

Работа с событиями

Урок 9





Кадочников Алексей

Frontend-разработчик

- ☀ Веб-разработчик со стажем более 9 лет
- ☀ Преподаватель GeekBrains с 2015 года
- ☀ Автор курсов по Frontend на портале Geekbrains
- ☀ Работал в таких компаниях, как VK и Wizard-C



План курса

1

Знакомство с JavaScript

2

Основы JavaScript

3

Функции в JavaScript

4

Циклы и массивы

5

Работа с объектами в JavaScript

6

Введение в DOM

7

Работа с DOM

8

Основы событий в JavaScript

9

Работа с событиями в JavaScript

10

Основы шаблонизации, работа с JSON

11











Работа с медиафайлами

12

Основы API, итоги курса



Что будет на уроке сегодня

-  Прерывание распространения события
-  Действия по умолчанию и отменяемые события
-  Генерация событий, пользовательские события
-  Как браузер использует DOM
-  Построение модели визуализации (Render tree)
-  Построение DOM-модели
-  Построение CSSOM-модели
-  Выполнение JavaScript
-  Создание макета (Layout)
-  Растеризация (Painting)



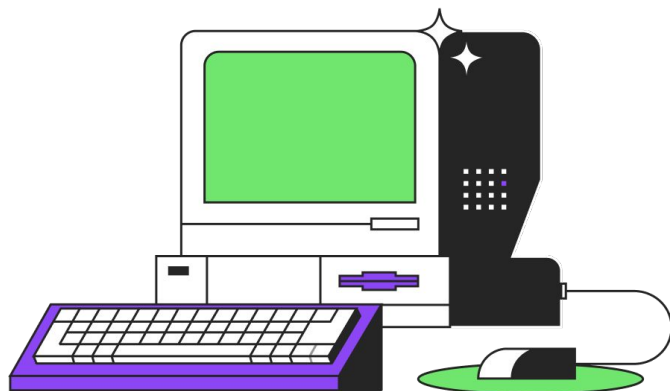
Прерывание
распространения
события





Прерывание распространения события

- `Event.stopPropagation()` — метод остановит дальнейшее распространение события вверх по DOM-дереву в текущей и последующих фазах.
- `Event.stopImmediatePropagation()`.





Действия
по умолчанию
и отменяемые события

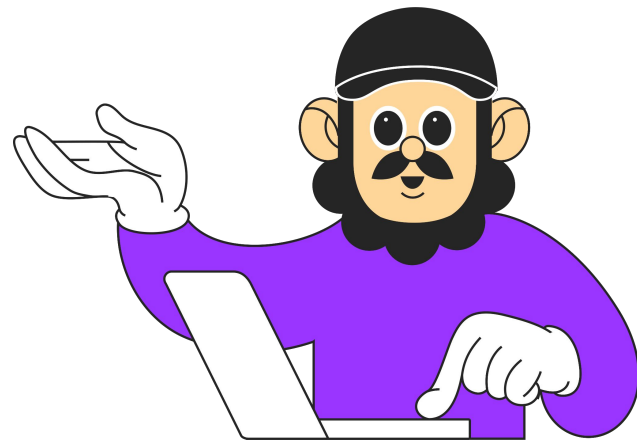




Действия по умолчанию и отменяемые события

Отменяемые объекты событий часто связаны с одним или несколькими действиями по умолчанию. Чтобы отменить действие по умолчанию, надо вызвать метод **preventDefault()** на объекте события.

Например, события **mousedown** отправляются сразу после того, как пользователь нажимает кнопки мыши или другого указательного устройства.





Действия по умолчанию и отменяемые события

Действия по умолчанию обычно выполняются после завершения отправки события, но в исключительных случаях также могут выполняться прямо перед отправкой события.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Привет, DOM-события!</title>
5   </head>
6   <body>
7     <form name="myForm">
8       <label>Тестовый checkbox: <input type="checkbox"></label>
9     </form>
10  </body>
11 </html>
```



Действия по умолчанию и отменяемые события

Действие по умолчанию, связанное с событием `click` для элементов `<input type="checkbox">`, переключает значение атрибута `checked` этого элемента. Если действие `click` по умолчанию для события отменяется, значение возвращается в прежнее состояние.

```
1 const checkbox = document.querySelector('input[type=checkbox]')
2
3 checkbox.addEventListener('click', (event) => {
4   console.log(event.target.checked) // true
5   event.preventDefault()
6 })
7
8
```



Действия по умолчанию и отменяемые события

Возможность отмены действия по умолчанию указывается в атрибуте `cancelable` объекта события. Вызов `Event.preventDefault()` останавливает все связанные действия по умолчанию объекта события. Атрибут **`Event.defaultPrevented`** указывает, отменялось ли уже это событие, например, через предварительного слушателя событий.





Действия по умолчанию и отменяемые события

Многие реализации дополнительно интерпретируют возвращаемое из слушателя событий значение, например, `false`. Это означает, что действие по умолчанию для отменяемых событий отменится, хотя обработчики `window.onerror` отменяются возвратом `true`.

```
1 const checkbox = document.querySelector('input[type=checkbox]')
2
3 checkbox.addEventListener('click', (event) => {
4   console.log(event.target.checked) // true
5   return false
6 })
7
```



Генерация событий,
пользовательские
события





Генерация событий, пользовательские события

Специализированные конструкторы типа `MouseEvent`, в отличие от `Event`, поддерживают такие дополнительные поля, как `view`, `clientX`, `clientY` и др. В случае с конструктором `Event`, который эти поля не поддерживает, требуется дополнительный код для их добавления в объект события.



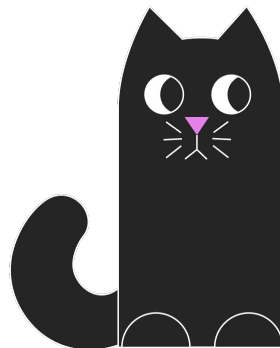
Генерация событий, пользовательские события

Встроенные события также генерируются методами **HTMLElement.click()**, **HTMLElement.focus()** и т. д.

Чтобы сгенерированное событие всплывало и было отменяемым, надо передать в опции поля **bubbles** и **cancelable**.

Отличить сгенерированное событие от собственного можно по полю `isTrusted` в обработчике события.

Важно! Обработчики сгенерированных событий выполняются синхронно.



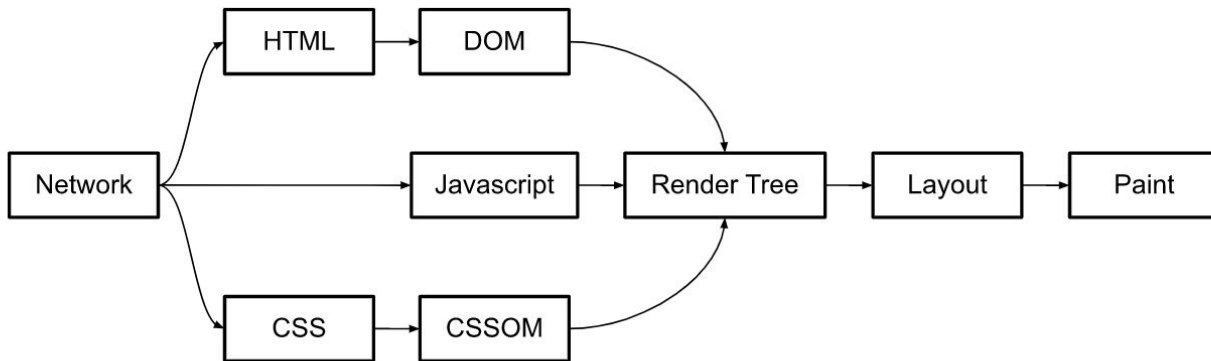


Как браузер
использует DOM





Как браузер использует DOM





Он разбивается на два больших этапа:

1. Анализ кода и построение модели визуализации (Render tree):

- Получение HTML-документа по сети или чтение с локального диска.
- Обработка HTML-кода и создание DOM, отправка дополнительных запросов.
- Обработка CSS-стилей и создание CSSOM.
- Выполнение синхронного JavaScript-кода.

2. Прямая отрисовка страницы:

- Создание макета страницы (Layout).
- Растеризация или визуализация страницы на экране (Painting).



Построение модели визуализации (Render tree)





Построение модели визуализации (Render tree)

Дерево рендера, или модель визуализации, — это специальная структура, состоящая из элементов, которые отобразятся на странице, и связанных с ними стилей. Такое дерево собирается из двух независимых структур данных (моделей).



Построение модели визуализации (Render tree)

DOM — объектная модель всех элементов страницы. Собирается на основе HTML-разметки страницы.

CSSOM — объектная модель таблицы стилей страницы. Собирается на основе CSS-разметки страницы.

Важно! Модель визуализации, состоящая из DOM и CSSOM, содержит только видимые элементы и не включает скрытых. Например, тех, в стилях которых указано *display: none*.



Построение DOM- модели





Построение DOM-модели делится на следующие этапы:

1. *Преобразование.* Браузер преобразует байты из HTML-файла, размещённого на диске или в сети, в символы, основываясь на приведённой в файле кодировке (например, UTF-8).
2. *Разметка.* На основании стандарта W3C HTML5 браузер выделяет среди символов такие теги в угловых скобках, как `<html>` и `<body>`. У каждого тега есть своё значение и набор правил.



Построение DOM-модели делится на следующие этапы:

3. *Создание объектов.* HTML-тегами браузер выделяет в документе объекты с конкретными свойствами.
4. *Формирование DOM.* Объекты образуют древовидную структуру, повторяющую иерархию HTML-файла, в котором одни теги помещаются в другие. Так, объект **p** помещается под **body**, а объект **body**, в свою очередь, — под **html** и так далее.



Построение CSSOM- модели





Построение CSSOM-модели

Объектная модель стилей (CSSOM) так же, как DOM, представлена в виде дерева объектов со связанными стилями для каждого узла, независимо от того, объявлены они явно или унаследованы неявно.





Выполнение JavaScript





Выполнение JavaScript

Для формирования модели визуализации браузер выполняет следующие действия:

1. Начиная с основания модели DOM, находит все видимые объекты.
 - a. Этот этап не затрагивает элементы, которые не будут видны на странице, например, теги скриптов, метатеги и т. д.
 - b. Он не затрагивает объекты, которые с помощью CSS помечены как невидимые.
2. Находит в CSSOM наборы стилей и присваивает их соответствующим объектам.
3. Формирует модель из видимых объектов, их содержания и стилей.



Создание макета (Layout)



Создание макета (Layout)

Макет — то, что определяет размер области просмотра и обеспечивает контекст для стилей CSS, которые зависят от него. Например, проценты или единицы просмотра.

Наиболее распространённое значение метаобласти просмотра — размер области просмотра, установленный в соответствии с шириной устройства.





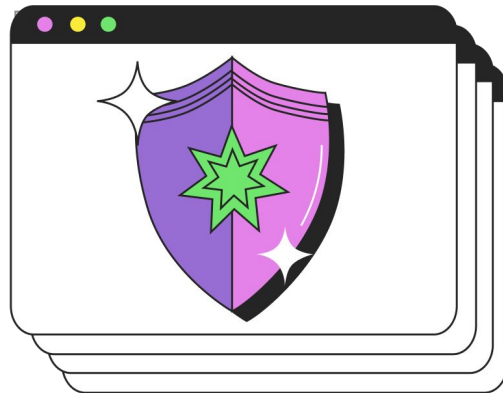
Создание макета (Layout)

Итак, браузер уже определил:

- какие объекты будут видны на странице;
- какие стили надо им присвоить.

Пришло время создать макет, то есть выяснить:

- какого размера окажутся объекты;
- как их надо расположить в области просмотра.





Создание макета (Layout)

Для этого браузер вычислит геометрическую форму объектов, проанализировав модель визуализации с самого начала. Рассмотрим простой пример:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta name="viewport" content="width=device-width,initial-
      scale=1">
5     <title>Здравствуй, мир!</title>
6   </head>
7   <body>
8     <div style="width: 50%">
9       <div style="width: 50%">Здравствуй, мир!</div>
10    </div>
11  </body>
12 </html>
```




Растеризация (Painting)

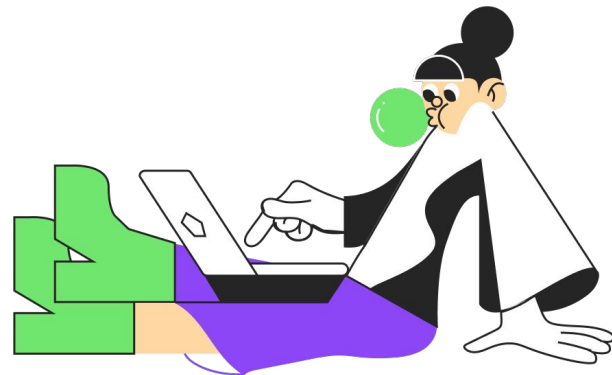




Растеризация (Painting)











Наконец, на этапе растеризации видимое содержимое страницы преобразуется в пиксели для отображения на экране.


Затраты времени на рисование зависят от размера модели DOM, а также от применяемых стилей. Воплощение одних стилей более «затратно», других — менее. Например, отрисовка сложного градиентного фона займёт больше времени, чем простая заливка фона одним цветом.





Итоги урока

-  Прерывание распространения события
-  Действия по умолчанию и отменяемые события
-  Генерация событий, пользовательские события
-  Как браузер использует DOM
-  Построение модели визуализации (Render tree)
-  Построение DOM-модели
-  Построение CSSOM-модели
-  Выполнение JavaScript
-  Создание макета (Layout)
-  Растеризация (Painting)

Спасибо 
за внимание

