JAVASCRIPT

Javascript (сокращённо JS) – это **язык программирования**, который изначально был придуман для браузера, чтобы придать страницам **интерактивность и динамичность**. Программы написанные на javascript называются **сценариями** или **скриптами**.

Занятие 2. Условный оператор. Управление стилями

1. Повторение
2. Как «подружить» CSS и JS
3. «Лампочка, гори!»
4. «Выключите свет»
5. Принимаем решения в коде
6. «Приручи дракона»
7. Домашнее задание

1. Повторение

“3 кита” в JS. Как взаимодействуют html + js?

Алгоритм

1. В HTML нужно создать элемент для вывода данных – указать для него уникальное имя

2. Найти этот элемент через JS

3. Записать ответ в элемент

1. Добавить атрибут id

```
<div id="output">
    <!-- место для данных -->
</div>
```

index.html

2. Найти элемент на странице по id

```
let outputNode = document.querySelector(`#output`);
```

3. Записать ответ в этот элемент

```
outputNode.innerHTML = `Это увидит
                        пользователь!`;
```

js/index.js

1. Повторение

Событие `click`.
Как обработать событие `click`?

Алгоритм

1. В HTML нужно создать элемент для вывода данных – указать для него уникальное имя

2. Найти этот элемент через JS

3. Записать ответ в элемент

index.html

```
1. Добавить кнопку с атрибутом id
<button id="click">Кнопка</button>
```

```
2. Найти кнопку на странице по id
let buttonNode = document.querySelector(`#click`);

3. Подписаться на событие Клик по кнопке
buttonNode.addEventListener(`click`, function () {
    console.log(`Меня нажали!`);
});
```

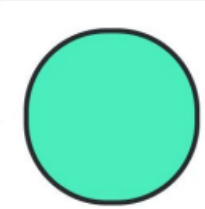
js/index.js

2. Как «подружить» CSS и JS

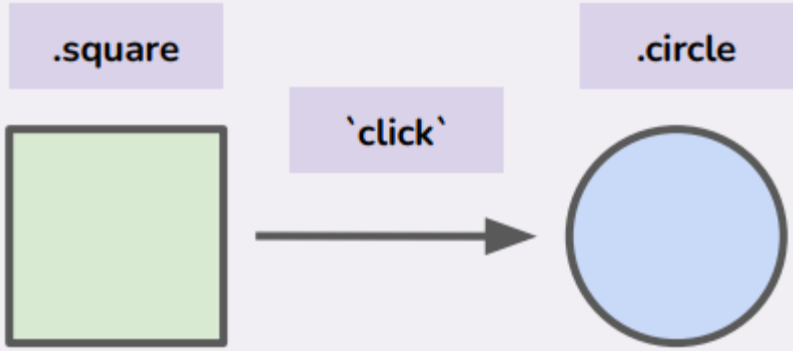
С помощью CSS мы можем переключить элемент из одного состояния в другое.

Для этого используют псевдоклассы.
У этого способа есть недостаток. Какой?

```
div {
    border: 4px solid black;
}
div:active {
    border-radius: 50px;
}
```



С помощью JS мы можем переключать элемент из одного состояния в другое с помощью CSS классов.



2. Как «подружить» CSS и JS

elem.classList – это специальный объект с методами для добавления/удаления одного класса.

Методы:
elem.classList.add/remove("class") – добавить/удалить класс

https://learn.javascript.ru/styles-and-classes#classname-i-classlist

```
По клику:
// добавить класс
node.classList.add("circle");
// удалить класс
node.classList.remove("square");
```

До

```
<div id="node"
     class="square">
</div>
```

После

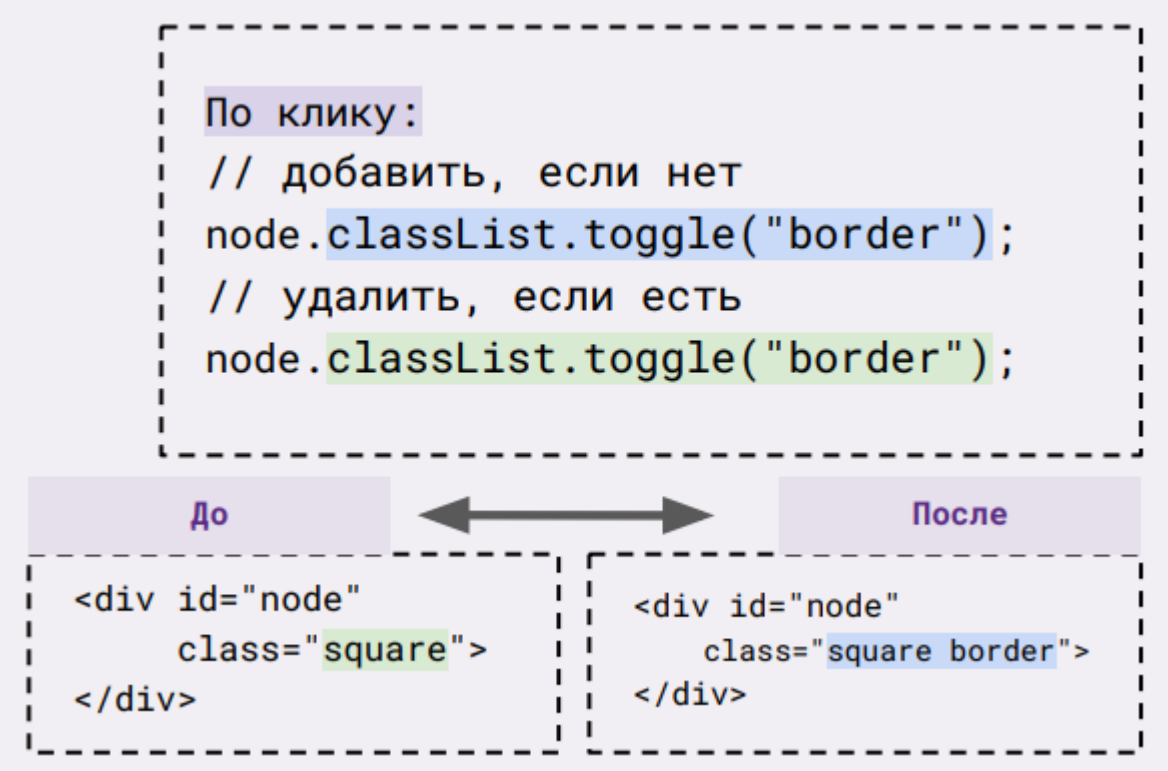
```
<div id="node"
     class="circle">
</div>
```

2. Как «подружить» CSS и JS

elem.classList – это специальный объект с методами для добавления/удаления одного класса.

Методы:
elem.classList.toggle("class") – добавить класс, если его нет, иначе удалить

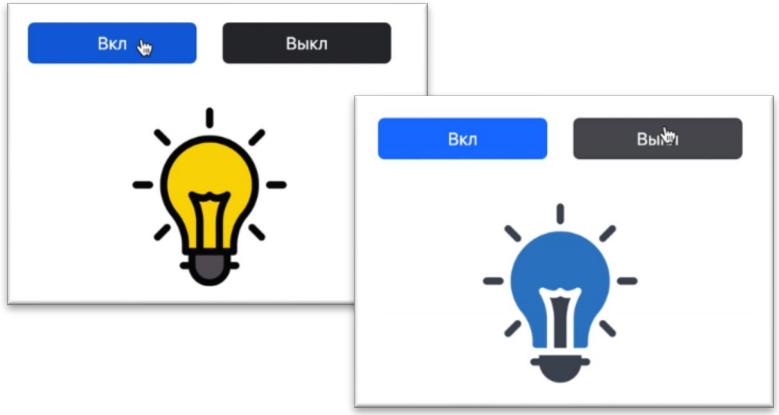
https://learn.javascript.ru/styles-and-classes#classname-i-classlist



3. Задача «Лампочка, гори!»

Сейчас на многих сайтах есть классная фича — тёмная и светлая тема. Попробуем сделать похожий механизм!

Сделай так, чтобы по клику на кнопки выключателя лампочка загоралась или гасла.



Алгоритм решения

1. Скачай архив lamp.

2. Добавь элементам в вёрстке атрибуты со следующими названиями:
Вкл – on
Выкл – off
Лампа – lamp. Какому тегу добавлять атрибут для лампы?

3. Заполни пропуски для кнопки ВЫКЛ

```
let offNode = [redacted](`#off`);
```

Добавь обработчик события клик:

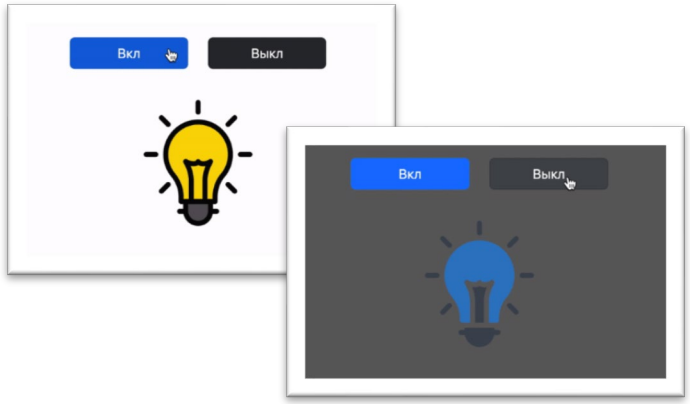
```
offNode.[redacted](`[redacted]`, function(){
    [redacted].innerHTML = `<img class="img img-fluid"
    src="assets/dark.png" />`
});
```

4. Настрой кнопку ВКЛ

4. Задача «Выключите свет»

С помощью CSS классов мы можем изменять внешний вид странички в ответ на действия пользователя

Никто не любит, когда свет включают резко, это бьёт по глазам.
Сделай так, чтобы фон страницы менялся плавно в течение 1 секунды с чёрного на белый и обратно.



Алгоритм решения

Продолжаем работать с проектом lamp.

1. Добавим css классы. В качестве фона страницы нам отлично подойдёт элемент с классом container. Добавь ему id="page" и напиши новые стили.

```
.dark {
background-color: black;
}
.container {
  transition: 1s;
}
```

2. При нажатии на кнопку ВЫКЛ нужно добавить класс dark, чтобы фон контейнера стал черным.

```
offNode.addEventListener(`click`, function(){
    pageNode.classList.add('dark');
    lampNode.innerHTML = `<img class="img img-fluid"
    src="assets/dark.png" />`
});
```

3. Сейчас лампочка появляется резко, хотя фон ещё плавно темнеет.

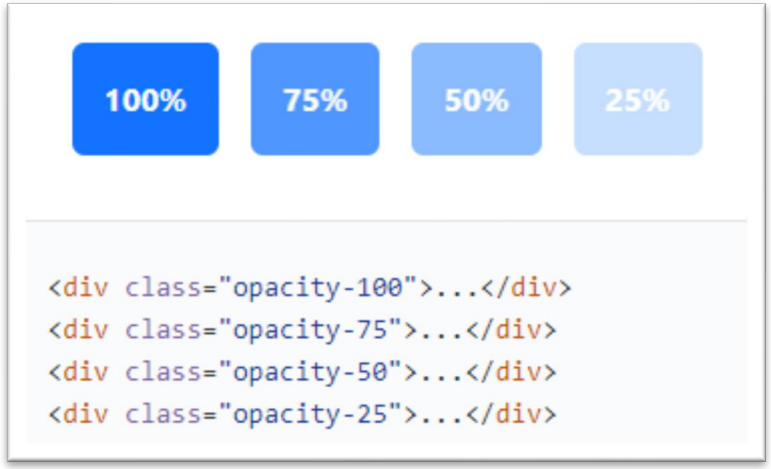
Давай попробуем их синхронизировать!

4. Задача «Выключите свет»

Алгоритм решения

Продолжаем работать с проектом lamp.

С помощью классов можно не только менять внешний вид, но ещё **показывать и скрывать элементы на странице**.



В этом поможет CSS свойство **opacity (**непрозрачность)

opacity: 0; делает элемент невидимым

opacity: 100; делает его видимым

opacity: 50; полупрозрачным :)

В Bootstrap есть целый класс, посвящённый этому свойству!

https://getbootstrap.com/docs/5.2/utilities/opacity/

Добавь для элемента лампочка класс opacity-0 (в функции ВЫКЛ), и удали для ВКЛ.

Чтобы лампочка появлялась и исчезала плавно, добавь для неё в CSS такое же время перехода, как и для фона страницы.

Так переходы будут выглядеть плавно для чувствительных глаз пользователей, и они точно останутся довольными :)

5. Принимаем решения в коде

Давайте напишем условие для входа в игровую локацию.

Она достаточно сложная, поэтому туда можно не всем игрокам.

```
если игрок не достиг 50 уровня,
    то не пускаем его в эту локацию
```

5. Принимаем решения в коде

ОПЕРАТОР **If**

Если

То

```
let level = 15;

if (level < 50) {
    console.log(`Эта локация
    для тебя закрыта`);
}
```

ОПЕРАТОР **If else**

Если

То

Иначе

```
let level = 60;

if (level < 50) {
    console.log(`Эта локация
    для тебя закрыта`);
} else {
    console.log(`Добро пожаловать,
    постарайся дойти до конца.`);
}
```

5. Принимаем решения в коде

ОПЕРАТОР **If + else if + else**

Если

Если

Иначе

```
let level = 95;

if (level < 50) {
    console.log(`Эта локация
    для тебя закрыта`);
} else if (level < 80) {
    console.log(`Добро пожаловать,
    постарайся дойти до конца.`);
} else {
    console.log(`Мобы тебя боятся,
    проходи мимо!`);
}
```

ОПЕРАТОРЫ **СРАВНЕНИЯ**

```
1.    7      !=    10
2.  "Женя"   !=  "Саша"
3.    5      ==    5
4.  "Катя"   ==  "Катя"
5.    12     >=     3
6.    5      >      1
7.    100    <=    1000
8.    100    <     1000
```

5. Принимаем решения в коде

Задача для начинающих: Помоги купить билет в самолёт!

Не хочу лететь в хвосте, поэтому билеты с 30 до 50 брать не буду.

А ещё не люблю число 13, не буду даже объяснять, почему…

Напиши скрипт на JS, который помогает определить пассажиру подходит ли ему, предложенное место для посадки в самолет или нет?

https://learn.javascript.ru/ifelse#instruktsiya-if

5. «Приручи дракона»

Напиши программу, которая побеждает дракона.

Благодаря оператору **if** мы можем создавать **МноГоХоДовочКи,** то есть программы, которые ведут себя по-разному в зависимости от ситуации. Например, в играх, в случае с боёвкой часто бывает, что некоторые атаки наносят больше урона, чем другие.

Вот задача для начинающих разработчиков:

Первая атака наносит - 1 урон (т.е. по клику на изображение дракона, количество его здоровья уменьшается на 1)

Вторая атака наносит - 3 урона (количество здоровья уменьшается еще на 3 и т.д.)

Третья атака наносит - 10 уронов

Четвёртая атака наносит - 1 урон

Пятая атака наносит - 3 урона

И так далее.....

Пока Здоровье дракона не станет отрицательным или равным нулю. В таком случае игру стоит остановить. На экране должна появиться картинка с изображением фрукта и сообщением о победе над драконом.

5. «Приручи дракона»

Реши задачу по алгоритму:

1. Скачай архив dragon

2. id должны быть у дракона (чтобы кликать по нему) и у здоровья (чтобы выводить в этот элемент, сколько здоровья осталось)

Найди эти элементы с помощью JS. (Не забудь сначала создать файл index.js и подключить его к файлу index.html)

3. Опиши с помощью синтаксиса JS

```
// объяви переменную health, в которой будем хранить здоровье дракона
// изначально у него 50 единиц здоровья

//enemyNode.addEventListener(`click`, function(){
    // уменьшай здоровье на кол-во уронов
    // показывай результат на экране
    // выглядеть должно так 'Здоровье: 50'
//});
```

5. «Приручи дракона»

4. Чтобы сделать механику атаки, нам нужно понять, какой удар был первым, вторым и третьим.

Для этого введём новую переменную — **счётчик атак**

5. Заполни по комментариям пропуски

Содержимое внутри if else заполни самостоятельно, используя комментарии как подсказки.

В результате при нажатии на дракона счётчик здоровья будет уменьшаться намного быстрее.

```
// объяви переменную count со значением 0
// с помощью неё ты будешь отсчитывать, какой сейчас удар
[redacted]
enemyNode.addEventListener(`click`, function(){
    // увеличивай при каждом клике счётчик атаки на 1
    // нам нужен счетчик, чтобы понимать на каком клике, сколько урона здоровья
    [redacted]

    if([redacted]){
        // если атака первая, уменьшай здоровье на 1
        [redacted]
        // count++;
    } else if([redacted]){
        // если атака вторая, уменьшай здоровье на 3
        health = health - 3;
        // count++;
    } else if(count == 3){
        // если атака третья, уменьшай здоровье на 10
        [redacted];
        // count++;
    } else if([redacted]){
        // если атака третья, уменьшай здоровье .... и т.д.
        [redacted];
        // count++;
    } else if([redacted]){
        [redacted];
        // сбрось счётчик атаки обратно к нулю
        [redacted];
    }
```

5. «Приручи дракона»

6. Сейчас дракона можно побеждать бесконечно, его здоровье улетает далеко за пределы нуля) Это, конечно, нужно немедленно поправить.

И поможет здесь условный оператор!

В конце игры поменяй картинку с драконом и выведи сообщение:
Вы побеждаете дракона. Он на ваших глазах уменьшается и становится маленьким и симпатичным. Возьмём его с собой?

Пользуйся подсказкой с комментариями.

```
if(health ){
    // продолжаем сражение
} else {
    // конец игры
}
```

5. Домашнее задание

Задача для начинающих: Угадайка!

Создай страницу с мини-игрой, где игроку нужно отгадать, что за слово было загадано, и написать его по-английски

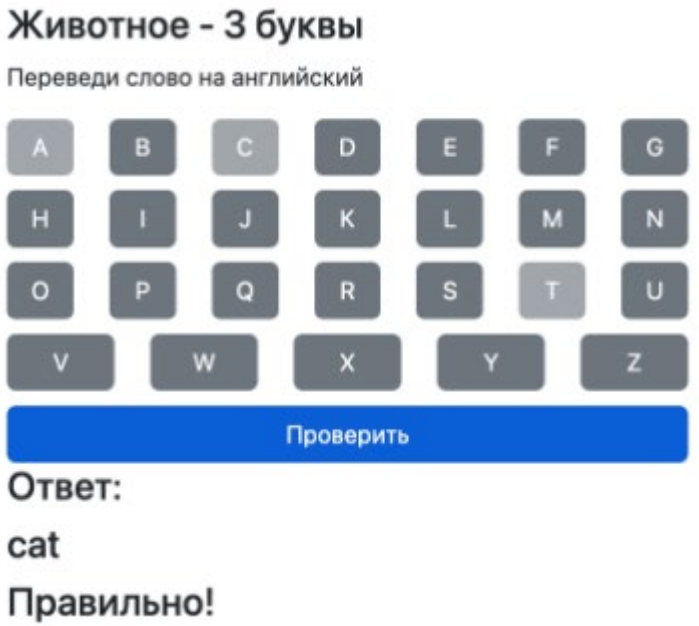
Вот некоторые требования к работе игры:

1. По нажатию на кнопку с буквой в элемент answer появляется новая буква

2. Каждую букву можно использовать только один раз. После нажатия на букву она становится ненажимаемой

3. При нажатии на кнопку Проверить под словом показывается результат

У игрока есть 3 попытки угадать слово.



5. Домашнее задание

Алгоритм скрипта игры Угадайка!

1. Скачай архив translator

2. Заставь кнопки печатать буквы всего за одну строчку. Дополни в скрипте функции Клик по кнопкам с буквами, чтобы по нажатию кнопки на экране печаталась соответствующая буква.

Понимаю, что работа одинаковая… неинтересная… можно упростить и т.п. Сделайте для 3-х правильных букв букв c, a, t.

3. Выключай кнопки после нажатия. Для этого вBootstrap есть специальный класс disabled. Если добавить его кнопке, по ней нельзя будет нажимать (на примере вышеуказанных 3-х букв)

При выполнении условия нажатия на кнопку, добавь кнопке класс disabled.

4. Выведи ответ. По нажатию на кнопку Проверить показывай, правильный ответ или нет.
Допустим, ответ в нашей игре это слово cat. Если пользователь напечатал правильно cat, напиши правильно. Если напечатал неверно (например, tac) – неправильно.

5. Самостоятельно сделай так, чтобы в игре было всего 3 попытки на угадывание слова.