Instrukcja 4

W celu lepszego zrozumienia treści niniejszej instrukcji warto zapoznać się z następującymi pojęciami:

Rodzic – obiekt (element strony) zawierający w sobie co najmniej jeden inny element strony. Dobrym przykładem takiego obiektu mogą być znaczniki <head> lub <body> - oba z nich mogą/zawierają w swoim ciele wiele innych elementów HTML.

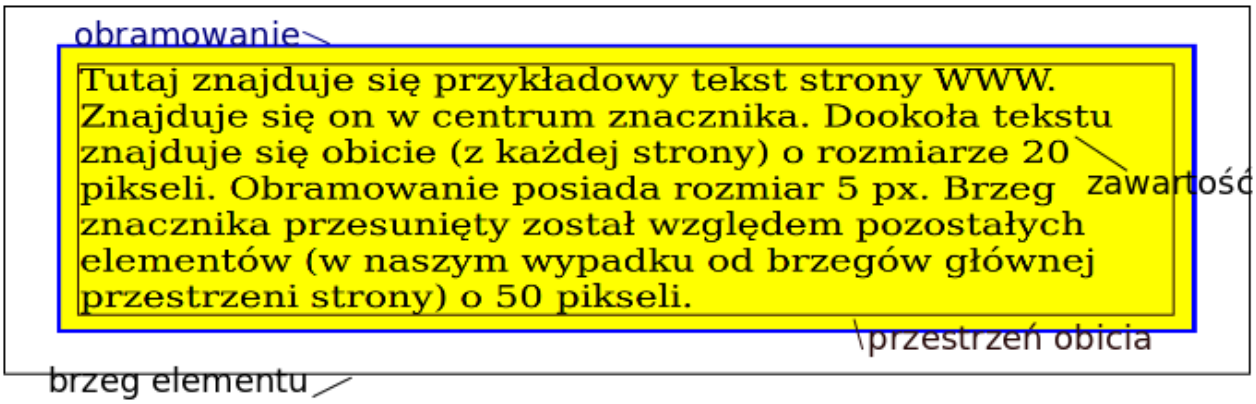
Dziecko – obiekt, który umieszczony jest w ciele innego znacznika HTML. Przeważnie element ten jest ograniczony wymiarowo do wymiarów rodzica; podobnie jak rodzic przyjmuje ustawienia widoczności oraz przezroczystości na stronie. Może natomiast posiadać różne ustawienia kolorów czy przezroczystości (chociaż sam zależny jest dodatkowo od przezroczystości rodzica).

Kontener – tożsame z rodzicem.

Stos – umiejscowienie poszczególnych elementów strony na osi z. Obiekty będące najniżej w hierarchii stosu będą automatycznie przykrywane przez obiekty znajdujące się wyżej. Daje to efekt głębi ekranu – czyli znane 3D (oś x – szerokość ekranu, oś y – wysokość ekranu oraz oś z – głębia ekranu). Domyślnie jeżeli mamy poniższą sytuację:
paragraf 1
paragraf 2
paragraf 3
to napis 'pararaf 1' będzie znajdował się na spodzie stosu (domyślnie wartość 0 na osi z), 'paragraf 2' będzie pośrodku (wartość 1), a 'paragraf 3' będzie na szczycie (wartość 2). Wszystkie te elementy domyślnie wyświetlą się jeden pod drugim (na osi y) przez co efekt nakładania się ich na siebie nie będzie widoczny. Jednak CSS umożliwia projektantowi przesuwanie elementów we wskazaną pozycję (szczegółowe omówienie znajduje się w 2 punkcie instrukcji) – dzięki temu można zaobserwować opisywany efekt. Oczywiście położenie na osi z można ustalać samemu (poprzez właściwość z-index); położenie określa się poprzez liczbę całkowitą (integer) będącą indeksem na opisywanym stosie (można ustalić wartość ujemną).

1. Blokowy układ strony

Każdy element HTML w arkuszach stylów kaskadowych opakowywany jest w swego rodzaju blok (pudło). Składa się on z kilku części.
Pierwszą, wewnętrzną częścią każdego z bloku jest jego właściwa zawartość (taka jak tekst czy obraz). Zawartość otoczona jest tzw. obiciem (padding). Dzięki obiciu możemy przesuwać zawartość elementu (tworząc margines) od obramowania oraz brzegu. Kolor obicia uzależniony jest od koloru tła elementu właściwego. Następne po obiciu jest obramowanie. Dzięki tej części można stworzyć graficzny efekt granicy edytowanego elementu HTML. Na samym końcu znajduje się brzeg elementu. Nie posiada on żadnej reprezentacji graficznej. Dzięki niemu można określić położenie wskazanego elementu względem innych elementów na stronie WWW.
Wszystkie wspomniane części bloku dzielą się na 4 strefy: góra, lewo, dół, prawo.



W języku CSS poszczególne części mają następujące nazwy:
- zawartość – content; nazwa ta NIE WYSTĘPUJE jawnie w CSS; reprezentowana poprzez wartości width (szerokość) oraz height (wysokość)
- obicie – padding; rozmiar tej części można podawać oddzielnie dla każdej strefy (padding-left, padding-right, padding-top, padding-bottom) bądź określić jednakowy dla wszystkich poprzez jedną właściwość padding
- obramowanie – border; rozmiar tej części można, podobnie jak dla obicia, podawać indywidualnie dla każdej ze stref bądź określać jej jednakową wartość dla wszystkich stref poprzez właściwość border.
- brzeg – margin; rozmiar można ustawić dla wszystkich stref poprzez właściwość margin lub dla każdej ze stref oddzielnie (jak w przypadku właściwości padding).

Przy ustalaniu rozmiaru każdej z części należy pamiętać, że ma ona wpływ na wielkość CAŁEGO elementu na stronie.

```
//plik styl.css
p {
	background-color: yellow;
	font-size:18px;
	padding: 20px;
	border: 5px;
	width: 700px;
	height: 200px;
	border-style: solid;
	border-color: blue;
	margin: 50px;
}
//plik index.html
<!DOCTYPE html>
<html lang="pl">
	<head>
		<meta charset='UTF-8'/>
		<title>Style CSS</title>
		<link rel="stylesheet" href="styl.css" type="text/css"/>
		<style type="text/css">
		</style>
	</head>
	<body>
	<p>Tutaj znajduje się przykładowy tekst strony WWW. Znajduje się on w centrum znacznika. Dookoła tekstu znajduje się obicie (z każdej strony) o rozmiarze 20 pikseli.
```

Obramowanie posiada rozmiar 5 px. Brzeg znacznika przesunięty został względem pozostałych elementów (w naszym wypadku od brzegów głównej przestrzeni strony) o 50 pikseli.
	</body>
</html>

W powyższym przypadku paragraf będzie miał łącznie 850 pikseli szerokości (wartość width + 2 * padding + 2 * border + 2 * margin) oraz 350 pikseli wysokości (wartość height + 2 * padding + 2 * border + 2 * margin). Wartość wysokości czcionki nie ma żadnego znaczenia dla ogólnej wielkości elementu. Wartości poszczególnych części bloku są przemnażane przez dwa ze względu na doliczanie odpowiednio lewej i prawej krawędzi (szerokość) lub górnej i dolnej krawędzi (wysokość) elementu.

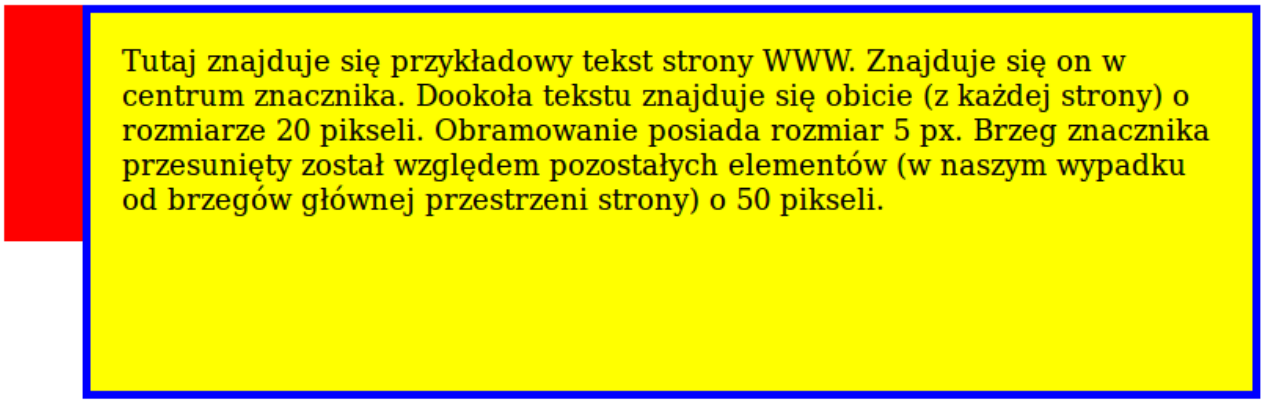
2. Przestrzeń elementów HTML w CSS

Wszystkie widoczne elementy HTML zajmują miejsce w ciele strony (w znaczniku body). Projektant strony może dowolnie ustalać ich pozycję, wielkość a nawet umiejscowienie na osi z (patrz stos).
Podstawowymi właściwościami każdego z obiektów HTML są:
- width – sztywno określa szerokość elementu
- height – sztywno określa wysokość elementu
- min-width – określa minimalną szerokość elementu (nawet jeżeli jego treść będzie mniejsza)
- min-height – określa minimalną wysokość elementu (nawet jeżeli jego treść będzie mniejsza)
- max-width – określa maksymalną szerokość elementu (nawet jeżeli jego treść będzie większa)
- max-height – określa maksymalną wysokość elementu (nawet jeżeli jego treść będzie większa)

Obiekt dla którego zostaną ustawione powyższe właściwości (wyłączając w to min-height oraz min-width) może zawierać w sobie kolejne elementy posiadające swoje ustawienia dotyczące jego wielkości. Jeżeli wielkość wpisanego elementu (dziecka) będzie większa od elementu nadrzędnego (rodzica) wtedy może się zdarzyć, że dziecko przekroczy granicę rodzica (nie zwiększając jego oryginalnie zaprojektowanej wielkości). Aby zapobiec takiej sytuacji dobrze jest zastosować dodatkowo właściwość overflow: auto. Dzięki temu w opisywanej sytuacji rodzic otrzyma pasek suwaków, dzięki którym można będzie przewinąć jego zawartość, a dziecko nie „wyjdzie” poza jego ramy.

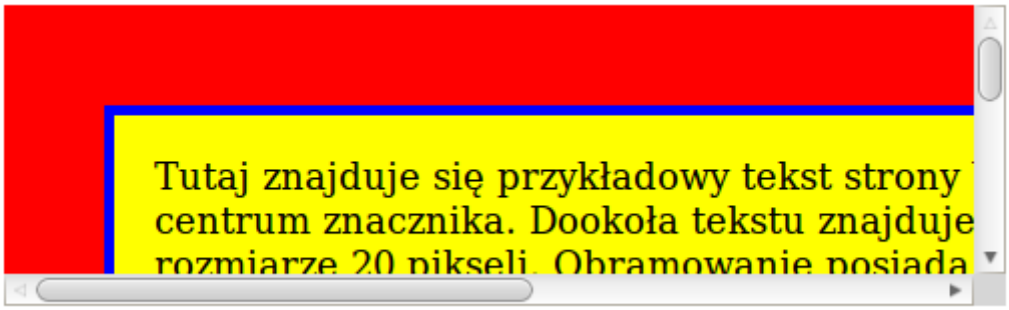
```
//plik styl.css
div {
	width: 500px;
	height: 150px;
	background-color: red;
}
p {
	background-color: yellow;
	font-size:18px;
	padding: 20px;
	border: 5px;
	width: 700px;
	height: 200px;
	border-style: solid;
	border-color: blue;
	margin: 50px;
}
//plik index.html
<!DOCTYPE html>
<html lang="pl">
```

```
	<head>
		<meta charset='UTF-8'/>
		<title>Style CSS</title>
		<link rel="stylesheet" href="styl.css" type="text/css"/>
		<style type="text/css">
		</style>
	</head>
	<body>
<div>
	<p>Tutaj znajduje się przykładowy tekst strony WWW. Znajduje się on w centrum znacznika. Dookoła tekstu znajduje się obicie (z każdej strony) o rozmiarze 20 pikseli. Obramowanie posiada rozmiar 5 px. Brzeg znacznika przesunięty został względem pozostałych elementów (w naszym wypadku od brzegów głównej przestrzeni strony) o 50 pikseli.</p>
</div>
	</body>
</html>
```



po zmianie stylu warstwy na następujący:

```
div {
	width: 500px;
	height: 150px;
	background-color: red;
	overflow: auto;
}
```

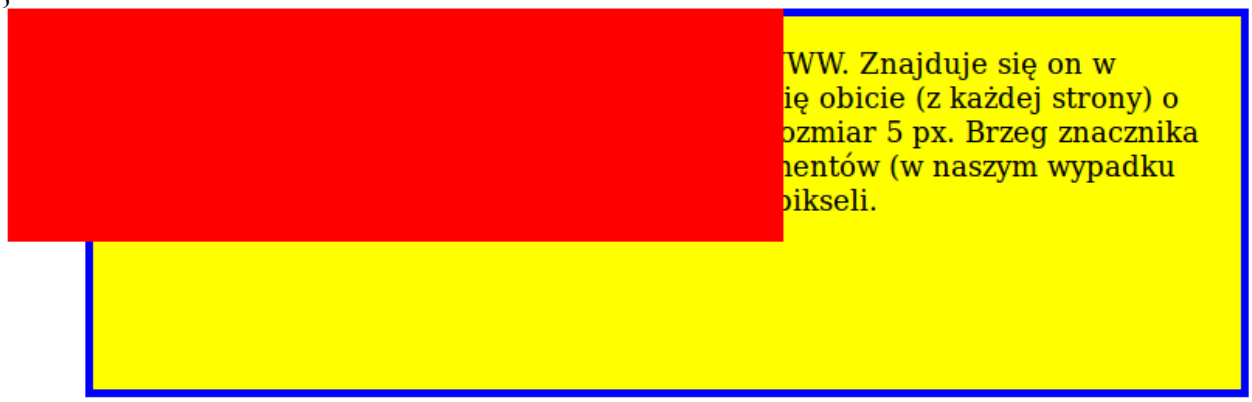


Dodatkową właściwością jest z-index. Dzięki niej możemy określić na jakim poziomie (względem obserwatora) znajdzie się element HTML względem pozostałych elementów. Umożliwia to np. „schowanie” jednego paragrafu tekstu za drugi bądź przeniesienie menu strony na najwyższy, względem pozostałych obiektów strony, poziom (by żaden element go nie przykrył).

```
//plik index.html
<!DOCTYPE html>
```

```
<html lang="pl">
	<head>
		<meta charset='UTF-8'/>
		<title>Style CSS</title>
		<link rel="stylesheet" href="styl.css" type="text/css"/>
		<style type="text/css">
		</style>
	</head>
	<body>
<div></div>
	<p>Tutaj znajduje się przykładowy tekst strony WWW. Znajduje się on w centrum znacznika. Dookoła tekstu znajduje się obicie (z każdej strony) o rozmiarze 20 pikseli. Obramowanie posiada rozmiar 5 px. Brzeg znacznika przesunięty został względem pozostałych elementów (w naszym wypadku od brzegów głównej przestrzeni strony) o 50 pikseli.</p>

	</body>
</html>
//plik styl.css
div {
	width: 500px;
	height: 150px;
	background-color: red;
}
p {
	background-color: yellow;
	font-size:18px;
	padding: 20px;
	border: 5px;
	width: 700px;
	height: 200px;
	border-style: solid;
	border-color: blue;
	margin: 50px;
	position: relative;
	top: -200px;
	z-index: -1;
}
```



Elementy mogą być pozycjonowane względem siebie na kilka sposobów. Najprostszym, a zarazem najczęściej stosowanym jest użycie właściwości float (opływ, przepływ). Obiekty mogą przepływać

jedynie poziomo – nie można określać opływu dla ułożenia pionowego.
Elementy mogą być łączone od prawej bądź od lewej strony. Przykładowo jeżeli dla znaczników ustawimy właściwość „float: right" wtedy pierwszy z nich będzie przyłączony do prawej krawędzi obiektu nadrzędnego strony (rodzica; jeżeli element znajduje się bezpośrednio w znaczniku body wtedy zakotwiczenie nastąpi do prawej strony ekranu), a następny będzie przyłączony do prawej krawędzi poprzednika (i tak kolejno następne). Jeżeli zdarzy się, że łączna szerokość elementów jest większa niż szerokość powierzchni obiektu nadrzędnego (rodzica) wtedy następne dodawane dziecko będzie przeniesione do następnej linii, a jego zakotwiczenie zostanie potraktowane jak pierwszego elementu (czyli zakotwiczy się do wskazanej krawędzi elementu nadrzędnego).
Poszczególne kontenery można wyłączyć z ciągu przepływu – wystarczy nadać im właściwość „clear: both".
Dostępne właściwości dla przepływu:
- clear – określa, która ze stron elementu ma zostać wyłączona z łączenia w przepływy (od której strony nie można będzie zakotwiczać elementów). Dostępne wartości:
1) left – przepływ nie będzie mógł zaczynać się od lewej strony
2) right – przepływ nie będzie mógł zaczynać się od prawej strony
3) both – przepływ nie będzie mógł zacząć się od żadnej ze stron (obiekt z tym ustawieniem będzie zawsze rozpoczynał się w nowej linii)
4) none – parametr zostaje wyzerowany (obiekt przyjmuje domyślne parametry przepływu i pojawia się w miejscu, w którym został ustawiony)

- float – określa krawędź, po której elementy mają być ze sobą łączone. Dostępne wartości:
1) left – element powinien przepływać od lewej strony do prawej
2) right – element powinien przepływać od prawej strony do lewej
3) none – parametr zostaje wyzerowany; wartość ta działa identycznie jak w przypadku właściwości clear

Oczywiście zarządzanie przepływem elementów nie jest efektywnym sposobem projektowania wyglądu strony. W celu precyzyjnego ustawiania składowych strony tam, gdzie dokładnie chcemy i potrzebujemy, służy parametr position. Dzięki niemu każdy z obiektów, taki jak tekst, obraz, czy cały kontener może być ustawiony w ściśle określonym miejscu edytowanego dokumentu. Ponadto może on pozostawać stale w jednym miejscu strony (pomimo przesuwania pozostałej treści strony), być ustawiony jednoznacznie (absolutnie) względem danego elementu strony (ustawienie względem pierwszego elementu znajdującego się w kontenerze (rodzicu); jeżeli takowy element nie istnieje opcja ta ustawia element względem głównego elementu dokumentu HTML bądź relatywnie względem oryginalnej pozycji, w jakiej miał być wyświetlony opisywany obiekt.
Dostępne właściwości dla pozycjonowania elementów HTML:
- position – określa rodzaj metody pozycjonowania obiektów HTML. Dostępne wartości:
1) static – domyślna metoda pozycjonowania; element pojawia się w miejscu wystąpienia (w tej samej kolejności) z domyślnym przepływem pozostałych treści elementów
2) absolute – element ustawiany jest względem pierwszego pozycjonowanego elementu na stronie (czyli nie mającego ustawionej właściwości position: static); jeżeli taka treść nie istnieje (wszystkie są statyczne) wtedy element ustawiany jest względem elementu głównego strony HTML
3) fixed – element ustawiany jest względem wymiaru okna przeglądarki
4) relative – element ustawiany jest względem swojej oryginalnej, pierwotnej pozycji, tj. przesunięcie go o 20 pikseli w lewo nastąpi względem jego oryginalnej pozycji na stronie.

- top, left, right, bottom – określa margines od (kolejno) górnego, lewego, prawego oraz dolnego brzegu elementu (czyli odpowiednie przesunięcie go bądź to od jego oryginalnej pozycji bądź to od pozycji najbliższego sąsiada). Dostępne wartości:
1) auto – przeglądarka sama określa wskazany brzeg; domyśle zachowanie
2) <wartość liczbowa> - określa sztywne odsunięcie od wskazanych krawędzi (w miarach CSS,

uwzględniając %). Można nadawać wartości ujemne

- clip – przycina zawartość elementu do wskazanego rozmiaru. Właściwość ta jest przydatna w przypadku, gdy element może posiadać zawartość z większym rozmiarem niż zostało założone w projekcie. Należy jednak pamiętać iż zawartość zostanie UCIĘTA, a nie przeskalowana. Dostępne wartości:
1) rect(top, right, bottom, left) – przycina obiekt od brzegu: górnego(top), prawego (right), dolnego (bottom), lewego (left). Wartości mogą być podawane we wszystkich miarach dostępnych w CSS3
2) auto – zawartość nie zostanie przycięta; domyślna konfiguracja.

- overflow – właściwość została opisana w instrukcji nr 2, punkt 3, podpunkt b)

Dzięki CSS można również chować (ukrywać) jak i usuwać wskazane elementy strony WWW. Elementy ukryte nadal zajmują swoje miejsce na stronie WWW podczas gdy usunięte je zwalniają. Usunięcie elementu uzyskuje się poprzez dodanie właściwości:
display: none;
Efekt ukrycia uzyskuje się natomiast poprzez:
visibility: hidden;
Właściwości display oraz visibility zostaną opisane w ostatnim punkcie instrukcji.

3. Pseudoklasy i pseudoelementy.

Pseudoklasy i/lub pseudoelementy stanowią integralną część znaczników (selektorów) HTML. Umożliwiają nadanie im specjalnych właściwości i efektów. Umożliwiają również zmiany właściwości pseudoklasy/pseudoelementu dla różnych klas wskazanego znacznika HTML.
Ogólna składania:
selector:pseudo-klasa {wlasciwosc: wartosc;}
Przy wykorzystaniu klasy selektora:
selector.wlasnaklasa:pseudo-klasa {wlasciwosc:wartosc;}

Najlepszym przykładem znacznika z pseudoklasami jest kotwica <a>; posiada jedną pseudoklasę na swój stan – przed odwiedzeniem (:link), po odwiedzeniu (:visited), mysz nad odnośnikiem (:hover) oraz gdy odnośnik jest aktualnie wybrany (:active).

```
a:link {
	color: red;
}
a:visited {
	color: orange;
}
a:hover {
	color: yellow;
}

a:active {
	color: black;
}
```

Trzeba pamiętać o kolejności podawania pseudoklas dla kotwicy (powinna być taka jak w przykładzie). Jeżeli ją zmienimy możemy nie uzyskać zamierzonego przez nas efektu.

Pozostałe pseudoklasy/pseudoelementy:
- :focus – służy do określenia stanu elementu w chwili, gdy zostanie zogniskowana na nim aktywność użytkownika (przykładowo wybrany zostanie element do wprowadzania tekstu)
- :first-letter – służy do zmiany stylu pierwszej litery każdego kolejnego elementu blokowego zawierającego tekst.
- :first-line – służy do zmiany stylu pierwszej linii każdego kolejnego elementu blokowego zawierającego tekst.
- :first-child – służy do zmiany stylu pierwszego elementu blokowego w danym kontenerze (rodzicu).
- :before – służy do wstawiania zawartości przed zawartością właściwą elementu, do którego pseudoklasa się odwołuje. Aby ją skutecznie wykorzystać należy w implementacji dodać właściwość content (opisana później)
- :after – służy do wstawiania zawartości po zawartości właściwej elementu (działa jak :before)
- :lang(język) – służy do zmiany stylu elementów, które napisane są we wskazanym języku. Atrybut „język" podaje się przeważnie w postaci dwuliterowego kodu, np. :lang(pl), :lang(it).

4. Tworzenie własnych atrybutów CSS.

Arkusze stylów umożliwiają tworzenie własnych atrybutów znaczników HTML. Będą one zmiennymi, dzięki którym można identyfikować kolejne pozycje strony WWW. Tworzone atrybuty mogą posiadać swoje wartości, za pomocą których możliwe jest odpowiednie indywidualne dostosowanie właściwości poszczególnych elementów strony wedle własnych potrzeb.
Atrybuty w CSS deklaruje się w następujący sposób:

```
[atrybut=wartosc]
{
wlasciwosc: wartosc;
}
```

Przykładowa, najprostsza deklaracja z definicją:

```
<!DOCTYPE html>
<html>
	<head>
	<style type="text/css">
		[zmienna] {
		color: green;
		}
	</style>
	</head>
	<body>
		<p zmienna="">Tekst z pierwszego paragrafu</p>
		<p>Tekst z drugiego paragrafu</p>
		<p zmienna="tekst">Tekst z trzeciego paragrafu</p>
	</body>
</html>
```

Jak widać nie gra roli jaką wartość podamy dla utworzonego przez nas atrybutu CSS (właściwość znacznika HTML). Oba paragrafy, dla których użyto utworzonego atrybutu otrzymały zielony kolor tekstu.

Atrybut może działać tylko na te znaczniki, dla których ustawiono jego odpowiednią wartość:

```
<!DOCTYPE html>
<html>
	<head>
	<style type="text/css">
		[zmienna=tekst] {
		color: green;
		}
	</style>
	</head>
	<body>
		<p zmienna="">Tekst z pierwszego paragrafu</p>
		<p>Tekst z drugiego paragrafu</p>
		<p zmienna="tekst">Tekst z trzeciego paragrafu</p>
	</body>
</html>
```

Nie gra roli czy podczas deklaracji atrybutu wartość podamy w cudzysłowie, apostrofach czy bez znaczników tekstu (jak w przykładzie) – tylko elementy HTML z utworzonym atrybutem oraz wskazaną wartością otrzymają odpowiedni styl CSS (ostatni paragraf).

Może się zdarzyć, że chcemy by dany styl został zastosowany wtedy, gdy atrybut zawiera (lecz niekoniecznie jest równy) daną wartość:

```
<!DOCTYPE html>
<html>
	<head>
	<style type="text/css">
		[zmienna~="tekst"] {
		color: green;
		}
	</style>
	</head>
	<body>
		<p zmienna="wartosc">Tekst z pierwszego paragrafu</p>
		<p>Tekst z drugiego paragrafu</p>
		<p zmienna="jakiś tekst ">Tekst z trzeciego paragrafu</p>
		<p zmienna="tekst">Tekst z czwartego paragrafu</p>
	</body>
</html>
```

Jak widać wystarczy dodać znak tyldy (~) przed znakiem równości by zastosować wskazany styl da dwóch paragrafów (jeden z nich zawiera wartość 'tekst' wraz z innym wyrazem, drugi natomiast posiada tylko wyraz 'tekst').

Jeżeli chcemy, by tworzony styl został zastosowany dla wskazanej grupy językowej (przykładowo en, en-us, en-gb itp.) można zastosować następującą konstrukcję atrybutu:

```
<!DOCTYPE html>
<html>
	<head>
	<style type="text/css">
		[lang|=en] {
		color: green;
```

```
		}
	</style>
	</head>
	<body>
		<p lang="en">Tekst z pierwszego paragrafu</p>
		<p lang="en-us">Tekst z drugiego paragrafu</p>
		<p lang="pl">Tekst z trzeciego paragrafu</p>
		<p lang="en-gb">Tekst z czwartego paragrafu</p>
		<p lang="gb">Tekst z piątego paragrafu</p>
	</body>
</html>
```

Należy pamiętać, że konstrukcja ta (operator |=) możliwa jest do zastosowania także dla innych atrybutów, jednak działa poprawnie jedynie z wartościami oddzielanymi znakiem '-'.

Można także określić konkretny znacznik, dla którego styl ma zostać zastosowany

```
<!DOCTYPE html>
<html>
	<head>
	<style type="text/css">
		span[znacznik] {
		color: green;
		}
		p[znacznik] {
		color: blue;
		}
	</style>
	</head>
	<body>
		<p znacznik="">Tekst z pierwszego paragrafu</p>
		<span>Tekst z drugiego paragrafu</span>
		<span znacznik="">Tekst z trzeciego paragrafu</span>
		<p>Tekst z czwartego paragrafu</p>
		<p znacznik="">Tekst z piątego <span znacznik="">paragrafu</span></p>
	</body>
</html>
```

W powyższym przykładzie atrybuty w nawiasach [] można stosować w dowolnej konfiguracji opisanej we wczesniejszych przykładach.

Jeżeli chcemy pobrać wartość tworzonego przez nas atrybutu i wykorzystać ją jako wartość dla innej właściwości CSS należy użyć funkcji attr() :

```
<!DOCTYPE html>
<html>
	<head>
	<meta charset='UTF-8'/>
	<style type="text/css">
		p[zmienna]:before {
		color: green;
		content: attr(zmienna);
		}
```

```
	</style>
	</head>
	<body>
		<p zmienna="wartosc">Tekst z pierwszego paragrafu</p>
		<p>Tekst z drugiego paragrafu</p>
		<p zmienna="jakiś tekst ">Tekst z trzeciego paragrafu</p>
		<p zmienna="tekst">Tekst z czwartego paragrafu</p>
	</body>
</html>
```

Pobranie wartości może nastąpić z dowolnego (niekoniecznie własnego) atrybutu HTML. Warunkiem jest jednak jego wcześniejsza deklaracja i definicja w modyfikowanym znaczniku na stronie WWW.

5. Wybrane przydatne właściwości CSS

1) cursor – jeżeli kursor myszy znajdzie się nad wskazanym elementem można dokonać zmiany jego wyglądu na jeden z predefniowanych (systemowy) bądź własny. Ustawienie własnego kursora: - cursor: url(Sciezka/nazwaPlikuGraficznego); - kursor przybierze postać obrazu spod wskazanego pliku graficznego. Trzeba pamiętać o odpowiednim dobrze pliku – by nie był za duży, a tym samym zbyt długo się nie ładował! Ponadto, jeżeli przeglądarka z jakichś przyczyn nie będzie mogła załadować wskazanego pliku dobrze jest podać po przecinku wartość auto (ustawia domyślne zachowanie kursora), np.
cursor: url(obrazek.png), auto;
Obrazków możemy podać kilka (w różnych formatach, na wypadek gdyby przeglądarka miała problem z którymkolwiek formatem):
cursor: url(obrazek.png), url(obrazek.cur), auto;
Pozostałe możliwe wartości dla kursora: default, crosshair, help, move, pointer, wait, text, progress, e-resize, w-resize, n-resize, ne-resize,nw-resize,s-resize, se-resize,sw-resize.

2) counter-increment – atrybut definiuje nowy licznik dla wskazanego znacznika. Za każdym wystąpieniem danego znacznika wartość licznika zmienia się o 1 (+1). Możliwe wartości to: none, lub nazwa/id licznika (dowolny, jednak UNIKATOWY ciąg znakowy BEZ spacji)

3) counter-reset – atrybut resetuje wskazany licznik (jako wartość podaje się jego nazwę).

4) content – właściwość ta ma zastosowanie głównie dla pseudoelementów :before oraz :after. Dzięki niej możemy wstawić pożądaną przez nas zawartość przed lub za wskazanym elementem strony WWW. Najprostrzą wartością tego parametru jest oczywiście podanie dowolnego ciągu znakowego. Pozostałe możliwe wartości:
- attr(nazwaAtrybutu) – ustawia wartość podanego atrybutu jako tekst przed/za wskazaną częścią strony
- open-quote – wstawia przed/za zawartością symbol (znak) otwieranego cudzysłowu
- close-quote – wstawia przed/za zawartością symbol (znak) zamykanego cudzysłowu
- no-open-quote/no-close-quote – jeżeli dla danego znacznika zostały zdefiniowane pseudoelementy :after lub :before z opcjami open-quote/close-quote wtedy, przykładowo dla jego jednej z klas tego znacznika można wyłączyć cudzysłowy poprzez podanie opisywanych wartości:
- url(sciezka/nazwapliku) – jak w przypadku atrybutu curosr, tak i tutaj można podać plik do załadowania przed/po wskazanym elemencie. W tym jednak przypadku plik może być dowolnego typu – obraz, dźwięk, wideo i inne.
- normal – tak jak w przypadku z cudzysłowami – jeżeli wcześniej dla któregokolwiek znacznika zdefiniowana została jakakolwiek akcja :after/:before to ustawienie dla klasy tego znacznika

niweluje ją (tożsame z wartością none)
- none – ustawia zerową zawartość dla :after/:before wskazanego znacznika.
- counter – wstawia zawartość aktualnego licznika zdefiniowanego dla wskazanego znacznika HTML

5) visibility – właściwość ta decyduje czy wskazany element WWW ma być widoczny na stronie czy nie (zmiana widoczności elementu NIE POWODUJE jego usunięcia – nadal zajmuje on swoje miejsce w przestrzeni HTML). Dostępne wartości:
- visible – domyślne ustawienie; sprawia, że element jest widoczny na stronie
- hidden – element na stronie przestaje być widoczny (nadal zajmuje miejsce)
- collapse – wartość działa jedynie dla elementów tabel. Usuwa wskazany wiersz bądź kolumnę jednak nie wpływa w żaden sposób na szablon tabeli. Uzyskana przestrzeń staje się dostępna dla pozostałej zawartości tabeli. Jeżeli wartość zostanie użyta dla elementu nie będącego częścią tabeli to spowoduje to jego ukrycie.

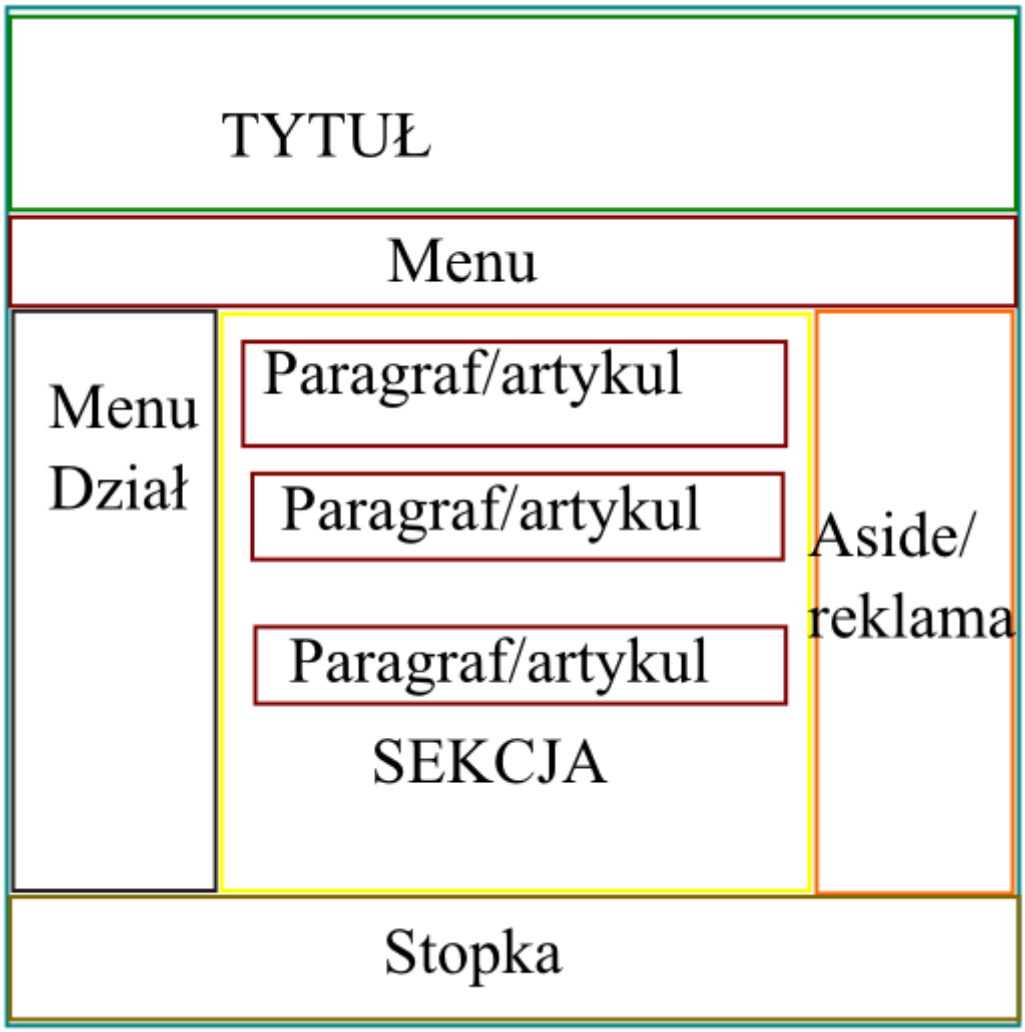
6) display – atrybut określa w jakim kontekście/w jaki sposób mają zostać wyświetlone elementy HTML. Dostępne wartości:
- inline – wyświetla elementy liniowo (jeden po drugim; przykładowo jest to domyślne zachowanie elementu blokowego). Domyślne ustawienie.
- block – wyświetla elementy jako elementy blokowe (jesto to domyślne zachowanie np. elementu blokowego <p>).
- inline-block – element z tą wartością będzie zachowywał się jako kontener elementów ustawiony liniowo. Elementy w nim zawarte będą zachowywać się jako elementy blokowe.
- inline-table – element wyświetlony zostanie jako część liniowa tabeli
- list-item – element zachowa się identycznie jak znacznik
- run-in – w zależności od kontekstu obiekt zostanie wyświetlony jako liniowy bądź blokowy
- table – element zachowa się identycznie jak znacznik <table>
- table-caption – równoważne ze znacznikiem <caption>
- table-column-group – równoważne ze znacznikiem <colgroup>
- table-header-group – równoważne ze znacznikiem <thead>
- table-footer-group – równoważne ze znacznikiem <tfoot>
- table-row-group – równoważne ze znacznikiem <tbody>
- table-cell – równoważne ze znacznikiem <td>
- table-column – równoważne ze znacznikiem <col>
- table-row – równoważne ze znacznikiem <tr>
- none – element nie zostanie wyświetlony (nie będzie także zajmował przestrzeni na stronie)

7) opacity – określa przezroczystość wskazanego elementu strony. Przyjmuje wartość od 0.0 (pełna przezroczystość) do 1.0 (nieprzezroczysty). Aż do wersji 8.0 Internet Explorer NIE ROZPOZNAJE tego atrybutu; w związku z powyższym dla tej przeglądarki należy użyć właściwości:

filter: Alpha(opacity=100)

gdzie wartość 100 oznacza nieprzezroczystość, a 0 – pełną przeźroczystość.

ZADANIA:
1. Należy przeprojektować posiadaną stronę (i jej wszystkie podstrony) w taki sposób, by strona posiadała swoje poszczególne elementy np. jak na poniższym zrzucie obrazu. Oczywiście projekt można potraktować jako inspirację i stworzyć własny.



2. Dobrym rozwiązaniem jest „ścieśnić” stronę tak, by np. zajmowała jednie 80% dostępnej szerokości (lepsza czytelność). Dodatkowo cała zawartość strony powinna zostać wyśrodkowana (jak wycentrować elementy na stronie za pomocą CSS można znaleźć na tej stronie http://www.w3schools.com/css/css_align.asp).
3. Proszę zmodyfikować stronę galerii. Do istniejących już obrazów należy dodać jeszcze 10 innych (można powtórzyć już istniejące zdjęcia – jednak w innej kolejności!). Mając do dyspozycji 20 obrazów należy pogrupować je w 4 galerie po 5 elementów każda. Galerie powinny zawierać opis (np. 'Galeria 1', 'Galeria 2' itd.). Dla każdego obrazu w danej galerii proszę ustawić jednakową wysokość i szerokość, jednak każda galeria powinna mieć inne ustawienia wielkości obrazów. Zdjęcia na stronie galerii powinny być uporządkowane poprzez atrybut float.
4. W formularzu kontaktowym proszę dodać podświetlenie aktualnie wybranego pola tekstowego na dowolny kolor.
5. Jeżeli mysz znajdzie się nad dowolnym elementem w menu strony to kursor ma przybierać postać inną niż domyślna.
6. Proszę spróbować przebudować stworzone poprzednio tabele HTML tak, by używały jedynnie stylów CSS.