Продвинутая работа с анимацией, создаем игру на html/css

Введение

Давайте как обычно разберёмся а каких знаний нам еще не хватает.

1. Работа с Анимацией
2. Новые возможности препроцессоров
3. Новые возможности добавления счетчика на сайт
4. Создание внешнего вида игры

В каждой из этих тем мы что-то знаем, но после данного урока вы поймёте насколько могут быть функциональными препроцессоры и как сильно они сокращают написание css, конечно разберемся с созданием анимаций, какая веб игра без анимаций? Можем прикрутить функционал подсчета для кликов пользователя и всё это только с использованием html/css. Надеюсь мне удалось вас вдохновить, поэтому давайте переходить к созданию проекта.

Что мы будем с вами разрабатывать? Это будет очень популярная игра из прошлого “Охота на уток”. Цель игры достаточно простая, вылетает утка, необходимо в нее попасть, если игрок попал, то утка пропадает, а счетчик попаданий увеличивается на единицу. Естественно утки летают по разным траекториям, мы можем настраивать скорость полета утки, так же мы сможем играть в эту игру на мобильном телефоне. Если сложить все технологии что будем использовать, то думаю уже хочется побыстрее приступить к изучению.

Счетчик кликов, на чистом html/css

Первое с чего мы начнем это с изучения новых возможностей, которые также появились в CSS3, конечно же это счетчик кликов. Давайте продумаем саму логику проекта и начнём с добавления самых простых элементов. Как мы в html можем добавить изображение и к нему применить клик, конечно же это <input type="checkbox">, пока это просто чекбокс, мы должны задать ему класс, так как у нас таких “уток” будет несколько и уникальный id, чтобы каждая картинка отвечала за свой чекбокс

```
<input id="duck1" class="check" type="checkbox">
```

Далее нам потребуется добавить блок, где будет находить сам счетчик, тут можно использовать любой блок, в моём примере это будет заголовок h1, но нужно помнить что заголовок h1 должен быть один на странице, так что если вы размещаете эту игру в какой-то сайт, то использовать h1 не рекомендуется.

```
<h1 class="counter-info"></h1>
```

Заметим что внутри заголовка мы не размещаем ничего, у него есть только класс, к которому мы будем обращаться в препроцессорном файле.

Теперь нам нужно понять как это всё будете работать в css, для начала, я хочу подключить препроцессор, он нам не нужен для данного функционала, но пригодится в дальнейшем, на будущее советую всегда подключать препроцессор при старте работы с проектом.

```
*
    margin: 0
    padding: 0


body
    counter-reset: count
```

Самым интересным для нас является counter-reset: count, всё остальное мы уже знаем. В данной ситуации самое простое определение, это то что мы создали сам счетчик и назвали его **count**. Очень похоже как в любом программировании, мы объявили переменную и задали ей имя. Мини уточнение, мы задали это свойство к body, это не обязательно, в нашем примере игра будет занимать всё доступное пространство на экрана, но если вы работаете с каким-то отдельным блоком, то counter-reset нужно будет задавать именно к блоку, к которому относится.

```
.check:checked
    counter-increment: count
```

Данная часть отвечает за увеличение счетчика, получается, когда нас чекбокс будет отмечен галочкой, то мы увеличиваем count на единицу.

Единственное чего нам не хватает, это вывести данное значение в наш html, тут нам как раз пригодится тот самый заголовок, который мы создали и псевдоэлемент befor

```
.counter-info::before
    content: counter(count)
```

С помощью свойство content (с которым мы уже знакомы достаточно давно) мы можем добавить результат подсчета нашего счетчика внутри блока h1

Поздравляю! Мы смогли добавить счетчик, теперь можно добавить несколько таких чекбоксов и каждый выбранный из них, будет увеличивать значение count на единицу.

Добавляем внешний вид проекта

Первое с чего мы начнем это внешний вид чекбокса, мы не будем менять отображение самого чекбокса, нам всего лишь нужно добавить label внутри которого будет изображение нашей утки. Хочется отметить что изображение может быть любым так что если вы создаете такой же проект, лучше использовать другую картинку.

```
<input id="duck1" class="check" type="checkbox">

<label for="duck1">

<img class="duck__img"
src="https://thumbs.gfycat.com/EasyMerryAnteater-size_restricted.gif">

</label>

<h1 class="counter-info"></h1>
```

Что мы может отметить, важно привязать нашу картинку к input, для этого интупу создаём id=”duck1” а внутри label пишем for=”duck1”, теперь при нажати на картинку, мы можем включать или выключать наш чекбокс

0

Получается что сам checkbox нам больше и не нужен, в итоге их можно скрыть

```
.check

   position: absolute

   visibility: hidden

   left: -9999999px
```

Дальше мы можем задать любые стили для нашего изображения, в нашем примере это просто значение ширины

```
.duck

   &__img

       width: 40px
```

Создадим блок duck для нашей конструкции и продублируем его, чтобы получилось 3 утки.

```
<div class="duck">

   <input id="duck1" class="check" type="checkbox">

   <label for="duck1"><img class="duck__img"
```

```
src="https://thumbs.gfycat.com/EasyMerryAnteater-size_restricted.gif"><
/label>

</div>

<div class="duck">

   <input id="duck2" class="check" type="checkbox">

   <label for="duck2"><img class="duck__img"


src="https://thumbs.gfycat.com/EasyMerryAnteater-size_restricted.gif"><
/label>

</div>

<div class="duck">

   <input id="duck3" class="check" type="checkbox">

   <label for="duck3"><img class="duck__img"


src="https://thumbs.gfycat.com/EasyMerryAnteater-size_restricted.gif"><
/label>

</div>
```

0

Вы можете нажать на любую из уток и увидеть что счетчик меняется

Осталось только добавить фон для нашего проекта и разместить счетчик

```
body

background-image:
url(https://i.ytimg.com/vi/i9R8MqilMW8/maxresdefault.jpg)

   background-position: center

   background-size: cover

   height: 100vh

   counter-reset: count

   position: relative
```

Добавляем для нашего сайта фон, растягиваем его на всю высоту экрана и задаём точку отсчета.

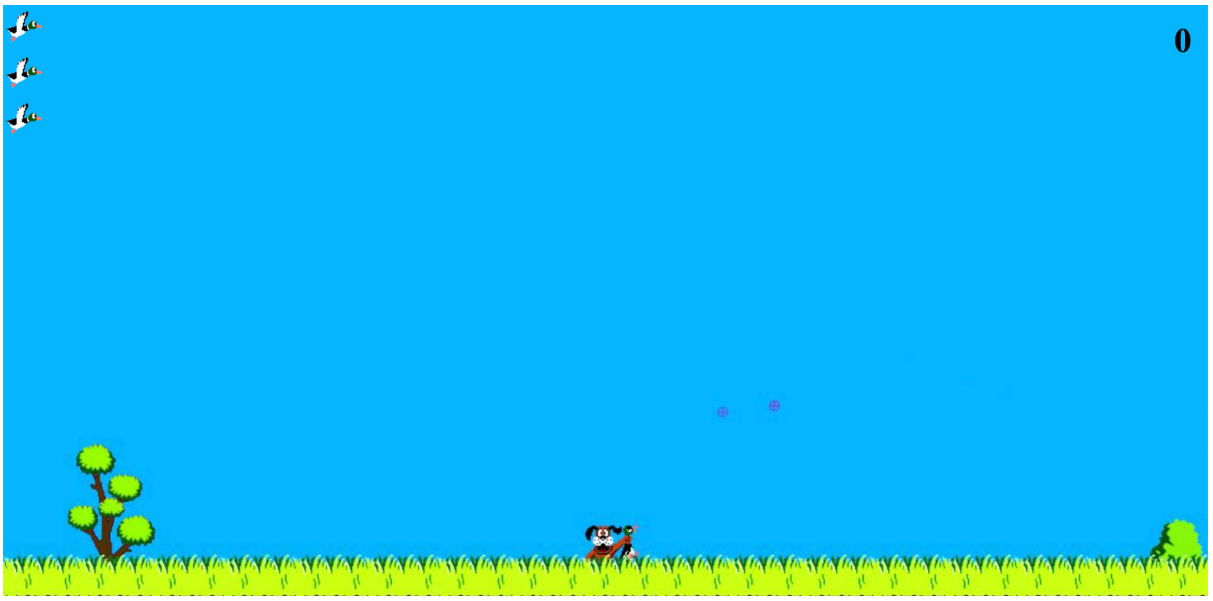
```
.counter-info

   position: absolute

   top: 16px

   right: 16px
```

Разместим наш счетчик в правый верхний угол, тут вы можете расположить значение где угодно, плюс вы можете задать любой текст внутри данного тега, может быть “Ваш счет: “



Добавление Анимации

Переходим к самой интересной части урока, конечно это добавление анимации. Первым делом нам нужно разобраться какого поведения полета уток мы ожидаем?

1. Утка должна вылетать из левой границы
2. Уток должно быть несколько.
3. Она должна двигаться не линейно
4. Всё поле она должна пролететь за 3 секунды
5. Каждая утка должна летать по уникальной траектории.

Мы с вами определили основные особенности, теперь давайте их реализуем.

Первым делом мы зададим для всех уток абсолютное позиционирование, чтобы элементы не толкались друг с другом и мы просто могли спрятать утки слева, за экраном.

```
.duck

   position: absolute
```

Теперь все утки в левом верхнем углу и им необходимо задать анимацию, для этого создадим keyframes, внутри которого и будет последовательность полета утки.

Внимание: Вы можете задать любое количество шагов анимации, мы будем использовать три.

```
@keyframes duckFly

   0%
```

```
    50%


    100%
```

Теперь нам потребуются препроцессоры, так как нам нужна рандомная точка в левой части и для этого мы будем использовать конструкцию **random()**. Это самая стандартная функция, которая возвращает случайное десятичное число от 0 до 1 и если вам необходимо случайное число от 0 до 100, вы можете написать random(100). Давайте подумаем какое число нас интересует? Мы не хотим чтобы утка вылетала из нижнего левого угла браузера, ведь там земля, поэтому я буду генерировать число от 0 до 90, 10% высоты экрана оставим для охотника. По левой границе мы должны спрятать изображение утки, мы знаем что ширина изображения равна 40px, поэтому логичным будет записать left: -40px, но изображение может быть больше, поэтому создадим переменную $widthDuck: 40px и если нужно будет поменять ширину то мы заменим значение переменной и ничего не сломаем.

Собираем всё воедино

```
@keyframes duckFly
    0%
        left: -$widthDuck
        top: random(90)+vh


    50%
        left: random(70)+vw
        top: random(90)+vh


    100%
        left: 100vw
        top: random(90)+vh
```

В 50% выполнения анимации отступ от левой границы может быть 50vw, но я ему решил так же добавить рандомной составляющей.

Мы продумали последовательность полета утки, осталось вызвать его

```
<div class="duck duck_1">
    <input id="duck1" class="check" type="checkbox">
    <label for="duck1"><img class="duck__img"
```

```
src="https://thumbs.gfycat.com/EasyMerryAnteater-size_restricted.gif"><
/label>

</div>
```

SASS

```
.duck

   position: absolute

   &_1

       animation: duckFly 3s linear 0s 1 normal forwards
```

Реализовали вызов анимации duckFly, время выполнения 3s, распределение вы можете оставить любое, тут для примера linear, и задержка для первой утки составляет 0s, количество итераций в нашем примере одна.

Теперь перед нами очень сложная задача, ведь уток может быть 3, а может быть 33, иили еще больше, для каждой из них мы должны вызвать анимацию и менять задержку вылета уток, чтобы они не вылетели все одновременно. Тут нам поможет цикл for из препроцессоров.

Добавление цикла for

Первым делом давайте разбираться с синтаксисом

```
@for $i from 1 through 3
```

Как мы можем заметить, если вы уже были знакомы хоть с одним языком программирования до этого, то мы создаем итератор $i, который идёт от 1, до значения 3, как раз то что нам необходимо, ведь у нас 3 утки, но если их станет 33, мне придётся поменять значение 3 на 33 и всё будет отлично работать.

```
.duck

   position: absolute

   @for $i from 1 through 3

       &_#{$i}

           animation: duckFly 3s linear 0s 1 normal forwards
```

Первым делом нам необходимо разобраться с селектором

```
.duck_1 {

}
```

```
.duck_2 {

}

.duck_3 {

}
```

Чтобы создать такую конструкцию, мы используем `&_#{$i}`

Где &_ возьмёт от родительского элемента .duck_ а дальше уже #{${i}} это как раз наша переменная, которая равная сначала 1, затем 2, 3.

На данный момент все 3 утки летают по одинаковой траектории, следовательно нам необходимо реализовать 3 анимации

```
$widthDuck: 40px
$iterationAnimation: 3


@for $i from 1 through $iterationAnimation
   @keyframes duckFly-#{$i}
       0%
           left: -$widthDuck
           top: random(90)+vh

       50%
           left: random(70)+vw
           top: random(90)+vh

       100%
           left: 100vw
           top: random(90)+vh
```

Вынесем в переменную `$iterationAnimation: 3` количество итераций цикла, так как он у нас используется при генерации последовательности и создании анимации.

На сколько удобно использовать цикл и не дублировать такой объемный код.

Осталось только вызвать определенную анимацию и увеличивать значения delay

```
.duck
```

```
    position: absolute
    left: -$widthDuck
    @for $i from 1 through $iterationAnimation
        &_#{$i}
            animation: duckFly-#{$i} 3s linear #{$i}s 1 normal forwards
```

`duckFly-#{$i}` Вызывает определенную анимацию

`#{$i}s` Задержка (1s, 2s, 3s) если вы хотите сделать старт без задержки `#{$i-1}s`

В итоге в css мы получаем

```
.duck_1 {
 -webkit-animation: duckFly-1 3s linear 1s 1 normal forwards;
         animation: duckFly-1 3s linear 1s 1 normal forwards;
}
.duck_2 {
 -webkit-animation: duckFly-2 3s linear 2s 1 normal forwards;
         animation: duckFly-2 3s linear 2s 1 normal forwards;
}
.duck_3 {
 -webkit-animation: duckFly-3 3s linear 3s 1 normal forwards;
         animation: duckFly-3 3s linear 3s 1 normal forwards;
}
```

Осталось только скрывать уток, при попадании в них

Тут вы можете проявить фантазию и выставить еще одну анимцию красивого исчезновения утки, но мы будем использовать стандартный способ

```
    &:checked + label
       display: none
```

Выводы

Мы с вами смогли реализовать веб игру и применить большую часть знаний из нашего курса. Чем больше вы работаете с препроцессорами и узнаете новые возможности HTML5/CSS3, тем лучше и качественней у вас будет получаться создание сайтов.

Сегодня в уроке мы узнали про создание счетчика, без использования языка программирования, добавление циклов в препроцессора sass, создание функционала веб игры, данная игра будет отлично работать на любых мобильных устройствах с сенсорным экраном и доступом в интернет.

Данный курс, получился максимально насыщенным и достаточно объемным, надеюсь вы нашли много новых и профессиональных приемах создания проектов любой сложности.