Конспект «Динамические стили элементов»

Свойство style

С помощью свойства `style` можно управлять стилями элемента. После `style` через точку указывают то CSS-свойство, которое нужно изменить. Чтобы изменить стиль элемента, указанному свойству нужно присвоить новое значение

```
let element = document.querySelector('p');



// Зададим абзацу зелёный цвет текста

element.style.color = 'green';
```

Стили, заданные с помощью свойства `style`, работают так же, как если бы их указали в разметке в атрибуте `style` самого элемента. Они имеют больший приоритет, чем CSS-правила из файла со стилями.

В JavaScript нельзя использовать дефисы в названиях свойств, вместо этого для разделения слов используется «верблюжий» стиль. Например:

CSS	JavaScript
font-size	fontSize
background-color	backgroundColor
border-left-width	borderLeftWidth

Если мы получаем данные из поля ввода, то чтобы задать единицы измерения, используем конкатенацию:

```
longread.style.fontSize = sizeSetting.value + 'px';

// Допустим, пользователь ввёл число 16

longread.style.fontSize = 16 + 'px'; // Результат: '16px'
```

Чтобы узнать, какие стили уже применяются к элементу, используйте метод `window.getComputedStyle`.

Обработчики onchange и oninput

Чтобы отслеживать изменения в поле ввода, можно использовать обработчики `onchange` и `oninput`. Разница между ними заключается в следующем:

- `onchange` срабатывает, если значение поля ввода изменилось и пользователь закончил ввод. Например, если пользователь передвинул ползунок и отпустил его. Или ввёл что-то в текстовое поле и убрал из него курсор.
- `oninput` срабатывает на каждое изменение значения, независимо от того, завершил пользователь ввод или нет. Например, он сработает на каждое изменение положения ползунка, даже если пользователь продолжает его двигать. И на каждый новый символ в текстовом поле, даже если пользователь продолжает вводить текст.

Когда мы меняем размер элементов, браузеру приходится перерисовывать страницу. Это трудоёмкая операция, и лучше выполнять её как можно реже. Используйте обработчик `oninput` с осторожностью.

Свойство type

Для ввода пароля используется специальное поле с типом `password`. Особенность этого поля в том, что введённый текст в нём маскируется. Обычно браузеры используют для этого звёздочки или кружочки. Подробнее о типах полей вы можете узнать из этой части.

Чтобы показать пароль, нужно превратить поле ввода пароля в текстовое поле ввода. Для этого надо изменить его тип на `text`. За тип в JavaScript отвечает свойство `type`. Чтобы изменить тип поля ввода, нужно записать в свойство `type` новое значение:

```
let input = document.querySelector('input');



// Сделаем input текстовым полем ввода

input.type = 'text';
```

Чекбокс и свойство checked

Чекбокс — это поле ввода, у которого может быть одно из двух состояний: включён или выключен. Обычно в браузерах на состояние чекбокса указывает галочка: если галочка стоит — чекбокс включён, если галочки нет — выключен. Когда галочку ставят или убирают, состояние чекбокса меняется. Чтобы отследить это событие из скрипта, используем обработчик событий `onchange`.

Чтобы проверить состояние чекбокса, используем свойство `checked`. Это свойство имеет булево значение: `true`, если чекбокс включён, и `false`, если нет.

```
// Проверяем, включён ли чекбокс

  if (showPassword.checked) {
```

```
    // Инструкции выполнятся, если чекбокс включён

  } else {

    // Инструкции выполнятся, если чекбокс выключен

  }
```

Операторы сравнения и умножение

В JavaScript есть несколько операторов сравнения. Эти операторы работают так же, как в математике:

```
1 > 2;  // Оператор «больше»

1 < 2;  // Оператор «меньше»

1 >= 2; // Оператор «больше или равно»

1 <= 2; // Оператор «меньше или равно»
```

Операторы «больше или равно» и «меньше или равно» обозначаются двумя символами: угловой скобкой и знаком равно.

Умножение в JavaScript обозначается звёздочкой:

```
console.log(3 * 10); // Выведет: 30
```

Конструкция else if

Конструкция `else if` позволяет добавить в условную конструкцию альтернативную ветку с условием. После `else if` в круглых скобках указывают условие, а в фигурных — инструкции, которые должны выполниться, если это условие вернёт `true`.

```
if (условие-1) {

  // Инструкции выполнятся, если истинно условие-1

} else if (условие-2) {

  // Инструкции выполнятся, если условие-1 ложно, а условие-2 истинно

} else {

  // Инструкции выполнятся, если оба условия ложны

}
```

JavaScript выполняет инструкции сверху вниз. Сначала он проверит `условие-1`. Если оно истинно, выполнятся инструкции из первой ветки. Если `условие-1` ложно, JavaScript

проверит `условие-2`. Если оно истинно, выполнятся инструкции из второй ветки. Если оба условия ложны, то выполнятся инструкции из ветки `else`.

Веток `else if` в условной конструкции может быть сколько угодно. Но чем их больше, тем запутаннее получается код.