Javascript Part 1

(www.iliasky.com/www/presentations/JS-1.html)

Printed on Sat, 21 Apr 2018 08:54:31 GMT

JavaScript

- JavaScript е скриптов език за уеб.
- Изпълнява се предимно в браузъра.
- Обектно-ориентиран език е, но:
  - Има прототипно базирано наследяване.
  - Има динамично типизиране.
  - Има функционални елементи(first-class functions).

JavaScript features

- EcmaScript е името на стандарта, който JS в браузърите се стремят да поддържат.
- Можем да разделим поддръжката на 3 фази:
  - Pre-ES5: стационарния компютър на баба ви
  - ES5: IE9+
  - ES6 (ES2015) +: Chrome + Firefox, но не и IE или мобилни
- Днес ще разгледаме първите 2, третата ще разгледаме другия път (тогава ще покажем и как да я подкарваме навсякъде).

JavaScript в HTML

```
<script type="text/javascript" src="file.js"></script>
<script type="text/javascript">...</script>
```

Променливи

```
var meaningOfLife = 42;       // typeof meaningOfLife = 'number'
var wantedGrade = 6.0;        // typeof wantedGrade = 'number'
var firstLine = "So long\n";  // typeof firstLine == 'string'
var secondLine = 'And thanks for all the fish!'; // no difference between ' and "
var happy = true;             // typeof happy = 'boolean'
var nothing;                  // typeof nothing = 'undefined'
```

Всичко останало - масиви, обекти, функции и тн - са обекти

Масиви

Индексирана последователност от елементи. Няма ограничение за хомогенност на елементите.

```
var fibonacci = [undefined, true, 1, 'two', 3.0, 5, 7];

alert(fibonacci[3]);          // 'two'
alert(fibonacci.length);      // 7
alert(fibonacci.indexOf(3));  // 4
alert(fibonacci.indexOf(4));  // -1
alert(fibonacci.slice(1, 3)); // [true, 1]

// push, pop, shift, unshift, sort are also available

// ES5 gives us forEach, map, filter, reduce etc

typeof fibonacci              // object
Array.isArray(fibonacci)      // true
```

Обекти (речници)

Или още map, хешове, речници, асоциативни масиви и други. Съдържа двойки с ключ и стойност. Ключовете са низове.

```
var person = {firstName: 'Chuck', lastName: 'Norris', powerLevel: 9001};

alert(person.firstName);    // Chuck
alert(person['lastName']);  // Norris
```

Първият начин (с точката) е за предпочитане.

Функции

```
function add(a, b) {
  return a + b;
}

var add = function (a, b) {
  return a + b;
};

// ES6 arrow function - doesn't work in 13% of browsers
var add = (a, b) ⇒ a + b;

console.log(add(1, 2, 5));  // returns 3, ignores last argument
```

Условни оператори

```
if (cond) {
  ...
} else if (cond) {
  ...
} else {
  ...
}

isHappy ? 'smile' : 'frown';

switch (n) {
  case 1: ... break;
  case 2: ... break;
  default: ...
}
```

Цикли

```
while (condition) { ... }

do { ... } while (condition);

var cars = ['Audi', 'BMW', 'Mercedes-Benz', 'Opel', 'Porsche', 'Volkswagen'];

for (var i = 0; i < cars.length; i++) {
  console.log(cars[i]);
}

cars.forEach(function(car, index){
  console.log(car);
});

cars.forEach(car ⇒ console.log(car));
```

Обхождане на речник

```
var person = {firstName: 'John', lastName: 'Doe', age: 25};
var txt = '';
for (var key in person) {
  txt += person[key] + ' ';
}

console.log(txt); // John Doe 25

Object.keys(person).reduce((a, key) ⇒ a + ' ' + person[key]);
```

Глобални обекти в браузъра

- Имаме няколко обекта, достъпни глобално на всяка страница.
- `window` (прозорец), `screen` (екран), `history` (история), `navigator` (браузър) - неинтересни
- `location` - чрез него взимаме информация за URL адреса или го сменяме с нов.
- `console` - чрез него извикваме `console.log`
- `document` - съдържа цялото ни DOM дърво (html) ?

Селектиране на елементи

```
<h1 id="page-title">Selection</h1>
<ul>
  <li class="food">Pizza</li>
  <li class="drink">Water</li>
  <li class="drink">Wine<li>
  <li class="food">Burger</li>
</ul>
```

```
var title = document.getElementById('page-title');   // HTMLElement - only 1 element!
var items = document.getElementsByTagName('li');     // HTMLCollection
var foods = document.getElementsByClassName('food'); // HTMLCollection

foods[0].textContent += ' Hut';         // Pizza Hut
title.innerHTML += ' <em>example</em>'; // Selection <em>example</em>
console.log(title.textContent);         // Selection example

// ES5 - use any CSS selector
var water = document.querySelector('.drink');     // only water
var drinks = document.querySelectorAll('.drink'); // water and wine
```

Работа с елементи

```
// Create
var text = document.createTextNode('text node content');
var div = document.createElement('div');

// Add
div.appendChild(text);
document.body.appendChild(div);

// Delete
var list = document.getElementById('myList');
list.removeChild(list.childNodes[0]);
```

Event handling

- Когато работите с браузъра постоянно се emit-ват събития за това.
- `click`, `mousemove`, `keypress`, `submit`, `DOMContentLoaded`, `hashchange` etc
- Можем да зададем handler функция за тези ситуации
- mybutton

```
// Create handler
var clickHandler = function(event) {
  console.log(event);     // See what you'll get in the console
};

var button = document.getElementById('mybutton');
button.addEventListener('click', clickHandler);
button.removeEventListener('click', clickHandler);

// Alternative - not very recommended
button.onclick = clickHandler; // this way we can have only one handler
```

JSON

- JSON = JavaScript Object Notation
- Тоест JSON е стринг, който изглежда почти идентично на JS обект. Има само 2 ограничения:
- Ключовете и стринговете задължително трябва да са в двойни кавички
- Стойностите могат да са само `string`, `number`, `object`, `array` или `true`, `false`, `null`
-

```
// Object
{
  key: 'string',
  num: 1024,
  arr: [1, 2, 3],
  obj: {naughty: null, nice: true},
  func: function () { return true; }
}
```

```
// JSON
{
  "key": "string",
  "num": 1024,
  "arr": [1, 2, 3],
  "obj": {"naughty": null, "nice": true}
}
```

- `JSON.stringify(obj) ⇒ json` и `JSON.parse(json) ⇒ obj`

AJAX

- AJAX = Asynchronous JavaScript and XML (но в днешно време е JSON)
- Дава ни възможност да получим данни от сървъра, без да презареждаме страницата.
- Всичко става асинхронно, тоест при изпращането на заявката се задава callback функция, която да обработи данните, когато те пристигнат.

AJAX - Demo

```
function ajax(url, settings){
  var xhr = new XMLHttpRequest();
  xhr.onload = function(){
    settings[xhr.status == 200 ? 'success' : 'error'](xhr.responseText);
  };
  xhr.open(settings.method || "GET", url, /* async */ true);
  xhr.send(settings.data || null);
}
var log = console.log.bind(console),
    err = console.error.bind(console),
    url = '../../blog/write-html-css-fast/index.html'; // same domain

ajax(url, {success: log, error: err});
```

Tricky notes

- Странностите в JS са почти неизброими
- Scope-ът на променливите (до ES5 включително) е само на ниво функция
- Пропуснати `;` не дават грешки, а се "самооправят"
- Функции и други променливи могат да се ползват преди момента на декларирането си (hoisting)
- WAT - short video **1:21**
- Не споменахме RegExp и Date - за първото има презентация тук, а второто няма да ни трябва сега.
- Ако в handler-а на form submit имаме `return false` или `event.preventDefault()` то формата няма да се изпрати.

Next time on W3Tech

- Scope
- `this`
- Patterns
- Prototypes
- ES6 features
- Node & Babel
- libs

Задача 1

Направете функция validate(), която да валидира следната форма:

- потребителско име - поне 3 символа и най-много 10 символа - букви, цифри и _
- парола - поне 6 символа, като трябва да има поне 1 главна, 1 малка буква и 1 цифра
- парола втори път - трябва да съвпадат
- бутон Submit изпращащ въведените данни към някакъв .php файл.

Ако има невалидно поле, то на страницата трябва да се появи съобщение за грешката и нищо да не се праща до сървъра.

Ако данните са валидни и се опитаме да изпратим формата, тя трябва да се изпрати (post до register.php).

Забележки:

- Popup прозорчето от alert НЕ се зачита като решение.
- Искаме JS решение - без PHP код.

Полезни ресурси:

- Списък с event-и в html5
- Списък с property-та на String

Задача 2

Използвайки ajax функцията от презентацията (или ваша модификация) искаме да изпращаме ajax request вместо нормалното изпращане на формата.

За целта ще изпратим JSON стринг от типа {"username": "...", "password": "..."}, където сме взели съответните стойности с javascript.

За да се уверите, че всичко работи - напишете си примерен register.php файл, където да разпакетирате данните от JSON към PHP асоциативен масив. След това можете да проверите дали масивът е наред чрез var_dump().

Каква е идеята:

- да предадем данните въведени от потребителя чрез javascript и в json формат, след което да ги извлечем като асоциативен масив в php.
- трябва да се научим да четем документации :(

Неща за търсене:

- json_encode и json_decode
- JSON.stringify и JSON.parse
- form events и input properties
- php://input и xhr Content-Type