

- **JavaScript** powstaje 1995r. – cel: wprowadzenie interaktywności na stronach WWW.
- Google w 2008r. tworzy silnik V8 (napisany w C++) dla interpretacji kodu JavaScript bez przeglądarki – dzięki temu powstaje Node.js.
- W 2009r. powstaje **Node.js** – pierwsze środowisko pozwalające na uruchamianie kodu JavaScript poza przeglądarką.

JavaScript: samodzielnie jest językiem synchronicznym, w połączeniu z przeglądarką lub Node.js staje się asynchroniczny.

Node.js: asynchroniczny, wielowątkowy (wewnątrz V8), menadżer pakietów NPM (Node Package Manager).



DOM
Document Object Model

Obiektowy model dokumentu (DOM):

- Standard ustalany przez organizację World Wide Web Consortium (W3C).
- Ma zdefiniowanych kilka wersji (poziomów, DOM Level)
- Umożliwia wykorzystanie języka JavaScript do eksplorowania zawartości dokumentu HTML i manipulowania nią. Można powiedzieć, że

  DOM jest połączeniem pomiędzy JavaScript a zawartością dokumentu HTML
- Przy użyciu DOM możliwe jest dodawanie, usuwanie i modyfikowanie elementów.
- Na czynności użytkowników można reagować przy pomocy zdarzeń.
- DOM pozwala również na uzyskanie kontroli nad arkuszami CSS.

BOM
Browser Object Model

Obiektowy model przeglądarki (BOM):

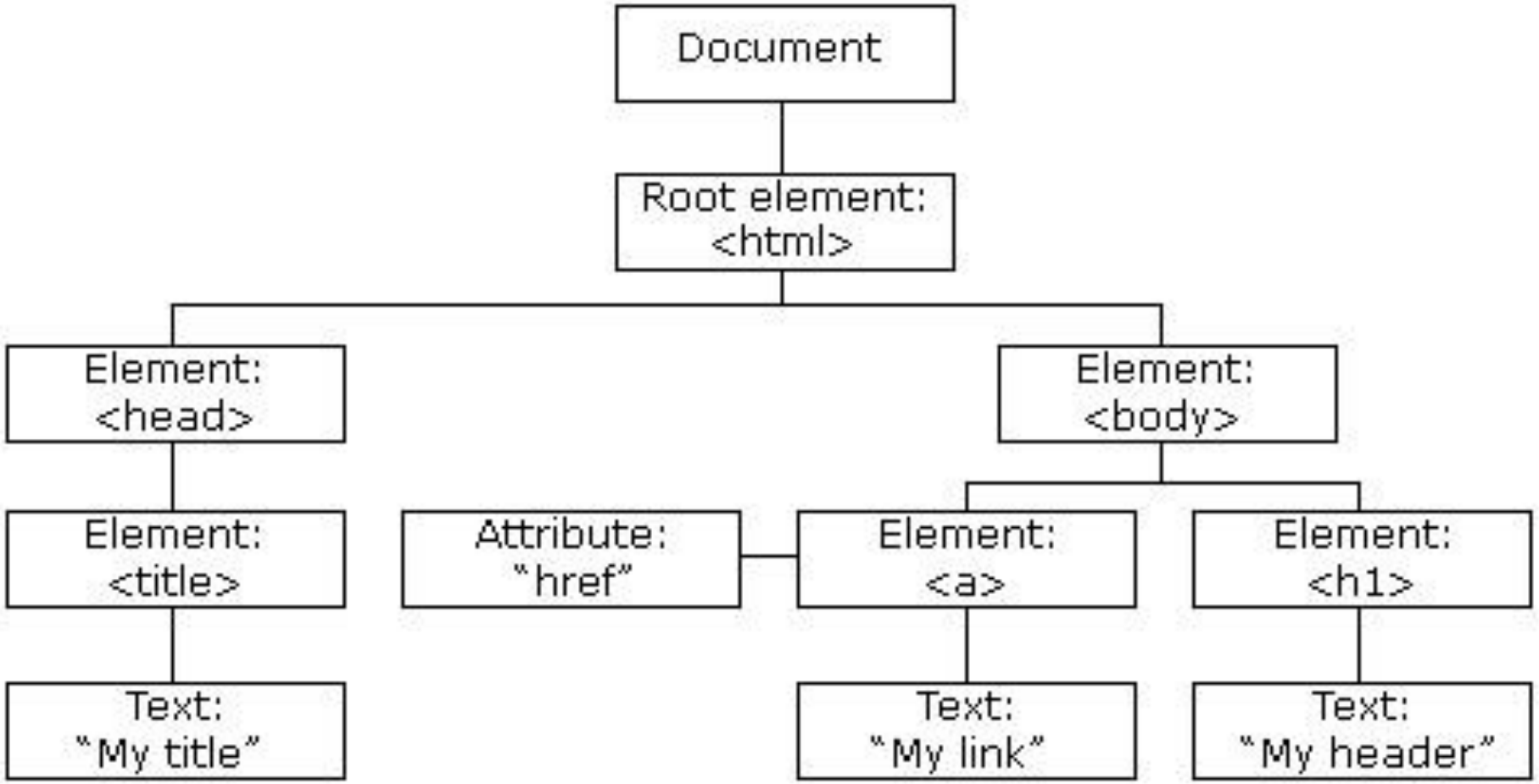
- Nie jest częścią żadnego standardu.
- Został zdefiniowany pewien zbiór obiektów wspólnych dla wszystkich przeglądarek – są to głównie obiekty dające dostęp do **przeglądarki** i **ekranu**. Odpowiadają im globalne obiekty window (okno) oraz window.screen (ekran).
- Mogą pojawiać się oddzielne obiekty definiowane w ramach przeglądarek poszczególnych producentów.

DOM
Document Object Model

- DOM jest zbiorem obiektów reprezentujących elementy dokumentu HTML.
- Każdemu obiektowi w modelu przypisane są właściwości i metody. Kiedy używa się ich do wprowadzania zmian w stanie obiektu, przeglądarka odzwierciedla te zmiany w odpowiednim elemencie HTML i aktualizuje dokument.
- Wszystkie obiekty DOM, które reprezentują elementy HTML, czyli obiekty HTMLElement obsługują zawsze ten sam zbiór podstawowych funkcji. Z bazowych funkcji obiekty mogą korzystać zawsze, niezależnie od tego, jaki rodzaj elementu dany obiekt reprezentuje. Dodatkowo, niektóre obiekty obsługują dodatkowe funkcje, pozwalające na wykonywanie operacji właściwych dla danych elementów HTML.

DOM

Document Object Model



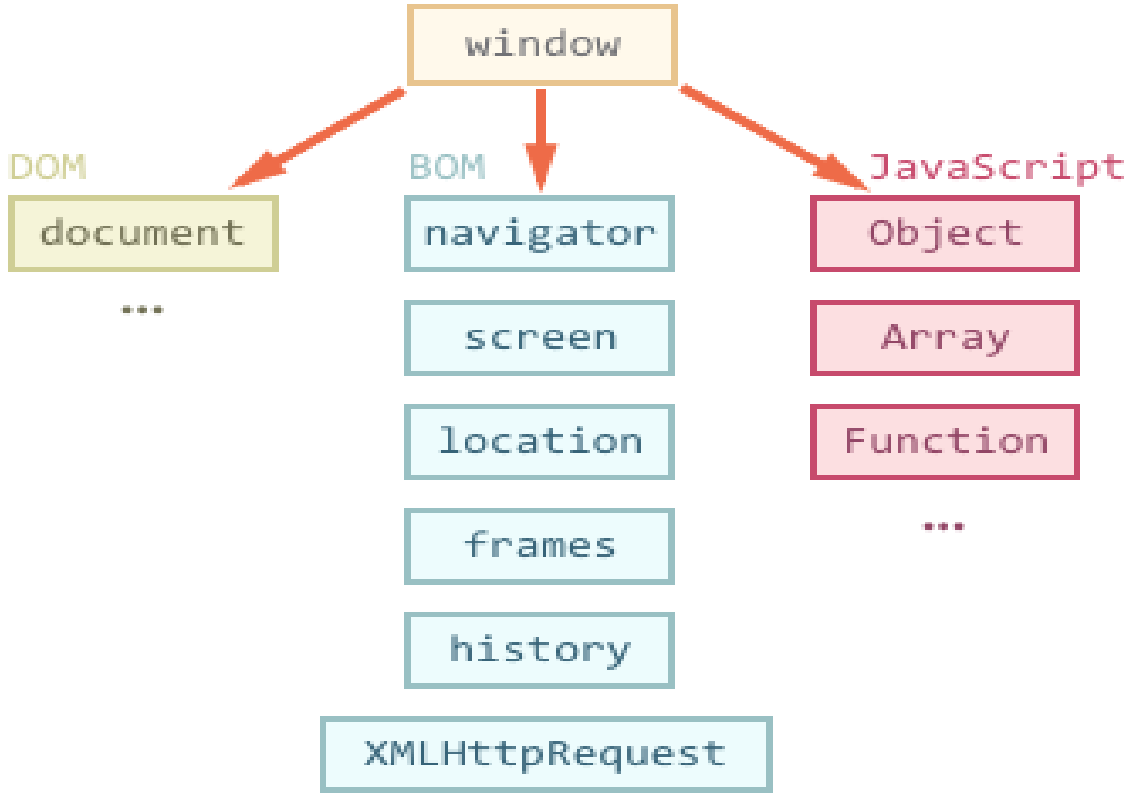
BOM/DOM

Składniki modelu

Obiekty:

- Document
- HTMLElement
- Text
- Location (BOM)
- Window (BOM)
- History (BOM)
- Screen (BOM)
- CSSStyleDeclaration

Zdarzenia np.: click, Focus, keydown, mouseenter, onload, onresize, submit, itp.



Źródło: https://javascript.plainenglish.io/developer-interview-question-what-is-the-browser-object-model-dfbf43b9b367

Obiekt Document

- Obiekt **Document** jest punktem wyjścia do pracy z funkcjami DOM, pozwala na uzyskiwanie informacji o bieżącym dokumencie, a także oferuje zbiór funkcji służących do eksploracji, nawigowania, przeszukiwania oraz manipulowania struktura i treścią dokumentu.
- Do obiektu **Document** uzyskuje się dostęp przy użyciu zmiennej globalnej document.

Przykład: *examples/html11.htm*

Obiekt Document
Dostęp do elementów HTML

Obiekt **Document** pełni funkcje wyjścia do pracy z obiektami reprezentującymi elementy w dokumencie.

Sposoby dostępu do obiektów:

- *Właściwości zwracające obiekty reprezentujące określone elementy lub typy elementów* tj. activeElement, body, head, forms, images, links, scripts, embeds plugins.
- *Pozyskanie nazwanego elementu przy użyciu notacji tablicowej i ’obiektowej”*.
- *Przeszukiwanie elementów* tj. getElementByID(id), getElementsByClassName(klasa), getElementsByName(nazwa), getElementsByTagName(tag), querySelector(css), querySelectorAll(css).
- *Nawigacja po drzewie DOM* tj. childNodes, firstChild, hasChildNodes(), lastChild, nextSibling, parentNode, previousSibling.

Przykład: *examples/html13.htm*

Obiekt Location

Właściwość document.location/window.location zwraca obiekt **Location**, który dostarcza szczegółowe informacje co do adresu dokumentu i pozwala na przechodzenie do innych dokumentów. Jest to równoznaczne z obiektem location.

Przykład: *examples/html12.htm*

Obiekt Window

Obiekt **Window** można pozyskać na dwa sposoby tj. poprzez właściwość document.defaultView (HTML5) i obiekt globalny window (BOM).

Podstawowe funkcje obiektu **Window** odnoszą się do okna, w którym wyświetlany jest bieżący dokument.

Możliwości:

- *Pozyskiwanie informacji o oknie.*
- *Interakcja z oknem.*
- *Wyświetlanie komunikatów.*
- *Pozyskiwanie ogólnych informacji tj. właściwości document, history, location.*
- *Praca z historią przeglądarki.*
- *Wykorzystanie czasu.*

Przykład: *examples/html14.htm*

Operowanie elementami DOM
Obiekt HTMLElement

Możliwości:

- *Właściwości*: checked, classList, className, dir, disabled, hidden, id, lang, spellcheck, tabIndex, tagName, title.
- *Operowanie klasami CSS (classList)*: add(klasa), contains(klasa), length, remove(klasa), toggle(klasa).
- *Operowanie atrybutami elementu*: attributes [name, value], dataset (HTML5), getAttribute(nazwa), hasAttribute(nazwa), removeAttribute(nazwa), setAttribute(nazwa,wartosc).
- *Operowanie zawartością tekstowa elementu - obiekt Text (dziecko danego elementu)*: appendData(tekst), data, deleteData(pozycja, liczba_znaków), insertData(pozycja, tekst), length, replaceData(pozycja, liczba_znaków, tekst), replaceWholeText(tekst), splitText(liczba), substringData(pozycja, liczba_znaków), wholeText.

Przykład: *examples/html15.htm*

Operowanie elementami DOM

Obiekt HTMLElement

Możliwości:

- *Modyfikowanie DOM*: appendChild(element), cloneNode(boolean), compareDocumentPosition(element), innerHTML, insertAdjecentHTML(położenie, tresc), insertBefore(nowyelement, element), isEqualNode(element), isSameNode(element), outerHTML, removeChild(element), replaceChild(element, element), createElement(tag), createTextNode(tresc).

Przykład: *examples/html16.htm*

DOM
Praca z arkuszami stylów CSS

Dostęp do arkuszy stylów CSS dokumentu uzyskuje się za pośrednictwem właściwości document.styleSheets, która zwraca zbiór obiektów reprezentujących powiązane z dokumentem arkusze stylów. Każdy arkusz stylów reprezentowany jest przez obiekt **CSSStyleSheet**.

Możliwości:

- *Składowe*: cssRules, deleteRule(pozycja), disabled, href, insertRule(reguła, pozycja), media, ownerNode, title, type.
- *Operowanie wybranymi stylami (cssRules)*: item(pozycja), length, cssText, parentStyleSheet, selectorText, style.
- *Operowanie obiektem style*: cssText, getPropertyCSSValue(nazwa), getPropertyPriority(nazwa), getPropertyValue(nazwa), item(pozycja), length, parentRule, removeProperty(nazwa), setProperty(nazwa, wartosc, priorytet), właściwości pomocnicze.

Przykład: *examples/html17.htm*

Zdarzenia

Zdarzenia w JavaScript pozwalają na tworzenie funkcji wywoływanych w odpowiedzi na zmianę stanu elementu.

Sposoby obsługiwania zdarzeń:

- *Wprowadzenie procedury obsługi zdarzeń inline* - przy użyciu atrybutu zdarzenia.
  Elementy obsługuj po jednym atrybucie zdarzenia na obsługiwane zdarzenie np.: atrybut *onmouseover* (globalne zdarzenie *mouseover*), wyzwalany kiedy użytkownik przeciąga kursor na obszar ekranu zajęty przez element. Większości zdarzeń odpowiada atrybut elementu *on[nazwa zdarzenia]*.

  Wady: ”rozwlekłe”, utrudniają czytanie kodu, odnoszą się tylko do jednego elementu.
- *Wprowadzenie procedury obsługi zdarzeń* - polega na definiowaniu funkcji i podaniu jej nazwy jako wartości atrybutów zdarzeń elementu.

  Zalety: brak powtórzeń kodu, lepsza czytelność.

  Wady: brak oddzielenia obsługi od elementów HTML.
- *Zastosowanie DOM i obiektu Event*

  Zalety: oddzielenie kodu obsługi zdarzenia od HTML, brak powtórzeń kodu.

Zdarzenia

Elementy, dla których można definiować zdarzenia to:

- zdarzenia związane z praca myszy i klawiatury,
- zdarzenia dotyczące elementów HTML tj. obiekt Document, obiekt Window„ formularzy, zdarzenia dotyczące ”fokusowania” i ”odfokusowywania” elementów, itd.

Przykład: *examples/html18.htm*

AJAX

Asynchronous JavaScript and XML

- **AJAX** (*ang.* Asynchronous JavaScript and XML) to technologia wykorzystująca asynchroniczny JavaScript i XML umożliwiająca asynchroniczne *wysyłanie* i *pobieranie* danych z serwera oraz przetwarzanie ich z użyciem JavaScript.
- Najważniejsza specyfikacja **AJAX**'a nosi nazwę obiektu JavaScript, który służy do definiowania i wydawania żądań: XMLHttpRequest. Ta specyfikacja ma dwa poziomy. Wszystkie popularne przeglądarki obsługują poziom pierwszy, który obsługuje podstawowe funkcje. Poziom drugi rozszerza bazowa specyfikacje o dodatkowe zdarzenia, funkcje ułatwiające prace z elementami form oraz obsługę pokrewnych specyfikacji.

Przykład: *examples/html19.htm*

AJAX
Możliwości

Zdarzenia:

- abort - wyzwalane przy przerwaniu zadania,
- error - wyzwalane w przypadku niepowodzenia w wykonywaniu zadania,
- load - wyzwalane, kiedy zadanie zostaje wykonane z powodzeniem,
- loadend - wyzwalane, kiedy zadanie zostaje wykonane z powodzeniem lub nie,
- loadstart - wyzwalane przy rozpoczęciu wykonywania zadania,
- progress - wyzwalane, by wskazać postępy w wykonywaniu zadania,
- readystatechange - wyzwalane na różnych etapach wykonywania zadania,
- timeout - wyzwalane, kiedy kończy się czas oczekiwania na odpowiedz.

Protokół HTTP:

- Zmiana metody HTTP zadania
- Wyłączenie buforowania treści
- Odczytywanie nagłówków odpowiedzi

Przykład: *examples/html19a.htm*

AJAX
Możliwości

- Przetwarzanie formularzy.
- XML (*ang* Extensible Markup Language).
- JSON (*ang.* JavaScript Object Notation)

  JSON określa się mianem odchudzonego XML, który charakteryzuje się ”dużą” rozwlekłością. Jest łatwy w zapisie i odczycie, bardziej zwięzły niż XML.

  JSON (JavaScript Object Notation, notacja obiektów JavaScript) opisuje dane za pomocą literałów obiektowych i tablicowych.

Przykład: *examples/html20.htm* i *examples/html21.htm*

PHP

PHP - Obiektowy język programowania zaprojektowany do generowania stron internetowych i budowania aplikacji webowych w czasie rzeczywistym.

Zastosowania:

- Tworzenie skryptów po stronie serwera WWW,
- Przetwarzanie danych z poziomu wiersza poleceń,
- Tworzenie programów pracujących w trybie graficznym (np. za pomocą biblioteki GTK+, używając rozszerzenia PHP-GTK).

Najczęściej wykorzystywana jest implementacja PHP wraz z serwerem WWW Apache oraz serwerem baz danych MySQL (lub inna relacyjna) i określana jest jako platforma AMP (w środowisku Linux – **LAMP**, w Windows – **WAMP**).

PHP
Kalendarium

- **1994r.** - początek, pierwsza wersja PHP/FI (Personal Home Page/Forms Interpreter) (Rasmusa Lerdorfa) - skrypty w Perlu, później w C.
- **8.06.1995r.** - publiczne udostepnienie kod źródłowego (PHP Tools 1.0).
- **11.1997r.** - oficjalne wydanie PHP/FI 2.0.
- **06.1998r.** - PHP 3.0 (Zeev Suraski, Andi Gutmans) - nowa architektura (zwiększenie wydajności), początki programowania obiektowego, modułowość tj. użytkownicy mogli rozszerzać funkcjonalność języka poprzez dodawanie nowych modułów.
- **1999r.** - Zend Engine - nowy silnik języka skryptowego, wokół którego zaczęto budować PHP 4 (Zeev Suraski, Andi Gutmans).
- **05.2000r.** - PHP 4.0 (Zeev Suraski, Andi Gutmans) - dużo nowych narzędzi, konstrukcji językowych oraz bezpieczniejszy system wejścia/wyjścia. Od strony administracyjnej pojawiło się oficjalne wsparcie dla wielu nowych serwerów.
- **04.2004r.** - PHP 5.0 (Zeev Suraski, Andi Gutmans) - nowy model programowania obiektowego (obiekt przestaje być zmienna, a jest referencja do właściwego obiektu), przebudowane wiele modułów np.: do obsługi XML-a i komunikacji z baza danych, udostępniono zbiór interfejsów znacznie rozszerzających możliwości klas użytkownika, zmiany oraz nowości w systemie modułów PHP.
- **06.2005r.-03.2010r.** - prace nad PHP 6.0, głównym celem było dążenie do ujednolicenia projektu, wprowadzenia dalszych możliwości wymaganych przez złożone projekty (m.in. pełne wsparcie unicode czy system cache'owania kodu) - zawieszony wskutek braku postępów w implementacji standardu Unicode oraz wewnętrznych sporów w gronie czołowych programistów.
- **2014r.- 2020r.** - prace nad PHP 7.0 - optymalizacja wydajności PHP przez refaktoring Zend Engine, przy zachowaniu zgodności języka, ma również zawierać ulepszona składnie zmiennych, wewnętrznie spójna i kompletna.
- **11.2020r.** - PHP 8.0.

PHP
Moduły

Cała funkcjonalność PHP zawarta jest w czterech zbiorach modułów różniących się od siebie dostępnością dla programisty.

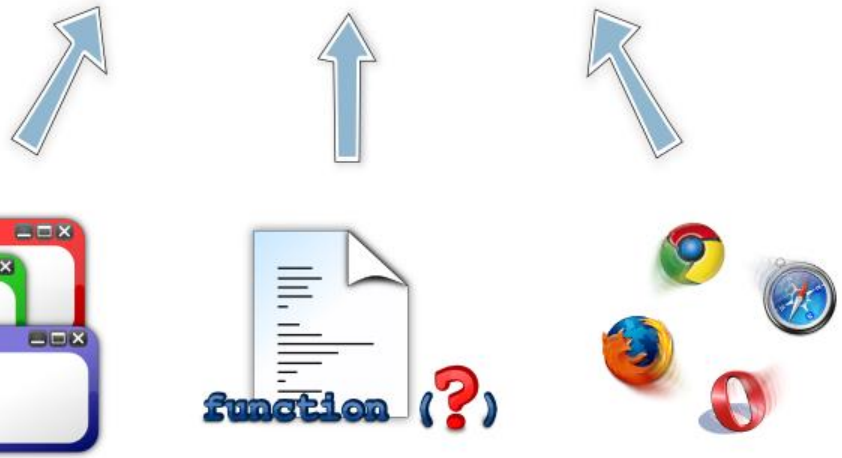
- **Moduły jadra** - cześć silnika PHP; zawsze aktywne.
- **Moduły oficjalne** - element każdej dystrybucji PHP; aktywowane ręcznie przez administratora serwera.
- **Repozytorium PECL** (*PHP Extension Community Library*) - darmowe moduły o otwartym źródle tworzone przez programistów z całego świata, przeznaczone do samodzielnej kompilacji. Począwszy od wydania PHP 5 do PECL przeniesionych zostało wiele wcześniejszych modułów oficjalnych, najczęściej tych niestabilnych lub rzadko używanych.
- **Repozytorium PEAR** (*PHP Extension and Application Repository*) – zbiór modułów realizujący typowe zadania klas o ujednoliconej budowie. Jest to framework i systemem dystrybucji rozszerzeń do języka PHP. Został rozpoczęty w 1999 roku przez Stiga S. Bakkena i w krótkim czasie dołączyło do niego wiele osób, które teraz tworzą społeczność zarządzającą projektem. Głównymi założeniami jego projektu było dostarczenie programistom PHP kolekcji otwarto-źródłowych rozszerzeń i prostego systemu ich dystrybucji w postaci tzw. paczek.

jQuery

Obecnie biblioteki JavaScript można podzielić na dwa rodzaje:

- Ułatwiające wprowadzenie dynamiki do istniejącego kodu HTML poprzez uproszczenie dostępu do DOM, CSS i AJAX, do tej grupy należy **jQuery** (udostępnia interfejs współpracujący z dokumentami HTML).
- Definiujące podstawowe elementy interfejsu i budujące cały interfejs użytkownika bezpośrednio w JavaScript np.: Angular, React, Vue.

- **jQuery** to jedna z popularnych bibliotek programistycznych dla języka JavaScript, ułatwiająca korzystanie z JavaScriptu, w tym manipulacje drzewem DOM.
- Witryny główna projektu: http://jquery.com
- Start: 2006r. (wersja 1.0 – pierwsze stabilne wydanie).



rozwój aplikacji internetowych

brak niektórych funkcji

problemy z interpretacją kodu

separacja kodu

podobieństwo do CSS

łańcuchy poleceń

zgodność z wieloma przeglądarkami

rozszerzenia

jQuery
Funkcjonalności

jQuery pozwala w wygodny i zrozumiały sposób korzystać z następujących funkcjonalności:

- selektory – umożliwiają wybranie dowolnego podzbioru węzłów modelu DOM,
- atrybuty – jQuery pozwala przetwarzać atrybuty węzłów dokumentu,
- manipulowanie modelem DOM,
- zmiana i przypisywanie stylu do elementów,
- rozbudowana obsługa zdarzeń, możliwość definiowania własnych,
- efekty – animacje,
- AJAX – prosty interfejs realizujący asynchroniczne zapytania.

jQuery
Użycie

- Typowe wykorzystanie biblioteki jQuery polega na przekazaniu selektora CSS funkcji $, której wynikiem jest tablica referencji do obiektów dopasowanych elementów (tablica może być pusta). Następnie na tej tablicy wykonuje się dodatkowe operacje poprzez metody obiektu jQuery.
- Wykorzystanie funkcji $ (lub w starszych wersjach: jQuery, $ jest standardowym aliasem obiektu jQuery) pozwala na tworzenie łańcuchów wywołań, gdyż funkcja ta i inne metody zwracają obiekt jQuery, co oznacza, że można łatwo łączyć je w łańcuch wywołań.
- Metody z prefiksem $. lub jQuery. są metodami samodzielnymi lub działają globalnie.
- jQuery oferuje również niezależny od przeglądarki interfejs do synchronicznych oraz asynchronicznych zadań HTTP (AJAX). Standaryzuje on różne implementacje obiektu XMLHttpRequest. Zapytania obsługuje się poprzez metody globalne: $.ajax (jQuery.ajax), $.post lub $.get. Metody te różnią się od siebie tym, ze $.post wysyła/pobiera dane za pomocą metody POST, a $.get pobiera dane za pomocą GET.

Przykład: *examples/html22.htm* i *examples/html23.htm*