ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Introduction

Internet programming

Ivan Kitanovski

Bojan Ilijoski

Why JavaScript?

- JavaScript is one of the **3 technologies** that every web programmer **must** know:
  - **HTML** to define the content.
  - **CSS** for specifying the style.
  - **JavaScript** for programming the behavior of the page.

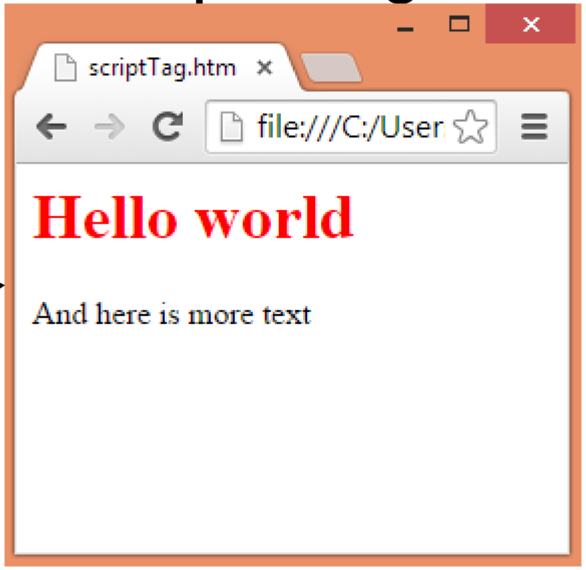
Where it is set JavaScript?

- JavaScript code is inserted between <script> tag:

```
<script type="text/javascript">
 //JavaScript kod
</script>
```

- Example

```
<html>
  <head>
     <script type = "text/javascript">
       <!--
          document.write("<h1 style=\"color: red\">");
          document.write("Hello world");
          document.write("</h1>");
       // -->
     </script>
  </head>
  <body> <p>And here is more text</p> </body>
</html>
```



Hello world

And here is more text

In newer browsers and in HTML5 you don't need to specify it because the predefined scripting language is JavaScript.

Work with browsers that don't support scripting languages.

- Some old browsers don't recognize the script tags.
- These browsers will ignore the script tags, but will display the code in the inserted JavaScript.
- In order to allow old browsers to ignore the entire code, HTML comments are to hide the script from the browser.

- syntax

```
<!–
script here
// -->
```

  - <!– start of HTML comment
  - For JavaScript to ignore the tag for an end of an HTML comment (-->), we use the JavaScript comment (//), which is applied to the end of the line.

Example

```
<html >
<head>
    <title>My first script</title>
</head>
<body bgcolor="#FFFFFF">
<h1>
    <script language="Javascript" type="text/javascript">
    <!-- Hide script from old browsers

            document.write("Hello, world!")

    // End hiding script from old browsers -->
    </script>
</h1>
</body>
</html>
```

Where can we put JavaScript

- It can be inserted in:
  - header (<head>)
  - body (<body>) of one HTML document
  - in the both places
- Functions should be defined in the header (<head>)
  - it provides the function to be loaded before it is used .

External JavaScript

- Can be placed in a separate .js file
  `<script src="myJavaScriptFile.js"></script>`
  - The external .js file allows using the same script to more HTML pages
  - The external .js file can’t contain <script> tag.

- Example:

```
<script src="myjavascript.js”
    language="JavaScript1.2”
    type="text/javascript">
</script>
```

Writing in console

- If the browser supports debugging
  - console.log()

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<script>
    console.log('Hello');
</script>
</body>
</html>
```

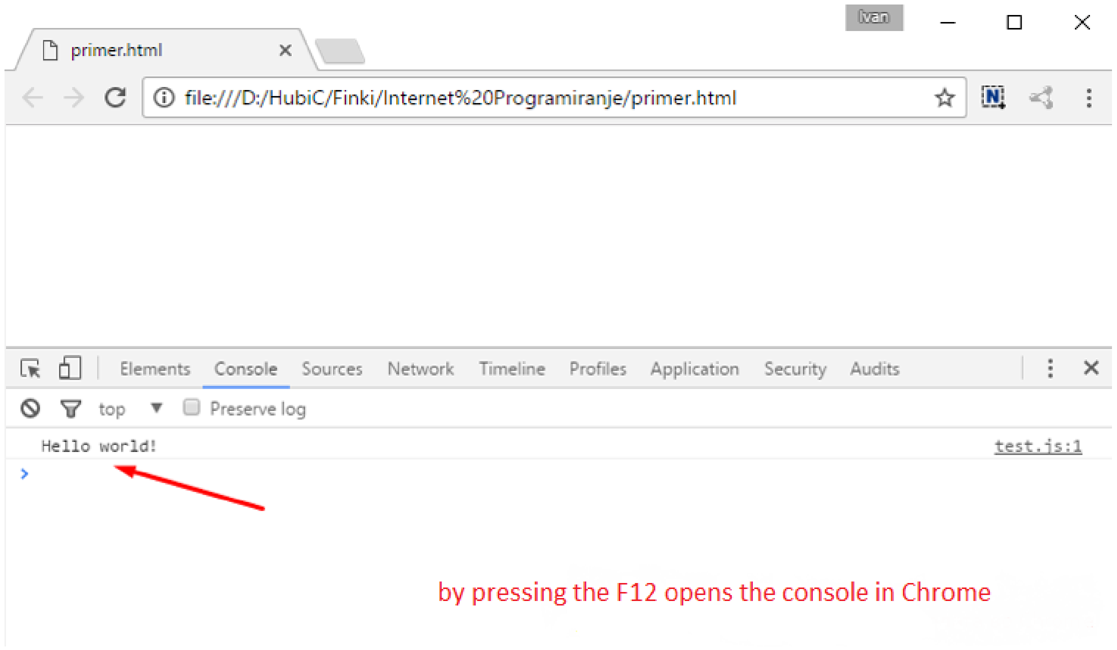
Example: External JS

example.html

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

test.js

```
console.log("Hello world!");
```



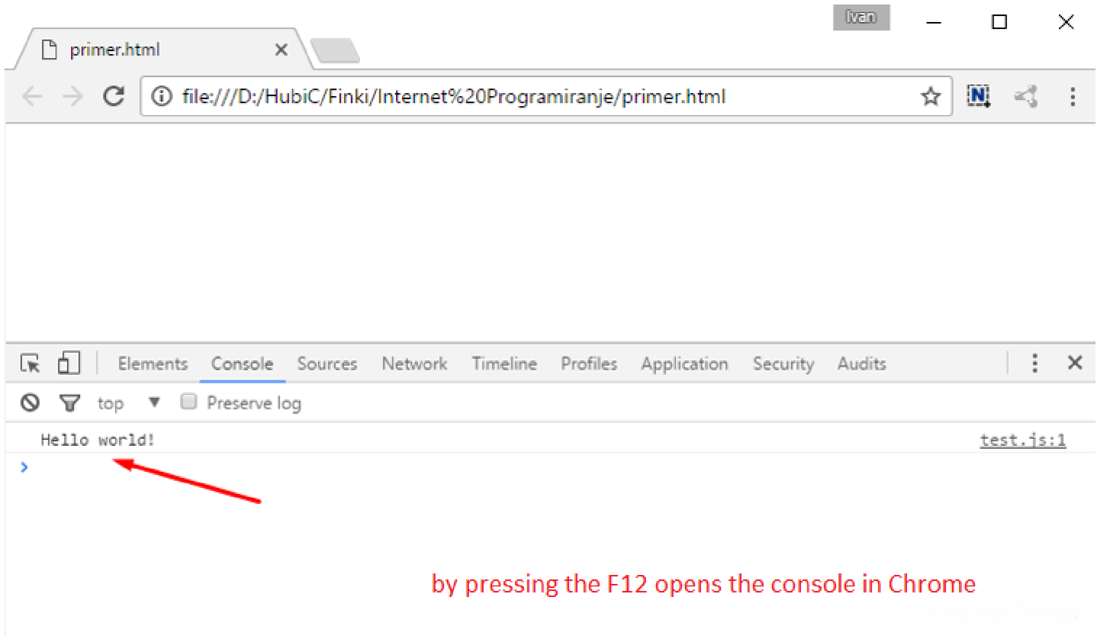
```
Hello world!                                                    test.js:1
```

by pressing the F12 opens the console in Chrome

Example: Directly in the scrip tag

example.html

```
<!DOCTYPE html>
<html>
<head>
        <script type="text/javascript">
                console.log("Hello world!");
        </script>
</head>
<body>
</body>
</html>
```



```
Hello world!                                   test.js:1
>
```

by pressing the F12 opens the console in Chrome

Variables

- The variables are declared with the var or let command
  - The word var is optional (its use is a good programming style)
  - The data type of the variables doesn't need to be declared, it is determined of the time execution of the script.
  - The variables can contain a value of every data type
    - Var pi;
    - Var name: Integer;   //(Javascript 2.0)
- The variables can be initialized using the sign =
    - var pi = 3.1416, x, y, name = "Dr. Dave" ;
  - Names of the variables must start with a letter, dash(_) or the sign($)
  - Capital letter and lowercase make a difference in the names of the variables.
- Constants are declared using the const command
  - Const capital = ‘Skopje’;

Simple data type

- JavaScript has 3 primitive data types:
  - numerical values (number),
  - text strings (string), and
  - logical values (boolean)
  - everything else is an object
- The numerical values are always in floating point format
  - Hexadecimal numbers begin with 0x
  - some platforms considered the number 0123 as octal, others considered it as decimal.

Primitive data types

- Text strings can be limited to single or double quotations marks.
  - they may also contain control marks - \n (newline), \" (double quote), etc.
  - Example:

    strFirst = "John";

    strLast = "Kennedy";

    strFull = strFirst + "F." + strLast;
- Booleam data can have ***true*** or ***false*** values
  - 0, "0", empty strings, undefined, null, and NaN are considered a boolean false
  - all other values are logical true

Example

example.html

```
<!DOCTYPE html>
<html>
<head>
       <script type="text/javascript">
              var pi = 3.14;
              var person = "John Doe";
              var answer = 'Yes I am!';
       </script>
</head>
<body>
</body>
</html>
```

Other data types

- Complex
  - Object
  - Array
- Special
  - Null
  - Undefined

Example

example.html

```
<!DOCTYPE html>
<html>
<head>
        <script type="text/javascript">
                var this_is_empty = null;
                var this_is_undefined;
                // ili this_is_undefined = undefined;
        </script>
</head>
<body>
</body>
</html>
```

Operators

- Operators are used to processing values
- Types of operators
  - arithmetic operators: + - * / % ++ --
  - comparison operators: < <= == != >= >
  - boolean operators: && || !
  - bitwise operators: & | ^ ~ << >> >>>
  - assignment operators: += -= *= /= %= <<= >>= >>>= &= ^= |=
  - string operators: +
- Conditional operator:

  condition ? *value_if_true* : *value_if_false*
- Additional operators:

  new typeof void delete

Operators

- Special relational operations for checking equality:
  - == and != try to convert the operands to the same type before performing the test
  - === and !== assume that the operands are unequal if they are of a different type

Operators

- The system will attempt to cast values to be able to perform the operation.
  - everything can be turned into a text string.
  - some text strings can be converted into numbers.
  - extra info on boolean values.
    - Ina numerical context, true is converted 1, and false is converted to 0
    - in a Boolean context, the defined values are considered as true, and undefined are considered false
    - in the context of text strings, true is converted into "true" and false is converted into "false”
  - nothing but a function, can be converted to function.
- Operators are executed in the context of operands
  - a * b   => provides a numerical context
  - e(x)   => function context (for e)
  - a + b   => undefined (string or number)

Example

example.html

```
<!DOCTYPE html>
<html>
<head>
       <script type="text/javascript">
       var x = 5;         // assign the value 5 to x
       var y = 2;         // assign the value 2 to y
       var z = x + y;     // assign the value 7 to z (x + y)
       </script>
</head>
<body>
</body>
</html>
```

Example

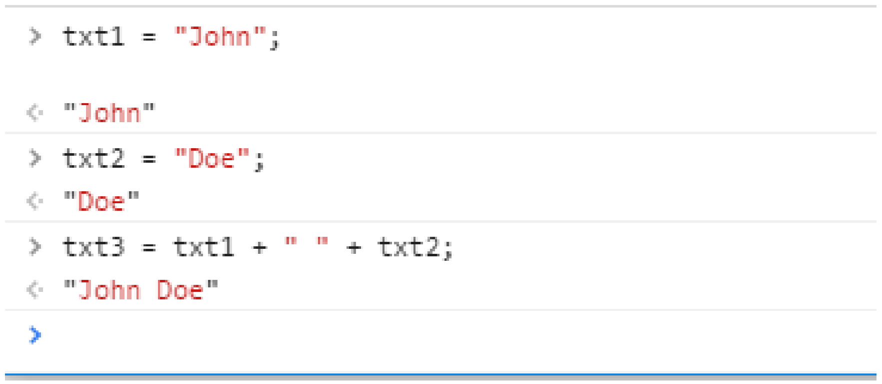
example.html

```
<!DOCTYPE html>
<html>
<head>
        <script type="text/javascript">
                var x = 10;
                x += 5
        </script>
</head>
<body>
</body>
</html>
```

Example

example.html

```
<!DOCTYPE html>
<html>
<head>
       <script type="text/javascript">
              txt1 = "John";
              txt2 = "Doe";
              txt3 = txt1 + " " + txt2;
              console.log(txt3);
       </script>
</head>
<body>
</body>
</html>
```

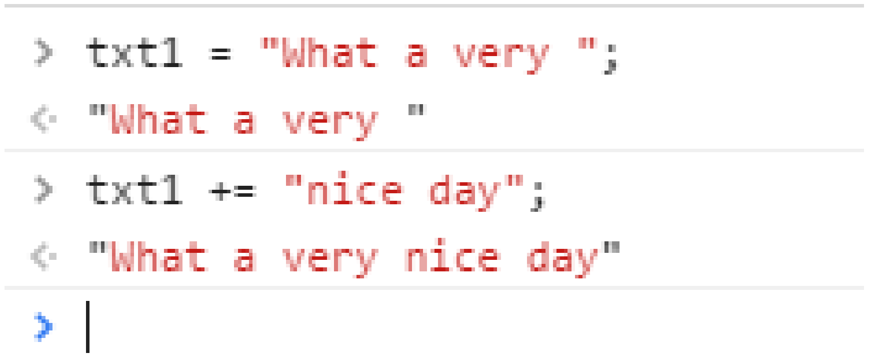


```
> txt1 = "John";
< "John"
> txt2 = "Doe";
< "Doe"
> txt3 = txt1 + " " + txt2;
< "John Doe"
>
```

Example

example.html

```
<!DOCTYPE html>
<html>
<head>
       <script type="text/javascript">
              txt1 = "What a very ";
              txt1 += "nice day";
       </script>
</head>
<body>
</body>
</html>
```

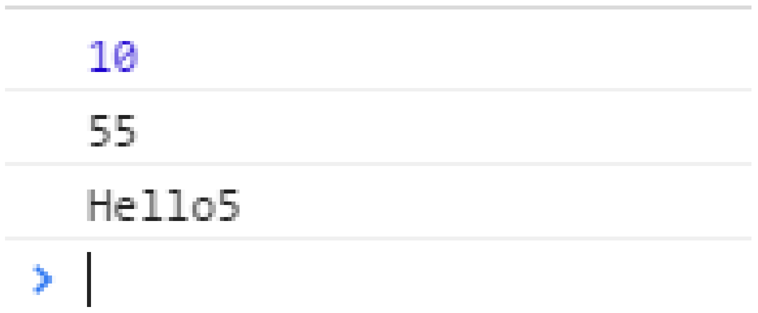


```
> txt1 = "What a very ";
<· "What a very "
> txt1 += "nice day";
<· "What a very nice day"
> |
```

Example

example.html

```
<!DOCTYPE html>
<html>
<head>
        <script type="text/javascript">
                x = 5 + 5;
                console.log(x);
                y = "5" + 5;
                console.log(y);
                z = "Hello" + 5;
                console.log(z);
        </script>
</head>
<body>
</body>
</html>
```

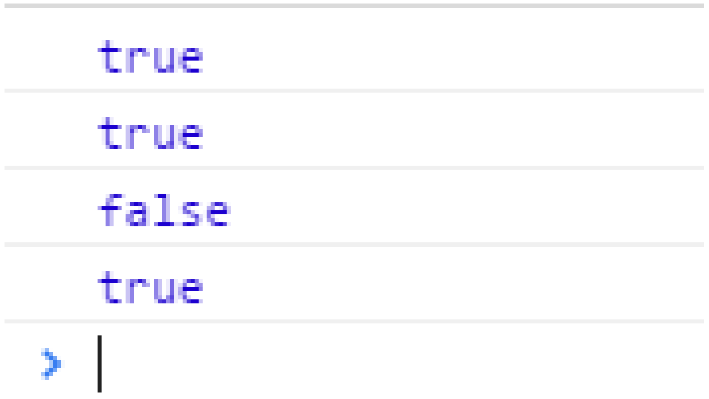


```
10
55
Hello5
>
```

Example

example.html

```
<!DOCTYPE html>
<html>
<head>
       <script type="text/javascript">
              a = 5 == 5;
              console.log(a);
              b = "5" == 5;
              console.log(b);
              c = "5" === 5;
              console.log(c);
              d = "5" === "5";
              console.log(d);
       </script>
</head>
<body>
</body>
</html>
```



```
true
true
false
true
>
```

Commands (1)

- Assignment commands:

  greeting = "Hello, " + name;

  nNum -= 3; nNum = nNum - 3;

  nNum *= 3; nNum = nNum * 3;

  nNum /= 3; nNum = nNum / 3;

  nNum %= 3; nNum = nNum % 3;

- Constraining a block of code:

  { *statement*; …; *statement* }

- Empty command: ;;or { }

Commands (2)

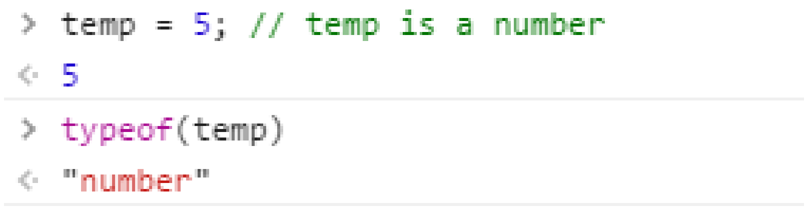
- Conditional statements
  - If, switch
- Commands to repeat a block of code
  - for, do-while, while. for-in
  - Example:

```
var person = {fname:"John", lname:"Doe", age:25};
var text = "";
var x;
for (x in person) {
    text += person[x];
}
```

- break, continue

Typeof / instanceof

- Typeof
  - Returns the type of the variable



```
> temp = 5; // temp is a number
<· 5
> typeof(temp)
<· "number"
```

- Instanceof
  - Returns true only if the variable is of the same type as the given object type

Comments

- Comments are identical to comments in C or Java::
  - from // to the end of the line
  - between /* and */

- Example

```
<script language="JavaScript">
<!-- definition of variables
var num_car= 25;
var passenger_per_car= 3;
//calculation of total number of people
var total_passenger= num_car * passenger_per_car
Alert(total_passenger);
// end of script -->
</script>
```