PWr Spotkanie 4
Aplikacje webowe na platformie .NET
Laboratorium – **Lista 4**

Wstęp.

Stworzenie strony WWW w wykorzystaniem języka JavaScript.

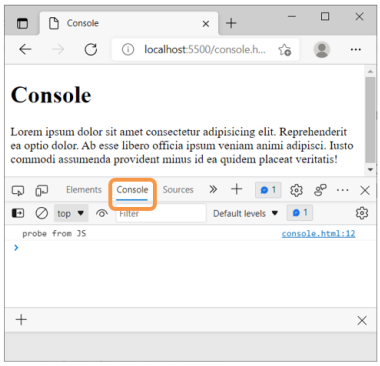
Wprowadzenie do języka JavaScript

Język JavaScript (w skrócie JS), to język skryptowy, który wykonuje się po stronie klienta. W zasadzie jedyny język, który jest dostępny. Składniowo przypomina język Java: pętle, instrukcje warunkowe, operator podstawienia, operacje matematyczne, relacyjne, logiczne, oczekiwany średnik na końcu instrukcji, operator kropki dla obiektów itp. Jednak nie wszystko jest takie samo. Np. konstrukcja metod i funkcji jest podobna, jednak w nagłówku nie występują typy parametrów. Ogólnie zmienne posiadają typ, ale nie jest on określany podczas deklaracji. Typ definiuje przypisane zmiennej wyrażenie. Kod, który jest programem w JavaScripcie umieszczony musi być w elemencie `<script>`. Inną możliwością jest umieszczenie w tym elemencie atrybutu `src` w którym podamy lokalizację pliku z kodem JS. Należy wówczas podać również typ `type="text/javascript"`.

Konsola JS

Każda zaawansowana przeglądarka pozwala na włączenie konsoli JS, która umożliwia sprawdzanie logów oraz błędów kompilacji i wykonania.

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" conte
6       <meta name="viewport" content="width=dev
7       <title>Console</title>
8   </head>
9   <body>
10      <h1>Console</h1>
11      <script>
12          console.log("probe from JS");
13      </script>
14    <p>Lorem ipsum dolor sit amet consectetur
15  </body>
16  </html>
```



Kompilacja odbywa się osobno w każdym elemencie `<script>`, aczkolwiek kod wykonywany we wcześniejszych takich elementach nadal jest dostępny w ramach maszyny wirtualnej JS (czyli zmienne zadeklarowane ze słowem **`var`**, globalne, funkcje i klasy).

Zmienne

Do deklaracji zmiennych służą 3 słowa kluczowe:

- Deklaracja zmiennych (poprzedzenie zmiennej słowem kluczowym):

- **var** (od zawsze)
- **let** (od ES6 2015)
- **const** (od ES6 2015)

Deklaracja zmiennej słowem `var` zachowuje się dość specyficznie dla tego języka. Otóż można wielokrotnie deklarować ten sam identyfikator zmiennej, oprócz tego tego typu zmienne są dostępne od początku jakiegokolwiek kodu JS (ale mają wartość **`undefined`**). Zmienne zadeklarowane jako **`let`** lub **`const`** są dostępne od momentu deklaracji do końca bloku, z tym, że te drugie nie mogą już być modyfikowane.
Zmienne ogólnie są w jednej tablicy mieszającej i można je również z niej usunąć. Próba użycia niezdefiniowanej zmiennej kończy się wyjątkiem.

Obiekty

Obiekty są pierwotnym elementem języka JS, natomiast pojęcie klasy jest wtórne. Klasa to w zasadzie specyficznie zapisana tablica mieszająca, czyli posiadająca pary klucz-wartość. Zapis obiektu to para nawiasów klamrowych, w którym przecinkiem rozdzielamy wspomniane pary klucz-wartość. Klucz od wartości oddziela dwukropek '`:`'. Wartością może być poprawne wyrażenie dowolnego typu. W nomenklaturze obiektowej klucz to pole. Podczas korzystania z obiektu można użyć notacji z kropką, czyli np. `osoba.imie`, jednak ponieważ jest to tak naprawdę tablica, możliwe jest też użycie zapisu z tablicą, jednak nazwa pola musi być wtedy w cudzysłowach: `osoba["imie"]`. Ponieważ nie są to klasyczne obiekty, można usunąć pole z obiektu poprzez zapis **`delete`** `osoba.imie`.

```
12      let person={ name:"Einstein", salary:2000};
13      console.log(person);
14      person.salary+=100;
15      console.log(person);
16      person.year=1900;
17      console.log(person);
18      person.name=undefined;
19      console.log(person);
20      delete person.salary;
21      console.log(person);
22      person["forname"]="Albert";
23      console.log(person);
24      person.name=() => "Name";
25      console.log(person);
26      console.log(person.name());
```

```
▶ {name: 'Einstein', salary: 2000}                    objects.html:13
▶ {name: 'Einstein', salary: 2100}                    objects.html:15
▶ {name: 'Einstein', salary: 2100, year: 1900}        objects.html:17
▶ {name: undefined, salary: 2100, year: 1900}         objects.html:19
▶ {name: undefined, year: 1900}                       objects.html:21
▶ {name: undefined, year: 1900, forname: 'Albert'}    objects.html:23
▶ {year: 1900, forname: 'Albert', name: f}            objects.html:25
Name                                                  objects.html:26
```

Funkcje/metody

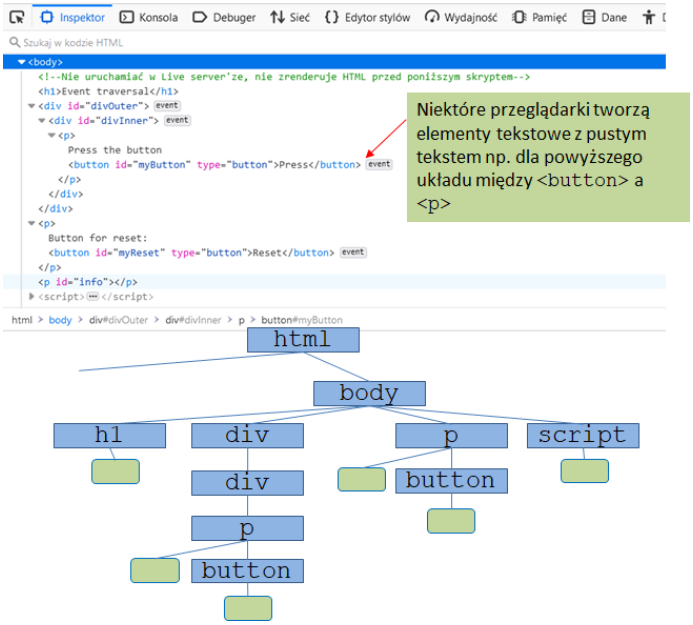
Funkcje globalne (lub metody w obiektach), to w zasadzie przyporządkowanie do identyfikatora zmiennej (lub pola obiektu) obiektu funkcyjnego. Zapis podobny jak w Javie to w zasadzie tylko skrót notacyjny. Powoduje to jednak, że nie mogą wystąpić dwie funkcje o tej samej nazwie. Kolejna definicja funkcji o tej samej nazwie nadpisuje poprzednią. Powoduje to też, że funkcję można wywołać z dowolną liczbą parametrów. Jeśli w definicji funkcji jest ich więcej, to brakujące będą mieć wartość `undefinded`. Z drugiej strony, jeśli w definicji jest ich mniej, do pozostałych można mieć dostęp przez specjalną tablicę `arguments` indeksowaną od 0.

Tablice

W JavaScripcie, jak opisano wcześniej, istnieją tablice mieszające, które są w sumie równoważne obiektom. Jednak można je również używać, jak tablice indeksowane liczbami całkowitymi od 0 w górę. To oznacza, że indeksem tablicy w JS może być ciąg znaków lub liczba. Chociaż nie jest to stosowane, indeksem w tym języku może być prawie każdy typ.

Model DOM

Cały dokument HTML jest pamiętany jako model DOM (ang. Document Object Model), czyli obiektowy model dokumentu. W języku JS oznacza to, że każdy element to obiekt, w którym istnieją pola i metody do manipulowania dokumentem. Taki model ma też układ hierarchiczny, gdyż w tym dokumencie elementy są wewnątrz innych elementów więc całość można uznać za drzewo dokumentów, w którym kolejność dzieci jest również ważna (bo elementy tego samego rodzica następują po sobie).



JavaScript i manipulowanie DOM

W obiekcie globalnym `document` istnieją metody umożliwiające otrzymanie dostępu do dowolnego elementu z DOM. Oprócz tego można zmieniać takiemu elementowi wszelkie atrybuty, ustawiać go w innym miejscu drzewa DOM. Można też stworzyć nowy element, wypełnić go atrybutami i wstawić w do DOM.

JavaScript i zdarzenia

Podczas działania strony mogą się zachodzić różne zdarzenia – ruch myszką, wciśniecie klawisza (np. podczas wpisywania danych do formularza), upływ czasu itp. W reakcji na część tych zdarzeń można użyć reguł styli CSS3, jednak nie wszystko jest w nich wykonalne. W takim przypadku przydatne mogą być możliwości przyporządkowania funkcji napisanej w JavaScripcie do wybranego zdarzenia. Można to wykonać za pomocą odpowiedniego atrybutu elementu zaczynającego się od słowa `on` (np. `onmousemove`) lub w kodzie JS wywołują metodę `addEventListener()` obiektu `windows`.

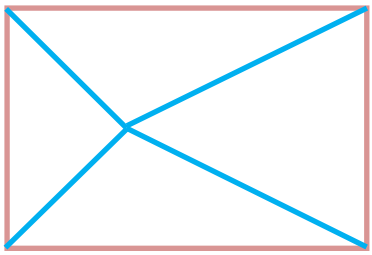
List zadań

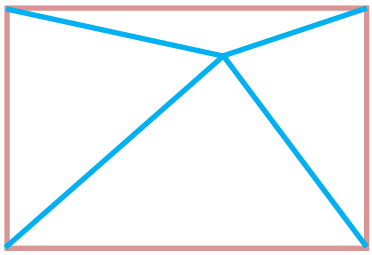
Proszę utworzyć witrynę zawierającą kilka stron zgodnych z standardem HTML 5 z odwołaniem do zewnętrznego arkusza stylów zgodnego z standardem CSS3. Wszystkie reguły CSS, z wyjątkiem punktu 1, powinny być umieszczone w zewnętrznym arkuszu stylów.

1. Stworzyć stronę WWW, gdzie **za pomocą języka JavaScript** oraz metod do tworzenia elementów DOM powstanie tabliczka mnożenia dla n losowych elementów z zakresu 1-99. Liczba *n* wierszy i kolumn z wynikami mnożenia powinna być podana przez użytkownika. Wartość n ma być zakresu (5 <= n <=20). Jeśli użytkownik poda cokolwiek innego, ma zostać przyjęta jakaś domyślna wartość z tego zakresu, ale na stronie ma się pojawić informacja, że podane dane były niewłaściwe, więc przyjęto n=X. Wyniki (mnożenia) parzyste w komórkach tablicy powinny być z klasą „`even`”, a nieparzyste z klasą „`odd`”. Wylosowane wartości mają się znaleźć w nagłówkach kolumn i wierszy. W pliku CSS nadać odpowiednio style, aby tablica wyglądała estetycznie i wizualnie wyróżniały parzystość wyników. Np. n=13:

	55	91	51	93	94	40	69	50	86	6	47	19	65
55	3025	5005	2805	5115	5170	2200	3795	2750	4730	330	2585	1045	3575
91	5005	8281	4641	8463	8554	3640	6279	4550	7826	546	4277	1729	5915
51	2805	4641	2601	4743	4794	2040	3519	2550	4386	306	2397	969	3315
93	5115	8463	4743	8649	8742	3720	6417	4650	7998	558	4371	1767	6045
94	5170	8554	4794	8742	8836	3760	6486	4700	8084	564	4418	1786	6110
40	2200	3640	2040	3720	3760	1600	2760	2000	3440	240	1880	760	2600
69	3795	6279	3519	6417	6486	2760	4761	3450	5934	414	3243	1311	4485
50	2750	4550	2550	4650	4700	2000	3450	2500	4300	300	2350	950	3250
86	4730	7826	4386	7998	8084	3440	5934	4300	7396	516	4042	1634	5590
6	330	546	306	558	564	240	414	300	516	36	282	114	390
47	2585	4277	2397	4371	4418	1880	3243	2350	4042	282	2209	893	3055
19	1045	1729	969	1767	1786	760	1311	950	1634	114	893	361	1235
65	3575	5915	3315	6045	6110	2600	4485	3250	5590	390	3055	1235	4225

2. Poczytać o elemencie `<canvas>`. Stworzyć kanwę, gdzie ruch myszką będzie powodował narysowanie po jednej linii od każdego z rogów do pozycji myszki. Po ruchu myszką linie te będą prowadziły do kolejnego miejsca:





Jeśli kursor myszki opuści element, linie mają zniknąć. Kod JavaScript ma działać poprawnie nawet jeśli zmienimy wymiary obiektu `<canvas>` (np. w stylach tego obiektu). Rozwiązanie powinno pozwalać wstawić element `<canvas>` w dowolne miejsce poprawnego dokumentu oraz wstawiać kilka poprawnie działających takich elementów.

Data I: Spotkanie 5 (0-100 punktów)
Data II: Spotkanie 6 (0-80 punktów)
Data III: Spotkanie 7 (0-50 punktów)