

# Язык JavaScript (ОСНОВЫ)

A large, bold, black 'JS' logo is centered on the page. The 'J' and 'S' are stylized with thick strokes and rounded terminals, typical of the JavaScript branding.

## ВОЗМОЖНОСТИ JAVASCRIPT

---

1. Математические операции — калькуляторы, расчет формул и т. д.;
2. Получение и обработка данных HTML-форм;
3. Взаимодействие с HTML-элементами на странице;
4. Обработка клиентских событий;
5. Отправка запросов на сервер и загрузка данных без перезагрузки страницы (технология AJAX);
6. Добавление анимации и различных графических эффектов на веб-страницы;
7. Разработка 2D и 3D игр и тд.

# JAVASCRIPT (ОСНОВЫ)

---

JavaScript - независимый язык, со своей спецификацией, которая называется ECMAScript.

Программы, написанный на языке JavaScript называются скриптами, это файлы с расширением js. Они могут напрямую подключаться к html и выполняются, как только загружается страница.

**JavaScript - это кросс-платформенный, объектно-ориентированный, интерпретируемый язык со слабой динамической типизацией.**

**Динамическая типизация** - приём при котором переменная связывается с типом в момент присваивания значения, а не в момент объявления переменной.

**Интерпретация** – это когда исходный код программы получает другой инструмент, который называют «интерпретатор», и выполняет его «как есть». При этом распространяется именно сам исходный код (скрипт).

**Компиляция** – это когда исходный код программы, при помощи специального инструмента, другой программы, которая называется «компилятор», преобразуется в другой язык, как правило – в машинный код. Этот машинный код затем распространяется и запускается.

# ВЫПОЛНЕНИЕ JAVASCRIPT ПРОГРАММЫ

**Текст программы интерпретируется**

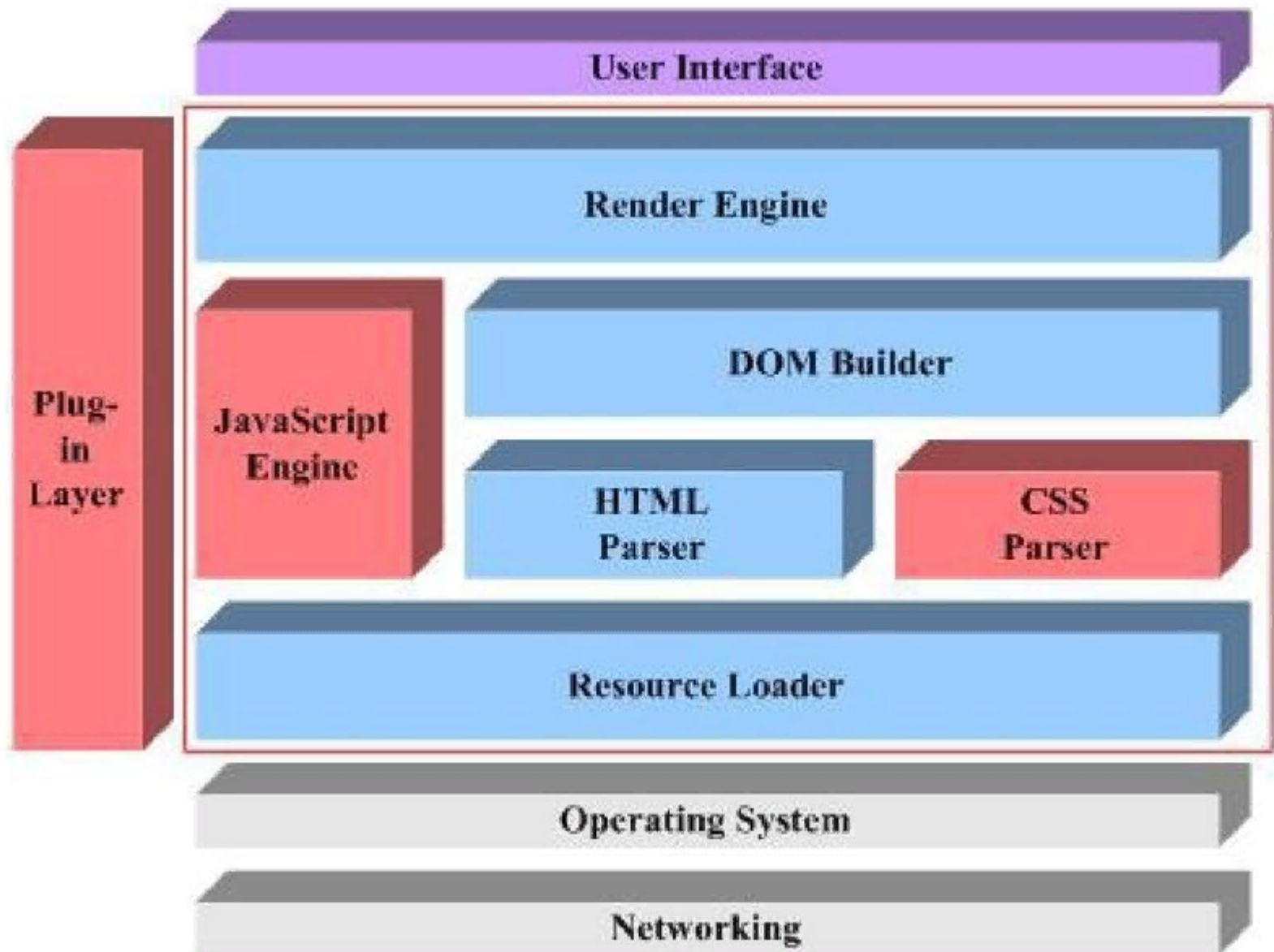
```
let age = document.getElementById('age');  
if (age < 18) {  
    alert("Доступ к сайту запрещен");  
} else {  
    alert("Добро пожаловать");  
}
```

**и выполняется.**

Современные интерпретаторы перед выполнением преобразуют JavaScript в машинный код или близко к нему, оптимизируют, а уже затем выполняют.

Интерпретатор JavaScript встроен в браузер.

# АРХИТЕКТУРА БРАУЗЕРА



**User Interface** - интерфейс пользователя обеспечивает стандартный набор функций (панель инструментов, вкладки, настройки, ввод информации, печать и т.д.)

**Render engine** – графический движок отображающий содержимое запрашиваемого ресурса. Его функция анализировать полученный HTML или XML, при этом учитывать CSS и JavaScript и создавать макет страницы который видит пользователь.

Ключевыми компонентами RE являются **HTML и CSS парсеры** – программные компоненты позволяющие отобразить страницу даже при наличии ошибок.

# АРХИТЕКТУРА БРАУЗЕРА

---

**Resource Loader** – компонент предоставляет функциональные возможности для обработки URL адресов и получении всех необходимых файлов, используя протоколы HTTP и FTP. Этот компонент осуществляет кэширование полученных данных.

**DOM Builder** – компонента которая создает объектную модель документа (дерево узлов) - интерфейс позволяющий программам и скриптам получить доступ к содержимому HTML, XML документов, а также изменять содержимое и оформление таких документов.

**JavaScript Engine** – отвечает за выполнение кода JavaScript. Результаты передает графическому движку для отображения содержимого документа



## ПОДКЛЮЧЕНИЕ JS СКРИПТА

Вариант 1 - использование на html странице

```
<script>
```

```
    alert("JS here");
```

```
</script>
```

Вариант 2 (более предпочтительный) -  
подключение внешних \*.js файлов

```
<script src="js/jsprogramm.js"></script>
```

## КОММЕНТАРИИ И ОТЛАДКА В JAVASCRIPT

---

Комментарии служат для заметок, не влияют на ход выполнения программы.

```
// однострочный комментарий  
/* многострочный комментарий */
```

Для вывода информации в консоль браузера используется

```
console.log("Сообщение");  
console.log(varName);
```

## ВЫПОЛНЕНИЕ СКРИПТА

Скрипт выполняется браузером сразу после загрузки.

```
<html>
<head></head>
<body>
  <p>Начало документа ...</p>
  <script>
    alert("Привет мир!!!");
    //document.write("<p>Привет мир!!!</p>");
  </script>
  <p>...Конец документа</p>
</body>
</html>
```

# ПЕРЕМЕННЫЕ В JAVASCRIPT

**Переменные** - поименованная выделенная область памяти.

JavaScript переменные являются "контейнерами" для хранения и извлечения информации.

Переменные состоят из оператора **var** / **let** / **const** и имени.

Требования к именам переменных:

1. Имя может состоять из: букв, цифр, символов \$ и \_
2. Первый символ не должен быть цифрой.
3. Имена переменных должны быть именами существительными и описывать сущность хранимого значения.
4. Регистр букв в именах переменных имеет значение.
5. Для имен переменных нельзя использовать зарезервированные слова (например, **let**, **function**, **return** и тд.).
6. Использование русских букв допустимо, но не рекомендуется

# ПЕРЕМЕННЫЕ В JAVASCRIPT

Объявление (создание)  
переменной:

**var имяПеременной;**  
либо **let имяПеременной;**

Например,  
**let name, login, age;**  
или **let name; let login; let age;**

Присвоить переменной  
значение:  
**имяПеременной = значение;**

Например,  
**name = "Евгений";**

Переопределение переменной:  
**имяПеременной = новое значение;**

Например,  
**name = "Григорий";**

Можно одновременно объявить  
переменную и присвоить значение:  
**var имяПеременной = значение;**

Например,  
**let login = "fistashka";**

После того, как переменной было  
присвоено значение, получить  
доступ к нему можно по этому  
имени.

# ТИПЫ ДАННЫХ В JAVASCRIPT

Число - number (Infinity, NaN)

```
var num = 8;
```

Строка - string

```
var str = 'Строка';
```

Логический тип - boolean

```
var boo = (true, false);
```

Значение неизвестно (ничего) - null

```
var unknownValue = null;
```

Значение не присвоено - undefined

```
var notSetValue;
```

Объект - object

```
var obj = { type: 'Object' };
```

Определение типа  
переменной:

```
typeof имяПеременной;
```

# ЧИСЛА В JAVASCRIPT

Числа могут быть целые и дробные, отрицательные и положительные. Например,

```
let num = -12;
```

```
let num2 = 13.89;
```

## NaN (Not-A-Number)

Если **математическая операция не может быть совершена**, то возвращается значение NaN. Значение NaN можно проверить функцией `isNaN(значение)`.

Функция `isNaN(значение)` возвращает (результатом работы функции является):

- **true** - если значение является NaN или не может быть преобразовано в число
- **false** - если значение число или может быть в него преобразовано

# МАТЕМАТИЧЕСКИЕ ОПЕРАТОРЫ JAVASCRIPT

---

+	сложение
-	вычитание
*	умножение
/	деление
%	взятие остатка от деления



# ЧИСЛА В JAVASCRIPT

Оператор сложения (+) используется для:

1. сложения чисел;

Например,

```
2 + 6; // 8
```

2. конкатенации (склеивание) строк; Если хотя бы один аргумент является строкой, то второй будет также преобразован к строке, после чего конкатенация слияние строк

Например,

```
console.log('2' + 6); // '26'
```

3. приведения значения к числу.

Например,

```
console.log(+ "67"); // 67
```

Остальные арифметические операторы работают только с числами и всегда приводят аргументы к числу.

## Деление на ноль (Infinity)

Результатом деления любого ненулевого числа на 0 будет Infinity (бесконечность).

Например,

```
console.log(5 / 0); // Infinity
```

Infinity больше любого числа.

Добавление к бесконечности не меняет её.

# ЧИСЛА В JAVASCRIPT

## Преобразование строки в число: `parseInt` и `parseFloat`.

Функции `parseInt` и `parseFloat` преобразуют строку символ за символом, пока это возможно. При возникновении ошибки возвращается число, которое получилось.

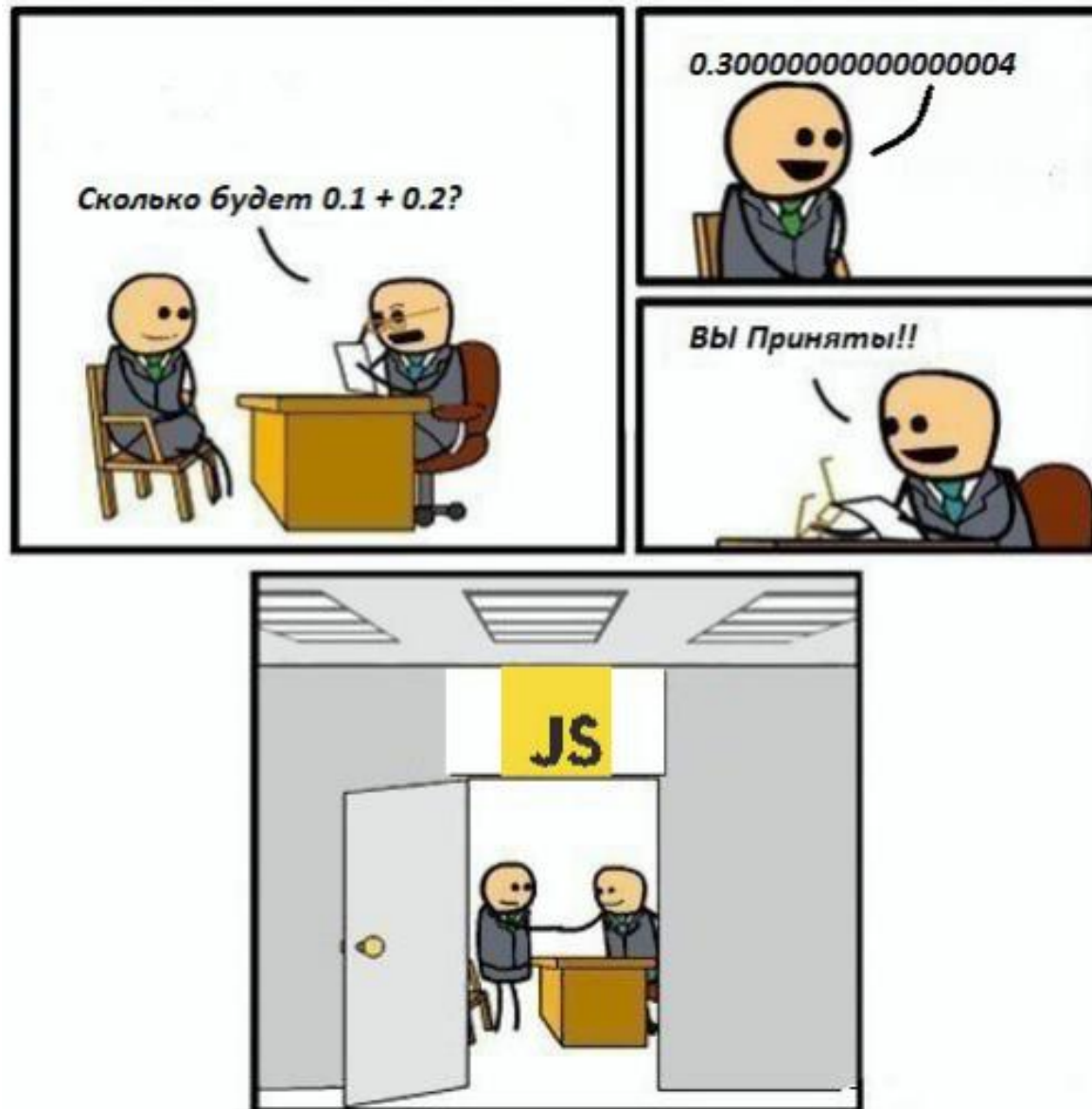
Функция **`parseInt`** читает из строки целое число, а **`parseFloat`** – дробное. Например,

```
parseInt('12.22'); // 12
parseInt('12em'); // 12
parseInt('12.22em'); // 12
parseInt('s12'); // NaN
```

```
parseFloat('12.22'); // 12.22
parseFloat('12em'); // 12
parseFloat('12.22em'); // 12.22
parseFloat('s12'); // NaN
```

Функция `parseInt` также позволяет указать систему счисления. Например, `parseInt("FXX123", 16); // 15`

# ЧИСЛА В JAVASCRIPT



# БУЛЕВЫЙ ТИП ДАННЫХ В JS

---

Логический (булевый тип данных) может принимать только два значения:

1. **true** (истина)
2. **false** (ложь).

Преобразование к true/false происходит в логическом контексте и при применении логических операторов.

При этом к **false** преобразуются:

- **null, undefined;**
- **пустые строки;**
- **0, NaN.**

Все остальные значения преобразуются к **true**.

# ОПЕРАТОРЫ СРАВНЕНИЯ В JAVASCRIPT

>	больше
<	меньше
==	равно (с приведением типов)
>=	больше или равно
<=	меньше или равно
!=	не равно
===	строгое равенство (без приведения типов)
!==	строгое неравенство

# ОПЕРАТОРЫ СРАВНЕНИЯ В JAVASCRIPT

Операторы сравнения **возвращают** значение логического типа:  
**либо true, либо false.**

При сравнении значений разных типов происходит приведение к числу, за исключением строгого равенства === (!==).

При сравнении с использованием строгого равенства приведение типов не происходит, значения сравниваются на полное совпадение.

Например,

```
console.log('3' == 3); // true
console.log('3' === 3); // false
console.log('3' === '3'); // true
console.log(3 === 3); // true
```

## ИНКРЕМЕНТ И ДЕКРЕМЕНТ В JAVASCRIPT

i++	инкремент (постфиксная форма) - увеличивает, и возвращает старое значение
++i	инкремент (префиксная форма) - сначала увеличивает, а потом возвращает значение
i--	декремент (постфиксная форма) - уменьшает, и возвращает старое значение
--i	декремент (префиксная форма) - сначала уменьшает, а потом возвращает значение



# ТЕРНАРНЫЙ ОПЕРАТОР

(условие) ? значение1 : значение2;

произойдёт проверка условия, если условие true, оператор вернет значение1, если false - значение2

Например,

```
let res;
```

```
let height = 23;
```

```
let width = '65px';
```

```
res = (height >= parseInt(width)) ? "высота больше или равна" :  
"ширина больше";
```

```
console.log(res);
```

## ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ: ALERT, PROMPT, CONFIRM

`alert(сообщение);`

выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «ОК».

Например,

```
alert("Привет мир");
```

`result = prompt(title, default);`

выводит модальное окно с заголовком `title`, полем для ввода текста, заполненным строкой по умолчанию `default` и кнопками ОК/CANCEL. Вызов `prompt` возвращает то, что ввёл посетитель – строку или специальное значение `null`, если ввод отменён.

Например,

```
var years = prompt('Сколько вам лет?', 100);
```

```
alert('Вам ' + years + ' лет!');
```

## ВЗАИМОДЕЙСТВИЕ С ПОЛЬЗОВАТЕЛЕМ: ALERT, PROMPT, CONFIRM

---

`result = confirm(question);`

`confirm` выводит окно с вопросом `question` с двумя кнопками: ОК и CANCEL. Результатом будет `true` при нажатии ОК и `false` – при CANCEL(Esc).

Например,

```
var isAdmin = confirm("Вы - администратор?");  
alert( isAdmin );
```

## ЧТО ПОЧИТАТЬ ДОПОЛНИТЕЛЬНО

<https://learn.javascript.ru/intro> - Введение в JavaScript

<https://learn.javascript.ru/hello-world> - Подключение скриптов

<https://learn.javascript.ru/structure> - Структура кода

<https://learn.javascript.ru/variables> - Переменные

<https://learn.javascript.ru/types> - Типы данных

<https://learn.javascript.ru/type-conversions> - Преобразования

ТИПОВ

<https://learn.javascript.ru/operators> - Операторы

<https://learn.javascript.ru/comparison> - Операторы сравнения

<https://learn.javascript.ru/uibasic> - Взаимодействие с

пользователем: alert, prompt, confirm

<https://learn.javascript.ru/ifelse#uslovnyy-operator> - Условный

оператор „?“