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:

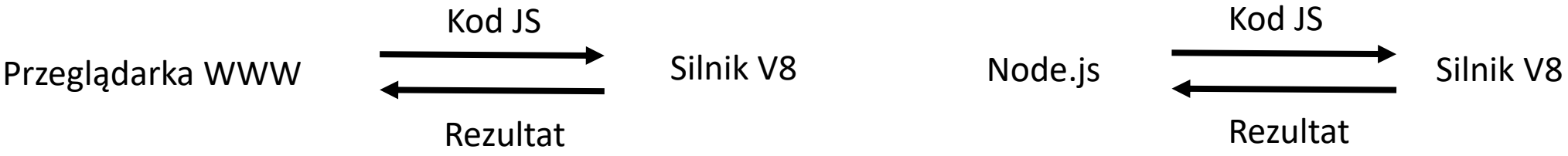
- Obiekty związane z obecnie załadownym dokumentem – DOM,
- Obiekty zewnętrzne tj. ekran, okno przeglądarki – BOM.

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):

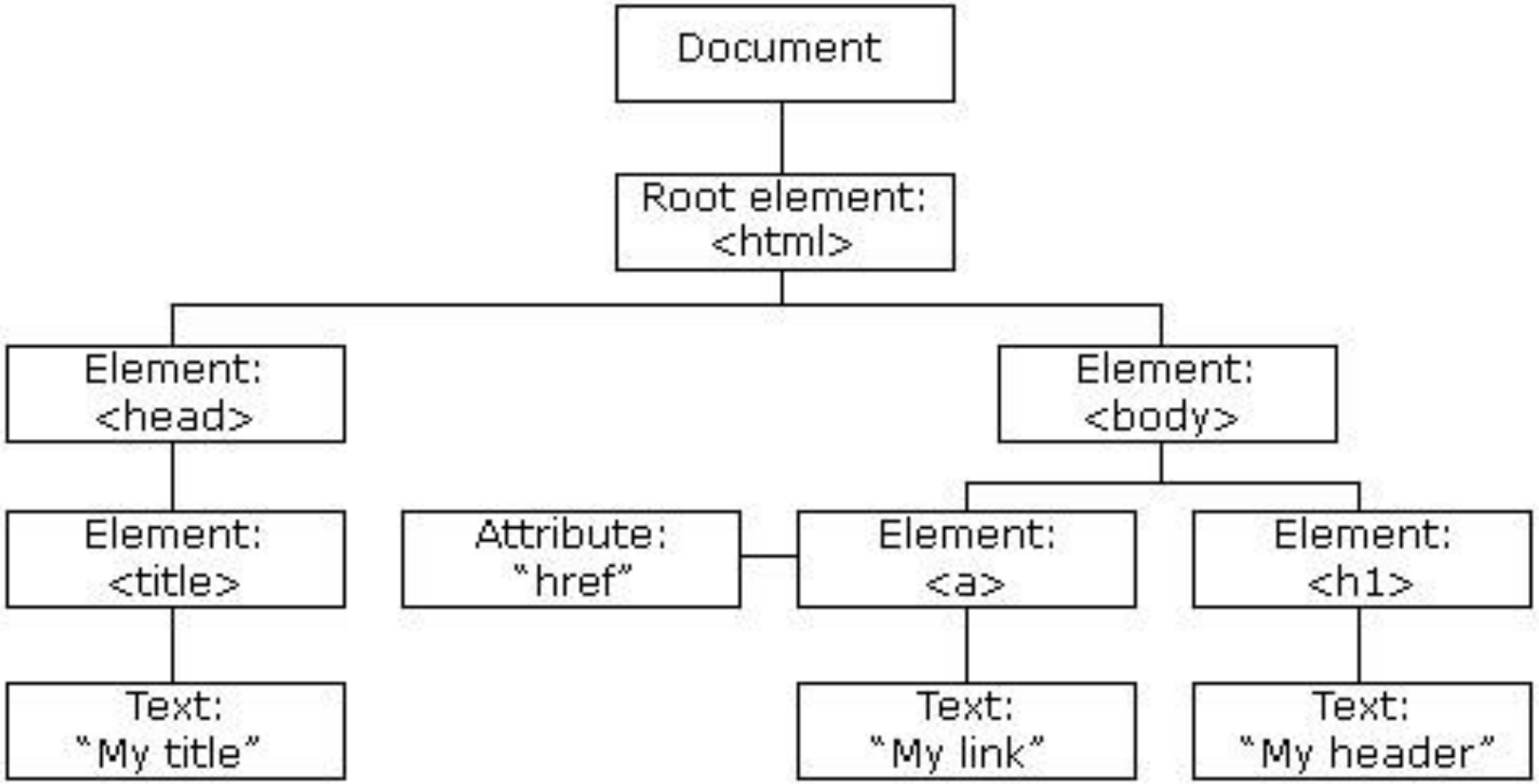
- Nie jest częścią żadnego standardu.
- Został zdefiniowany pewien zbiór obiektów wspólnych dla wszystkich przeglądarek – są to głównie obiekty dające dostęp do **przeglądarki** i **ekranu**. Odpowiadają im globalne obiekty window (okno) oraz window.screen (ekran).
- Mogą pojawiać się oddzielne obiekty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



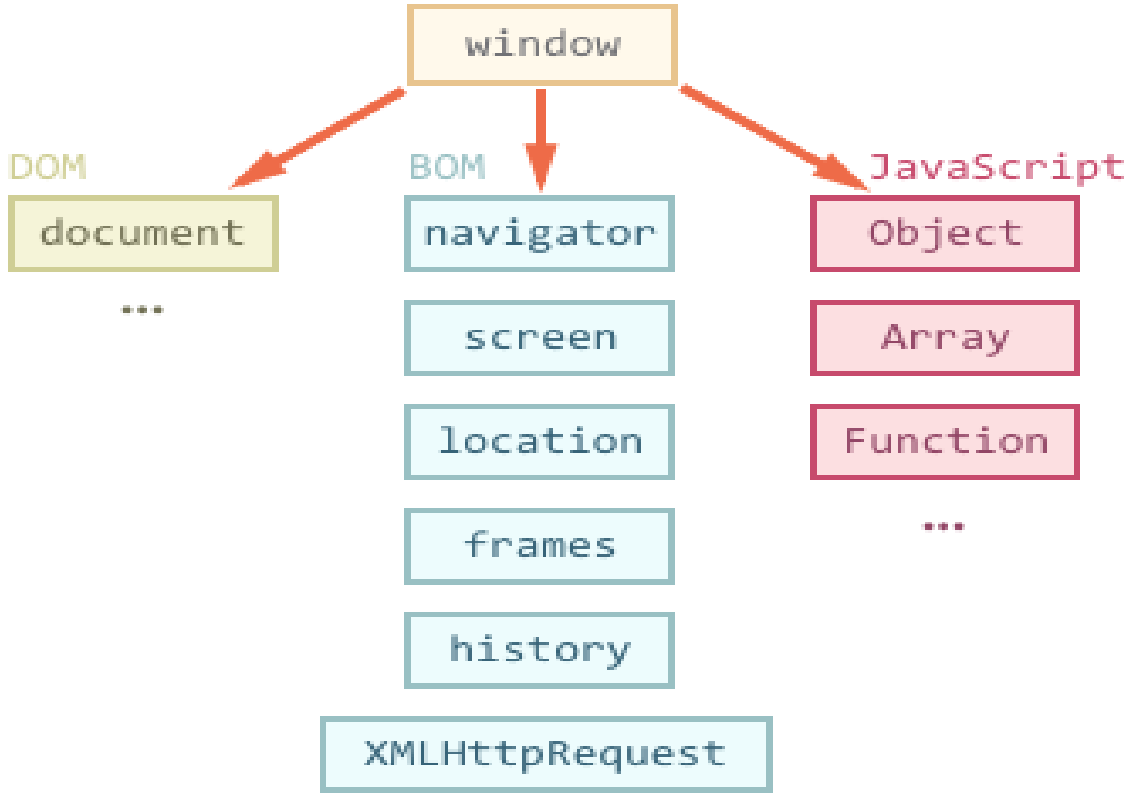
BOM/DOM

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.

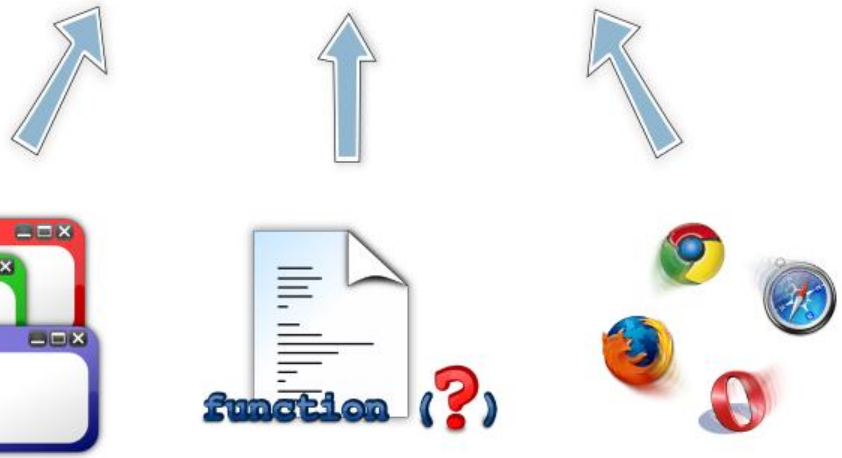
- **Moduły jadra** - cześć silnika PHP; zawsze aktywne.
- **Moduły oficjalne** - element każdej dystrybucji PHP; aktywowane ręcznie przez administratora serwera.
- **Repozytorium PECL** (*PHP Extension Community Library*) - darmowe moduły o otwartym źródle tworzone przez programistów z całego świata, 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*