Лекция "Введение в JavaScript"

Что такое JavaScript?

Определение:
JavaScript — это язык программирования, предназначенный для создания интерактивности на веб-страницах.

JavaScript — это язык программирования, который помогает сделать веб-страницы интерактивными. Представьте себе сайт, который выглядит красиво, но не реагирует на ваши действия — например, при нажатии на кнопку ничего не происходит, а формы для ввода данных работают только как текстовые поля. Именно для того, чтобы сайты «оживали», и был создан JavaScript. Он позволяет добавлять интерактивность: всплывающие окна, меню, которые разворачиваются при наведении, автоматическую проверку формы перед отправкой.

Этот язык работает прямо в вашем браузере, поэтому для его использования вам не нужно ничего устанавливать — все современные браузеры, такие как Google Chrome, Firefox, Safari или Edge, уже поддерживают его. Когда вы открываете сайт, который использует JavaScript, ваш браузер читает код и выполняет его. Например, если на сайте есть слайдер с фотографиями, JavaScript управляет тем, как эти фотографии сменяют друг друга.

JavaScript появился в 1995 году, и его изначальной задачей было выполнение простых действий на веб-страницах. Однако со временем язык стал настолько мощным, что теперь с его помощью можно создавать сложные веб-приложения, мобильные приложения и даже управлять серверной частью сайтов. Благодаря JavaScript интернет стал таким, каким мы привыкли его видеть: интерактивным, удобным и гибким. Если бы не JavaScript, многие сайты оставались бы статичными, напоминая цифровые версии книг или плакатов, которые нельзя изменить или взаимодействовать с ними.

Сегодня JavaScript считается одним из самых популярных и востребованных языков программирования в мире. Его знают и используют миллионы разработчиков, а благодаря простоте освоения начать учить его может даже человек без опыта в программировании.

Ключевые особенности:

- Позволяет оживлять веб-страницы, добавляя кнопки, формы, анимации.
- Работает в браузере, что делает его доступным на большинстве устройств.

- Без JavaScript веб-страницы были бы статичными.

Зачем учить JavaScript?

JavaScript стоит изучать, потому что он является неотъемлемой частью современной веб-разработки и делает сайты не просто функциональными, но и удобными для пользователей. Представьте себе сайт без интерактивности: кнопки не реагируют на нажатия, формы не проверяют данные перед отправкой, галереи не пролистываются, а меню не разворачиваются. Всё это работает благодаря JavaScript. Этот язык превращает статичные страницы в живые и динамичные приложения, которые мы используем каждый день.

Его значение трудно переоценить. JavaScript используется везде, где требуется взаимодействие с пользователем через браузер. Если вы когда-либо заполняли форму на сайте, общались через онлайн-чат или искали что-то на интерактивной карте, скорее всего, за этим стоял JavaScript. А ещё он даёт возможность создавать полноценные серверные приложения благодаря технологиям вроде Node.js, что делает его универсальным инструментом для разработки.

Ещё один важный момент — это востребованность JavaScript-разработчиков. Почти каждая компания, у которой есть сайт или приложение, нуждается в специалистах, способных сделать их продукт современным и удобным (все же хотят получать 100500 денег в секунду 😁). А так как JavaScript работает практически во всех браузерах и поддерживается всеми устройствами, знание этого языка значительно расширяет карьерные перспективы. К тому же, его использование не ограничивается только веб-разработкой: например, JavaScript применяется для создания мобильных приложений, разработки игр и даже работы с искусственным интеллектом.

Изучив JavaScript, вы освоите инструмент, который позволяет не только творить, но и быть востребованным на рынке труда. Это язык, который продолжает развиваться вместе с технологиями и остаётся актуальным.

Как работает JavaScript?

Когда вы открываете веб-страницу в браузере, браузер загружает не только HTML и CSS, но и код JavaScript. Этот **код выполняется непосредственно в вашем браузере**, и его основная цель — сделать страницу интерактивной. Представьте, что без JavaScript веб-страницы были бы просто статичными изображениями, которые не реагируют на ваши

действия 🤨. Например, без JavaScript на сайте не работали бы кнопки, которые меняют цвет при наведении, формы, которые проверяют правильность введённых данных, или динамичные анимации.

JavaScript позволяет странице реагировать на действия пользователя, такие как клики на кнопки, ввод текста в поля или прокручивание страницы. Например, при нажатии на кнопку с помощью JavaScript можно изменить текст на странице, показать или скрыть дополнительные элементы, а также выполнить другие действия без перезагрузки всей страницы. Это помогает сделать веб-сайт более удобным и современным, потому что всё происходит быстро и без необходимости загружать страницу снова.

Одним из примеров работы JavaScript является ситуация, когда вы нажимаете кнопку на странице, и текст изменяется сразу, без обновления страницы. Этот процесс скрывает от вас все технические детали, и вы просто видите результат. Важно отметить, что JavaScript работает "в фоне" и не мешает пользователю взаимодействовать с остальной частью страницы. Он выполняет задачи асинхронно, то есть задачи могут быть выполнены в фоновом режиме, не останавливая работу других частей страницы.

Пример простого кода

Простой пример кода, который мы рассмотрим, выглядит так:

```
alert("Привет, мир!");
```

Этот код делает несложную, но важную вещь — он показывает пользователю всплывающее окно с текстом «Привет, мир!». Это очень простой способ для начинающего разработчика увидеть результат работы JavaScript в действии. Когда браузер видит этот код, он не выполняет никаких сложных операций или вычислений. Вместо этого он просто выводит на экран окошко с сообщением. Это может показаться элементарным, но на самом деле такие простые элементы взаимодействия с пользователем лежат в основе множества динамических и интерактивных функций на современных сайтах.

Всплывающие окна, которые создаются с помощью функции `alert()`, — это базовый способ показать информацию пользователю. Вы могли бы использовать их, например, чтобы предупредить о каком-то важном событии или напомнить о какой-то информации. Эта функция работает мгновенно, и всё, что она делает — это выводит сообщение и ожидает, пока пользователь закроет окно. Это простое поведение, но оно демонстрирует, как JavaScript может взаимодействовать с пользователем через браузер.

В будущем, когда мы будем писать более сложные программы, принцип работы с такими окнами будет полезен для создания различных типов взаимодействий с пользователями, например, уведомлений или диалоговых окон (хотя есть более удобные для этого вещи, которые мы рассмотрим ближе к концу курса).

Переменные в JavaScript

Переменные в JavaScript — это как контейнеры, в которых хранятся данные. Данные могут быть самыми разными: числа, строки, логические значения или даже более сложные объекты. Например, если вам нужно сохранить имя пользователя, то вы создаёте переменную, которая будет хранить это имя. Переменные позволяют вам работать с данными в программе, изменять их, выполнять операции и принимать решения в зависимости от значений, которые они содержат.

Когда вы создаёте переменную в JavaScript, вы используете специальные ключевые слова, такие как `let`, `const` или `var`. Эти ключевые слова помогают компилятору понять, как именно должна работать переменная.

`let` — это ключевое слово, которое используется для создания переменной, значение которой можно изменить в процессе работы программы. Это наиболее современный способ создания переменной, и его рекомендуется использовать в большинстве случаев. Например, если вы создаёте переменную для хранения имени пользователя и хотите, чтобы оно могло изменяться в дальнейшем, вы можете сделать это с помощью `let`. Если вы хотите изменить имя, например, в ответ на действия пользователя, переменная, объявленная с `let`, будет как раз тем, что вам нужно.

```
let name = "Максим";
name = "Екатерина";  // Имя можно изменить
```

Кроме `let`, в JavaScript есть ещё одно ключевое слово для создания переменных — **`const`**. Переменная, созданная с помощью `const`, не может быть изменена после того, как ей присвоено значение. Это удобно, когда вы хотите, чтобы значение переменной оставалось постоянным в течение всей программы, например, когда храните какие-то константы или неизменяемые данные.

```
const pi = 3.14;
// pi = 3.1415; // Ошибка: значение константной переменной нельзя изменить
```

Наконец, есть старый способ создания переменных с помощью **`var`**. Он используется в

более старых версиях JavaScript и отличается тем, что переменные, созданные с помощью `var`, имеют свою область видимости в рамках всей функции, а не только блока кода. Из-за этого `var` иногда ведёт себя непредсказуемо, особенно в больших программах, и его стараются избегать в современном коде (просто прочитайте что такое есть и можно забыть))).

```
var age = 25;
age = 26;  // Значение можно изменять, но есть особенности с областью видимости
```

В современных проектах обычно используют `let` и `const`, так как они делают код более понятным и безопасным. `let` полезен для переменных, значения которых могут изменяться, а `const` — для значений, которые не должны изменяться после присвоения.

Переменная — это как коробка, в которой мы можем хранить информацию. В программировании она позволяет нам сохранять разные значения, чтобы потом их использовать. Например, в коде ниже мы создаём переменную с именем `name`, в которую записываем текст "Максим". Этот текст можно потом показать пользователю на экране. В JavaScript для создания переменных мы обычно используем команду `let`. Команда `alert` вызывает всплывающее окно, в котором мы и увидим значение нашей переменной.

Пример кода:

```
let name = "Максим";
alert(name);
```

Что происходит в этом примере? Переменная `name` хранит значение "Максим". Когда программа доходит до команды `alert(name)`, она вызывает всплывающее окно, в котором появляется текст из переменной — "Максим". Это очень простой пример, но он показывает, как мы можем использовать переменные для хранения и отображения данных. В реальных приложениях переменные могут хранить гораздо более сложную информацию, но сама идея остаётся той же.

Пример с переменной

Пример:

```
let name = "Максим";
alert(name);
```

Результат:

- Всплывающее окно с текстом "Максим".

Основные типы данных в JavaScript

В JavaScript есть несколько типов данных, которые используются для хранения информации. Понимание этих типов важно, поскольку каждый из них имеет свои особенности и способы использования. Давайте разберемся в каждом из них более подробно, с примерами.

Строки

Строки в JavaScript — это текстовые данные. Это может быть как одно слово, так и целое предложение. Строки всегда заключаются в кавычки. В JavaScript можно использовать как одинарные ('), так и двойные (") кавычки, чтобы обозначить строку. Иногда также используются шаблонные строки, которые заключаются в обратные кавычки (`), и они позволяют вставлять переменные и выражения прямо в строку.

Пример строки:

```
let greeting = "Привет, мир!";
let name = 'Максим';
let sentence = `Меня зовут ${name}`;
```

Здесь `greeting` — это строка, которая содержит текст "Привет, мир!". Переменная `name` также является строкой. В строке `sentence` используется шаблонная строка, где через `${name}` вставляется значение переменной `name` .

Строки могут быть разных длин и содержать пробелы, символы, буквы, цифры и даже специальные символы, такие как новая строка или табуляция. Важно помнить, что строки в JavaScript неизменяемы. Это означает, что, если мы изменяем строку, фактически создается новая строка, а не изменяется старая.

Числа

Числа в JavaScript могут быть как целыми (целые числа), так и дробными (с плавающей точкой). Числа не заключаются в кавычки, как строки, и могут быть записаны как обычные цифры, например 10 или 3.14, либо в экспоненциальной записи.

Пример целого числа:

```
let age = 30;
```

Здесь `age` — это целое число 30.

Пример числа с плавающей точкой:

```
let price = 19.99;
```

Здесь `price` — это число с плавающей точкой, представляющее цену товара.

Числа могут выполнять различные математические операции, такие как сложение, вычитание, умножение, деление. Важно помнить, что JavaScript имеет ограничения по точности для чисел с плавающей точкой, что может привести к небольшим погрешностям при расчетах.

Булевы значения

Булевы значения — это логические значения, которые могут быть только два: `true` (истина) и `false` (ложь). Этот тип данных используется для принятия решений в программах. Например, в условных операторах, таких как `if` , или в циклах, таких как `while` , булевы значения позволяют решать, выполнить ли какой-то блок кода.

Пример булевого значения:

```
let isRaining = true;
let isSunny = false;
```

Здесь переменная `isRaining` хранит значение `true` , что может означать, что на улице идет дождь. Переменная `isSunny` хранит значение `false` , что означает, что на улице не солнечно.

Булевы значения часто используются в условиях. Например, можно проверить, является ли число положительным, с помощью булевой проверки:

```
let number = 5;
if (number > 0) {
    console.log("Число положительное");
}
```

В данном примере выражение `number > 0` возвращает `true`, если число больше нуля, и выполняет код внутри блока.

Массивы

Массивы — это структуры данных, которые позволяют хранить несколько значений в одной переменной. Эти значения могут быть разного типа: числа, строки, объекты и даже другие массивы (так называемые "многомерные массивы"). Массивы обычно используются, когда нужно работать с набором данных, например, с коллекцией элементов, такими как список пользователей или список товаров. Массивы в JavaScript могут содержать одновременно элементы разных типов. Это дает гибкость при разработке, но также требует внимательности при манипуляциях с такими данными.

Массивы обозначаются квадратными скобками (`[]`), а элементы внутри массива разделяются запятыми.

Пример массива:

```
let numbers = [1, 2, 3, 4, 5];
let fruits = ["яблоко", "банан", "апельсин"];
```

В примере `numbers` — это массив чисел, а `fruits` — массив строк, представляющих названия фруктов. Каждый элемент массива можно получить по индексу (порядковому номеру), начиная с нуля. Например, чтобы получить первое число из массива `numbers`, нужно написать:

```
let firstNumber = numbers[0]; // 1
```

Массивы могут быть изменяемыми. Это значит, что вы можете добавить, удалить или изменить элементы в массиве после его создания.

Объекты

Объекты — это более сложные структуры данных, которые позволяют хранить данные в виде пар "ключ-значение". Ключи всегда являются строками, а значениями могут быть данные любого типа: строки, числа, массивы, другие объекты и т.д. Объекты широко используются для представления различных сущностей, таких как пользователи, товары, автомобили и т.д.

Объект обозначается фигурными скобками (`{}`), внутри которых находятся пары "ключ-значение", разделенные двоеточием. Каждая пара разделяется запятой.

Пример объекта:

```
let person = {
    name: "Максим",
    age: 22,
    isStudent: true
};
```

Здесь `person` — это объект, который описывает человека. У него есть три свойства: `name` (имя), `age` (возраст) и `isStudent` (студент ли он). Чтобы получить значение какого-либо свойства объекта, нужно обратиться к нему через точку:

```
let name = person.name; // "Максим"
let age = person.age; // 22
```

Объекты могут содержать не только простые данные, но и функции, которые называются методами. Например, можно добавить метод, который будет выводить информацию о человеке (но про это мы поговорим позднее):

```
let person = {
    name: "Максим",
    age: 22,
    greet: function() {
        console.log("Привет, меня зовут " + this.name);
    }
};
person.greet(); // "Привет, меня зовут Максим"
```

Метод `greet` выводит строку, обращаясь к свойству `name` объекта через ключевое слово `this`.

Операции с переменными в JavaScript

Операции с переменными — это фундаментальная часть программирования, позволяющая выполнять различные вычисления, манипулировать данными и управлять логикой программы. В JavaScript переменные могут быть использованы для хранения данных, а операции с ними помогают изменять или использовать эти данные в различных контекстах.

Переменные в JavaScript могут хранить различные типы данных: числа, строки, объекты, массивы и так далее. Операции с переменными могут быть арифметическими,

строковыми, логическими и т. д. Давайте рассмотрим основные виды операций, которые можно выполнять с переменными.

Арифметические операции

Арифметические операции включают в себя стандартные математические действия, такие как сложение, вычитание, умножение, деление и так далее. Рассмотрим примеры:

- **Сложение (+)** — складывает два числа или соединяет строки.

  Пример:

  ```
  let a = 5;
  let b = 3;
  let result = a + b;  // Результат будет 8
  ```

  Также, если складывать строки, то они просто объединяются:

  ```
  let greeting = "Hello, ";
  let name = "world!";
  let message = greeting + name;  // Результат будет "Hello, world!"
  ```

- **Вычитание (-)** — вычитает одно число из другого.

  Пример:

  ```
  let a = 10;
  let b = 4;
  let result = a - b;  // Результат будет 6
  ```

- *Умножение ()** — умножает два числа.

  Пример:

  ```
  let a = 6;
  let b = 4;
  let result = a * b;  // Результат будет 24
  ```

- **Деление (/) и остаток от деления (%)** — делит одно число на другое и возвращает остаток от деления соответственно.

  Пример деления:

```
let a = 10;
let b = 2;
let result = a / b;  // Результат будет 5
```

Пример остатка:

```
let a = 10;
let b = 3;
let result = a % b;  // Результат будет 1
```

- **Инкремент (++) и декремент (--)** — увеличивает или уменьшает значение переменной на 1 соответственно.

  Пример инкремента:

```
let a = 5;
a++;  // Теперь a равно 6
```

  Пример декремента:

```
let a = 5;
a--;  // Теперь a равно 4
```

Операции с переменными строкового типа

Строки в JavaScript могут быть объединены (конкатенированы) с помощью оператора `+`. Это важный аспект работы с текстом.

Пример:

```
let firstName = "John";
let lastName = "Doe";
let fullName = firstName + " " + lastName;  // Результат: "John Doe"
```

Кроме того, можно использовать строковые шаблоны (template literals), которые позволяют удобно вставлять переменные в строки:

```
let firstName = "John";
let lastName = "Doe";
let fullName = `${firstName} ${lastName}`;  // Результат: "John Doe"
```

Логические операции

Логические операции используются для работы с булевыми значениями (`true` или `false`).

- **Оператор "И" (&&)** — возвращает `true` , если оба операнда истинны.

  Пример:

  ```
  let a = true;
  let b = false;
  let result = a && b;  // Результат будет false
  ```

- **Оператор "ИЛИ" (||)** — возвращает `true` , если хотя бы один операнд истинный.

  Пример:

  ```
  let a = true;
  let b = false;
  let result = a || b;  // Результат будет true
  ```

- **Оператор "НЕ" (!)** — инвертирует логическое значение.

  Пример:

  ```
  let a = true;
  let result = !a;  // Результат будет false
  ```

Операции с объектами

С объектами в JavaScript можно выполнять различные операции, но чаще всего это доступ к значениям через ключи или изменение этих значений. Например:

- Доступ к свойствам объекта:

  ```
  let person = {name: "John", age: 30};
  let name = person.name;  // Результат будет "John"
  ```

- Изменение свойств объекта:

  ```
  person.age = 31;  // Теперь age равно 31
  ```

- Добавление новых свойств:

```
person.city = "New York";  // Добавляется новое свойство city
```

Операции с массивами

Массивы в JavaScript — это тоже объекты, и с ними можно выполнять операции, такие как доступ к элементам по индексам, добавление новых элементов, удаление и так далее.

- Доступ к элементам массива:

```
let arr = [1, 2, 3];
let firstElement = arr[0];  // Результат будет 1
```

- Изменение элементов массива:

```
arr[1] = 10;  // Теперь второй элемент массива будет 10
```

- Добавление элемента в конец массива:

```
arr.push(4);  // Теперь массив [1, 2, 3, 4]
```

- Удаление элемента из конца массива:

```
arr.pop();  // Теперь массив [1, 2, 3]
```

Операции с переменными и типами данных

Очень важным аспектом работы с переменными в JavaScript является то, что тип данных переменной может влиять на результат операции. Например, если попытаться сложить строку с числом, JavaScript приведет число к строке и выполнит конкатенацию:

```
let result = "Возраст: " + 30;  // Результат будет "Возраст: 30"
```

Если же вы хотите выполнить математическое сложение, важно убедиться, что оба операнда имеют числовой тип данных:

```
let result = 10 + 30;  // Результат будет 40
```

Мем

картинку найдете сами в интернете :)

Мем, связанный с операциями над строкой и числом в JavaScript, основан на своеобразной и часто непредсказуемой работе языка с типами данных. В JavaScript тип данных переменной может динамически изменяться, и язык пытается автоматически преобразовать значения в подходящий тип в зависимости от контекста. Это называется **"автоматическое приведение типов"**. Иногда результаты таких операций выглядят забавно или неожиданно, что и породило множество мемов.

Один из самых известных примеров выглядит так:

```
1 + "1" // Результат: "11"
1 - "1" // Результат: 0
```

В первом случае происходит конкатенация строки `"1"` и числа `1`. JavaScript преобразует число в строку, так как операция `+` при работе с одним строковым операндом понимается как операция объединения строк.

Во втором случае строка `"1"` преобразуется в число, так как операция `-` применима только к числам, и результатом становится `0`.