JavaScript for Enterprise Development

Week 1

CONTENTS

HTML

Simple example

Let's take a look at simple example of HTML element:

```
<article>My First HTML Element</article>
```

HTML tags label pieces of content such as "heading", "paragraph", "table", and so on. Browsers do not display the HTML tags, but use them to render the content of the page

HTML tags usually come in pairs like `<p>` and `</p>`. The first tag in a pair is the start (opening) tag, the second tag is the end (closing) tag.

What is HTML?

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language;
- HTML describes the structure of Web pages using XML markup;
- HTML elements are the building blocks of HTML pages;
- HTML elements are represented by tags.

There is many historical revisions of HTML standard, but we will focus on latest and actual **HTML5** specification.

Web page structure

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>
```

Elements is base building blocks of HTML document, displayed by web browsers.

Elements could be nested and therefore form hierarchical structure.

This structure reflects semantic and visual representation of document.

Web page structure

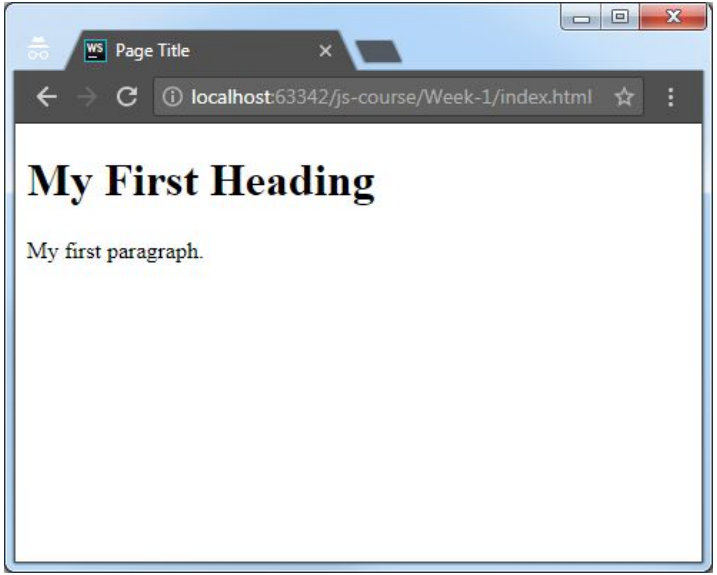
```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>
```

- `<!DOCTYPE html>` defines document as HTML5 (special case of non-self-closing single tag);
- `<html>` element - root element of an HTML page;
- `<head>` holds meta information about document;
- `<title>` element specifies a title for the document;
- `<body>` element contains the visible page content;
- `<h1>` element defines a large heading;
- `<p>` element defines a paragraph.

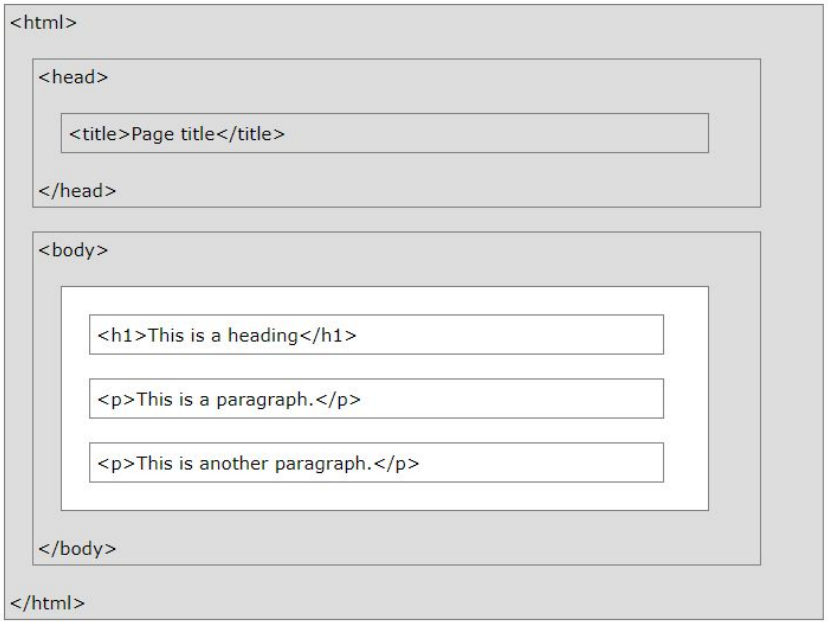
Results

Let's save our web page markup as index.html file and open it in Google Chrome browser:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>My First Heading</h1>
    <p>My first paragraph.</p>
  </body>
</html>
```



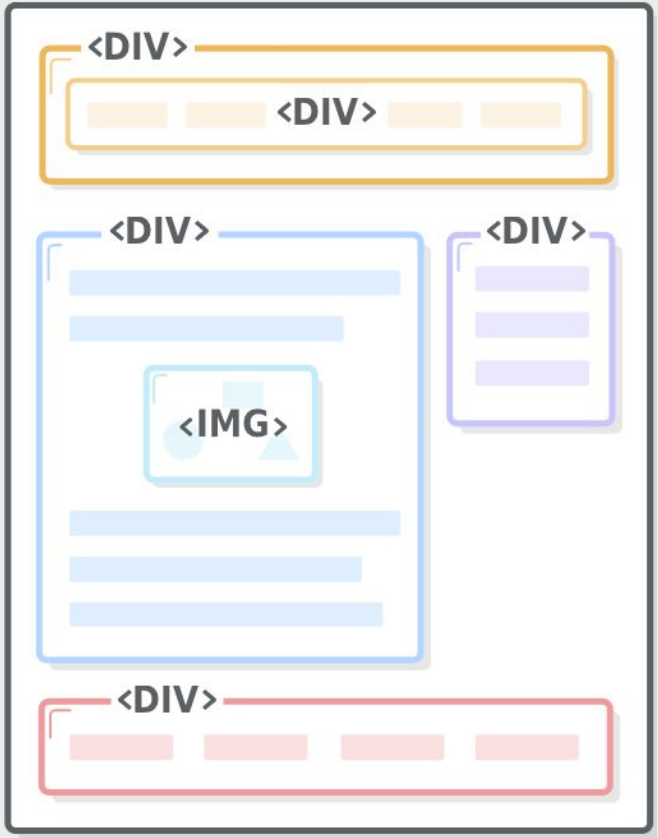
HTML page semantics

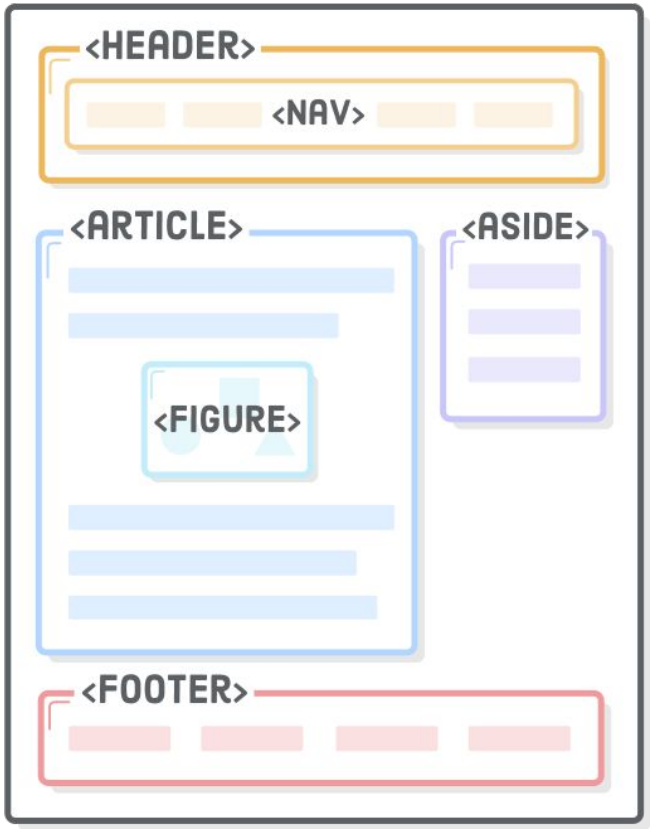


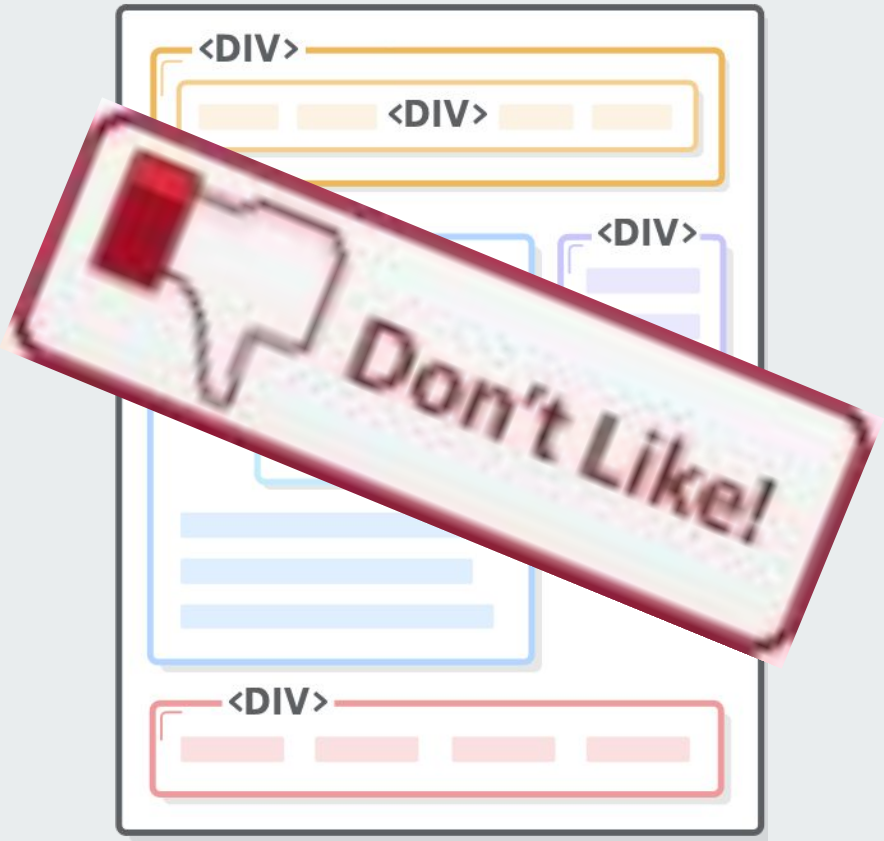
Keep in mind:

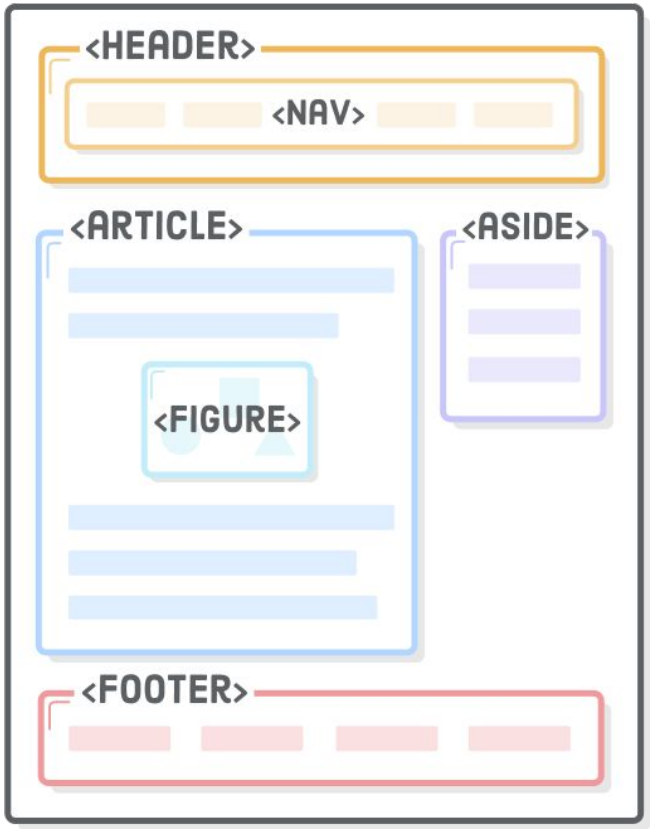
HTML at first represents semantic structure of document, not it`s appearance.

"Semantic HTML" refers to the idea that all your HTML markup should convey the underlying meaning of your content.


<DIV>
<DIV>
<DIV>
<DIV>
<IMG>
<DIV>


<HEADER>
<NAV>
<ARTICLE>
<ASIDE>
<FIGURE>
<FOOTER>


<DIV>
<DIV>
<DIV>
Don't Like!
<DIV>


<HEADER>
<NAV>
<ARTICLE>
<ASIDE>
<FIGURE>
<FOOTER>

Semantic web is one of the most popular web developer job interviews topic.

You can learn more about this at Semantic HTML article at internetingishard.com.

Practical details and showcases could be found in Inhuman UI lecture of Vadim Makeev

Job interview most popular questions

Note on working with HTMl

1. You could use any lightweight text editor in such simple cases like above, but when it comes to large and complicated projects we recommend using professional IDE like **WebStorm**, **IntelliJ IDEA**, **Atom** or **VS Code**.

   Essential requirement for HTML text editor are HTML syntax highlight and UTF-8 support.

2. Files, containing HTML, have **.html** or **.htm** extension. A common filename for web application entry point is **index.html**;

3. UTF-8 encoding is recommended.

HTML Document

As we saw before, all HTML documents starts with a type declaration: `<!DOCTYPE html>`.

The HTML document itself begins with `<html>` and ends with `</html>`.

The `<head>` section contains meta information about page and additional non-visualizable data. Some of useful tags within head tag includes `<meta>`, `<script>`, `<style>` and `<title>`.

The visible part of the HTML document is between `<body>` and `</body>`. There is various amount of "presentational" tags, which could be used inside of body.

Some of them: `<span>`, `<img>`, `<table>`, `<video>`, `<input>`, `<button>`.

Into elements

An HTML element usually consists of a start and end tag, with the content inserted in between:

```
<p>My first paragraph</p>
```

HTML elements with no content are called empty elements. Empty elements do not have an end tag and have a little distinct syntax. Take a look at line break element as example:

```
<br />
```

Nested HTML Elements

```
<!DOCTYPE html>
<html>
 <!-- This is a comment -->
  <body>
   <h1>My First Heading</h1>
   <hr />
   <p>My first paragraph.</p>
  </body>
</html>
```

Obviously, HTML elements can be nested.

Indentations and line breaks are optional (browser will ignore them), but significantly improves readability.

Take a note: Browser is not case-sensitive about HTML tags: `<P>` means the same as `<p>`, but we recommend using lowercase in HTML.

Later React library syntax will make us to use lowercase for standard in-browser tags anyway.

HTML Attributes

Attributes provide additional information about HTML elements.

Example: HTML links are defined with the `<a>` tag. The link address is specified in the href attribute:

```
<a href="https://www.google.com">This is a link</a>
```

One more example: The filename of the image source is specified in the src attribute:

```
<img src="img_girl.jpg">
```

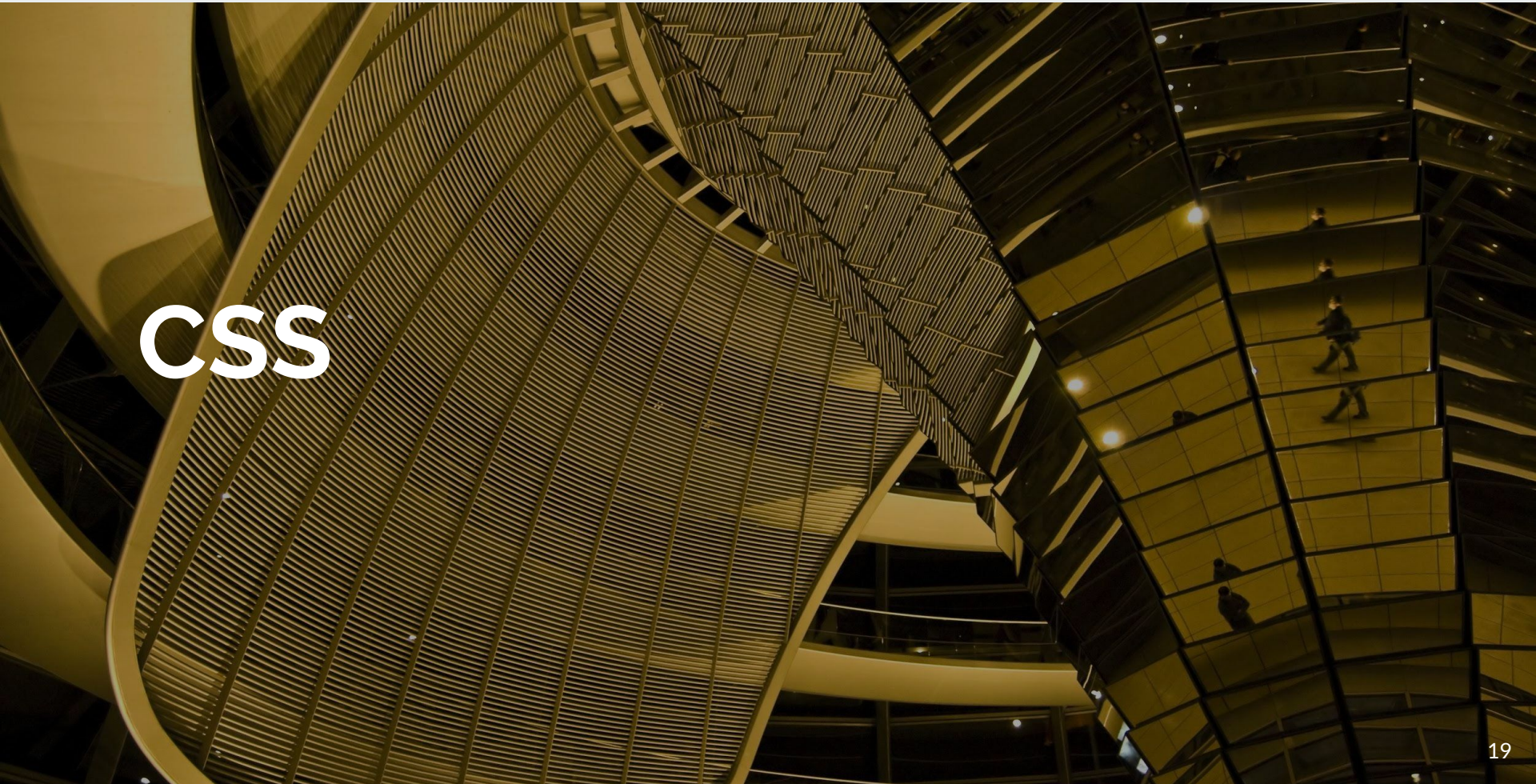
HTML Attributes

The style attribute is used to specify the styling of an element, like color, font, size etc.:

```
<p style="color:red;">I am a red paragraph</p>
```

Summary:

Each tag have it's own attributes. Some of them are optional (like `style`), some are mandatory (like `src` on `<img>` tag). Attributes allow specify element representation and behaviour.


CSS

CSS

CSS stands for Cascading Style Sheets. It describes how HTML elements are to be displayed on screen, paper, or in other media.

CSS can be added to HTML elements in 3 ways:

- **Inline** - by using the style attribute in HTML elements
- **Internal** - by using a `<style>` element in the `<head>` section
- **External** - by using an external CSS file

The most common way to add CSS is to keep the styles in separate CSS files.

Inline CSS

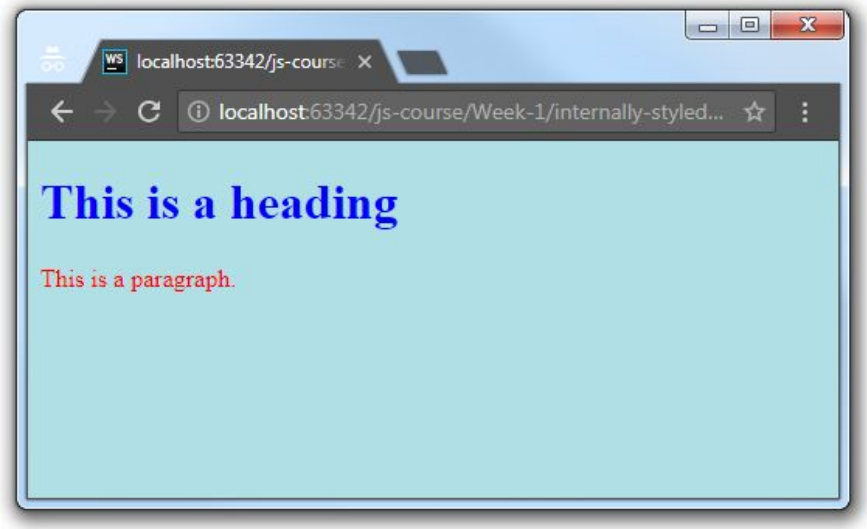
Is used to apply a unique style to a single HTML element with use of style attribute of an element

```
<h1 style="color:blue;">
  This is a Blue Heading
</h1>
```

Internal CSS

Defines a style for a single HTML page and defined in the `<head>` section, within a `<style>` element:

```html
<!DOCTYPE html>
<html>
  <head>
    <style>
      body {background-color: cyan;}
      h1   {color: blue;}
      p    {color: red;}
    </style>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```



External CSS

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet we should add a link to it in the <head> section of the HTML page.

```html
<!DOCTYPE html>
<html>
  <head>

  <link rel="stylesheet" href="styles.css">

  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

styles.css

```css
body {
  background-color: cyan;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

“id” attribute

To define a specific style for single unique element, add an id attribute to the element:

```
<p id="application">I am different</p>
```

then define a style for the element with the specific id:

```
#application {
  color: blue;
}
```

“class” attribute

To define a style for special types of elements, add a class attribute to the element:

```
<p class="error">I am different</p>
```

then define a style for the elements with the specific class:

```
p.error {
  color: red;
}
```

OR

```
.error {
  color: red;
}
```

Combining rules

To specify which elements should receive the rule in previous examples we specified tags, which will be used as filtering rule.

E.g.: `h1 {color: blue;}` rule will select all `h1` elements on the page and make them blue.

Besides tag/element selectors there is also class selectors, id selectors, attribute selectors and their combinations. Rules could overlap each other.

In each case rule priority will be determined by **selector specificity**

Specificity is a weight that is applied to a given CSS declaration, determined by the number of each selector type in the matching selector.

Job interview most popular questions

Deep dive

CSS is pretty simple, but drastically diverse technology.

There is two major aspects of web development, based on CSS:

1. Browsers support. Most of CSS rules are cross-browser, but some of modern standards has not been adopted by all browsers (*hello, Internet Explorer*)
2. CSS organisation on huge projects. Rules could interfere and even exclude each other. Therefore, you have to adopt one of techniques to help you with this complexity. Insights and ideas on webpage elements hierarchy, semantic layout and styling could be found in BEM methodology.


JavaScript

JavaScript

JavaScript makes HTML pages more dynamic and interactive.

```
<!DOCTYPE html>
<html>
<body>
   <button
      type="button"
      onclick="document.getElementById('demo').innerHTML = Date()"
   >
      Click me to display Date and Time.
   </button>
   <p id="demo"></p>
</body>
</html>
```

<script> tag

The `<script>` tag is used to define a client-side script (JavaScript).

The `<script>` element either contains scripting statements, or it points to an external script file through the src attribute (just like with external css).

```
<!DOCTYPE html>
<html>
<head>
   <script type="text/javascript" src="script.js">
   </script>
</head>
...
```

index.html

```
<button type="button" onclick="handleClick()" >
<p id="demo"></p>
```

script.js

```
var handleClick = function () {
    var el = document.getElementById('demo');
    el.innerHTML = Date();
};
```

index.html

```
<button type="button" onclick="handleClick()" >
<p id="demo"></p>
```

script.js

```
var handleClick = function () {
    var el = document.getElementById('demo');
    el.innerHTML = Date();
};
```

index.html

```
<button type="button" onclick="handleClick()" >
<p id="demo"></p>
```

script.js

```
var handleClick = function () {
    var el = document.getElementById('demo');
    el.innerHTML = Date();
};
```

JS + HTML + CSS

Sooner or later it became difficult to control and keep in mind all two-directional connections between all elements on the page and JavaScript instructions.

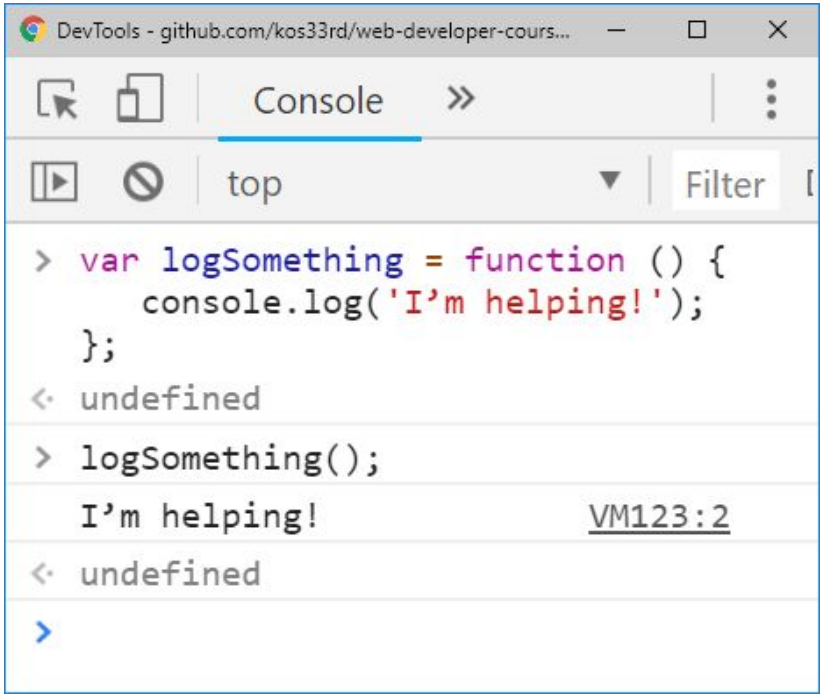
Reasonable approach is to split whole web page into more or less isolated pieces and consider such ones as an whole combination of HTML, CSS and JS - a component.

This approach is being called **Component-based**. It has been adopted by such web libraries and frameworks, as React, Angular, Vue and even became web standard called **Web components**.

Debugging

script.js

```
var logSomething = function () {
    console.log('I'm helping!');
};
```



JavaScript: Base syntax and Data types

Variables, Assignment and Strings

es5.js

```
var stringOne = "I'm a string";
var stringTwo = 'I\'m also a string';
```

es6.js

```
var stringOne = "I'm a string";
let stringTwo = 'I\'m also a string';
const stringLiteral = `I have ${1 + 2} assignments`;
```

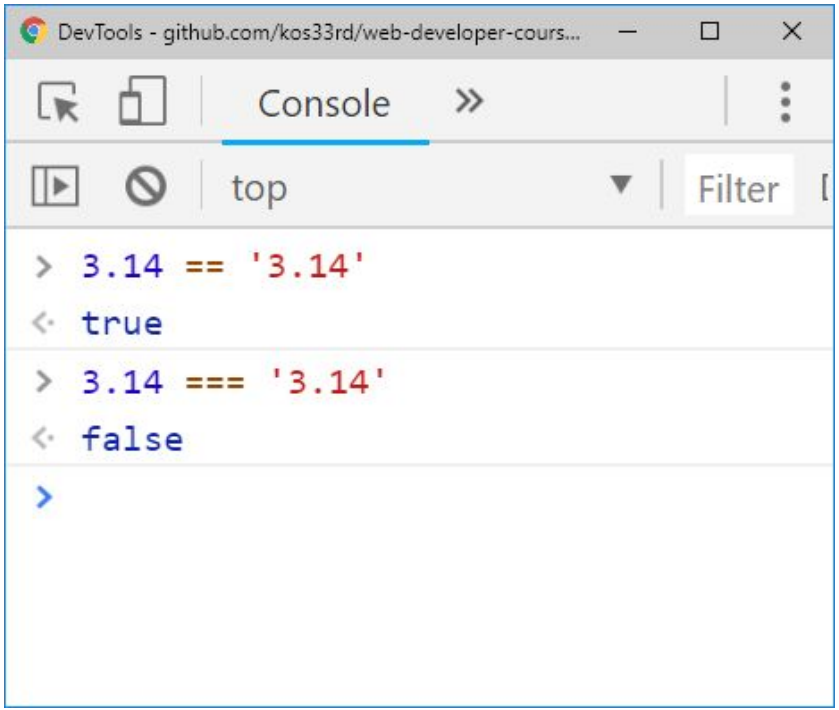
Primitive data types

```
var person = "John Doe";
var pi = 3.14;
var hasCat = true;
var userId = null;

var middleName; // middleName is undefined
var lastName = void 0 //es6, also undefined

var privateMethod = Symbol(); //es6
```

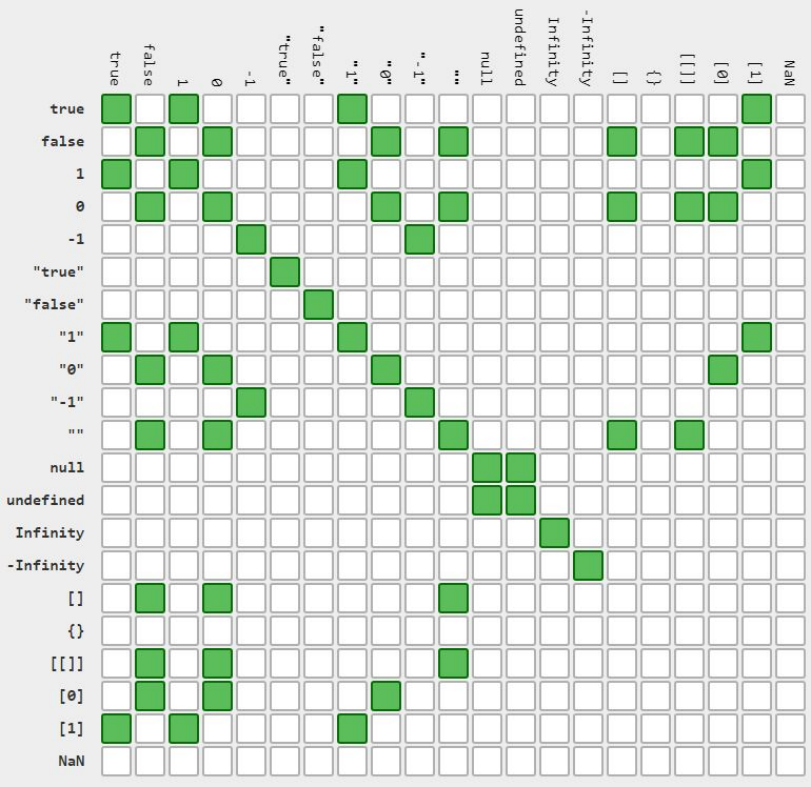
Other built-in data types

```
var personData = {
 creationDate: new Date(),
 friends: [],
 hasFriends: function() { /*...*/ },
};

var bitmap = new Int32Array(1024); //es6
/*...*/
```

Type coercion



Type casting



It’s… complicated

Type casting

General advices:

1. Explicit is better than implicit. Do not substract an Array from undefined.
2. Learn your language. Type casting in JS is actually pretty rational and logical.
3. Try to avoid some rare data types and values such as NaN, Null, Infinity and etc.

Operator precedence

Level	Operators	Notes
1	() [] .	call, member (including typeof and void)
2	! ~ − ++ −−	negation, increment
3	* / %	multiply/divide
4	+ −	addition/subtraction
5	<< >> >>>	bitwise shift
6	< <= > >=	relational
7	== !=	equality
8	&	bitwise AND
9	^	bitwise XOR
10	|	bitwise OR
11	&&	logical AND
12	||	logical OR
13	?:	conditional
14	= += −= *= /= %= <<= >>= >>>= &= ^= |=	assignment
15	,	comma

Type coercions and Operator priorities questions are pretty common at Web Developer interviews.

Job interview most popular questions

Functions

es5.js

```
function calcProd (p1, p2) {
 return p1 * p2;
}
```

es5.js

```
var calcSum = function(p1, p2) {
 return p1 + p2;
}
```

es6.js

```
const divByTwo = (divisor) => {
 return divisor / 2;
}
```

es6.js

```
const divByTwo = divisor =>
 divisor / 2;
```

Loops

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

while (j < 10) {
  console.log(`It's ${j}`);
  j++;
}
```

Conditions

```
if(haveAccess) {
 console.log('Allow');
}

switch (new Date().getDay()) {
 case 5:
   console.log("Friday!");
   break;
 default:
   console.log(':|');
}
```

JavaScript: Browser API and Events

DOM manipulations

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can add and remove existing HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

Practical details will be highlighted in lab exercises

HTML Events

An HTML event can be something the browser does, or something a user does, e.g.:

- An HTML web page has finished loading
- An HTML input field was changed
- An HTML button was clicked

HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.

```
<button onclick="this.innerHTML = Date()">The time is?</button>
```

Most common HTML Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Window object

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser. It's available in JavaScript by the ***window*** global variable.

All global JavaScript objects, functions, and variables automatically become members of the window object.

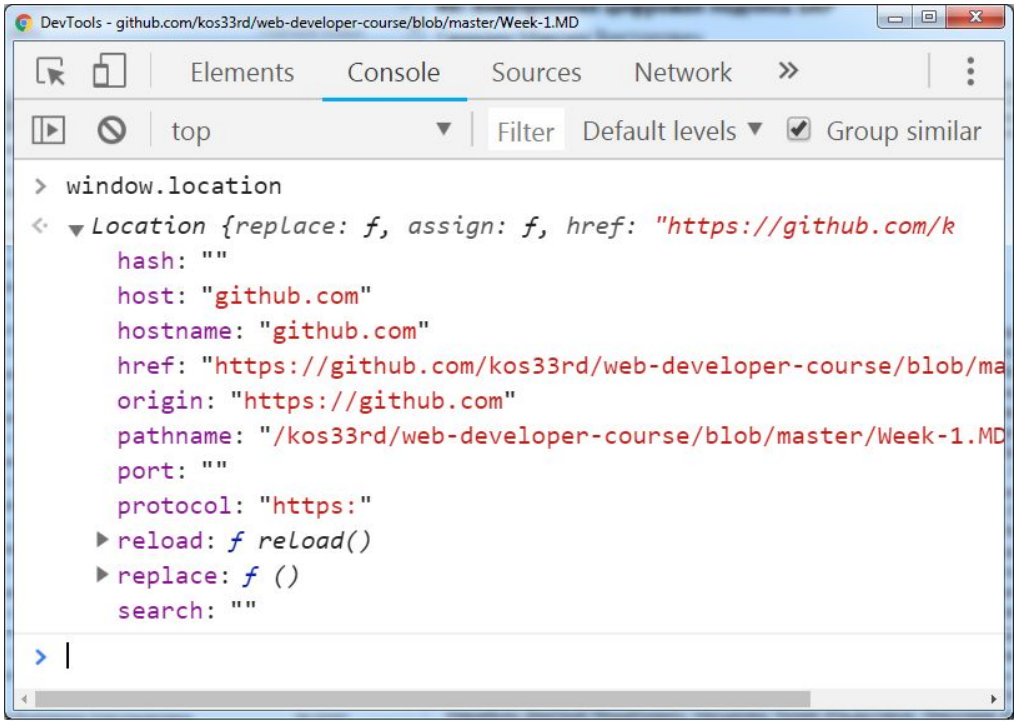
Apparently previously used in examples ***document*** is also a part of a window object:

```
window.document === document
```

Window object

```
window.open(); // open a new window

window.close(); // close the current window

window.screen.width; // 1920

window.screen.height; // 1080

window.alert("Annoying!");

window.console.log('Message');

window.console.warn('Warning');

window.console.error('Error');
```

Window navigation



Timing events

The window object allows execution of code at specified time intervals. These time intervals are called timing events. The two key methods to use with JavaScript are:

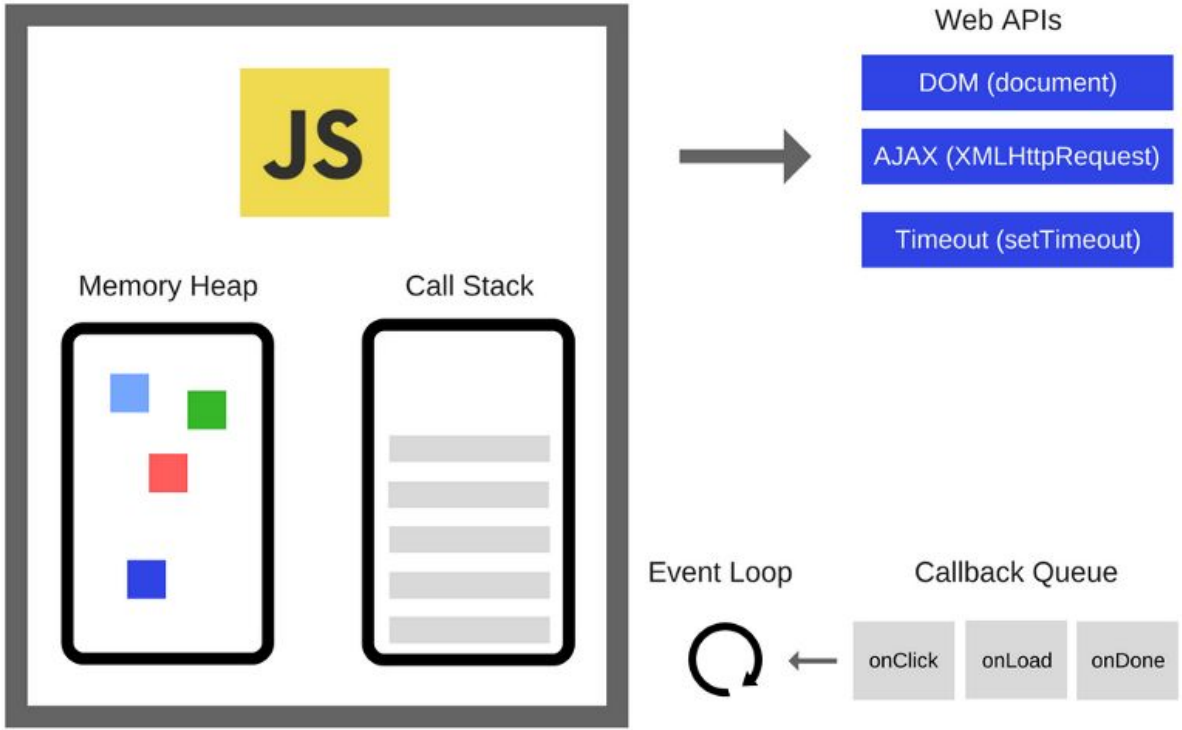
```
setTimeout(yourFunction, milliseconds);
```

Executes a function, after waiting a specified number of milliseconds.

```
setInterval(yourFunction, milliseconds);
```

Same as setTimeout(), but repeats the execution of the function continuously.

Short spin-off about event loop



We recommend future readings and practical exercises about event loop.

Job interview most popular questions

Client-Server communication

```
function loadDoc() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState === 4 && this.status === 200) {
            console.log(this.responseText);
        }
    };
    xhttp.open("GET", "https://github.com", true);
    xhttp.send();
}
```

Modern API

```
fetch('https://github.com')
    .then(function (response) {
        console.log(response)
    });

axios.get('https://github.com')
    .then(function (response) {
        console.log(response);
    });
```

Assignment

Assignment for the next week

- Make your course project application scaffold:
  - index html page
  - base styles
  - javascript entry point
- Design a wireframe of your application with main states / use cases / screens
- Make a PR

That’s all!

Resources

- Lectures, resources and course project requirements: https://github.com/kos33rd/web-developer-course
- Github accounts to send PRs with complete tasks: @kos33rd, @AVVlasov
- News, announcements, resources and useful links: http://moodle.innopolis.university