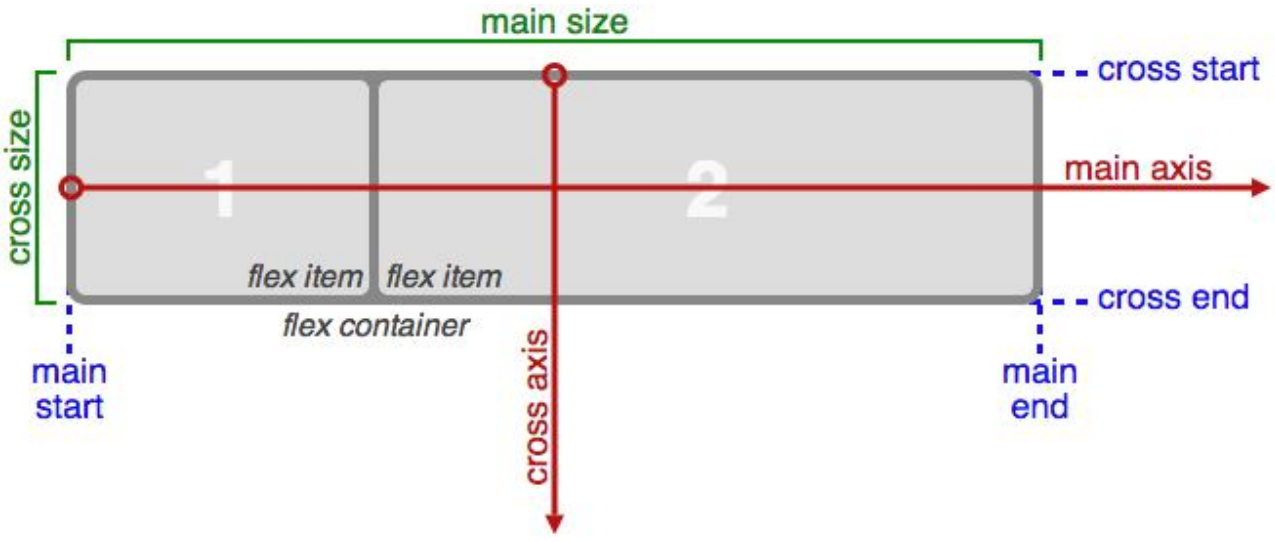
Лекція 5.
Адаптивний дизайн, float та flex властивості блочної верстки

CSS параметри Flexbox

Модуль Flexbox-лейаута (flexible box - «гнучкий блок», на даний момент W3C Candidate Recommendation) ставить задачу запропонувати більш ефективний спосіб верстки, вирівнювання і розподілу вільного місця між елементами в контейнері, навіть коли їх розмір невідомий і / або динамічний (звідси слово «гнучкий»).

Головна задумка flex-верстки в наділення контейнера здатністю змінювати ширину / висоту (і порядок) своїх елементів для найкращого заповнення простору (в більшості випадків - для підтримки всіх видів дисплеїв і розмірів екранів). Flex-контейнер розтягує елементи для заповнення вільного місця або стискає їх, щоб запобігти виходу за межі.



Оскільки flexbox - це цілий модуль, а не просто одиничне властивість, він об'єднує в собі безліч властивостей. Деякі з них мають застосовуватися до контейнера (батьківського елементу, так званому flex-контейнеру), в той час як інші властивості застосовуються до дочірніх елементів, або flex-елементам .

display: flex | inline-flex
Застосовується до: батьківського елементу flex-контейнера.

Визначає flex-контейнер (інлайновий або блоковий в залежності від обраного значення), підключає flex-контекст для всіх його безпосередніх нащадків.

```
display: other values | flex | inline-flex;
```

flex-direction
Застосовується до: батьківського елементу flex-контейнера. Встановлює головну вісь main-axis, визначаючи тим самим напрямок для flex-елементів, що розміщуються в контейнері.

```
flex-direction: row | row-reverse | column | column-reverse
row(за замовчуванням): зліва направо для ltr, справа наліво для rtl;
row-reverse: Справа наліво для ltr, зліва направо для rtl;
column: Аналогічно row, зверху вниз;
column-reverse: Аналогічно row-reverse, від низу до верху.
flex-wrap
```

flex-wrap
Застосовується до: батьківського елементу flex-контейнера.

Визначає, чи буде контейнер однорядковим або багаторядковим, а також напрямок поперечної осі, що визначає напрямок, в якому будуть розташовуватися нові рядки.

```
flex-wrap: nowrap | wrap | wrap-reverse
nowrap(за замовчуванням): однорядковий / зліва направо для ltr,
справа наліво для rtl;
wrap: Багатостроковий / зліва направо для ltr, справа наліво для rtl;
wrap-reverse: Багатостроковий / справа наліво для ltr, зліва направо
для rtl.
```

flex-flow
Застосовується до: батьківського елементу flex-контейнера.

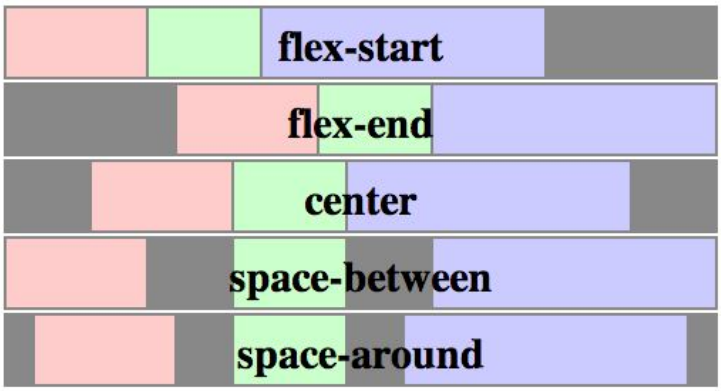
Це скорочення для властивостей flex-directioni flex-wrap, разом визначають головну і поперечну осі. За замовчуванням приймає значення row nowrap.

```
flex-flow: <'flex-direction'> || <'flex-wrap'>
```

justify-content
Застосовується до: батьківського елементу flex-контейнера.

Визначає вирівнювання відносно головної осі. Сприяє ефективному розподілу вільне місце в разі, коли елементи рядки не «тягнуться», або тягнуться, але вже досягли свого максимального розміру. Також дозволяє до певної міри керувати вирівнюванням елементів при виході за межі рядка.

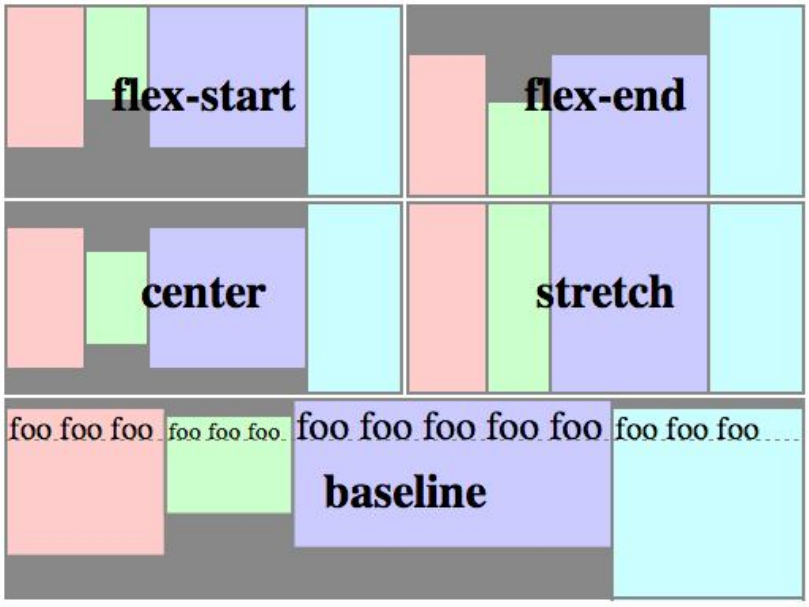
```
justify-content: flex-start | flex-end | center | space-between | space-around
flex-start (За замовчуванням): елементи зсуваються до початку рядка;
flex-end: Елементи зсуваються до кінця рядка;
center: Елементи вирівнюються по центру рядка;
space-between: Елементи розподіляються рівномірно (перший елемент на початку рядка, останній - в кінці);
space-around: Елементи розподіляються рівномірно з рівною відстанню між собою і кордонами рядки.
```



align-items
Застосовується до: батьківського елементу flex-контейнера.

Визначає поведінку за замовчуванням для того, як flex-елементи розташовуються відносно поперечної осі на поточному рядку. Вважайте це версією justify-contentдля поперечної осі (перпендикулярної до основної).

align-items: flex-start | flex-end | center | baseline | stretch
flex-start: Межа cross-startдля елементів розташовується на позиції cross-start;
flex-end: Межа cross-endдля елементів розташовується на позиції cross-end;
center: Елементи вирівнюються по центру поперечної осі;
baseline: Елементи вирівнюються по своїй базової лінії;
stretch(за замовчуванням): елементи растягіваются, заповнюючи контейнер (з урахуванням min-width/ max-width).

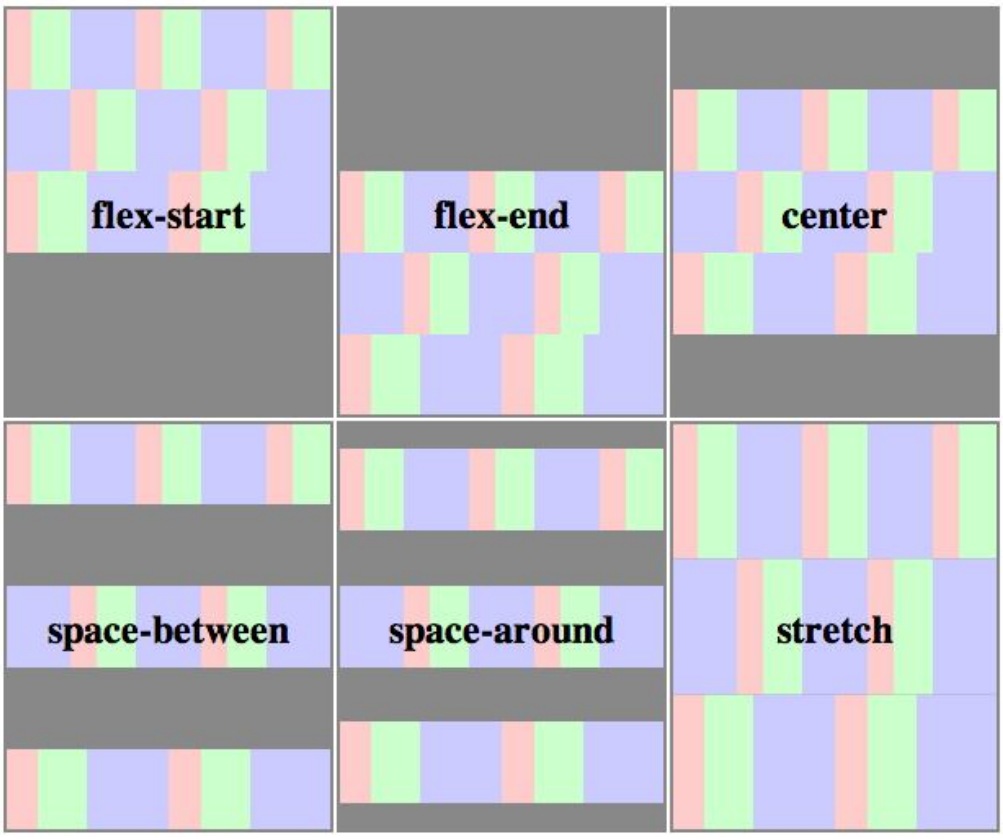


align-content
Застосовується до: батьківського елементу flex-контейнера.

Вирівнює рядки flex-контейнера при наявності вільного місця на поперечної осі аналогічно тому, як це робить justify-contentна головній осі.

Зауваження : це властивість не працює з однорядковим flexbox.

align-content: flex-start | flex-end | center | space-between | space-around | stretch
flex-start: Рядки вирівнюються щодо початку контейнера;
flex-end: Рядки вирівнюються щодо кінця контейнера;
center: Рядки вирівнюються по центру контейнера;
space-between: Рядки розподіляються рівномірно (перший рядок на початку рядка, остання - в кінці);
space-around: Рядки розподіляються рівномірно з рівною відстанню між собою;
stretch (За замовчуванням): рядки розтягуються, заповнюючи вільний простір.



order
Застосовується до: дочірньому елементу / flex-елементу.

За замовчуванням flex-елементи розташовуються в початковому порядку. Проте, властивість orderможе управляти порядком їх розташування в контейнері.

```
order: <integer>
```

flex-grow
Застосовується до: дочірньому елементу / flex-елементу.

Визначає для flex-елемента можливість «виростати» при необхідності. Приймає безрозмірне значення, що служить в якості пропорції. Воно визначає, яку частку вільного місця всередині контейнера елемент може зайняти.

Якщо у всіх елементів властивість flex-growзадано як 1, то кожен нащадок отримає всередині контейнера однаковий розмір. Якщо ви задали одному з нащадків значення 2, то він забере в два рази більше місця, ніж інші.

```
flex-grow: <number> (по умолчанию 0)
```

Негативні числа не приймаються.

flex-shrink
Застосовується до: дочірньому елементу / flex-елементу.
Визначає для flex-елемента можливість стискатися при необхідності.

```
flex-shrink: <number> (default 1)
```

Негативні числа не приймаються.

flex-basis
Застосовується до: дочірньому елементу / flex-елементу.

Визначає розмір за замовчуванням для елемента перед розподілом простору в контейнері.

```
flex-basis: <length> | auto (default auto)
```

flex
Застосовується до: дочірньому елементу / flex-елементу.

Це скорочення для flex-grow, flex-shrinki flex-basis. Другий і третій параметри (flex-shrink, flex-basis) є обов'язковими. Значення за замовчуванням - 0 1 auto.

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

align-self
Застосовується до: дочірньому елементу / flex-елементу.

Дозволяє перевизначити вирівнювання, заданий за замовчуванням або в align-items, для окремих flex-елементів.

Зверніться до опису властивості align-itemsдля кращого розуміння доступних значень.

```
align-self: auto | flex-start | flex-end | center | baseline | stretch
```

CSS параметри плаваючого розміщення

```
float: left | right | none | inherit
```

Адаптивний дизайн:

Адаптивний веб-дизайн— дизайн веб-сторінок, що забезпечує оптимальне відображення та взаємодію сайту з користувачем незалежно від роздільної здатності та формату пристрою, з якого здійснюється перегляд сторінки.

Метою адаптивного веб-дизайну є практичне відображення інформації та зручна навігація на всіх пристроях із доступом до інтернету (від стаціонарних ПК до мобільних телефонів). За технологією адаптивного веб-дизайну не потрібно створювати окремі версії веб-сайту. Один сайт може працювати на всьому спектрі пристроїв.

Етапи створення адаптивного дизайну

1. Підключити html5.js для браузерів які не підтримують HTML5
Internet Explorer нижче 9-ї версії не підтримує нові елементи містяться в HTML5, такі як <header>, <article>, <footer>, <figure> та інші. Тому підключаємо Javascript файл html5.js в HTML документ, який дозволить IE розуміти нові елементи.

Приклад:
```
<! - [if lt IE 9]>
   <script src = "http://sitename.com/ html5.js"> </ script>
<! [endif] ->
```

2. Задати новим текам HTML5 блоковий тип відображення на сайті
Наступний CSS зробить HTML5 елементи (article, aside, figure, header, footer, etc.) блоковими.

Приклад:
```
article , aside, details, figcaption, figure, footer, header, hgroup,
menu, nav, section {display : block ;}
```

3. Описуємо основну структуру в CSS
Гумовий макет на основі пропорцій (fluid grid). Основна ідея - формула для обчислення пропорцій в процентах «target / context = result». Наприклад, у нас є макет psd з шириною 1000px. У ньому є два блоки: один зліва шириною 270px, інший праворуч 730px. Зверстаємо ми їх так:

Приклад:

```
.leftcolumn  {
        width :  27 % ;  / * 270px / 1000px = 0,27 * /
        float :  left ;
}

.rightcolumn  {
        width :  73 % ;  / * 730px / 1000px = 0,73 * /
        float :  right ;
}
```

Якщо всередині лівого стовпчика буде ще один блок і, скажімо, на макеті він шириною в 170px, то для нього зміняться мета і контекст, і код буде виглядати ось так:

Приклад:

```
.leftcolumn  .some-div  {
        width :  62 , 962 963 % ;  / * 170px / 270px = 0.62962963 * /
        float :  left ;
}
```

4. Підключаємо css3-mediaqueries.js для браузерів які не підтримують @media

Internet Explorer 8 і більше ранніх версій не підтримує CSS3 media queries. Ви можете включити її, додавши Javascript файл css3-mediaqueries.js .

Приклад:

```
<! - [if lt IE 9]>
   <script src = "http://sitename.com/respond.js"> </ script>
<! [endif] ->
```

5. Описуємо в CSS @media із використанням гумової, адаптивної верстки

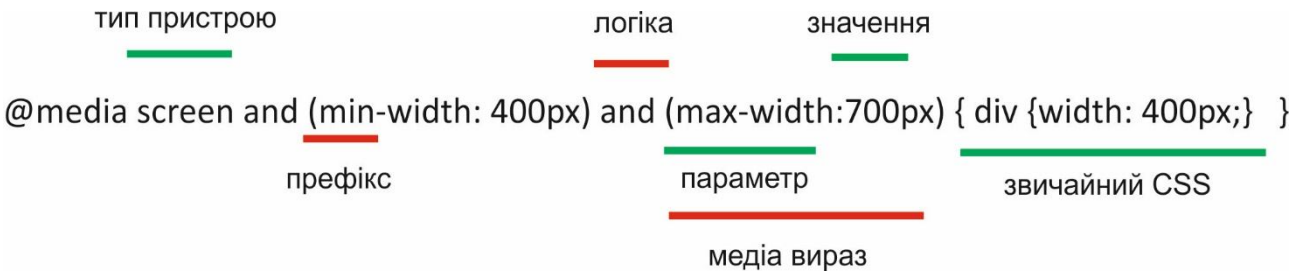
Опис медіа запитів як правило виконують в основному файлі таблиць стилів сайту. Правила @mediaзавжди прописуються в кінці файлу css-стилів.

Приклад:
основний CSS{} @media{}

Також можливо задати окремий CSS файл в мета тегах із заданням параметрів пристрою як би це потрібно було б прописати в основному файлі таблиць стилів

Приклад:
<link rel="stylesheet" media="mediatype and|not|only (media feature)" href="mystylesheet.css">

Основні параметри @media
Правило @media дозволяє вказати тип носія, для якого буде застосовуватися зазначений стиль. Як типів виступають різні пристрої, наприклад, принтер, КПК, монітор та ін. В табл. 1 перераховані деякі з них.



Синтаксис:
@media тип1 [, тип2] { Опис стилів}

Опис синтаксису:
Після ключового слова @media йде один або кілька типів носія, перерахованих в табл. 1; якщо їх більше одного, то вони поділяються між собою комою. Після чого йдуть обов'язкові фігурні дужки, всередині яких йде звичайне опис стильових правил.

Значення типу носія:

Тип носія являє собою тип пристрою, наприклад, принтери, екрани.

all	Підходить для всіх типів пристроїв.
print	Призначений для сторінкових матеріалів і документів, що переглядаються на екрані в режимі попереднього перегляду друку.
screen	Призначений в першу чергу для екранів кольорових комп'ютерних моніторів.
speech	Призначений для синтезаторів мови.

CSS2.1 і Media Queries 3 визначали кілька додаткових типів, таких як aural, braille, embossed, projection, tty, tvi handheld, але вони прийняті застарілими в Media Queries 4 і не будуть використовуватися.

Логічні оператори

За допомогою логічних операторів можна створювати комбіновані медіазапроси, в яких буде перевірятися відповідність декільком умовам.

and	Оператор andп ов'язує один з одним різні умови:	@media (min-width: 600px) and (max-width: 800px) {/* css-стили */;}
not	Оператор not дозволяє спрацювати медіа запитом в протилежному випадку. Ключове слово notдодається в початок медіа запиту і застосовується до всього запитом цілком, тобто запит	@media not screen and (color), print and (color)
only	Оператор only дозволяє підключати стилі для браузерів, які не підтримують медіа запити, наприклад:	@media only screen and (color) { /* css-стили */;}
кома	Оператор кома працює за аналогією з логічним оператором or.	@media screen, projection { /* css-стили */;}

Технічні характеристики носіїв

До характеристик медіаносія відносяться перевіряються параметри пристрою. Значення, які використовуються при завданні характеристик, є контрольними точками.

width	Перевіряє ширину області перегляду. Значення задаються в одиницях довжини, px, emi т.д., наприклад, (width: 800px). Зазвичай для перевірки використовуються мінімальні і максимальні значення ширини. min-widthзастосовує правило якщо ширина області перегляду більше значення, зазначеного в запиті, max-width- ширина області перегляду менше значення, зазначеного в запиті.
height	Перевіряє висоту області перегляду. Значення задаються в одиницях довжини, px, emi т.д., наприклад, (height: 500px). Зазвичай для перевірки використовуються мінімальні і максимальні значення висоти. min-heightзастосовує правило якщо висота області перегляду більше значення, зазначеного в запиті, max-height- висота області перегляду якого менше значення, зазначеного в запиті.
aspect-ratio	Перевіряє співвідношення ширини до висоти області перегляду. Широкоекранний дисплей зі співвідношенням сторін 16: 9 може бути позначений як (aspect-ratio: 16/9). min-aspect-ratioперевіряє мінімальне співвідношення, max-aspect-ratio- максимальне співвідношення ширини до висоти області перегляду.
orientation	Перевіряє орієнтацію області перегляду. Приймає два значення: (orientation: portrait)і (orientation: landscape).
resolution	Перевіряє дозвіл екрана (кількість пікселів). Значення також можуть перевіряти кількість точок на дюйм (dpi) або кількість точок на сантиметр (dpcm), наприклад, (resolution: 300dpi). min-resolutionперевіряє мінімальний дозвіл екрана, max-resolution- максимальне.
color	Перевіряє кількість біт на кожен з колірних компонентів пристрою виведення. Наприклад, (min-color: 4)означає, що екран конкретного пристрою повинен мати 4-бітну глибину кольору. min-colorперевіряє мінімальну кількість біт, max-color-максимальна кількість біт.
color-index	Перевіряє кількість записів в таблиці підстановки квітів. Як значення вказується позитивне число, наприклад, (color-index: 256). min-color-indexперевіряє мінімальну кількість записів, max-color-index- максимальна кількість записів.

monochrome	Перевіряє кількість бітів на піксель монохромного пристрою. Значення задається цілим позитивним числом, наприклад, (min-monochrome: 8). min-monochromeперевіряє мінімальну кількість бітів, max-monochrome- максимальна кількість бітів.
-webkit-device-pixel-ratio	Задає кількість фізичних пікселів пристрої на кожен css-піксель.

При складанні медіа запитів потрібно орієнтуватися на так звані переломні точки дизайну , тобто такі значення ширини області перегляду, в яких дизайн сайту істотно змінюється, наприклад, з'являється горизонтальна смуга прокрутки. Щоб визначити ці точки, потрібно відкрити сайт в браузері і поступово зменшувати область перегляду.

Щоб адаптувати дизайн сайту під різні пристрої, необхідно задати різні стилі для різних дозволів екранів, використовуючи такі контрольні точки (не обов'язково все) :

Приклад:

```
/* Для мобільних пристроїв. Android ,  iPhone і так далі.*/
@media only screen and (min-width: 480px) { }
/* Планшети, мобільні пристрої в горизонтальному режимі */
@media only screen and (min-width: 768px) { }
/* Планшети в горизонтальному режимі,  нетбуки ,  ноутбуки ,
десктоп */
@media only screen and (min-width: 992px) {}
/* Десктоп з великим розміром екрану,  телевізори.*/
@media only screen and (min-width: 1382px) { }
```

Повний приклад:

```
/ * Smartphones (вертикальна і горизонтальна орієнтація) ---------- * /
@media only screen and (min-width: 320px) and (max-width: 480px) {}
/ * Smartphones (горизонтальна) ---------- * /
@media only screen and (min-width: 321px) {}
/ * Smartphones (вертикальна) ---------- * /
@media only screen and (max-width: 320px) {}
/ * IPads (вертикальна і горизонтальна) ---------- * /
```

```
@media only screen and (min-width: 768px) and (max-width: 1024px) {}
/ * IPads (горизонтальна) ---------- * /
@media only screen and (min-width: 768px) and (max-width: 1024px) and
(orientation: landscape) {}
/ * IPads (вертикальна) ---------- * /
@media only screen and (min-width: 768px) and (max-width: 1024px) and
(orientation: portrait) {}
/ * IPad 3 ********** /
@media only screen and (min-width: 768px) and (max-width: 1024px) and
(orientation: landscape) and (-webkit-min-device-pixel-ratio: 2) {}
@media only screen and (min-width: 768px) and (max-width: 1024px) and
(orientation: portrait) and (-webkit-min-device-pixel-ratio: 2) {}
/ * Настільні комп'ютери і ноутбуки ---------- * /
@media only screen and (min-width: 1224px) {}
/ * Великі екрани ---------- * /
@media only screen and (min-width: 1824px) {}
/ * IPhone 4 ---------- * /
@media only screen and (min-width: 320px) and (max-width: 480px) and
(orientation: landscape) and (-webkit-min-device-pixel-ratio: 2) {}
@media only screen and (min-width: 320px) and (max-width: 480px) and
(orientation: portrait) and (-webkit-min-device-pixel-ratio: 2) {}
/ * IPhone 5 ---------- * /
@media only screen and (min-width: 320px) and (max-height: 568px) and
(orientation: landscape) and (-webkit-device-pixel-ratio: 2) {}
@media only screen and (min-width: 320px) and (max-height: 568px) and
(orientation: portrait) and (-webkit-device-pixel-ratio: 2) {}
/ * IPhone 6 ---------- * /
@media only screen and (min-width: 375px) and (max-height: 667px) and
(orientation: landscape) and (-webkit-device-pixel-ratio: 2) {}
@media only screen and (min-width: 375px) and (max-height: 667px) and
(orientation: portrait) and (-webkit-device-pixel-ratio: 2) {}
/ * IPhone 6 + ---------- * /
@media only screen and (min-width: 414px) and (max-height: 736px) and
(orientation: landscape) and (-webkit-device-pixel-ratio: 2) {}
@media only screen and (min-width: 414px) and (max-height: 736px) and
(orientation: portrait) and (-webkit-device-pixel-ratio: 2) {}
/ * Samsung Galaxy S3 ---------- * /
@media only screen and (min-width: 320px) and (max-height: 640px) and
(orientation: landscape) and (-webkit-device-pixel-ratio: 2) {}
```

```
@media only screen and (min-width: 320px) and (max-height: 640px) and
(orientation: portrait) and (-webkit-device-pixel-ratio: 2) {}
/ * Samsung Galaxy S4 ---------- * /
@media only screen and (min-width: 320px) and (max-height: 640px) and
(orientation: landscape) and (-webkit-device-pixel-ratio: 3) {}
@media only screen and (min-width: 320px) and (max-height: 640px) and
(orientation: portrait) and (-webkit-device-pixel-ratio: 3) {}
/ * Samsung Galaxy S5 ---------- * /
@media only screen and (min-width: 360px) and (max-height: 640px) and
(orientation: landscape) and (-webkit-device-pixel-ratio: 3) {}
@media only screen and (min-width: 360px) and (max-height: 640px) and
(orientation: portrait) and (-webkit-device-pixel-ratio: 3) {}
```

6. Описуємо параметри тегів контенту для гумової верстики

Гумові зображення (fluid images) Підлаштовують свої розміри під блок батька. Основна ідея в неочевидно застосуванні властивості {max-width: 100%}. Зображення з img {max-width: 100%} ніколи не вилізе зі свого блоку-батька. Якщо блок-батько менше, ніж розміри img, то зображення пропорційно зменшиться. Цей принцип можна застосувати як для img, так і для embed, object, video.

Приклад:

```
.video  embed , .video  object , .video  iframe  {width :  100 % ;height :
auto ;}
```

7. Повідомити браузеру використовувати ширину по замовчуванню самого пристою

Приклад:

```
<meta  name = "viewport"  content = "width = device-width; initial-scale =
1.0" >
```

Існує два підходи до побудови адаптивного веб дизайну. Розробка починається або з мобільного інтерфейсу, а далі відбувається адаптація інтерфейсу для інших розширень, або, навпаки, розробка починається від вигляду на екранах стаціонарних ПК і закінчується інтерфейсом на мобільних телефонах.

Відносні одиниці вимірювання

Область перегляду сторінки може бути монітором, екраном мобільного або яким завгодно пристроєм. Щільність пікселів на різних екранах також різна, тому потрібні гнучкі одиниці виміру, що працюють всюди. Адаптивний дизайн - саме той випадок, коли відносні одиниці виміру на зразок відсотків стають дійсно корисними. За допомогою відсотків можливо поставити блоку ширину в 50%, і на будь-якому пристрої він буде займати лише половину екрану.

Контрольні точки

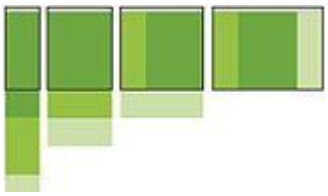
Контрольні точки дозволяють змінювати розташування блоків на сторінці тільки в разі використання екрану з певними розмірами. Наприклад, на стаціонарних комп'ютерах на сторінці буде три колонки, а на мобільних телефонах - тільки одна. Контрольні точки визначає контент сторінки, тобто розробник поступово, починаючи від певної точки, змінює розмір і коли розташуванню контенту потрібні зміни встановлює контрольну точку. Існує інший підхід, який базується на створення контрольних точок для найбільш поширених пристроїв, проте зважаючи на збільшення кількості нестандартних розширень екранів цей підхід не є популярним.

Шаблони

Перетічний

Популярний і, не зважаючи на це, простий шаблон. Макет складається з декількох колонок, розміри яких на екранах великої та середньої ширини залишаються незмінними, а змінюються лише поля. На невеликих екранах відбувається зміна розміру контенту та стовпці розміщуються один під одним.

Спадні стовпці



Використовується в макетах, що складаються з декількох стовпців, які займають всю ширину екрану. Коли ширина вікна стає занадто малою для відображення всього контенту, шаблон розміщує стовпці один за одним по вертикалі. З часом це призводить до того, що всі стовпці будуть розташовані вертикально один під одним. Вибір контрольних точок для цього шаблону залежить від контенту і визначається для кожного варіанту дизайну окремо.

Рухомий макет

Даний шаблон є найбільш адаптивним, оскільки в ньому передбачено наявність декількох контрольних точок для екранів різної ширини. Основною відмінністю цього макета є те, що замість розміщення стовпців один під одним рухається сам контент. Через значні відмінності між основними контрольними точками, підтримка цього макета є більш складним завданням, крім того, доводиться змінювати не тільки загальний макет контенту, але і його елементи.

Крихітні зміни

Даний шаблон вносить невеликі зміни в макет, наприклад регулює розмір шрифту, змінює розмір зображень чи переміщує контент. Він добре працює на макетах, що складаються з одного стовпчика, як односторінкові лінійні веб-сайти і статті з великою кількістю тексту.

Поза тлом

У всіх вище перелічених шаблонах присутня тенденція до розміщення елементів контенту вертикально один під одним. Даний шаблон використовує інший підхід. Контент, який використовується рідко, наприклад елементи навігації або меню, розміщується за межами екрану і використовуються тільки тоді, коли це дозволяє зробити розмір екрану. На невеликих екранах контент можна відкрити за допомогою додаткової кнопки (часто використовується іконка гамбургера.

Джерела:
http://htmlbook.ru - бібліотека, довідник по тегам та атрибутам HTML CSS
http://javascript.ru - довідник по мові програмування JavaScript

https://habrahabr.ru
https://html5book.ru