Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа № 5

ДИНАМИЧЕСКИЕ WEB-СТРАНИЦЫ. ОБРАБОТКА СОБЫТИЙ.

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

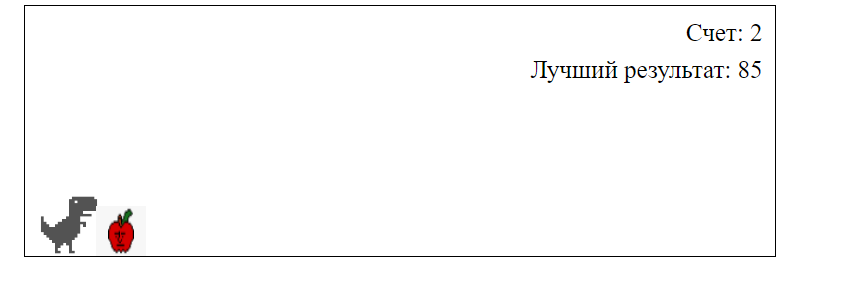
Проверила: старший преподаватель

Первова Наталья Викторовна

Чебоксары, 2024

**Цель работы.**

Изучить принципы разработки динамических web-страниц с использованием клиентских сценариев JavaScript и объектной модели документа. Научиться создавать кроссбраузерные html-страницы. Познакомиться с уровнями и особенностями DOM API и обработкой событий DOM различных уровней.

**Сценарий разрабатываемой игры и её интерфейс.**  
Игровая веб страница разработана по теме динозавра из браузера Chrome. Динозавр должен уворачиваться от кактусов и птеродактилей и по возможности подбирать яблоки, они дают доп очки.  
**Интерфейс:**  
  
**Описание интерфейса всех предполагаемых игровых уровней (состояний) в терминах DOM-модели.**

1. **Контейнер игры (container)**
   * Содержит все элементы игры, такие как персонаж, препятствия и счет.
2. **Персонаж (dino)**
   * Состояния:
     + **Нормальное состояние**: изображение динозавра (dinoImg).
     + **Прыжок**: добавляется класс jump, изменяющий положение динозавра.
     + **Обработка нажатия клавиши**: при нажатии на стрелку вниз картина динозавра меняется на картинку пригнувшегося динозавра, а при отжатии кнопки изначальная картинка возвращается.
3. **Препятствия**
   * **Кактус (cactus)**
     + Состояния:
       - **Скрыт**: display: none;
       - **Отображен**: display: block;
   * **Птеродактиль (ptic\_1)**
     + Состояния:
       - **Скрыт**: display: none;
       - **Отображен**: display: block;
   * **Яблоко (apple)**
     + Состояния:
       - **Скрыто**: display: none;
       - **Отображено**: display: block;
4. **Счет (score)**
   * Отображает текущий счет игрока.
   * Обновляется каждую секунду, пока игра активна.
5. **Лучший результат (bestScore)**
   * Отображает лучший результат, сохраненный в localStorage.
6. **Модальное окно (modal)**
   * Состояния:
     + **Скрыто**: не создается в DOM
     + **Отображено**: создается в DOM с сообщением о завершении игры и кнопкой для перезапуска.
   * Содержит:
     + Сообщение о завершении игры (#gameOverMessage).
     + Кнопка перезапуска (#restartButton).

**Краткое описание игрового сценария с указанием: используемых структур данных, переменных и их назначения, перехватываемых событий и последовательности их обработки.**

**Используемые переменные:**

* score: хранит текущий счет игрока (число).
* bestScore: хранит лучший результат, сохраненный в localStorage (число).
* isGameActive: булева переменная, указывающая, активна ли игра (true/false).
* timer, timer2: используются для установки интервалов (обновление счета и генерация объектов).
* dino, dinoImg, cactus, apple, ptic\_1: ссылки на соответствующие элементы DOM, представляющие персонажа и объекты игры.

**Перехватываемые события:**

* + **keydown**: перехватывает нажатие клавиш для управления динозавром.
    - **Пробел** (Space): вызывает функцию jump(), чтобы динозавр прыгнул.
    - **Стрелка вниз** (ArrowDown): изменяет изображение динозавра и его положение, чтобы он присел.
  + **keyup**: восстанавливает исходное состояние динозавра, когда клавиша стрелки вниз отпускается.

**Событие клика мыши:**

* Клик на модальном окне (при окончании игры) для закрытия окна.

**Последовательность обработки событий:**Если игра активна, каждую секунду увеличивается счет на 1.

Каждую секунду с помощью рандомайзера отображается кактус, птеродактиль или яблоко.

При нажатии на пробел вызывается функция jump, которая вызывает анимация прыжка.

При нажатии на стрелку вниз динозавр приседает, а при отжатии кнопки возвращается в исходное состояние.

Каждые 10 мс проверяется, не произошло ли столкновение динозавра с предметами.

Если произошло столкновение с птеродактилем или кактусом, то игра останавливается и отображается модальное окно.

При нажатии в модальном окне на кнопку “начать заново” счет сбрасывается и игра начинается сначала.

**Перечисление и описание используемых средств JavaScript (объектов, свойств, функций DOM API) для взаимодействия с DOM-деревом**

**document.getElementById()**

Возвращает ссылку на элемент по его идентификатору (ID); идентификатор является строкой, которая может быть использована для идентификации элемента; она может быть определена при помощи атрибута id в HTML или из скрипта.

Синтаксис

element = document.getElementById(id);

Параметры

Id - чувствительная к регистру строка, являющаяся уникальным идентификатором искомого элемента.

Возвращаемое значение

ссылка на объект типа Element соответствующий указанному ID или null, если элемент с указанным ID не найден в документе.

**Глобальная функция setInterval()**

Метод setInterval() доступен из интерфейсов Window и WorkerGlobalScope. Он циклически вызывает функцию или участок кода с фиксированной паузой между каждым вызовом. Уникальный идентификатор intervalID, возвращаемый методом, позволяет впоследствии удалить запущенный setInterval c помощью clearInterval().

Синтаксис

var intervalID = scope.setInterval(func, delay[, param1, param2, ...]);

var intervalID = scope.setInterval(code, delay);

Параметры

funct

функция, которая будет вызываться каждые delay миллисекунд. Ожидается, что функция не принимает параметры и ничего не возвращает.

code

Этот необязательный синтаксис позволяет вам включать строку вместо функции, которая компилируется и выполняется каждые delay миллисекунд. Однако такая форма не рекомендуется по тем же причинам, которые делают eval() угрозой безопасности.

delay

Время в миллисекундах (одна тысячная секунды), на которое таймер выполнит задержку между вызовом функции. Если задано значение меньше 10, то будет использовано число 10.

param1, ..., paramN

Дополнительные параметры, передаваемые в функцию func.

Возвращаемое значение

Возвращаемый intervalID представляет из себя ненулевое числовое значение, которое идентифицирует таймер, созданный вызовом setInterval(); Это значение может быть передано в clearInterval() для отмены таймера.

**parseInt()**

Функция parseInt() принимает строку в качестве аргумента и возвращает целое число в соответствии с указанным основанием системы счисления.

Синтаксис

parseInt(string, radix);

Параметры

string

Значение, которое необходимо проинтерпретировать. Если значение параметра string не принадлежит строковому типу, оно преобразуется в него (с помощью абстрактной операции ToString). Пробелы в начале строки не учитываются.

radix

Целое число в диапазоне между 2 и 36, представляющее собой основание системы счисления числовой строки string, описанной выше. В основном пользователи используют десятичную систему счисления и указывают 10. Всегда указывайте этот параметр, чтобы исключить ошибки считывания и гарантировать корректность исполнения и предсказуемость результата. Когда основание системы счисления не указано, разные реализации могут возвращать разные результаты.

Возвращаемое значение

Целое число, полученное парсингом (разбором и интерпретацией) переданной строки. Если первый символ не получилось сконвертировать в число, то возвращается NaN.

**EventTarget.addEventListener()**

Метод EventTarget.addEventListener() регистрирует определённый обработчик события, вызванного на EventTarget.

EventTarget может быть Element, Document, Window, или любым другим объектом, поддерживающим события (таким как XMLHttpRequest).

Синтаксис

target.addEventListener(type, listener[, options]);

target.addEventListener(type, listener[, useCapture]);

Параметры

type

Чувствительная к регистру строка, представляющая тип обрабатываемого события.

Listener

Объект, который принимает уведомление, когда событие указанного типа произошло. Это должен быть объект, реализующий интерфейс EventListener или просто функция JavaScript.

options Необязательный

Объект options, который определяет характеристики объекта, прослушивающего событие. Доступны следующие варианты:

capture: Boolean указывает, что события этого типа будут отправлены зарегистрированному обработчику listener перед отправкой на EventTarget, расположенный ниже в дереве DOM.

once: Boolean указывает, что обработчик должен быть вызван не более одного раза после добавления. Если true, обработчик автоматически удаляется при вызове.

passive: Boolean указывает, что обработчик никогда не вызовет preventDefault(). Если всё же вызов будет произведён, браузер должен игнорировать его и генерировать консольное предупреждение. Пример Улучшение производительности прокрутки с помощью passive true

useCapture Необязательный

Если равно true, useCapture указывает, что пользователь желает начать захват. После инициализации захвата все события указанного типа будут отправлены в зарегистрированный listener перед отправкой в какой-либо EventTarget под ним в дереве DOM. События, восходящие вверх по дереву, не будут вызывать обработчиков, которым назначено использовать захват. Смотрите DOM Level 3 Events для более детального объяснения. Значение useCapture по умолчанию равно false.

Возвращаемое значение

Undefined

**GlobalEventHandlers.onclick**

Свойство onclick возвращает обработчик события click на текущем элементе.

Синтаксис

элемент.onclick = functionRef;

где functionRef это функция - зачастую это имя функции, которая объявлена где-то в другом месте или же функциональное выражение. См. "JavaScript Guide:Functions".

Единственный аргумент, переданный в определённую функцию обработчик события это MouseEvent объект. this внутри обработчика будет указывать на элемент, на котором было вызвано событие.

**Math.random()**

Статический метод Math.random() возвращает псевдослучайное число с плавающей запятой, которое больше или равно нулю и меньше единицы с приблизительно равномерным распределением в этом диапазоне. В дальнейшем это число можно «отмасштабировать», привести к нужному диапазону. Выбор начального числа для алгоритма генерации случайных чисел происходит автоматически (зависит от реализации) и не может быть изменён пользователем.

Синтаксис

Math.random()

Параметры

Нет.

Возвращаемое значение

Псевдослучайное число с плавающей запятой от 0 (включительно) до 1 (не включая).

**Node.appendChild**

Аннотация

Node.appendChild() добавляет узел в конец списка дочерних элементов указанного родительского узла. Если данный дочерний элемент является ссылкой на существующий узел в документе, то функция appendChild() перемещает его из текущей позиции в новую позицию (нет необходимости удалять узел из родительского узла перед добавлением его к какому-либо другому узлу).

Это означает, что узел не может находиться в двух точках документа одновременно. Поэтому, если у узла уже есть родитель, он сначала удаляется, а затем добавляется в новую позицию. Node.cloneNode() можно использовать для создания копии узла перед добавлением его в новый родительский элемент. Обратите внимание, что копии, сделанные с помощью cloneNode , не будут автоматически синхронизироваться.

Если данный дочерний элемент является DocumentFragment, то все содержимое DocumentFragment перемещается в дочерний список указанного родительского узла.

Синтаксис

var child = element.appendChild(child);

element родительский элемент.

child это элемент вставляется в конец element.

Возвращаемое значение

Возвращается дочерний элемент (aChild), кроме тех случаев, когда child это DocumentFragment, в таком случае возвращается пустой DocumentFragment.

**Перехватываемые браузерные события были описаны ранее.**

**Описание объекта Event в виде конспекта**

**Event**

Интерфейс Event представляет собой любое событие, которое происходит в DOM; некоторые из них генерируемые пользователем (клик мышью или нажатие клавиши на клавиатуре), а некоторые - генерируемые API (события, обозначающие завершение процесса анимации, приостановка видео и т.д.). Существует много типов событий, некоторые из них используют интерфейсы, базирующиеся на главном интерфейсе Event. Event содержит общие свойства и методы для всех событий.

Конструктор

Event()

Создаёт объект Event и возвращает его вызывающему.

Свойства

Event.bubbles Только для чтения

Логическое значение, указывающее, всплыло ли событие вверх по DOM или нет.

Event.cancelBubble

Историческое название синонима Event.stopPropagation(). Если установить значение в true до возврата из обработчика события (Event Handler), то событие не будет распространяться дальше (например, на обработчики для родительских узлов).

Event.cancelable Только для чтения

Логическое значение, показывающее возможность отмены события.

Event.composed Только для чтения

Логическое значение, показывающее может или нет событие всплывать через границы между shadow DOM (внутренний DOM конкретного элемента) и обычного DOM документа.

Event.currentTarget Только для чтения

Ссылка на текущий зарегистрированный объект, на котором обрабатывается событие. Это объект, которому планируется отправка события; поведение можно изменить с использованием перенаправления (retargeting).

Event.deepPath Non-standard

Массив DOM-узлов, через которые всплывало событие.

Event.defaultPrevented Только для чтения

Показывает, была ли для события вызвана функция event.preventDefault().

Event.eventPhase Только для чтения

Указывает фазу процесса обработки события.

Event.explicitOriginalTarget Non-standard Только для чтения

Явный первоначальный целевой объект события (Mozilla-специфичный). Не может содержать анонимного контента.

Event.originalTarget Non-standard Только для чтения

Первоначальный целевой объект события до перенаправлений (Mozilla-специфичный). Может быть из анонимного контента.

Event.returnValue Non-standard

Нестандартная альтернатива (оставшаяся от старых версий Microsoft Internet Explorer) для Event.preventDefault() и Event.defaultPrevented.

Event.scoped Только для чтения

Логическое значение, показывающее всплывает ли данное событие через shadow root (внутренний DOM-элемента). Это свойство было переименовано в composed.

Event.srcElement Non-standard

Нестандартный синоним (остался от старых версий Microsoft Internet Explorer) для Event.target.

Event.target Только для чтения

Ссылка на целевой объект, на котором произошло событие.

Event.timeStamp Только для чтения

Время, когда событие было создано (в миллисекундах). По спецификации это время от начала Эпохи (Unix Epoch), но в действительности в разных браузерах определяется по-разному; кроме того, ведётся работа по изменению его на DOMHighResTimeStamp тип.

Event.type Только для чтения

Название события (без учёта регистра символов).

Event.isTrusted Только для чтения

Показывает было или нет событие инициировано браузером (например, по клику мышью) или из скрипта (например, через функцию создания события, такую как event.initEvent)

Методы

Event.createEvent()

Создаёт новое событие, которое затем должно быть проинициализировано вызовом его метода initEvent().

Event.initEvent() Устарело

Инициализация значений созданного с помощью Document.createEvent() события. Если событие уже отправлено, то эта функция ничего не делает.

Event.preventBubble() Non-standard

Предотвращает всплытие события. Устаревшая, используйте вместо неё event.stopPropagation.

Event.preventCapture() Non-standard

Устаревшая, используйте вместо неё event.stopPropagation.

Event.preventDefault()

Отменяет событие (если его возможно отменить).

Event.stopImmediatePropagation()

Для конкретного события не будет больше вызвано обработчиков. Ни тех, которые привязаны к этому же элементу (на котором работает обработчик, который вызывает этот Event.stopImmediatePropagation()), ни других, которые могли бы вызваться при распространении события позже (например, в фазе перехвата - capture).

Event.stopPropagation()

Остановка распространения события далее по DOM.

Event.getPreventDefault() Non-standard

Нестандартная. Возвращает значение Event.defaultPrevented. Используйте вместо неё Event.defaultPrevented.

Вывод:  
Изучил принципы разработки динамических web-страниц с использованием клиентских сценариев JavaScript и объектной модели документа. Научился создавать кроссбраузерные html-страницы. Познакомился с уровнями и особенностями DOM API и обработкой событий DOM различных уровней.