**Css – podstawy**

**Podstawowy patern:**selector{

Property: value;

}

**Gdzie piszemy css’a** – w atrybutach html’a jako style=”property: value” (niepolecane).   
**<style></>** - w headzie. Można umieszczać tutaj standardowego css’a

**<link href=”{link do pliku .css ze stylami}”>** - umieszczany w headzie. Używany do podpinania plików z określonym zachowaniem css

**Podstawowe własności:**

**Color** – zamiena kolor tekstu. Wartości kolorkami

**Background-color** – zamienia kolor bloku za tekstem (dla h1 to np. do końca wiersza)

**Rodzaje zapisu kolorów:   
nazwowe** – podstawowe kolory mają określoną nazwę (np. red, blue)

**RGB** – rgb({wartość red},{wartość green},{wartość blue}) -> rgb(173,20,219)

**Hex(hexadecimal)** - #ffff00 2\*{red},{green},{blue}. Jeśli każdy z elementów jest duplikatem (#00ffee) można zapisać jako #0fe

**Podstawowe właściwości tekstu:**

**text-align** – przesuwa nasz tekst. Wartości to right/left/center/justify

**font-weight** – określa grubość tekstu od 100-900 (400 normal/700 bolded). Wartość dodatkowe to normal, bold, lighter, bolder

**text-decoration** – określa dodatkowe elementy do tekstu (podkreślniki etc.). Wartości to underline/overline/overline/none. Do tych elementów można dodać określenie koloru jak i ich własne właściwości jak np. dotted/wavy

**line-height** – wysokość pomiędzy wierszami (bez entera)

**letter-spacing** – odległość pomiędzy literkami

**Font size i wielkości:**

**px** – średnio rekomendowane. Różni się w zależności od maszyny

**font-family** – podajemy jako wartość nazwę fontu którego chcemy użyć. Możemy stworzyć jako taką rodzinę dodając przecinki po wartościach jako backup. Istnieją też gotowe rodziny które można wrzucić

**Strona z kolorkami** - https://coolors.co/palettes/trending

**Selectors**

**Podstawowe selectory:**

**\*{…}** – zaznacza wszystko

**img{…}** – zaznacza wszystkie zdjęcia

**h1,h2{…}** – można szybko edytować kilka elementów na raz

**#{id}{…}** – każdemu elementowi możemy dodać atrybut id. W html’u używany do m.in. labelów. Tutaj możemy edytować te elementy indywidualnie. Nie przesadzać z używaniem

**.{id klasy}{…}** – każdemu elementowi możemy dodać atrybut class. Jedno id może występować w wielu obiektach i dzięki temu na raz można je zmieniać. Częściej używane

**Wyszukiwanie elementów na podstawie ich pozycji:**

**{typ/typy narzędne oddzielone spacją} {typ docelowy}{…}** – możemy wprowdzać zmiany tylko na elementach znajdujących się wewnątrz jakiś elementów. Np. **.temp li a** odnosi się do *anchortagów* w *elementach* *listy* w klasie *temp*

**{typ sąsiadujący} + {typ docelowy}{…}** – dotyczy elementów, które następują tuż po czymś  
**{typ nadrzędny} > {typ docelowy}{…}** – dotyczy elementów które znajdują się bezpośrednio pod nimi, a nie jeszcze niżej w hierarchi

**Zaawansowane:**

**{typ}[ {atrybut} = ”” ]{…}** – określa elementy, których dodatkowo jest spełniony atrybut. Dodatkowo używamy „=” jako porównanie, „\*=” jako zawieranie

**{typ}:{atrybut}{…}** – określa elementy w momencie jakiejś akcji (pseudo classes). Np:   
**{typ}:hover** – co ma się dziać gdy najedziemy **{typ}:active** – co ma się dziać gdy klikniemy

**{typ}:nth-of-type(value/$n){…}** – co ile coś ma się dziać. Jeśli damy liczbę to dane wystąpienie zostanie zamienione, jeśli jakieś n to co ileś wystąpień

**{typ}::{atrybut}{…}** – określa elementy w tym danym bycie (pseudo elements). Np:

**{typ}::selection** – decyduje o zaznaczaniu elementu na stronie (przytrzymanie lewego klawisza myszy)

**{typ}::placeholder** – placeholder

**{typ}::first-letter** – pierwsza litera

**{typ}::first-line** – pierwszy wiersz

**Specificity:**

Kolejność ma znaczenie. Elementy bardziej specyficznie określone mają większy priorytet. Id > class > element. Istotne jedynie na wypadek konfliktu. Bez niego element z mniej specyficznego jest użyty do bardziej specyficznego (przechodzi)

**!important** – można użyć (lepiej nie) po *properties:value* ***!important;*** daje nam to opcje nadpisania wszystkich priorytetów

Bazowo większość właściwości przekazywane są od rodziców, jeśli w pojedynczych typach nie zostały określone

**Box model**

Zasadniczo każdy element jest tak naprawdę boxem. Do niego zalicza się wnętrze, padding i obramowanie.

**Context box:**

**width** – kontroluje szerokość wnętrza

**height** – kontroluje wysokość wnętrza

**Border:**

Istotne w ramach podkreślania istotnych elementów. Choćby jeśli chodzi o pokazanie na czym jest myszka, czy obecnie wybrane okno. Do stworzenia takowej potrzebne są wszystkie poniższe elementy:

**border-width** – określa grubość ramki

**border-color** – kolor ramki. Wartości to sandardowe kolory

**border-style** – typ ramki. ***solid*** – stała; ***dotted*** - kropki

**border-radius** – określa zaokrąglenie boxu. Wartości w px lub %. Procent tutaj to kombinacja height i width

***Do każdego z tych elementów możemy się odwoływać pojedynczo(left/right etc)***

**box-sizing** – to czy ramka ma wychodzić poza context box czy nie. Jeśli ma być do środka to ***border box***. Bazowo tak nie jest

**border: {*width*} {*style*} {*color*}** – łatwiejszy sposób stworzenia ramki

**Padding:**

Używane do ładniejszego rozmieszczania elementów i nie nakładania na siebie elementów.   
**padding: {wartości}** – mamy parę różnych opcji wprowadzania. ***value = {px}*** dodaje po każdej ze stron; ***value = {px px}*** vertical i horizontal; ***value = {px px px px}*** top, right, bottom, left (zgodnie ze wskazówkami zegara). Można też oczywiście użyć ***padding-left*** etc

**Margin:**

Używane do oddzielania elementów od siebie na zewnątrz (jest za ramką).   
**margin: {wartości}** - mamy parę różnych opcji wprowadzania. ***value = {px}*** dodaje po każdej ze stron; ***value = {px px}*** vertical i horizontal; ***value = {px px px px}*** top, right, bottom, left (zgodnie ze wskazówkami zegara). Można też oczywiście użyć ***padding-left*** etc

***Ważne:*** *Body* bazowo ma ustawione pewne margin. Warto na początku tworzenia stony je usunąć, żeby co najwyżej w przyszłości ręcznie je dodać

**Display:**

Pozwala nam zmieniać rodzaj z block na inline etc.

**display: {wartości}** – przykładowe wartości to **none; block**; **inline**; **inline-block** – dodaje działanie margin i height

***Ważne:*** na inline nie działa ***height*** oraz pionowo ***margin***. Na **block** oczywiście działa

**CSS units:**

**Relatywne:**

em używane kiedy chcemy mieć coś skalarnego, rem łatwiej przewidzieć **%** - jest relatywne do odpowiednich wartości (np. wielkości rodzica w przypadku ***height*** etc)

**line-height** – procent wielkości tekstu

**em** – polega na pobieraniu wielkości od rodzica. 1em oznacza tą samą wielkość, 2em 2x większą etc. Przydatne do ***margin***

**rem** – zasadniczo jak px tylko, że zawsze ma taką samą wartość. Możemy też to zmieniać w .css **html {font-size: [value]}**

**Other CSS properties**

**Opacity & alpha chanels:   
rgba(...,{alpha value})** – dodatkowa wartość do standardowego rgb. Od 0-1, gdzie 0 jest całkowicie przeźroczyste, a 1 pełne

**opacity** – kontroluje przeźroczystość wszystkiego znajdującego się w danym obiekcie. Wartości od 0 do 1

**Position:**

Decyduje o zachowaniu i pozycji elementu. Wartości to:   
**static** – bazowa wartość. Element zachowuje się „normalnie”  
**relative** – na podstawie pierwotnej pozycji można dokonywać zmian w pozycji

**absolute** – usunięte z normalnego flow (tak jakby lata i nie wpływa na strukture innych elementów). Poyzcja względem rodzica, lub też po prostu pozycji pliku (lewy górny)

**fixed** – względem pozycji rodzica lub początku pliku jeśli nieokreślona. Zostaje na ekranie w tym samym miejscu  
**sticky** - Najpierw zachowuje się jednak jako static, a później zmienia się w fixed przy scrolowaniu. Jednak będzie się poruszać tylko w ramach swojego własnego kontenera  
**Ważne** – wszystkie pozycje wyżej zapisane (poza *static*) możemy zmieniać korzystając z **top**, **bottom**, **left**, **right**. Które decydują o przesunięcu **OD** danej własności

**Transitions:**Używane do przejść pomiędzy dwoma stylami jednego bytu

**transition** – Przykładowe wartości:   
**{sekundy}** - wartości to sekundy(2s) określające jak długo ma trwać zmiana

**{property} {sekundy}** – używane do zmiany pojedynczych właściwości. Property może być **all.** Warto oddzielnie je zapisywać

**{property} {sekundy} {delay}**

**transition-timiny-function** – do określenia sposobu zmiany. Przykładowe wartości to:   
**linear** – standardowe

**ease-int**

**steps(n)**

**Ważne:** grupujemy zapisując je obok siebie używając przecinka

**Transform:**

**transform-origin** – środek wykonanania zmiany

**transform** - Używane do przesuwania elementów. Przykładowe wartości:

**rotate({deg})** – obracanie elementów

**scale()** – do zmieniania wielkości obiektu. Wartości to jakby *k* w matematyce

**translate({x},{y})** – do zmieniania pozycji obiektu

**Ważne:** grupujemy zapisując je obok siebie używając spacji

**How to center a div:**body{

display:flex;

alling-content:center;

justify-content:center;

height: 100vh;

}

**Shadows:**

**box-shadow** – używane do twożenia cieni pod obiektami. Wartości to **{x} {y} {blur radius} {spread radius} {color}**. Np *box-shadow: 2px 2px 2px 1px red;*

**cursor** – używane do zmienienia ikonki kursona. Wartości to m.in. ***pointer, move***

**Background:**Istnieje więcej opcji edytowania tła. Standardowo można zmieniać kolor, ale także wrzucać zdjęcia w różny sposób etc.

**STRONA Z DARMOWYMI ZDJĘCIAMI: https://unsplash.com/**

**background-image: url({url})** – podanie linku do zdjęcia

**background-size** – pozwala zmieniać sposób wyświetlania zdjęcia. Wartości:  
**cover** – tak duże jak możliwe. Przycina

**contain** – tak duże jak możliwe. Nieprzycina (wrzuca cały + kopie części)

**%** - procent wielkości bazowego obrazka

Możemy jeszcze decydować o tym czy **dany obrazek się powtarza czy nie**

**background** – uproszczony sposób zapisu. Można mieszać ze sobą kolejność, ale jeśli chcemy użyć właściwości ***position*** i ***size*** to musi to wyglądać na zasadzie ***{position}/{size}*** np.   
*background: url(„link”) no-repeat center/cover;*

**Google Fonts:**

**fonts.google.com**

Fonty są drogie. Google daje popularne i ładne fonty za darmo. Wrzucamy linki do nich w headzie programu tak samo jak wrzucamy plik css

**Ważne:** Nawet jeśli obiekty są inline, to gdy w html’u dodamy nową linię między nimi, tworzy się między nimi znak spacji. Jeśli chcemy brak pustego miejsca, trzeba wszystko zapisać w jednej lini

**Flexbox**

**Co to?**

Narzędzia do rozkładania elementów po stronie.

**Podstawowe właściwości:  
display: flex;** - włącza właściwości dla danego kontenera.

**flex-direction** – typowo elementy ustawiane są od lewej do prawej. To pozwala nam zmieniać to zachowanie:   
**row** – standardowe **row-reversed** – od prawej do lewej

**column** – od góry do dołu

**column-reversed** – od dołu do góry

**Ważne** – jeśli elementy się nie mieszczą to ich wielkość jest odpowiednio zmniejszona

**justify-content** – przesunięcie elementów z pozostawionym układem. Wartości to:   
**start** – podstawowa opcja. Elementy ustawione od początku do końca

**end** – przesunięcie do końca

**center** - wyśrodkowane

**space-between** – dodanie takich samych przestrzeni pomiędzy elementami (nie wliczając krańców)

**space-evenly** – takie jak powyższe, ale z odstępami na krańcach

**flex-wrap** – decyduje o tym, czy elementy gdy się nie mieszczą mają się zmniejszać (***nonewrap***), czy mają wchodzić na nową linę (***wrap***).

**Cross axis:**

Mówimy o tej przestrzeni jako tej prostopadłej do obecnego ustawienia sortowania (if columns then x, if rows than y)

**aling items** – decyduje o wyrównaniu względem ***cross axis***. Wartości to:   
**start** – bazowo przytula elementy do góry

**end** – bazowo przytula elementy do dołu

**center** – wyśrodkowuje

**aling content** – działa tylko w przypadku ***flex-wrap***. Używane do rozdzielania przestrzeni między nowo powstałymi wierszami. Nadpisuje ***aling items***:

**start** – podstawowa opcja. Elementy ustawione od początku do końca

**end** – przesunięcie do końca

**center** - wyśrodkowane

**space-between** – dodanie takich samych przestrzeni pomiędzy elementami (nie wliczając krańców)

**space-evenly** – takie jak powyższe, ale z odstępami na krańcach

**aling self** – dajemy tą właściwość elementowi w kontenerze. Takie same wartości jak to ***aling content***

**Flex:**

**flex-basis** – ustawia wartość długości podstawy. W zależności czy columns czy rows w ***flex-direction*** to albo width, albo heigth

**flex-grow** – właściwość sprawiająca, że element zajmie całe pozostałe możliwe miejsce. Wartości to liczby oznaczające priorytet i odpowiednio od tego wychodzi ile miejsca zajmują. Można ograniczać wielkości używając ***max-width & max-heigth***

**flex-shrink** – tak samo jak ***flex-grow***, ale sprawia, że możesz kontrolować zmniejszanie się obiektów

**flex** – uproszczony sposób zapisania wszystkich elementów powyżej. W zależności od ilości wartości przekazanych zostaną one uznane jako:

**{number}** - grow  
**{number} {size}** – grow, basis

**{number} {number} {size}** – grow, shrink, basis

**Responsive Design:**

**@media ({min-width/max-width}: {value})** – w tym elemencie określamy pojedyncze właściwości w zależności od długości widocznego miejsca. Można pomiędzy () dać ***and*.**

**@media (orientation: {landscape/portrait}**