Стандартная структура документа:

<!DOCTYPE html>

<html lang="en-US">

<head>

<title>Introduction to Web Pages</title>

</head>

<body>

Welcome

</body>

</html>

Два основных типа элементов:

**Block Elements** –

* <br> element
* <body> element
* <p> elements
* <div>

**Inline Elements**

* strong
* ins
* del
* sup
* sub
* i
* em

| **Format** | **Markup** | **Result** |
| --- | --- | --- |
| Bold | <b> | **Bold** Text |
| Strong | <strong> | **Strong** Text |
| Italic | <i> | *Italic* Text |
| Emphasized | <em> | *Emphasized* Text |
| Small | <small> | Small Text |
| Marked | <mark> | Marked Text |
| Deleted | <del> | D Text |
| Inserted | <ins> | Inserted Text |
| Subscript | <sub> | Subscript Text |
| Superscript | <sup> | Superscript Text |

Internal links:

<img id="coolImage" src="images/interestingImage.jpg" alt="Interesting Image" />

<a href="#coolImage">Navigate to Cool Image</a>

#### Input Types

There are a wide variety of types that can be used with the **input** element. This table shows the types and their respective functionality:

| **Type** | **Functionality** |
| --- | --- |
| button | This renders a button that is typically used with JavaScript frameworks (such as Angular or Knockout) |
| checkbox | This renders a checkbox that can be independently clicked. The checkbox has a value of on or off |
| file | This renders a field for file selection and a Browse button to select the file on the client device |
| hidden | This does not render anything but it will store a value. This can be used to store values that need to be sent to the server when the form is saved |
| image | This renders an image that functions as a submit button (click on image to save form) |
| password | This renders a text input where the characters are masked for privacy. |
| radio | This renders a radio button that can be selected as part of a group of radio buttons. |
| text | This renders a basic text input field. |
| submit | This renders a button that submits the HTML form. |

### Input Attributes

Multiple attributes were introduced that allowed for the most common and simple validation requirements. If all validation requirements weren't met, the browser would simply not save your HTML form. There were also new attributes introduced that enhanced the functionality of forms in HTML5.

#### Required

The **required** attribute specifies that a text input must have valid data before the HTML form is saved:

<input name="email\_address" type="email" required="required" />

#### Pattern

The **pattern** attribute specifies a regular expression that the value of the input field must match before the HTML form is saved:

<input name="zip\_code" type="text" pattern="\d{5}(-\d{4})?" required="required" />

#### Readonly

The **readonly** attribute specifies that a particular field is readonly and cannot be modified by the user:

<input name="profile\_url" type="url" readonly="readonly" />

#### Disabled

The **disabled** attribute specifies that a particular field is disabled and cannot be modified by the user:

<input type="submit" disabled="disabled" />

#### Min/Max

The **min** and **max** attributes are used with the **range** input type to specify boundaries for the selected numeric value:

<input name="issue\_quantity" type="range" min="1" max="15" />

They can also be used with **date** inputs:

<input name="service\_date" type="date" min="2000-01-01" max="2999-12-31">

#### Autocomplete

The **autocomplete** attribute is used with various text inputs to toggle the auto-complete feature found in most modern browsers.

<input name="api\_key" type="text" autocomplete="off" />

#### Placeholder

The **placeholder** attribute renders a "watermark" that is shown when the input is empty. This hint text can be used to give the user further instructions for the specific field.

<input name="account\_name" type="text" placeholder="Account name must contain at least 4 characters." />

### New Input Types in HTML5

HTML5 introduced a new set of input **types** that can be used in your HTML form for data capture validation without the need for additional code or JavaScript:

| **Type** | **Functionality** |
| --- | --- |
| color | This renders a color picker. |
| date | This renders a date control that allows you to select year, month and day without time. |
| datetime-local | This renders a date control that allows you to select year, month, day and time without time-zone information stored. |
| email | This renders a text input field and validates that the e-mail address is valid. |
| month | This renders a date control that allows you to select year and month only. |
| number | This renders a text input field that only allow numeric input. |
| range | This renders a control (typically a slider in most browsers) that allows a user to select an imprecise number. |
| search | This renders a text input field used for search. |
| tel | This renders a text input field used for telephone numbers. |
| time | This renders a date control that allows you to enter time without time-zone information stored. |
| url | This renders a text input field and validates that the url is valid. |
| week | This renders a date control that allows you to select year and week only. |

## **SVG**

Scalable Vector Graphics (SVG) is an image format that can draw 2D graphics directly in the browser. SVG is based on XML and supports features such as interactivity, transitions and animations. SVG was one of the many standards designed by the World Wide Web Consortium (W3C). Typically SVG graphics are stored in XML files and edited using vector-based image manipulation programs. Many browsers can display SVG graphics in a manner similar to how they display other image formats.

HTML5 introduced the ability to embed SVG graphics directly in web pages. Now you can create SVG shapes in your HTML web page and manipulate them directly with CSS or JavaScript.

Css

The coding of CSS style rules can be done in three places, namely:

* **Inline** - done in the HTML tag.
* **Internal Style Sheet** - coded at the beginning of a HTML document i.e. inside the <head></head> tags, and closed by the <style type=“text/css”> </style> tags.
* **External Style Sheet** - this is a separate file with a .css extension which serves as a reference for multiple HTML pages to use. A link is defined in the header of the HTML pages pointing browsers to where to look for the styles.
* **<link rel=”stylesheet” href=””>**

.class {}

#id {}

Child:

section > p {

font-weight: bold;

}

body {

color: red !important;

}

## JavaScript

<script src="/path/to/script.js"></script>

if (day > 5) {

greeting = "Have a good weekend!";

} else if (day < 2) {

greeting = "Are you having a case of the Mondays!";

} else {

greeting = "Welcome to work";

}

var countries = ["USA", "JPN", "RUS", "ENG"];

for (var i = 0; i < countries.length; i++) {

text += countries + "<br />";

}

## **jQuery**

$('a#example').html('Click Me'); //id

$('button.blueButton').css('background-color', 'blue'); /class

JavaScript

Чтобы определить константу используем ключевое слово const, имя константы пишем большими буквами

Чтобы использовать локальную переменную для блока, используем ключевое слово let

++x Увеличивает х и выводит тоже значение

х++ выводит х, а потом увеличивает его

Promise:

https://courses.edx.org/courses/course-v1:W3Cx+JS.0x+1T2017/courseware/c383695587a642409d48e3a7d808cf8c/0c071e4b3e8040c5987afbe506249a24/?child=first

<https://learn.javascript.ru/promise>

<https://learn.javascript.ru/fetch>

Javascript имеет не блочную, а область видимости внутри функций

Объекты:

Если свойство объекта начинается с числа, то к нему можно обращаться только через квадратные скобки, через точку нельзя !!!

Тоже самое, если свойство имеет пробелы

**Explanation**

No, it's not correct! While adding and removing properties after the object has been declared/created (line 1 of the source code), the syntax for declaring properties INSIDE an object (with ":") is no more valid. You must use the "=" operator for the assignment. The correct code is:

let pacman = {};

pacman.color **=** 'yellow';

pacman.shape **=** 'pizza';

With JavaScript version 5 (and previous versions), you can define a pseudo-class template called "a constructor function". The syntax is the same as for creating a function, except that, by convention, its name is Capitalized. The first letter of the function name is in uppercase. It is a good way to know, when you read someone else's code, that this is not a regular function, but a constructor function. Its name is a noun, the name of the class of objects you are going to build. Example: Person, Vehicle, Enemy, Product, Circle, Ball, Player, Hero, etc.

You also build new objects using the new keyword.

[Course](https://courses.edx.org/courses/course-v1:W3Cx+JS.0x+1T2017/course/)  [Module 4: Structuring data](https://courses.edx.org/courses/course-v1:W3Cx+JS.0x+1T2017/course/#block-v1:W3Cx+JS.0x+1T2017+type@chapter+block@1f5a7a2285994fbd87d23829a53d29d2)  [4.5 Exercises - Module 4](https://courses.edx.org/courses/course-v1:W3Cx+JS.0x+1T2017/course/#block-v1:W3Cx+JS.0x+1T2017+type@sequential+block@46a6fe30eb904f8cace75feb2930200a)  Exercices (13-17)

 Previous

1. other Intro. exercises - Module 4
2. problem Exercises (1-6)
3. problem Exercises (7-12)
4. problem Exercices (13-17)

Next

## Exercices (13-17)

 Bookmark this page

### 13. Order is important (part 1)

1/1 point (graded)

let x = sum(3, 5);

function sum(x, y) {

return (x + y);

}

The function sum is declared after we called it at the first line of the above code. Is this correct? Will it work?

Yes correct

No

**Explanation**

In JavaScript, you can call a function BEFORE it has been declared in your source code. This is called "hoisting": it's like if all function declarations were moved to the top before being executed. It works and it's correct.

var p = new Person();

class Person {...}

The class Person is declared after being used at the first line of the above code. Is this correct? Will it work?

No correct

Yes

**Explanation**

Unlike functions, classes must be declared BEFORE using them.

An important difference between function declarations and class declarations is that function declarations are "hoisted" and class declarations are not. It means that you can call a function BEFORE it has been declared in your source code. This is not the case with ES6 classes!

You first need to declare your class and then access it, otherwise a line of code, such as the one shown in the example, will give a ReferenceError

#### here is a new one: objects can also be created by functions that return objects (factories)

1. function getMousePos(event, canvas) {
2. var rect = canvas.getBoundingClientRect();
3. var mxx = event.clientX - rect.left;
4. var my = event.clientY - rect.top;
6. **return { // the getMousePos function returns an object. It’s a factory**
7. **x: mx,**
8. **y: my**
9. **}**
10. }

And here is how you can use this:

1. var mousePos = getMousePos(evt, canvas);
3. console.log("Mouse position x = " + mousePos.x + " y = " + mousePos.y);

JavaScript is a "pass by value" language, unlike some other languages, which are "pass by reference" languages. This means that when you pass a variable to a function as argument, the value of the variable is copied into the argument.

Example:

1. var x = 2;
2. function sum(a, b) {
3. a = a + b;
4. return a;
5. }
6. sum(x, 3); // returns 5
7. x; // 2 <- but x equals 2

When working with objects, the reference of the object is copied into the argument. That means you can modify the referenced object. But if you change the reference (for example by assigning a new object), the original variable (which now points to another object) will not be modified.

Example 1:

1. var obj = { x: 2 }
2. function add(a, b) {
3. a.x += b;
4. }
5. add(obj, 3);
6. obj.x; // 5 <- The referenced object is modified

### Comparing two objects

Comparing two objects will only return true if they point to the same object (i.e., if they have the same reference).

**When this environment is a Web server   
(and this is the case for all examples we have seen in this course),**  
**this global object is named window.**

**The “global variables” defined with the keyword var are properties of this window object,   
and we can say the same of predefined functions like prompt, alert, etc.**

**However, at the top level of programs and functions,   
let, unlike var, does not create a property on the global window objec**

### The most useful methods of the built-in class Array

#### Most useful methods you can use on arrays: sort(), join(), slice(), splice(), push()and pop()

* + sort: sort the elements in the array. Either alphabetically if they are strings, or in ascending order if they are numbers. There is also a way to sort the elements using other criteria, which is explained a bit further on in the course. With a call to var b = a.sort(), a is also sorted. The sort method sorts the array + returns it.
  + join: adds a string between each element and returns the result as a string
  + slice: returns a sub-array without modifying the original array
  + splice: modifies the array, it removes “a slice” and it also adds new elements
  + push: appends an element at the end of the array and returns the new length
  + pop: removes the last element and returns it

Есть массив с несколькими элементами, нужно преобразовать его в строку разными способами.

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | // массив **var** arr = [1, 2, 3, 4, 5];  // склеивание через запятую **var** str = arr.join(); // 1,2,3,4,5  // склеивание без разделителя **var** str2 = arr.join(''); // 12345  // склеивание через " . " **var** str3 = arr.join(' . '); // 1 . 2 . 3 . 4 . 5  // склеивание через " - " **var** str4 = arr.join(' - '); // 1 - 2 - 3 - 4 - 5 |

var a = [1,2,3,4,5];

var str1 = a.join('');

b=str1.indexOf('5');

index=mass.join('').indexOf('elem'); /// РАБОТАЕТ ТОЛЬКО С ОДНОЗНАЧНЫМИ ЧИСЛАМИ и СИМВОЛАМИ!!!!

Math.random() returns a float value between 0 and 1.

**100 \* Math.random(); // between 0 and 100**

#### To get a number between a min and a max value, use this formula: val = ((max - min) \* Math.random()) + min

Math.round(6\*Math.random()) – целое число от 1 до 6

Число от 7 до 12: Math.round((12-7)\* Math.random()+7)

## [Доступ к объекту через this](https://learn.javascript.ru/object-methods" \l "доступ-к-объекту-через-this)

Для полноценной работы метод должен иметь доступ к данным объекта. В частности, вызов user.sayHi() может захотеть вывести имя пользователя.

**Для доступа к текущему объекту из метода используется ключевое слово this**.

Значением this является объект перед «точкой», в контексте которого вызван метод

Такой синтаксис, при котором функция объявляется в контексте выражения (в данном случае, выражения присваивания), называется Function Expression, а обычный синтаксис, при котором функция объявляется в основном потоке кода – Function Declaration.

Функции, объявленные через Function Declaration, отличаются от Function Expression тем, что интерпретатор создаёт их при входе в область видимости (в начале выполнения скрипта), так что они работают до объявления.

Обычно это удобно, но может быть проблемой, если нужно объявить функцию в зависимости от условия. В этом случае, а также в других ситуациях, когда хочется создать функцию «здесь и сейчас», используют Function Expression.