JavaScript курс

Введение в мир JS

Dynamic HTML или DHTML — это способ (подход) создания интерактивного веб-сайта, использующий сочетание статичного языка разметки HTML, встраиваемого (и выполняемого на стороне клиента) скриптового языка JavaScript, CSS (каскадных таблиц стилей) и DOM (объектной модели документа). Он может быть использован для создания приложения в веб-браузере: например для более простой навигации или для придания интерактивности форм. DHTML может быть использован для динамического перетаскивания элементов по экрану. Также он может служить как инструмент для создания основанных на браузере видеоигр. DHTML приложения, которые вполне автономны в браузере, без серверной поддержки, такой как база данных, иногда вынуждены обращаться к Single Page Applications, или SPA.

DHTML = HTML + CSS + JavaScript

- **HTML** - язык разметки, он определяет положение контента на странице, используя различные теги (заголовки, параграфы,листы)
- **CSS** - каскадные таблицы стилей. Определяет правила и стили оформления элементов HTML документа (Шрифты, заливку, позицию, отступы, видимость, переходы)
- **Javascript** - определяет динамическое поведение на странице, позволяет интерактивно реагировать на действия пользователя в документе:
    - Валидация форм
    - Изменение структуры HTML элементов на странице
    - Выполнение сложных вычислений
    - Обмен информацией с удаленным сервером, без перезагрузки страницы
    - Графика В настоящее время активно используется не только на стороне клиента но и на нем часто строится серверная часть приложений.

История возникновения JS

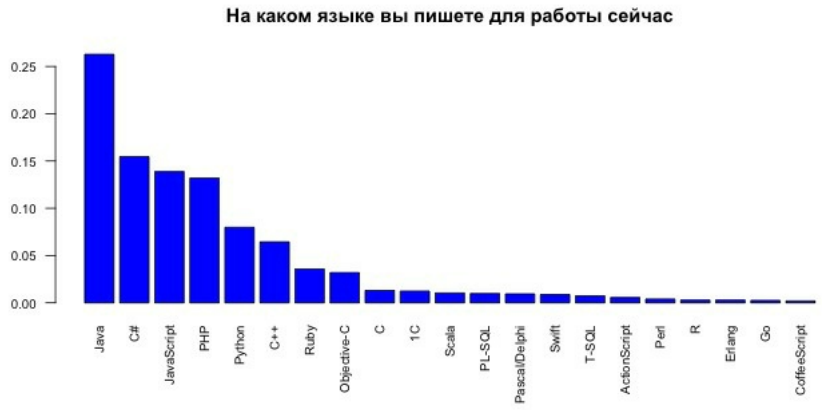
В 1992 году компания Nombas начала разработку встраиваемого скриптового языка Cmm (Си-минус-минус), который, по замыслу разработчиков, должен был стать достаточно мощным, чтобы заменить макросы, сохраняя при этом схожесть с Си, чтобы разработчикам не составляло труда изучить его. Главным отличием от Си была работа с памятью. **В новом языке всё управление памятью осуществлялось автоматически**: не было необходимости создавать буфера, объявлять переменные, осуществлять преобразование типов. В остальном языки сильно походили друг на друга: в частности, Cmm поддерживал стандартные функции и операторы Си. **В конце ноября 1995** года Nombas разработала версию CEnvi, внедряемую в веб-страницы. Страницы, которые можно было изменять с помощью скриптового языка, получили название Espresso Pages — они демонстрировали использование скриптового языка для создания игры, проверки пользовательского ввода в формы и создания анимации. Espresso Pages позиционировались как демоверсия, призванная помочь представить, что случится, если в браузер будет внедрён язык Cmm. Работали они только в 16-битовом Netscape Navigator под управлением Windows

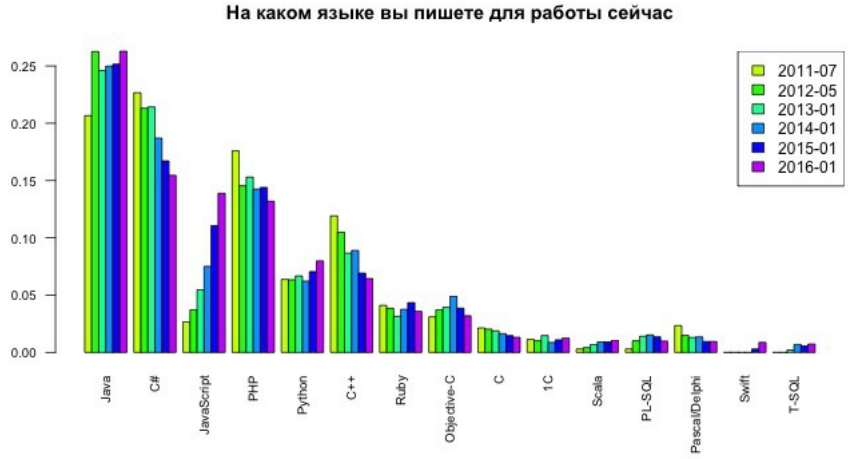
Перед Бренданом Эйхом, нанятым в компанию Netscape **4 апреля 1995** года, была поставлена задача внедрить язык программирования Scheme или что-то похожее в браузер Netscape. Помимо Брендана Эйха в разработке участвовали сооснователь Netscape Communications Марк Андрессен и сооснователь Sun Microsystems Билл Джой: чтобы успеть закончить работы над языком к релизу браузера, компании заключили соглашение о сотрудничестве в разработке. Они ставили перед собой цель обеспечить «язык для склеивания» составляющих частей веб-ресурса: изображений, плагинов, Java-апплетов, который был бы удобен для веб-дизайнеров и программистов, не обладающих высокой квалификацией.

Язык предназначался как для программирования на стороне клиента, так и для программирования на стороне сервера. На синтаксис оказали влияние языки Си и Jav, 4 декабря 1995 года LiveScript переименовали в JavaScript.

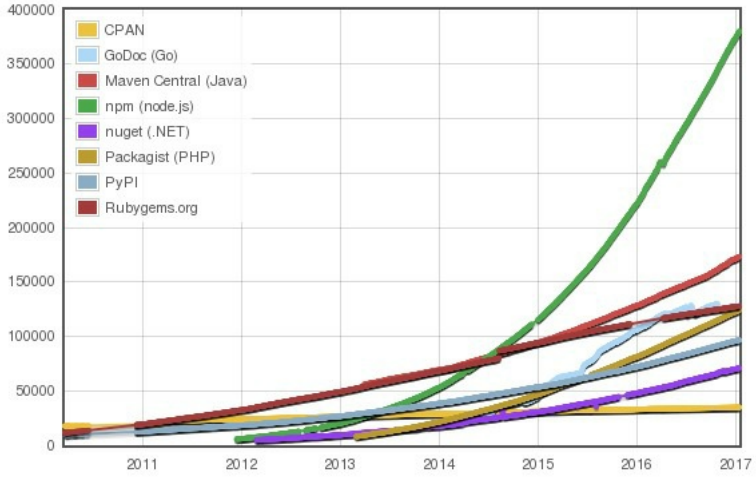
Анонс JavaScript со стороны представителей Netscape и Sun состоялся накануне выпуска второй бета-версии Netscape Navigator. В нём декларируется, что 28 лидирующих ИТ-компаний выразили намерение использовать в своих будущих продуктах JavaScript как объектный скриптовый язык с открытым стандартом.

Популярность





Согласно TIOBE Index, базирующемуся на данных поисковых систем Google, MSN, Yahoo!, Википедия и YouTube, в апреле 2015 года JavaScript находился на 6 месте (год назад на 9)[31]. По данным Black Duck Software (англ.)[32] в разработке открытого программного обеспечения доля использования JavaScript росла. 36 % проектов, релизы которых состоялись с августа 2008 по август 2009 гг., включают JavaScript, наиболее часто используемый язык программирования с быстрорастущей популярностью. 80 % открытого программного обеспечения использует Си, C++, Java, Shell и JavaScript. При этом JavaScript — единственный из этих языков, чья доля использования увеличилась (более чем на 2 процента, если считать в строках код



На конференции Node.js Interactive озвучены заслуживающие внимания достижения проекта NPM, в рамках которого сформирован крупнейший в мире репозиторий пакетов (если не принимать во внимание, что многие пакеты включают лишь несколько строк кода). В настоящее время в NPM размещено более 380 тысяч пакетов. Для сравнения в репозитории Apache Maven присутствует 173 тысячи пакетов, в Rubygems.org - 127 тысяч, в Packagist (PHP) - 123 тысячи, в PyPI - 96 тысяч, в nuget (.NET) - 70 тысяч, в CPAN - 34 тысячи. За последние 28 дней через NPM было установлено 18 миллиардов пакетов и загружено 6 миллиардов пакетов (66% установок осуществлено из кэша, поэтому число загрузок меньше, чем число установок). Еженедельно примерно 160 новых разработчиков публикуют в NPM свои первые пакеты. Число зарегистрированных пользователей NPM составляет 314 тысяч, из которых 102 тысячи являются активными разработчиками, публикующими свои пакеты. Примечательно, что несмотря на то, что NPM создавался для серверного использования с Node.js, около 80% пользователей используют NPM в том числе при разработке фронтэндов (клиентские web-приложения), а 20% - только на стороне фронтэнда.

Возможности языка

- подписка на события и их обработка
- Манипулировние DOM деревом
- Работа с cookies и browser storage

- Обработка исключений
- Сетевой обмен данными

"Движки" и особенности

Втроен в браузер и зависит от производителя: V8 в Chrome, Chakra в IE, Spidermonkey в Firefox, JavaScriptCore в Safari.

- Является объектно-ориентированным языком, только в последней версии использование подхода для работы с объектами соответствует подходу в других языках
- Имеет ряд свойств присущих функциональным языкам, позволят использовать функции как объекты первого порядка: передавать как аргументы, возвращать и присваивать переменным
- Автоматическое приведение типов
- Автоматическая сборка мусора
- Анонимные функции
- Компилируется в момент исполнения

Первый скрипт

```
<html>
    <body>
      <script type="text/javascript">
        alert('Hello JavaScript!');
      </script>
    </body>
</html>
```

Использование JavaScript на HTML странице

Javascript код должен быть заключен в теги:

- в `<head>` тег `<script>`
- в `<body>` тег `<script>`
- Во внешних файлах и импортиорован в документ посреждством указания источника

```
<script src="scripts.js" type="text/javascript">
<!-- Код размещенный здесь не вызовется -->
</script>
```

В каких случаях вызывается?

- Выполянется во время загрузки страницы или когда браузер запускает событие
  - Все инструкции выполняются во время загрузки страницы
  - Некоторые операторы определяют функции которые могут быть вызваны позже
- Может быть добавлен как обработчик событий через атрибут тегов непосредственно к HTML элементу
  - выполняется когда событие вызывается браузером

    ```
    <img src="logo.gif" onclick="alert('clicked!')" />
    ```

Пример выполнения после возникновения события

```
<html>
    <head>
        <script type="text/javascript">
          function test (message) {
            alert(message);
          }
        </script>
    </head>

    <body>
      <img src="logo.gif"
        onclick="test('clicked!')" />
    </body>
</html>
```

Использование внешних файлов

- HTML страница

```
<html>
    <head>
      <script src="sample.js" type="text/javascript">
      </script>
    </head>
    <body>
      <button onclick="sample()" value="Call JavaScript"
        function from sample.js"/>
    </body>
</html>
```

- Внешний файл

```
function sample() {
  alert('Hello from sample.js!')
}
```

JavaScript синтаксис

Очень похож на синтаксис C#

- Операторы (+, *, = , !=, &&, ++, ...)
- Переменные (не типизируются)
- Условия (if, else)
- Циклы (for, while)
- Массивы (array[] и ассоциативные массивы array['abc'])
- Функции (могут возвращать значения)

Стандартные всплывающие диалоговые блоки

- Блок `alert` имеет кнопку `[OK]` - просто показывает уведомление
  `alert("Какой то текст");`
- Блок подтверждения `confirm` - имееет кнопки `[OK]` и `[Cancel]`
  `confirm("Вы уверены?");`
- Блок ввода - содержит текст поле ввода и значение по умолчанию
  `prompt ("введите значение", 10);`

Встроенные обьекты браузера

Через эти обьекты Javascript получает доступ к браузеру и содержимому веб страницы

- **window** - Верхний уровень DOM дерева, представляет окно браузера
- **document** - содержит информацию о текущем загруженном на странице документе
- **screen** - Хранит параметры дисплея
- **browser** - Содержит информацию о браузере

Объект Math

Объект Math позволяет работать с математическими функциями

```
for (i = 1; i <= 5; i++) {
  var x = Math.random();
  x = 10 * x + 1;
  x = Math.floor(x);
  console.log(
      "Random number (" +
      i + ") in range " +
      "1..10 --> " + x);
}
```

Объект Date

Объект Date() позволяет работать с календарными функциями

```
var now = Date();
var result = "сейчас " + now
```

Задержка выполнения

```
function bang() {
    console.log("Прошло 5 секунд")
}

var timer = setTimeout(bang, 5000);
```

`clearTimeout(timer);` **- останавливает таймер и предотвращает вызов функции**

Повтор выполнения через определенные промежутки времени

```
z = 0
function bang() {
    z++
    console.log("Прошла " + z + " секунда")
}

function clear() {
    clearInterval(timer);
    z = 0;
}

var timer = setInterval(bang, 1000);
// через 5 секунд останавливаем функцию
var stop = setTimeout(clear, 5060);
```

Пример таймера

```
<script type="text/javascript">
  function timerFunc() {
    var now = new Date();
    var hour = now.getHours();
    var min = now.getMinutes();
    var sec = now.getSeconds();
    document.getElementById("clock").value =
      "" + hour + ":" + min + ":" + sec;
  }
  setInterval(timerFunc, 1000);
</script>
<input type="text" id="clock" />
```

Дебажинг

- Современные браузеры имеют консоль, в которую вываливается служебная информация и сыпятся ошибки от исполняемых скриптов, ошибки могут отличаться в различных браузерах.
- Некоторые системы предоставляют возможность детального отслеживания ошибок, установку контрольных точек и наблюдателей
- существует отличное приложение Firebug для Firefox
- Node.js позволяет скидывать в лог информацию

- Вывод информации в консоль, в зависимости от типа сообщение может быть отформатировано стилистически или скрипт приостановить свое выполнение.
    - debug("Сообщение или переменная")
    - info("Сообщение или переменная")
    - log("Сообщение или переменная")
    - warn("Сообщение или переменная")
    - error("Сообщение или переменная")