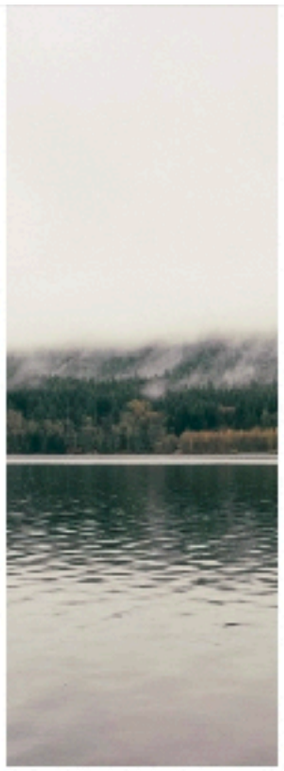
CSS grid layout

Vira Huskova







https://caniuse.com/#search=grid

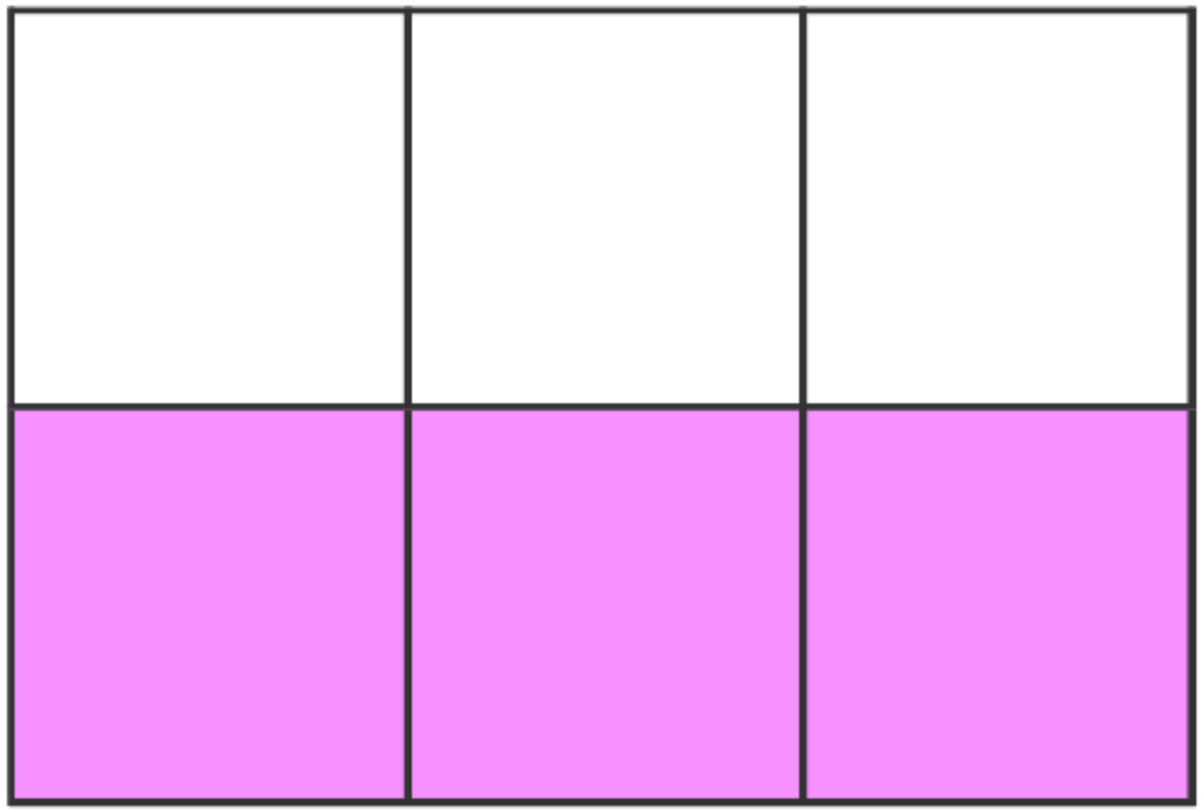
CSS Grid это новая модель шаблона, оптимизированная для двумерных шаблонов. Это идеальная модель для шаблонов сайтов, форм, галерей и всего, что требует точного и отзывчивого позиционирования.

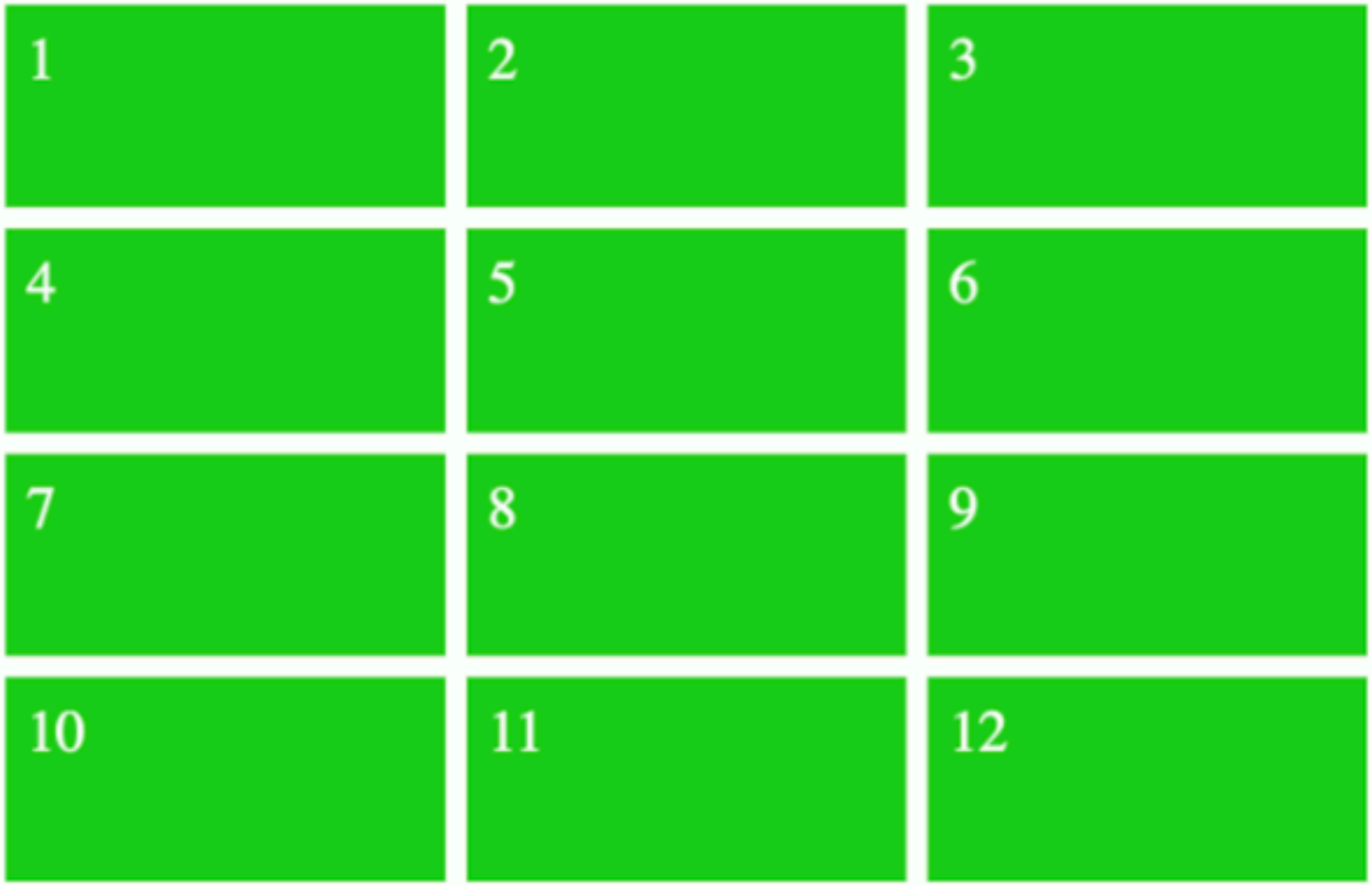
Grid шаблон работает по системе сеток. Grid это набор пересекающихся горизонтальных и вертикальных линий, которые создают размеры и позиционируют систему координат для контента в grid-контейнере.

Чтобы создать Grid, нужно выставить элементу **display: grid**.

Это автоматически сделает всех прямых **потомков** этого элемента — grid элементами.

Первым шагом является определение того, сколько колонок и рядов есть в grid. Но даже это опционально.





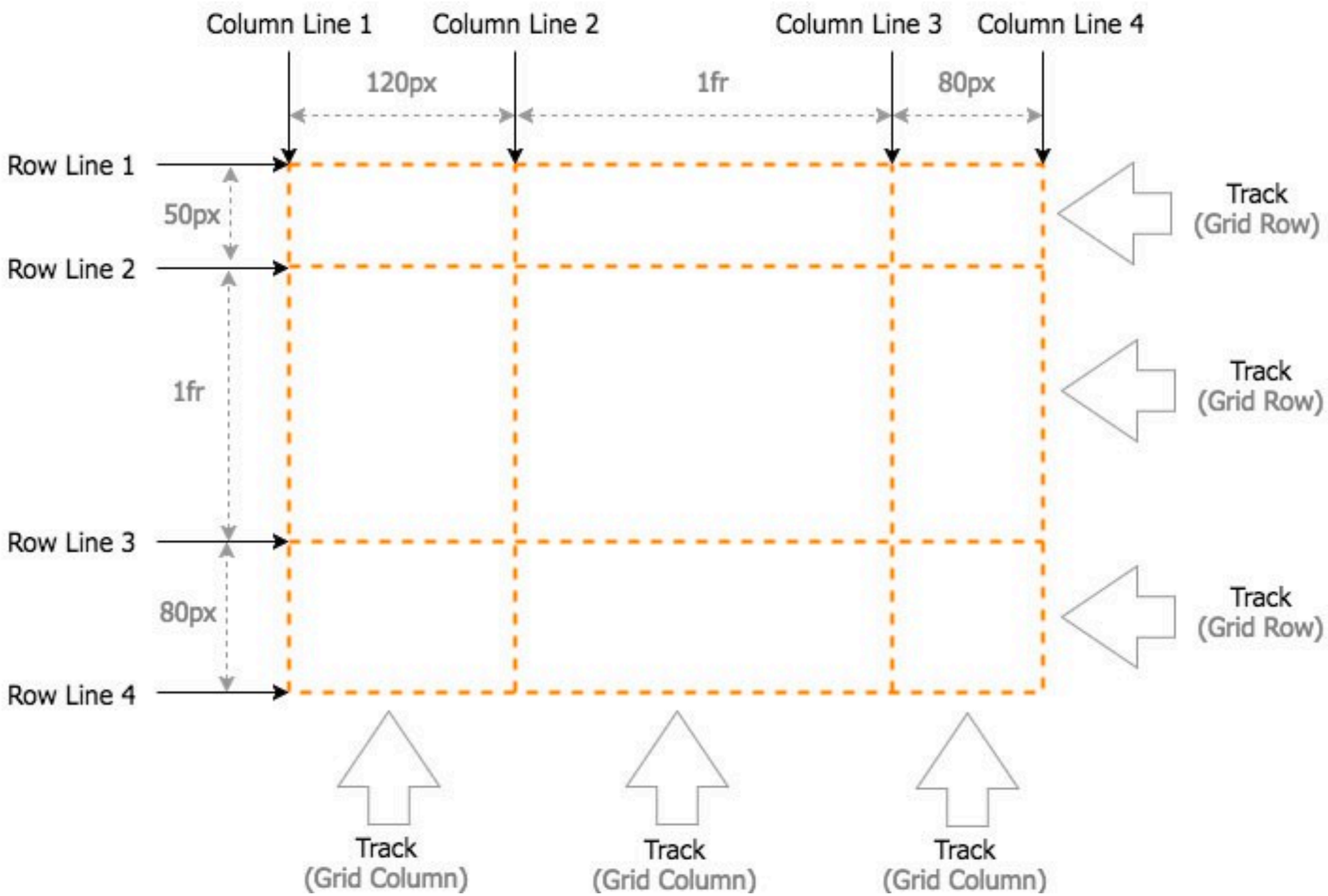
Это пример грида с четырьмя рядами и тремя колонками. Он состоит из 12 grid элементов. Каждый из этих элементов зеленый и между ними есть небольшое расстояние.

Все эти `grid` элементы одного размера, но они могли бы быть любого размера.

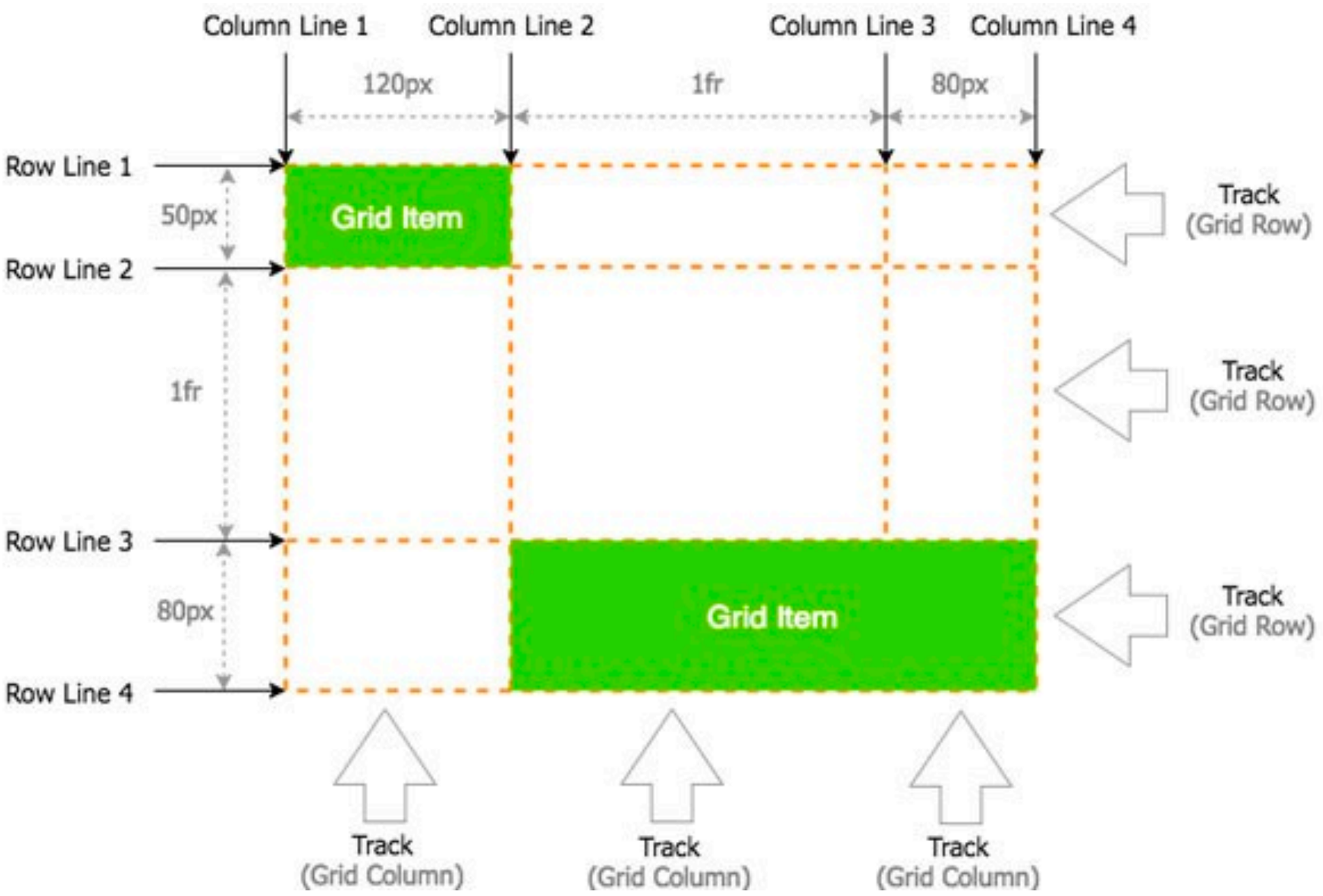
Некоторые могли бы охватывать несколько столбцов и рядов, другие могли бы оставаться размеров с одну ячейку.

Расстановка Grid-элементов

Все сходится к грид-линиям.



Грид линии это горизонтальные и вертикальные линии в гриде. Каждая строка и колонка имеет грид линию с каждой своей стороны. У каждой грид линии есть числовой индекс, к которому вы можете ссылаться во время расстановки грид элементов.



https://codepen.io/lostsou41216364/pen/EdGwNg

https://codepen.io/lostsou41216364/pen/xyyEZZ

`display: grid`

Превращает элемент в grid контейнер.

Имеем грид-контейнер и грид-элементы.

Значения гридов создают блочный контейнер.

Можно использовать display: inline-grid, что создать строчный грид-контейнер.

Можно использовать display: subgrid, чтобы создать подсетку, это значение используется на самих grid элементах.

`grid-template-rows: 1fr 1fr 1fr`

Выстраивает ряды в гриде. Каждое значение представляет размер ряда. В этом случае все значения равны 1fr.

Для этого можно было бы использовать разные значения, такие как `100px`, `7em`, `30%` и так далее.

Можно назначать имена строкам вместе с их размерами (через атрибуты).

`grid-template-columns: 1fr 1fr 1fr`

Тоже самое, что и выше, только определяет колонки в гридах.

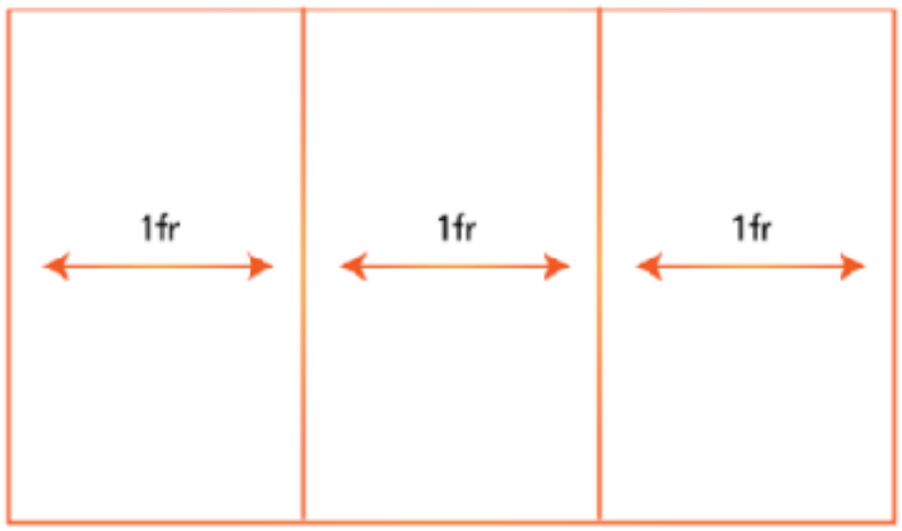
FR

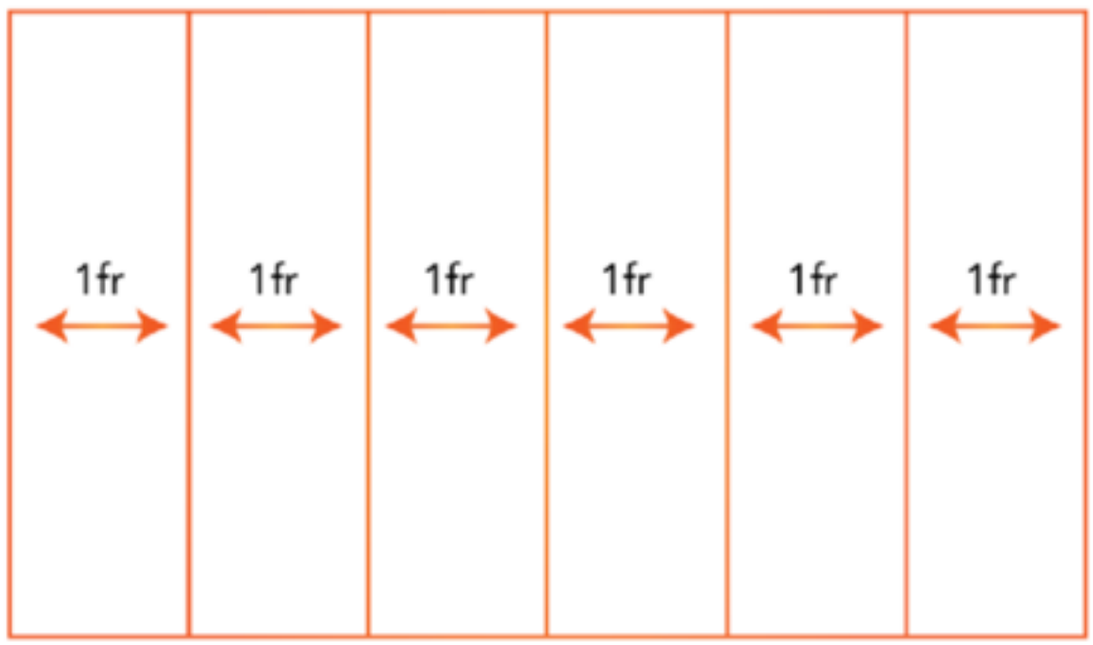
Fractional unit в контексте Grid разработки `fr` - это единица гибкости.

С единицей гибкости `(fr)` не нужно что-то больше пересчитывать.

На то она и единица гибкости, чтобы быть гибкой.

Если указать ширину в `1fr`, то можно дальше добавлять столько элементов, сколько возможно и `fr` об этом позаботится. Ширина каждого элемента будет равномерно разделена среди дочерних элементов.





Нет привязки к целым значениям. Можно указывать такие значения как: `1.5fr 3fr 4.5fr .`

Общая доля равна 1.5fr + 3fr + 4.5fr = 9fr

Если родительский контейнер имеет фиксированную ширину 900px

Первое значение, 1.5fr будет иметь ширину из расчета 1.5fr/9fr * 900px. Что выдает в итоге 150px.
Второе значение, 3fr будет иметь ширину 3fr/9fr * 900px. Что выдаст нам в итоге 300px.
Третье значение, 4.5fr будет иметь ширину 4.5fr/9fr * 900px. Что выдаст в итоге 450px.

`grid-gap: 2vw`

Выставляет разрыв. То есть пробелы между грид элементами.

Тут используется vw единица длины, которая относительна ширине viewport, но также можно использовать `10px`, `1em` и т. д.

`Grid-gap` свойство это сокращение для grid-row-gap и grid-column-gap свойств.

Функция repeat()

Можно использовать функцию repeat() для повторяющихся объявлений значения размера элемента. Для примера, вместо того, чтобы делать это:

```
grid-template-rows: 1fr 1fr 1fr 1fr 1fr;
```

```
grid-template-rows: repeat(5, 1fr);
```

Шаблон сайта на гридах

Гриды включают в себя интуитивный «ASCII-графический» синтаксис, в котором вы можете виртуально «видеть» шаблон в коде, по-этому становится очень легко создавать и изменять ваш шаблон.

Даже значительные изменения могут быть сделаны за несколько секунд. Этот интуитивный синтаксис также помогает с адаптивным веб-дизайном.

https://codepen.io/lostsou41216364/pen/KGGgYw

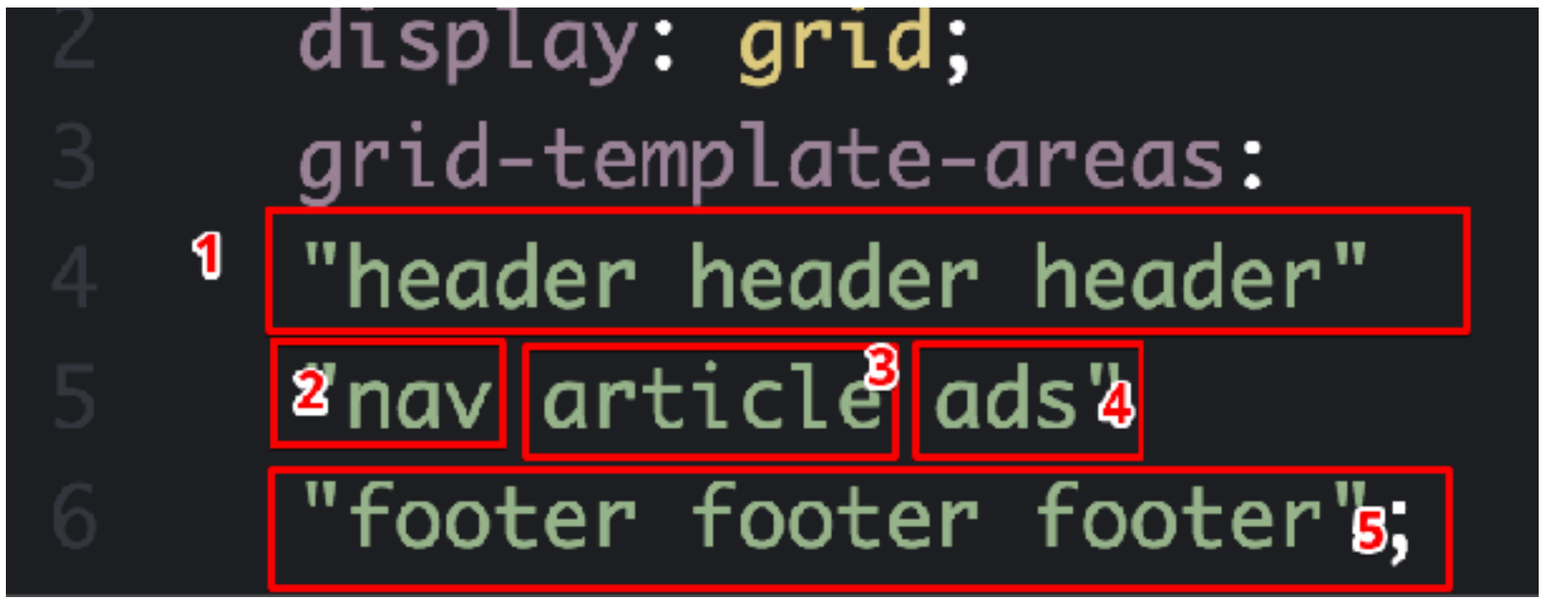
`<body>` грид-контейнер, таким образом все другие элементы станут грид-элементами.

```
display: grid;
grid-template-areas:
"header header header"
"nav article ads"
"footer footer footer";
```

Этот кусок определяет наш шаблон.

3x3 грид (три ряда и три колонки).

Таким образом у нас получается пять грид областей на девяти грид ячейках, так как некоторые грид-области занимают несколько ячеек.



Шапка занимает весь первый ряд в три ячейки, а подвал занимает весь нижний ряд, также забирая три ячейки.

Навигационная, контентная и рекламная секции, все вместе делят место во втором ряду, где каждому из этих элементов достается по одной ячейке.

Назначаем каждую из этих грид-областей каждому элементу

```
#pageFooter {
    grid-area: footer;
}
#mainArticle {
    grid-area: article;
}
#mainNav {
    grid-area: nav;
}
#siteAds {
    grid-area: ads;
}
```

Свойство `grid-area` это свойство, которое позволяет размещать грид-элементы в гриде.

В нашем случае, мы просто отсылаемся к названиям, которые мы предварительно указали в grid-template-areas.

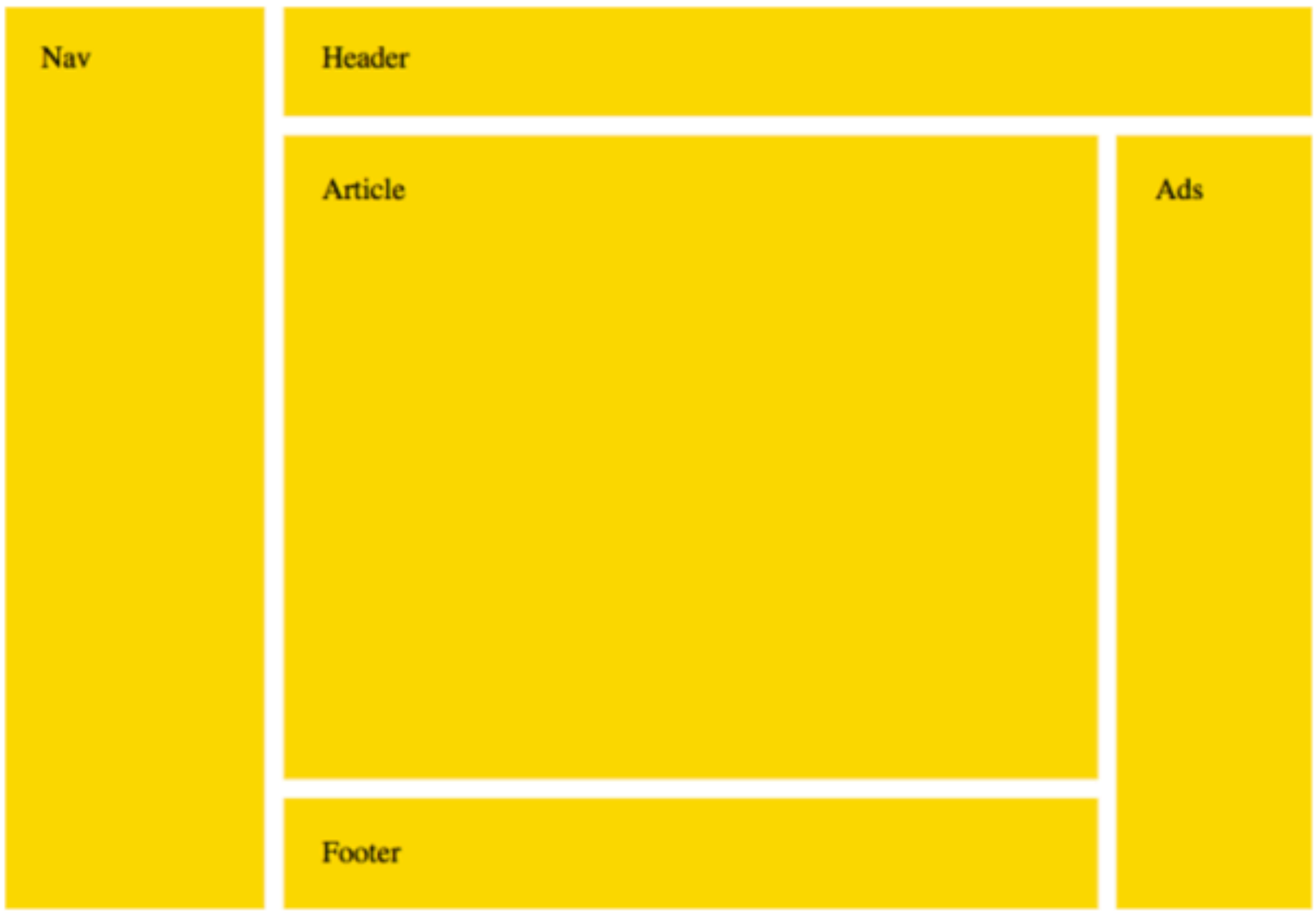
Размеры по строкам и колонкам

```
grid-template-rows: 60px 1fr 60px;
grid-template-columns: 20% 1fr 15%;
```

Первая и третья строки — обе в `60px` высотой, а вторая строка забирает все оставшееся место.

Первый столбец равен `20%`, а третий `15%`. Второй же забирает все оставшееся место.

Можно изменить шаблон просто перераспределив грид-области в `grid-template-areas`.




```
grid-template-areas:
"nav header header"
"nav article ads"
"nav footer ads";
```

https://codepen.io/lostsou41216364/pen/MPPbjb

Адаптивный Grid

В шаблоне на гридах есть значения `auto-fill` и `auto-fit`, которые позволяют создавать грид с множеством треков определенного размера, которые будут помещаться в контейнер.

Это может означать то, что грид является адаптивным, то есть в нем элементы меняют свои позиции с тем, как меняется размер окна браузера.

Auto fill

https://codepen.io/lostsou41216364/pen/VEEmNO

+ изменение окна браузера

Код, отвечающий за это

```
grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
```

В нем колонкам выдается минимальный размер в `150px` и максимум по оставшемуся месту (1fr).

Такие треки будут повторяться столько раз, сколько нужно для того, чтобы уложиться в контейнер.

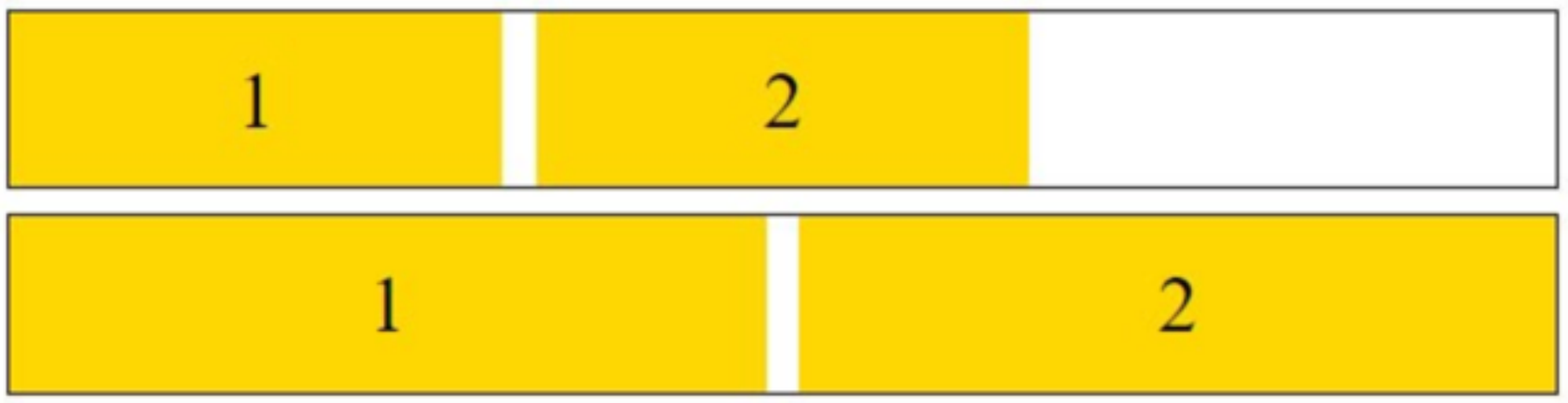
`Repeat()` функция повторяет трек такое количество раз, которое заданное первым параметром.

Использование `auto-fill` заставит трек повторяться столько раз, пока они не заполнят контейнер.

Auto fit

`auto-fit` работает практически также, как и `auto-fill`.

Разница только в том, что `auto-fit` стягивает все пустые треки в конце размещения, в то время как `auto-fill` нет.



https://codepen.io/lostsou41216364/pen/qJJRdw

`Auto-fill` оставляет пустые треки в конце по указанным размерам, а auto-fit растягивает пустой трек, что ведёт к заполнению треков растянутыми элементами для заполнения пространства.

Grid + media query

Одной из сильных сторон гридов является то, что можно создать совершенно отличный шаблон за секунды.

Это делает гриды идеальными для медиа запросов. Мы можем просто переназначить значения в ASCII-графике и обернуть результат в медиа запрос.

https://codepen.io/lostsou41216364/pen/Zqqgzw

Это трех колоночный шаблон на большом `viewport` и он сжимается в одноколоночный на маленьких устройствах. Таким образом, этот пример будет выглядеть по-другому в зависимости от размера экрана.

В любом случае, вот релевантный код для трехколоночного шаблона, для широких вьюпоротов.

Стандартное отображение

```
grid-template-areas:
“header header header”
“nav article ads”
“footer footer footer”;
```

Мобильное отображение

```
grid-template-areas:
“header”
“article”
“ads”
“nav”
“footer”;
```

Таким образом, все дело состоит в переназначении значений в свойстве `grid-template-areas`.

```
@media all and (max-width: 575px) {
body {
grid-template-areas:
“header”
“article”
“ads”
“nav”
“footer”;
grid-template-rows: 80px 1fr
    70px 1fr 70px;
grid-template-columns: 1fr;
}
```

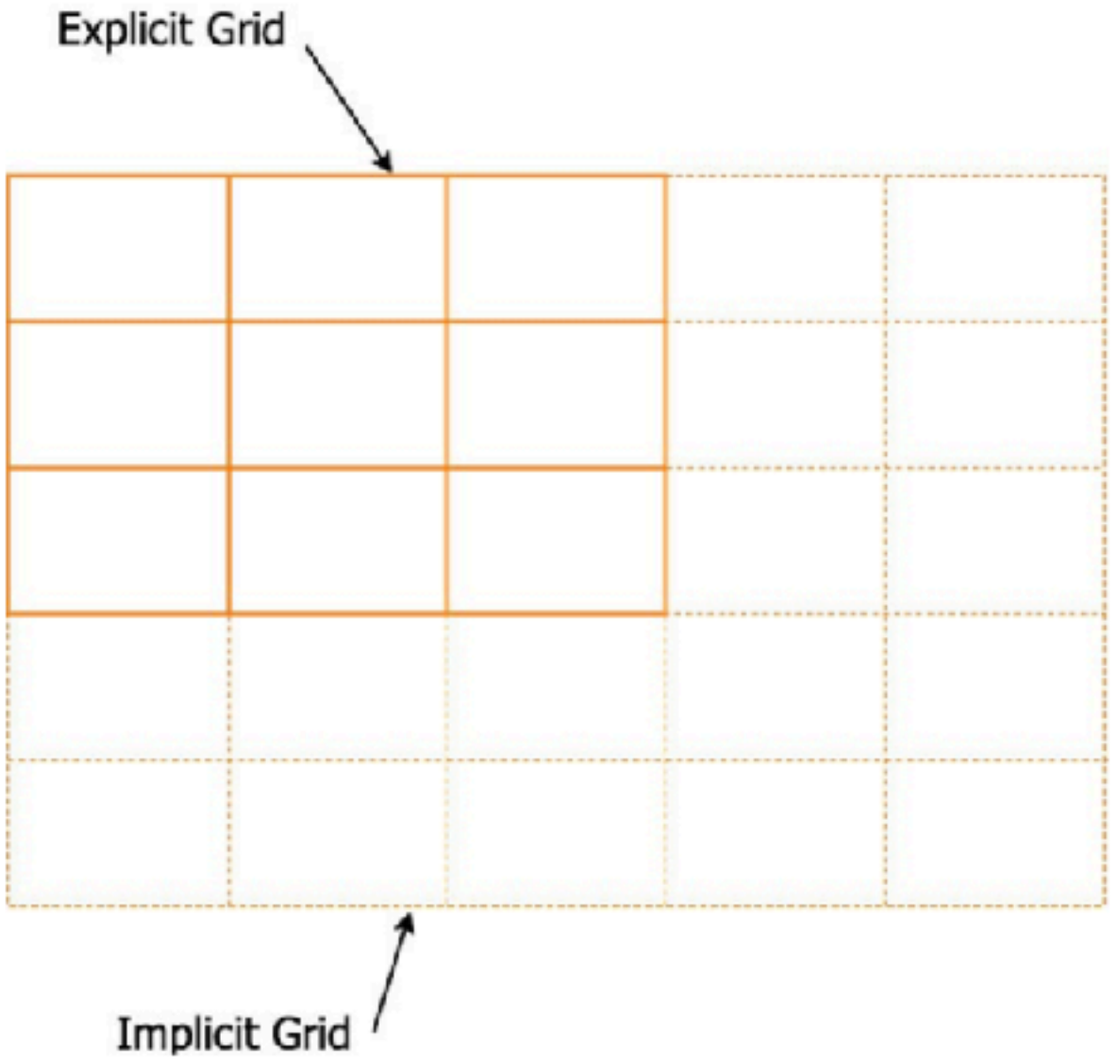
Необходимо подкорректировать значение в `grid-template-rows` и в `grid-template-columns`, чтобы они подходили под новый шаблон.

В частности, тут должна быть только одна колонка и она должна занимать все свободное место. А так как все грид элементы будут в одну кучу, назначим 5 строк и определим высоты.

Явные и неявные гриды

CSS Grid использует концепцию явного грида и неявного. Это ключевая концепция, которой нужно остерегаться при создании гридов, в противном случае под конец выходит скопление строк и колонок, существовании которых не планировалось.

Явный грид, это грид, который вы определяете в grid-template-rows, grid-template-colums и в grid-template-areas.



Тем не менее, можно получить элементы, которые не умещаются в «явно» опредленный грид.

Для примера, вы определили грид, который может уместить только шесть элементов, но сам контейнер на самом деле состоит из девяти элементов. Только шесть элементов будут умещаться в явный грид и три останутся. И вот где начинаются неявные гриды.

Неявные гриды автоматически генерируются грид-контейнером, всякий раз когда грид-элементы располагаются за пределами явного грида.

Контейнер генерирует неявные грид треки, добавляя неявные строки в грид. Эти строки вместе с явными гридами и формируют неявные.

https://codepen.io/lostsou41216364/pen/dgwzdX

Явно определены две строки и две колонки, уместив четыре град элемента.

А присутствует шесть грид элементов, -> был создан неявный грид, для того, чтобы уместить два дополнительный элемента.

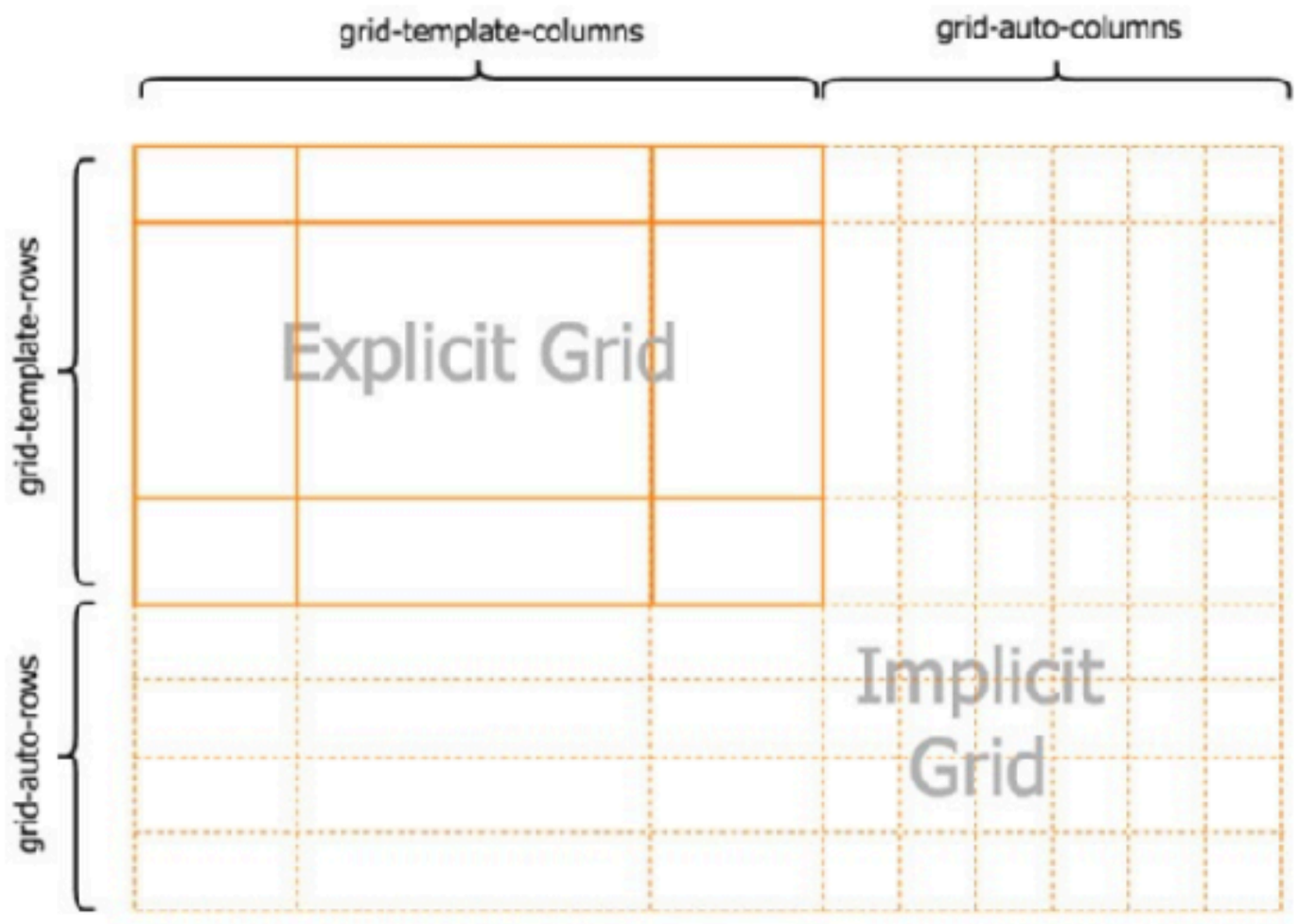
Размер для неявных гридов

Дополнительный ряд отличается высотой от предыдущих двух.

Мы выставили высоту строки grid-template-rows в свойстве, но применяется она только для явных гридов.

Высота строки на неявных гридах должна выставляться с помощью свойства grid-auto-rows.

Неявный ряд использует размер трека auto, который основывается на контенте.



Явный грид использует `grid-template-rows` и `grid-template-columns`

Неявный грид использует `grid-auto-rows` и `grid-auto-columns`

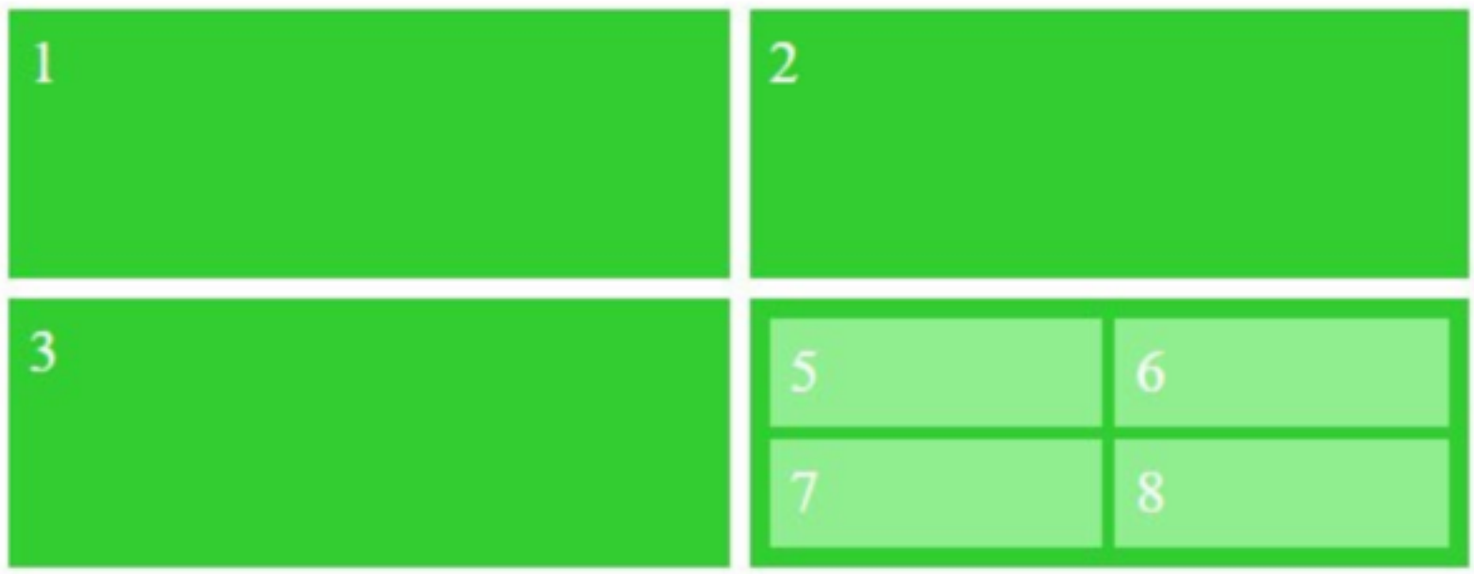
Сделаем явные и неявные строки одной высоты (`60px`).

https://codepen.io/lostsou41216364/pen/qJLXGa

Вложенный грид

Грид элементы могут сами становиться гридами в CSS Grid. То есть можно вкладывать грид-элемент в другой грид-элемент, тем самым создавая вложенный грид.

Чтобы создать такой вложенный грид, все что вам нужно сделать — это применить `display: grid` (или `display: inline-grid`) к грид элементу и он сам станет гридом. Вы также можете использовать `display: subgrid`для создания подгрида.



https://codepen.io/lostsou41216364/pen/BqvwmM

Наследование

Большинство грид свойств не наследуются, что означает то, что ваш вложенный грид не будет наследовать значения своего родительского грида.

Это позволяет вам вносить изменения в родительский грид, без непреднамеренного влияния на вложенный грид.

Для примера, вы выставили `grid-auto-flow: column` на родительском гриде, но вы не выставили свойство на вложенный грид.

В этом случае, вложенному гриду будет выставлено значение row, потому что это изначальное значение для этого свойства.

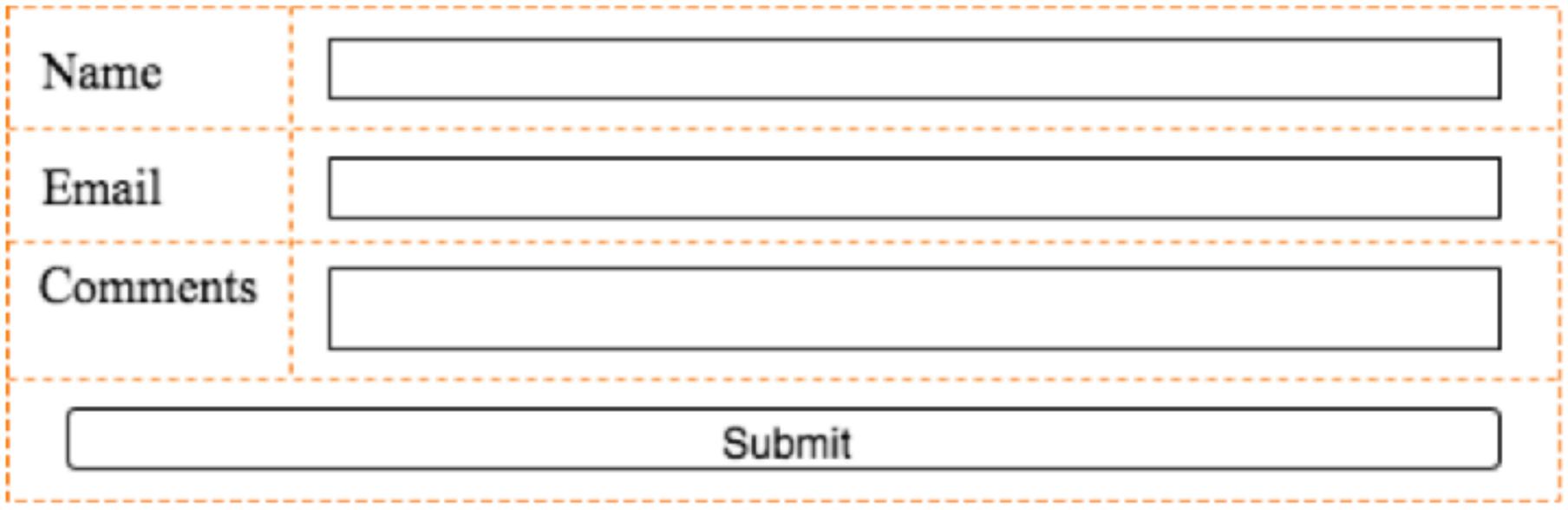
На родительском гриде числа идут вертикально вниз по колонкам, вместо того, чтобы располагаться горизонтально вдоль строки, но вложенный грид все таки идет в горизонтальном направлении вдоль строки.

https://codepen.io/lostsou41216364/pen/pxqWqg

Формы с авто-размещением

Вы можете использовать явные гриды как приемущество при создании форм или любого другого скопления элементов, которым требуется выравнивание в виде гридов.

Для примера, вы можете использовать явные гриды для создания такой формы:



И когда вы будете добавлять элементы формы в разметку, явный грид будет добавлять строки, чтобы уместить их.

https://codepen.io/lostsou41216364/pen/rqoYVz

Нет нужды в дополнительной разметке, чтобы правильно все расставить.

Также нет нужды в добавлении дополнительных классов для элементов формы.

Добавляем только один класс для `<form>` элемента, но хотя даже и это опциоально.

Автоматически можно добавлять новые элементы формы и они будут автоматически подстроены под гриды, так как они будут размещены в явном гриде.

Это является возможным, так как сделали саму форму гридом (к `.myForm` применено `display: grid`). И потом указали, что `labels` идут в одну колонку, а controls в другую.

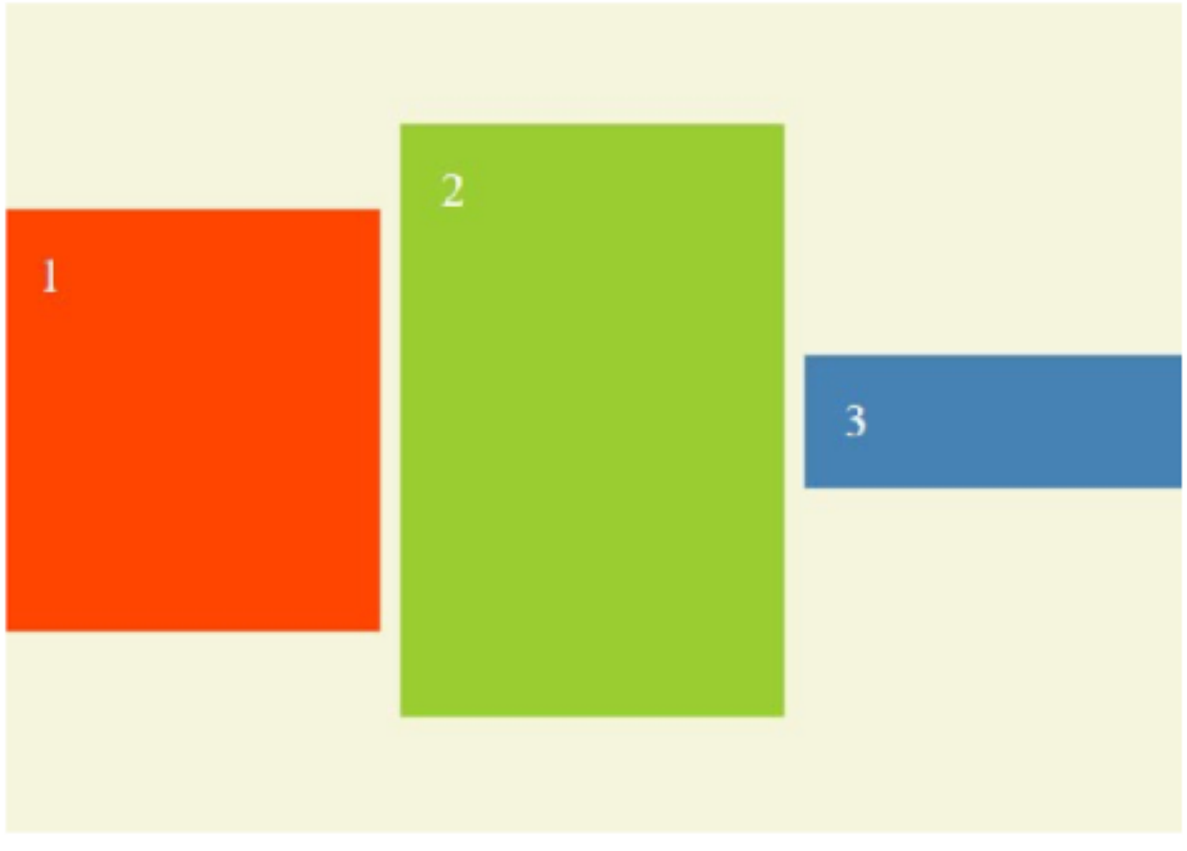
https://codepen.io/lostsou41216364/pen/PyXOEq

CSS Grid выравнивание

В основном, большинство свойств выравнивания работает также на грид элементах, как и на других элементах.

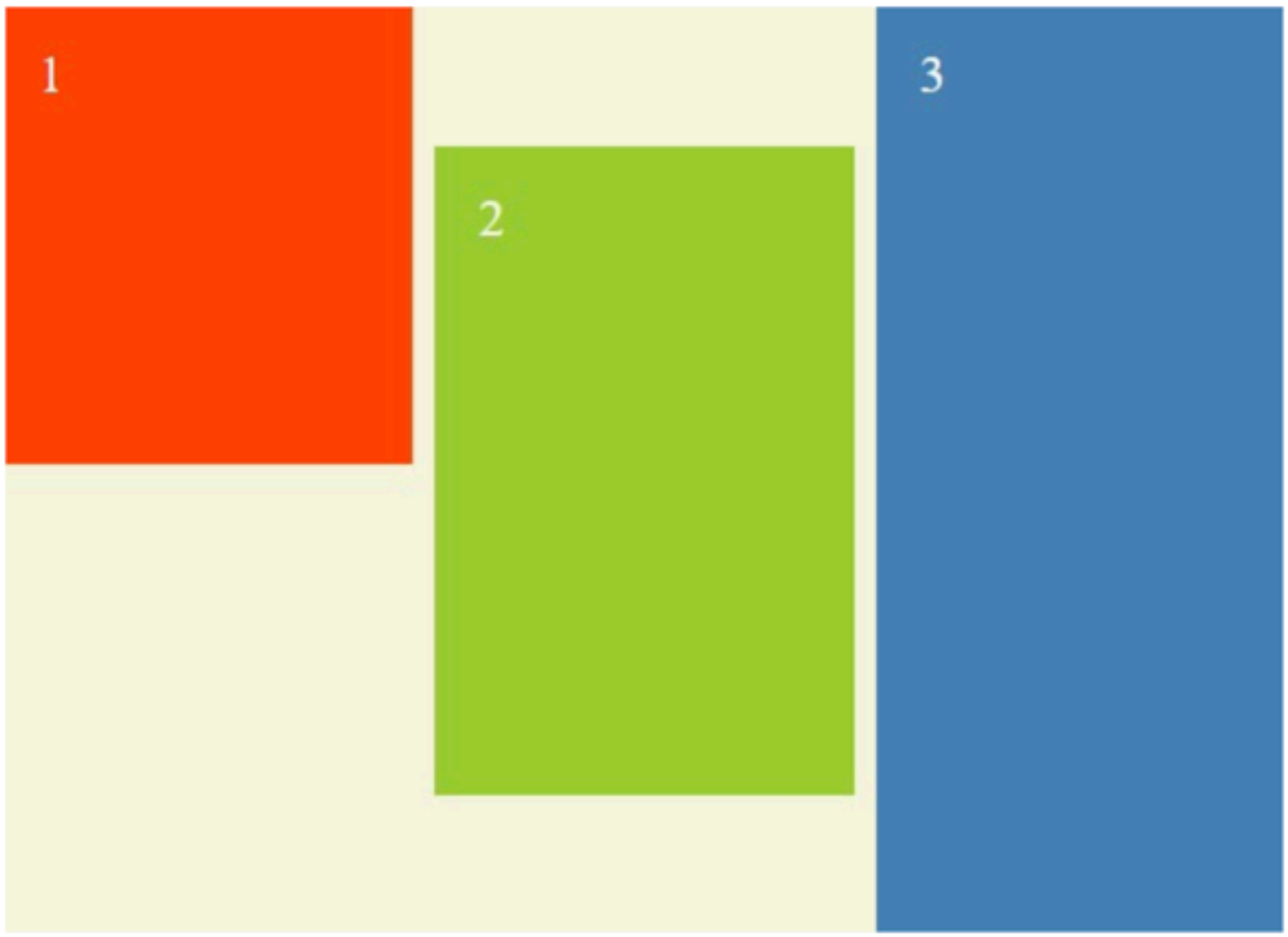
Но также есть некоторые свойства выравнивания, которые применяются только для гридов и флексов.

Свойство `align-items` указывает стандартное значение `align-self` для всех грид элементов участвующих в грид контейнере в контексте форматирования оного.




```
align-items: center;
```

```
align-items: baseline;
```



Свойство justify-items

Это свойство указывает стандартное значение для `justify-self` значения всех грид элементов в грид контейнере.

```
justify-items: center;
```

Свойство justify-self

Это свойство может использоваться для выравнивания индивидуальных грид элементов вдоль строчной/линейной/главной осей.