

**#1**

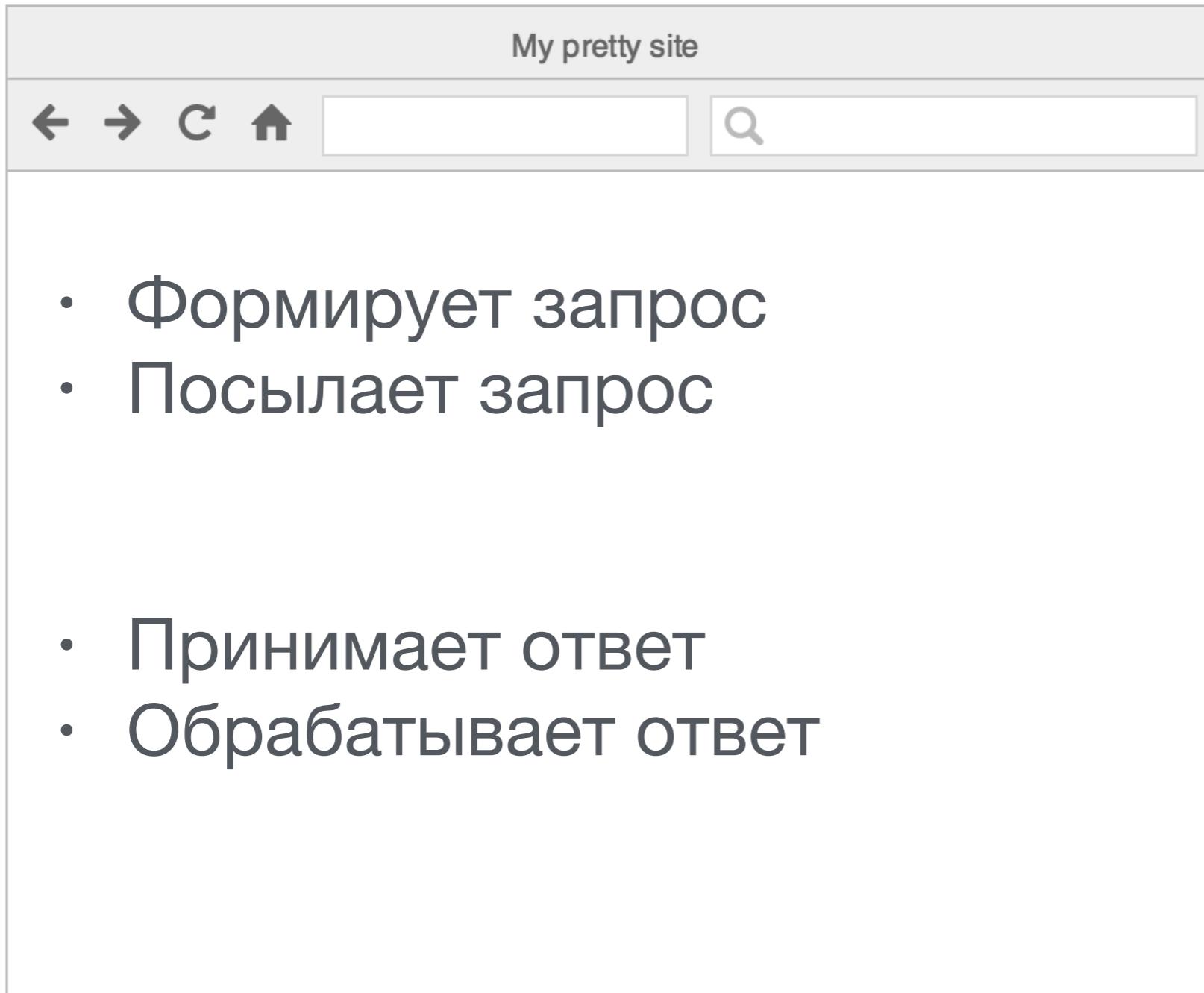
**HTML, SVG, CSS**

**и**

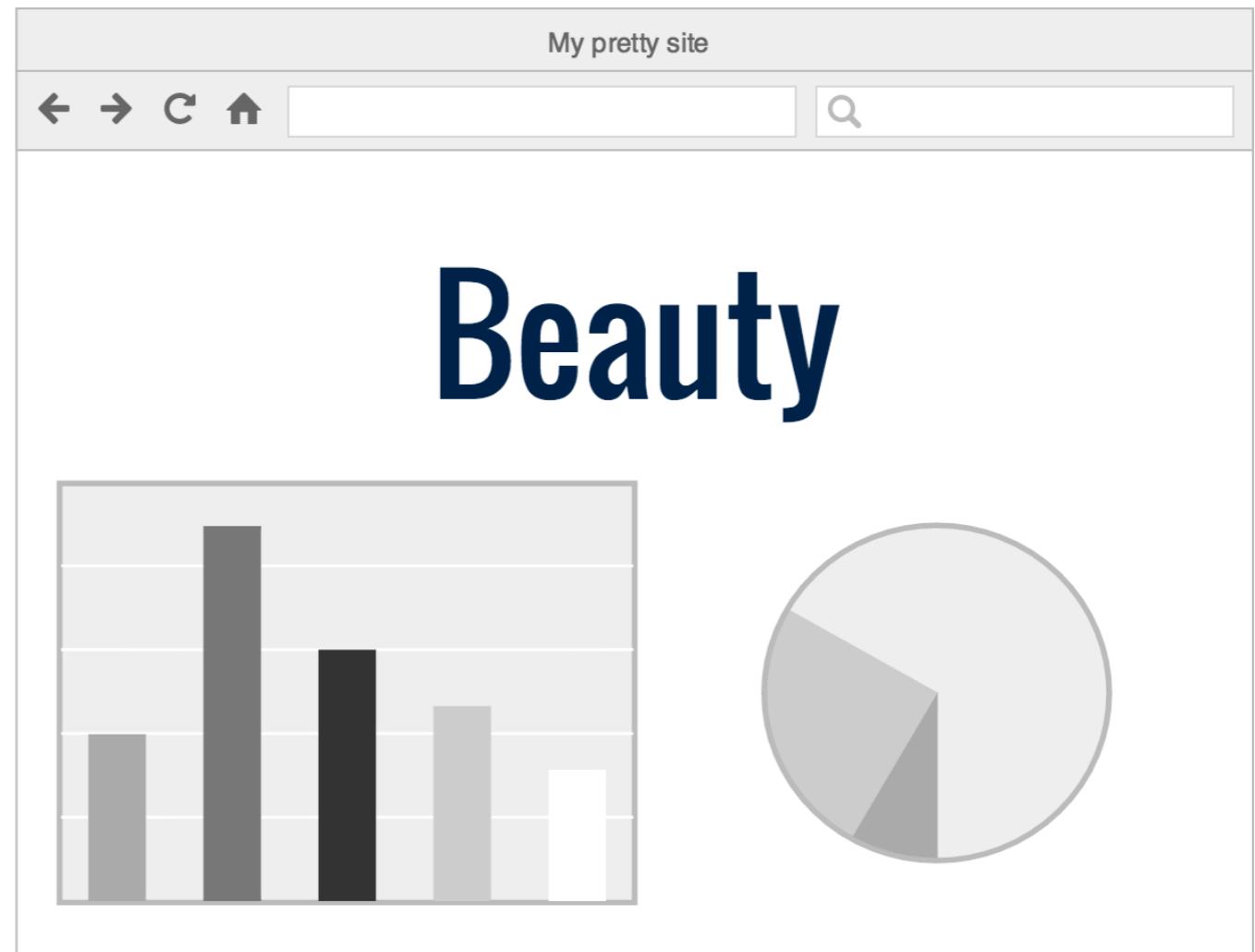
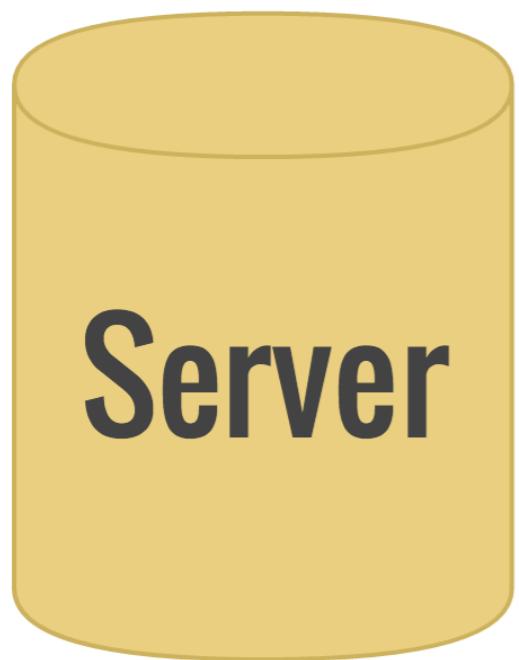
**JAVASCRIPT**

# Представление данных в вебе.

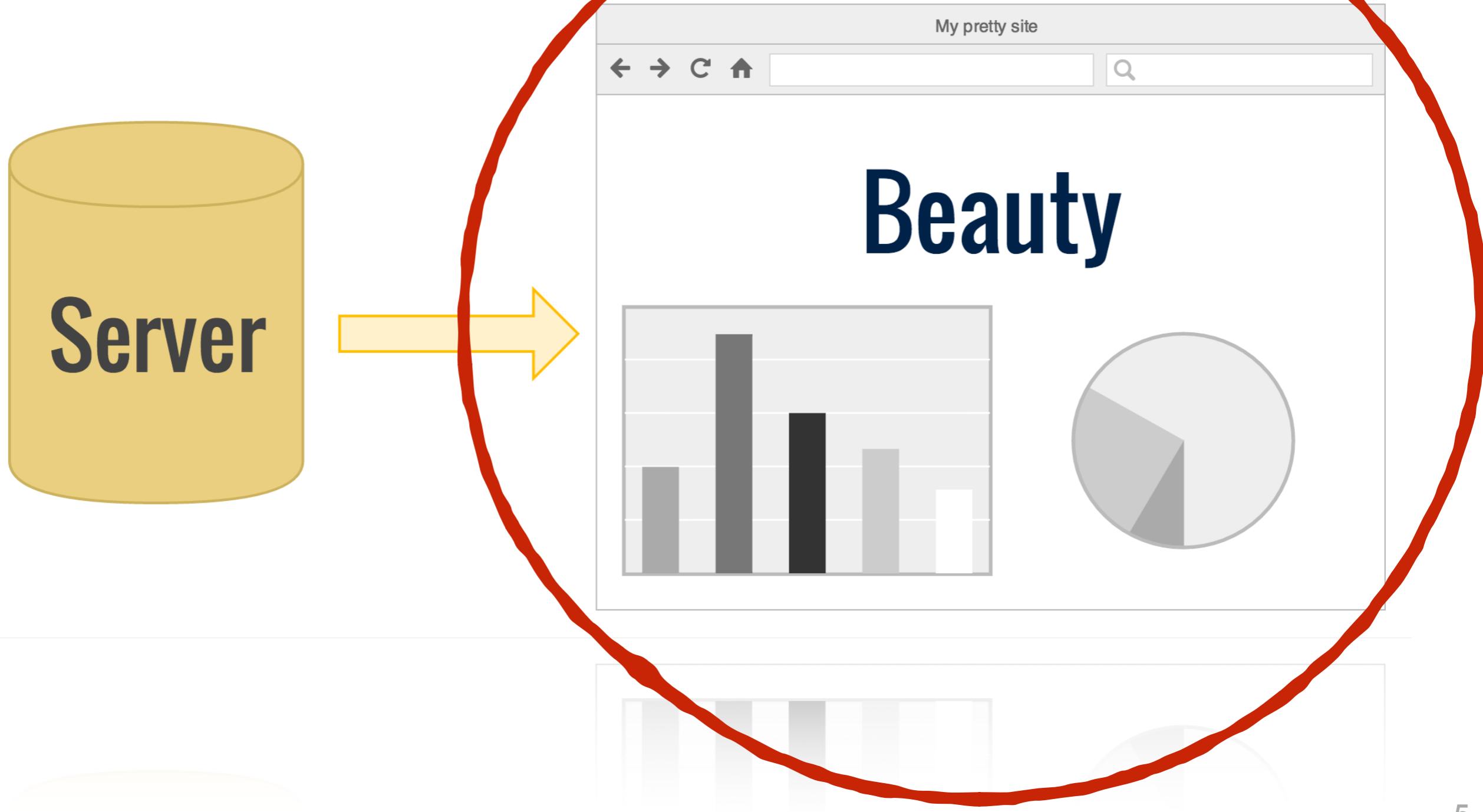
# Как работает браузер ?



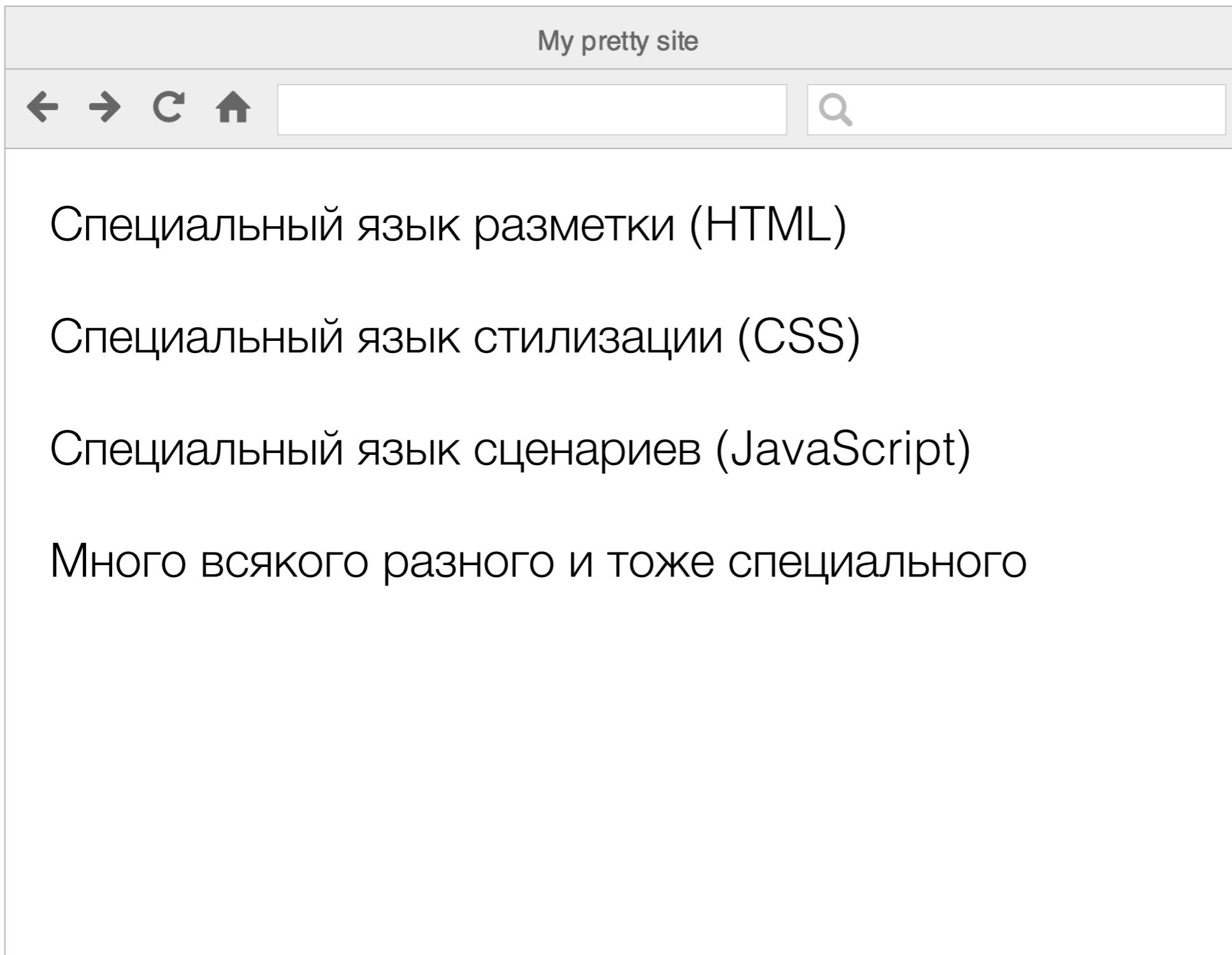
# Как работает браузер ?



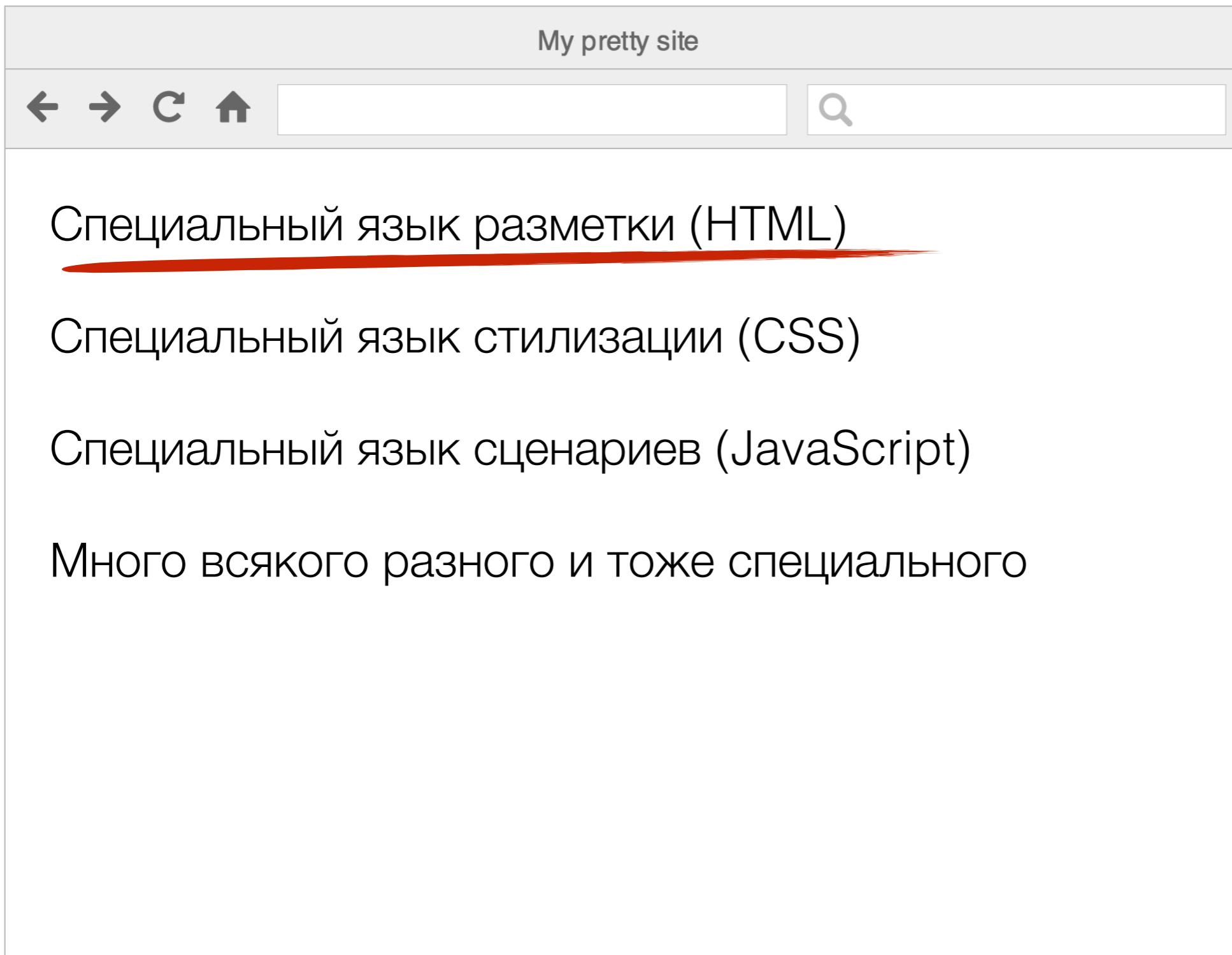
# Как работает браузер ?



# Как работает браузер ?



# Как работает браузер ?



вообще-то

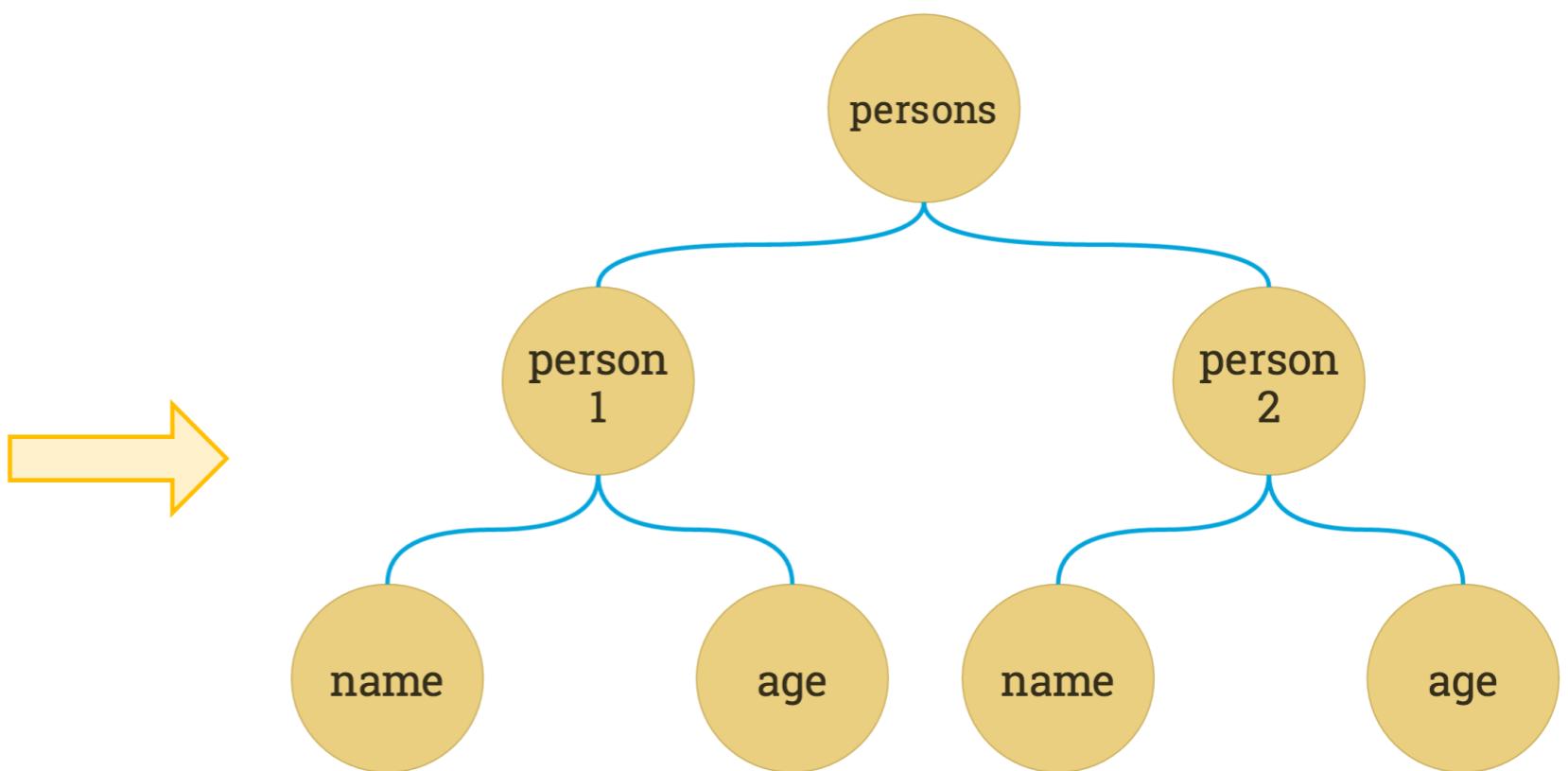
**HTML ∈ XML**

# XML

*extensible markup language*

```
<persons>
  <person id="1">
    <name>John</name>
    <age>26</age>
  </person>

  <person id="2">
    <name>Jane</name>
    <age>24</age>
  </person>
</persons>
```

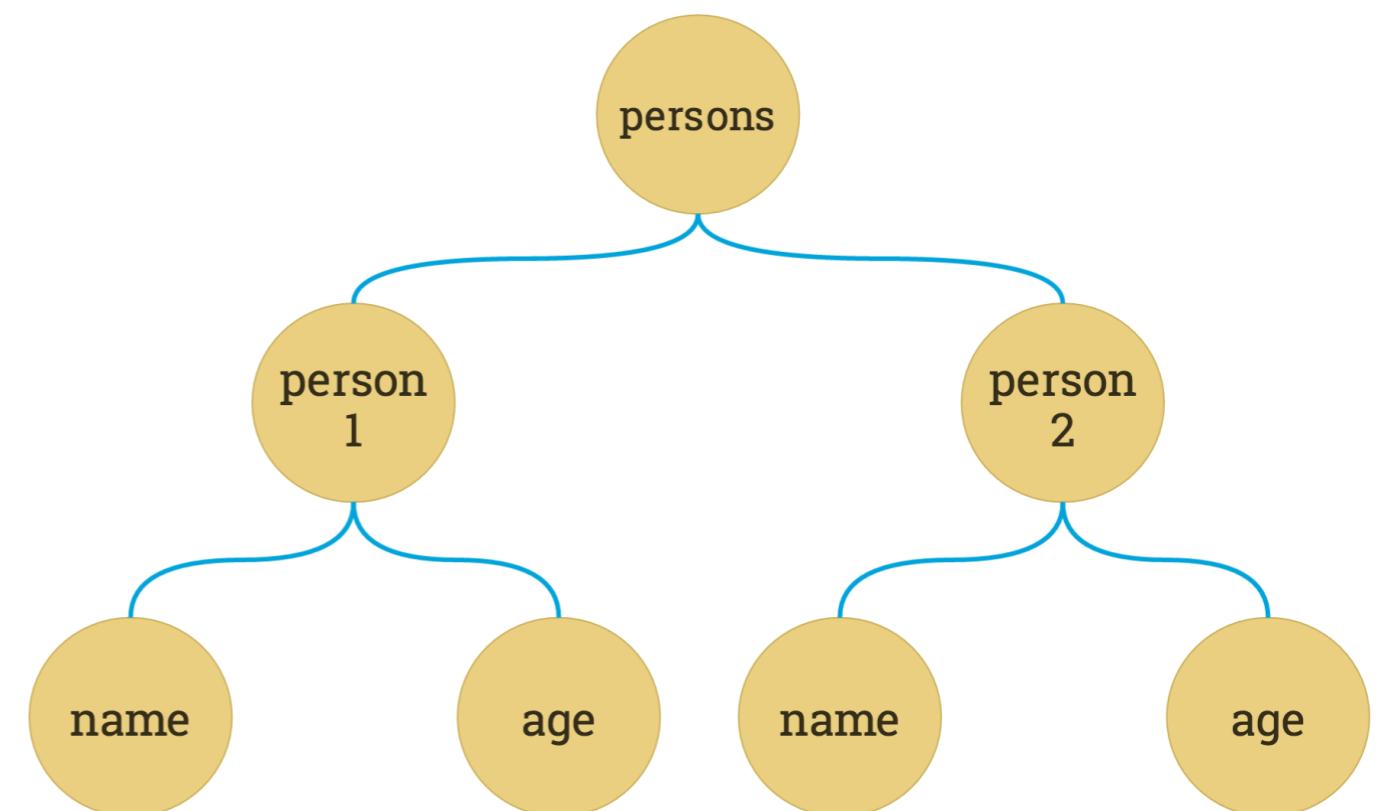


# XML

*extensible markup language*

```
<persons>
  <person id="1">
    <name>John</name>
    <age>26</age>
  </person>

  <person id="2">
    <name>Jane</name>
    <age>24</age>
  </person>
</persons>
```



# HTML

*HyperText Markup Language*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Привет, Moscow Coding School! </title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1> Заголовок </h2>
    
  </body>
</html>
```

# HTML

*HyperText Markup Language*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Привет, Moscow Coding School! </title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1> Заголовок </h2>
    
  </body>
</html>
```

**Тэг, элемент, узел, node**

# HTML

*HyperText Markup Language*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Привет, Moscow Coding School! </title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1> Заголовок </h2>
    
  </body>
</html>
```

**Тэг, element, узел, node**

# HTML

*HyperText Markup Language*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Привет, Moscow Coding School! </title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1> Заголовок </h2>
    
  </body>
</html>
```

**Тэг, element, узел, node**

# HTML

*HyperText Markup Language*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Привет, Moscow Coding School! </title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1> Заголовок </h2>
    
  </body>
</html>
```

**Тэг, element, узел, node**

# HTML

*HyperText Markup Language*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Привет, Moscow Coding School! </title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1> Заголовок </h2>
    
  </body>
</html>
```

**Атрибут (имя)**

# HTML

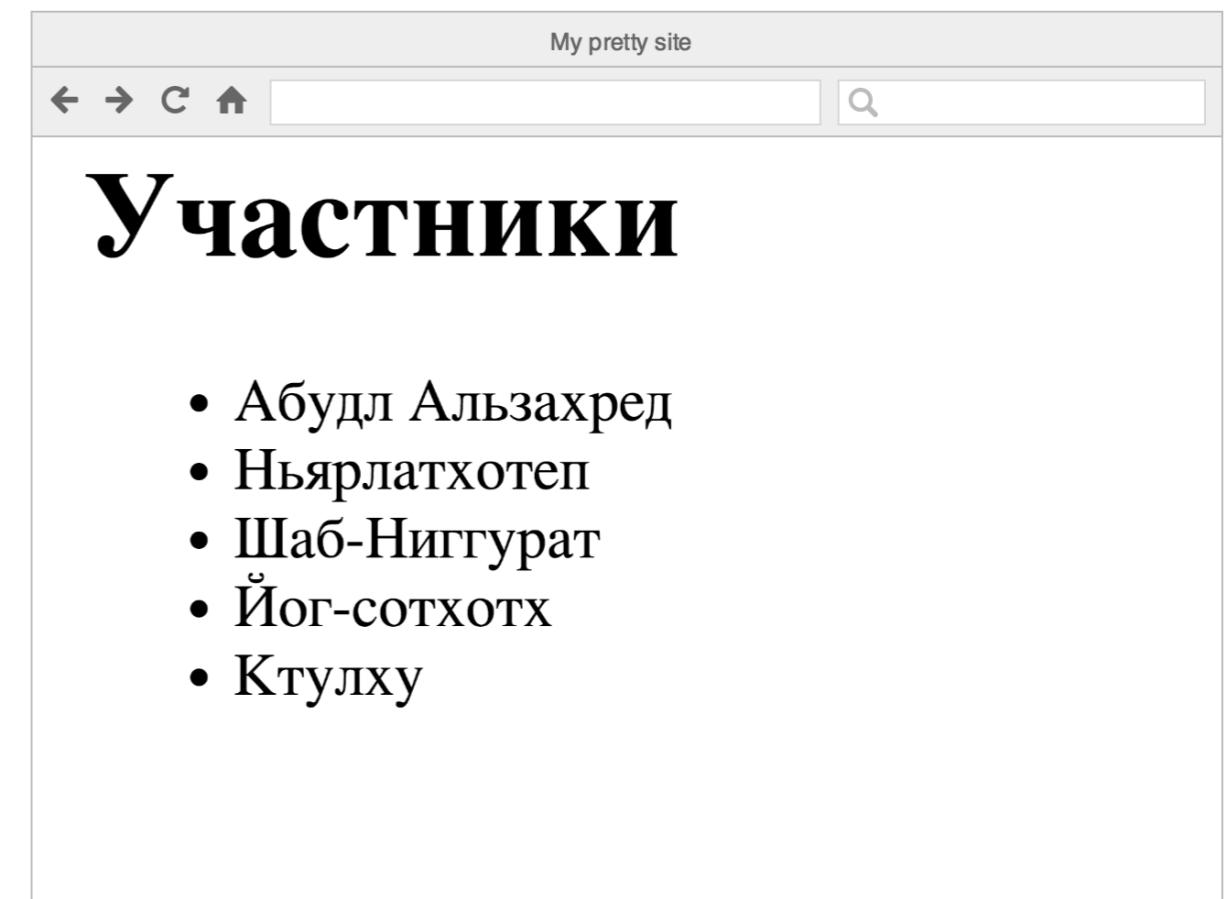
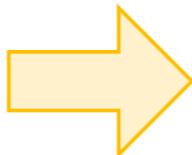
*HyperText Markup Language*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title> Привет, Moscow Coding School! </title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1> Заголовок </h2>
    
  </body>
</html>
```

**Атрибут (значение)**

# HTML - это иерархия документа

```
<h1>Участники</h1>
<ul>
  <li>Абдул Альзахред</li>
  <li>Нъярлатхотеп</li>
  <li>Шаб-Ниггурат</li>
  <li>Йог-сотхотх</li>
  <li>Ктулху</li>
</ul>
```



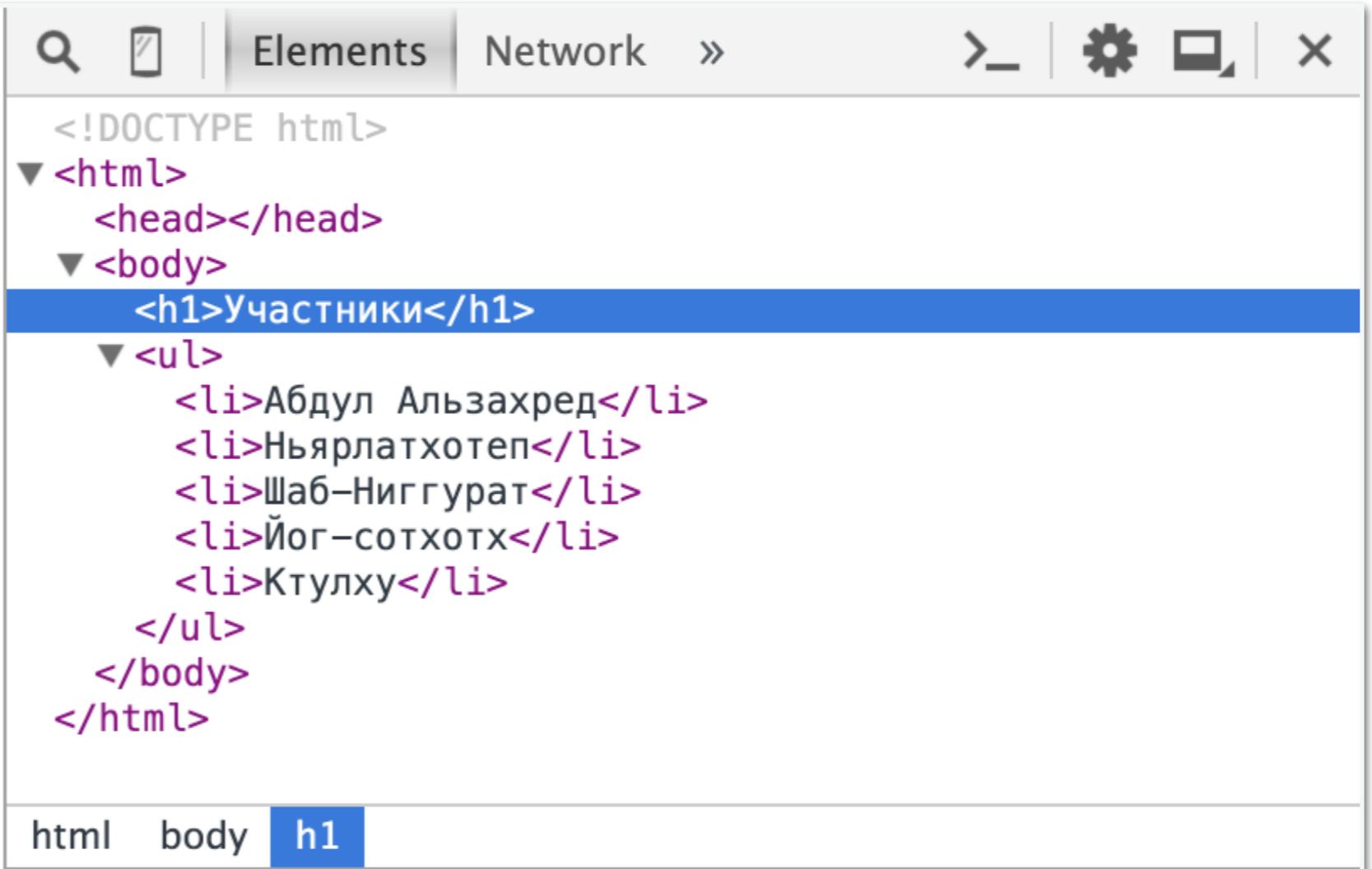
# Объектная модель документа

```
01: <!DOCTYPE html>
02: <html>
03:   <head>
04:     <title> Привет, Moscow Coding School! </title>
06:   </head>
07:   <body>
08:     <h1>Участники</h2>
09:     <ul>
10:       <li>Абдул Альзахред</li>
11:       <li>Нъярлатхотеп</li>
12:       <li>Шаб-Ниггурат</li>
13:       <li>Йог-сотхотх</li>
14:       <li>Ктулху</li>
15:     </ul>
16:   </body>
17: </html>
```

# Объектная модель документа

## Участники

- Абдул Альзахред
- Ньярлатхотеп
- Шаб-Ниггурат
- Йог-сотхотх
- Ктулху



The screenshot shows the 'Elements' tab of a browser's developer tools. The DOM tree is displayed, starting with the document type declaration <!DOCTYPE html>. Below it is the <html> element, which contains the <head> and <body> elements. The <body> element contains an <h1> element with the text 'Участники'. This <h1> element is highlighted with a blue selection bar. Below it is a <ul> element, which contains five <li> elements, each listing one of the participants from the bullet list on the left. The <ul> element is also highlighted with a blue selection bar. At the bottom of the developer tools interface, there is a navigation bar with tabs labeled 'html', 'body', and 'h1', where 'h1' is currently selected.

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>Участники</h1>
    <ul>
      <li>Абдул Альзахред</li>
      <li>Ньярлатхотеп</li>
      <li>Шаб-Ниггурат</li>
      <li>Йог-сотхотх</li>
      <li>Ктулху</li>
    </ul>
  </body>
</html>
```

```
01: <!DOCTYPE html>
02: <html>
03:   <head>
04:     <title> Привет, Moscow Coding School! </title>
06:   </head>
07:   <body>
08:     <h1 class="heading main-heading">Участники</h2>
09:     <ul id="main-list">
10:       <li>Абдул Альзахред</li>
11:       <li>Нъярлатхотеп</li>
12:       <li>Шаб-Ниггурат</li>
13:       <li>Йог-сотхотх</li>
14:       <li>Ктулху</li>
15:     </ul>
16:   </body>
17: </html>
```

```
01: <!DOCTYPE html>
02: <html>
03:   <head>
04:     <title> Привет, Moscow Coding School! </title>
06:   </head>
07:   <body>
08:     <h1 class="heading main-heading">Участники</h2>
09:     <ul id="main-list">
10:       <li>Абдул Альзахред</li>
11:       <li>Нъярлатхотеп</li>
12:       <li>Шаб-Ниггурат</li>
13:       <li>Йог-сотхотх</li>
14:       <li>Ктулху</li>
15:     </ul>
16:   </body>
17: </html>
```

The screenshot shows the Chrome DevTools Elements tab interface. At the top, there is a toolbar with icons for selecting elements, zooming, and other tools. Below the toolbar is a status bar with the text "Select an element in the page to inspect it". The main area displays the DOM tree:

```
<!DOCTYPE html>
▼ <html>
  ▶ <head>...</head>
  ▼ <body style="height: 619px;">
    <h1 class="heading main-heading">Участники</h1>
    ▼ <ul id="main-list">
      <li>Абдул Альзахред</li>
      <li>Нъярлатхотеп</li>
      <li>Шаб-Ниггурат</li>
      <li>Йог-сотхотх</li>
      <li>Ктулху</li>
    </ul>
  </body>
</html>
```

The element `<h1 class="heading main-heading">Участники</h1>` is selected and highlighted with a blue background. In the bottom navigation bar, the path `html body h1.heading.main-heading` is shown, with the last part "h1.heading.main-heading" highlighted in blue.

# Визуализация

только HTML, только хардкор

<b>Table Heading Cell</b>	<b>Table Heading Cell</b>
Table Body Cell	Table Body Cell
Table Body Cell	Table Body Cell

```
01: <table>
02:   <thead>
03:     <tr>
04:       <th>Table Heading Cell</th>
05:       <th>Table Heading Cell</th>
06:     </tr>
07:   </thead>
08:   <tbody>
09:     <tr>
10:       <td>Table Body Cell</td>
11:       <td>Table Body Cell</td>
12:     </tr>
13:     <tr>
14:       <td>Table Body Cell</td>
15:       <td>Table Body Cell</td>
16:     </tr>
17:   </tbody>
18: </table>
```

<table>

Table Heading Cell	Table Heading Cell
Table Body Cell	Table Body Cell
Table Body Cell	Table Body Cell

# <thead>

The diagram illustrates a table structure with the following characteristics:

- Header Row:** The first row is highlighted with a large red rounded rectangle spanning the width of the table. It contains two **Table Heading Cells**, both labeled "Table Heading Cell".
- Body Rows:** There are three additional rows below the header. The second row contains two **Table Body Cells**, both labeled "Table Body Cell". The third row also contains two **Table Body Cells**, both labeled "Table Body Cell".
- Cell Content:** All cells contain the text "Table Body Cell" or "Table Heading Cell" centered within them.
- Table Structure:** The table is defined by a dark blue border around the entire structure, and internal grid lines divide it into four columns and four rows.

Table Heading Cell	Table Heading Cell
Table Body Cell	Table Body Cell
Table Body Cell	Table Body Cell

<tbody>

Table Heading Cell	Table Heading Cell	<tr>
Table Body Cell	Table Body Cell	<tr>
Table Body Cell	Table Body Cell	<tr>

<p><b>Table Heading Cell</b></p> <p><b>&lt;th&gt;</b></p>	<p><b>Table Heading Cell</b></p> <p><b>&lt;th&gt;</b></p>
<p>Table Body Cell</p> <p><b>&lt;td&gt;</b></p>	<p>Table Body Cell</p> <p><b>&lt;td&gt;</b></p>
<p>Table Body Cell</p> <p><b>&lt;td&gt;</b></p>	<p>Table Body Cell</p> <p><b>&lt;td&gt;</b></p>

<b>Table Heading Cell</b>	<b>Table Heading Cell</b>
Table Body Cell	Table Body Cell
Table Body Cell	Table Body Cell

```
01: <table border="1" cellpadding="10">
02:   <thead>
03:     <tr>
04:       <th>Table Heading Cell</th>
05:       <th>Table Heading Cell</th>
06:     </tr>
07:   </thead>
08:   <tbody>
09:     <tr>
10:       <td>Table Body Cell</td>
11:       <td>Table Body Cell</td>
12:     </tr>
13:     <tr>
14:       <td>Table Body Cell</td>
15:       <td>Table Body Cell</td>
16:     </tr>
17:   </tbody>
18: </table>
```

<b>Table Heading Cell</b>	<b>Table Heading Cell</b>
Table Body Cell	Table Body Cell
Table Body Cell	Table Body Cell

# SVG

вообще-то

**SVG ∈ XML**

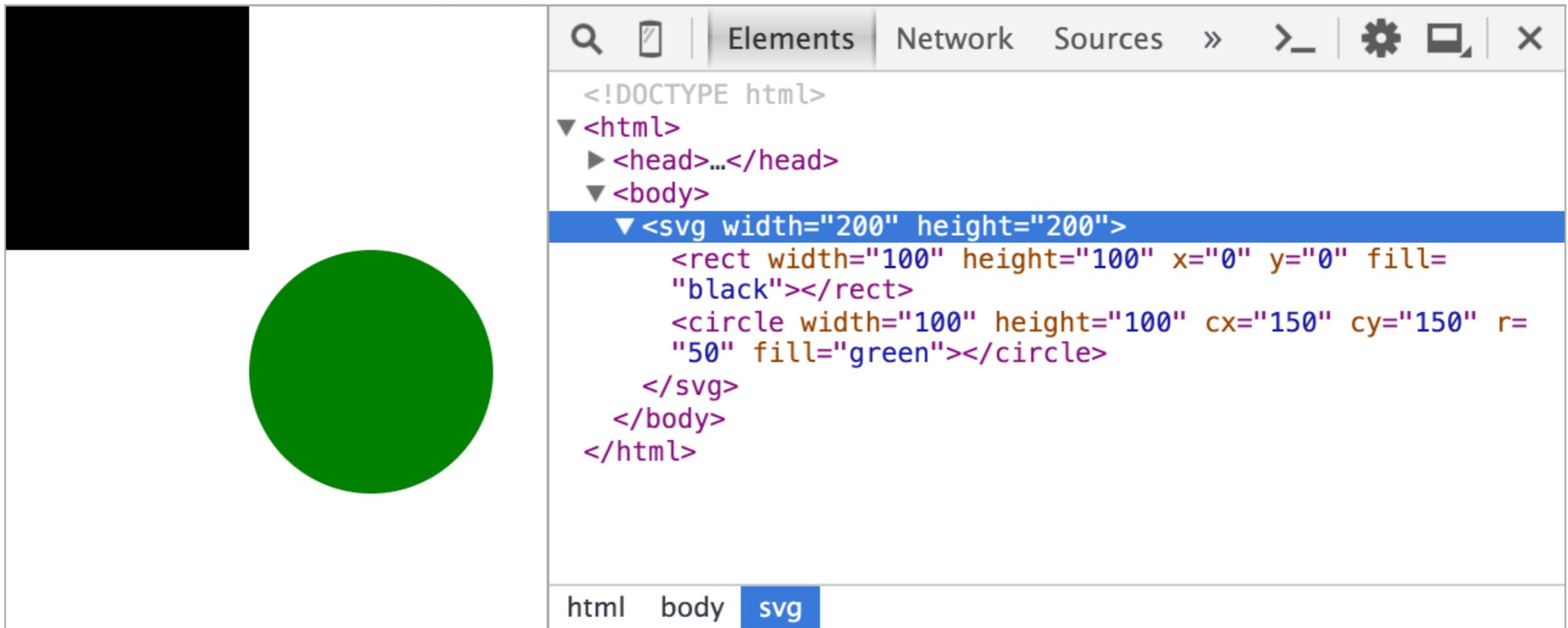
# SVG

## *Scalable Vector Graphic*

```
01: <svg width="200" height="200">
02:   <rect width="100" height="100"
03:     x="0" y="0" fill="black" />
04:
05:   <circle cx="150" cy="150" r="50" fill="green" />
06: </svg>
```

# SVG

*Scalable Vector Graphic*

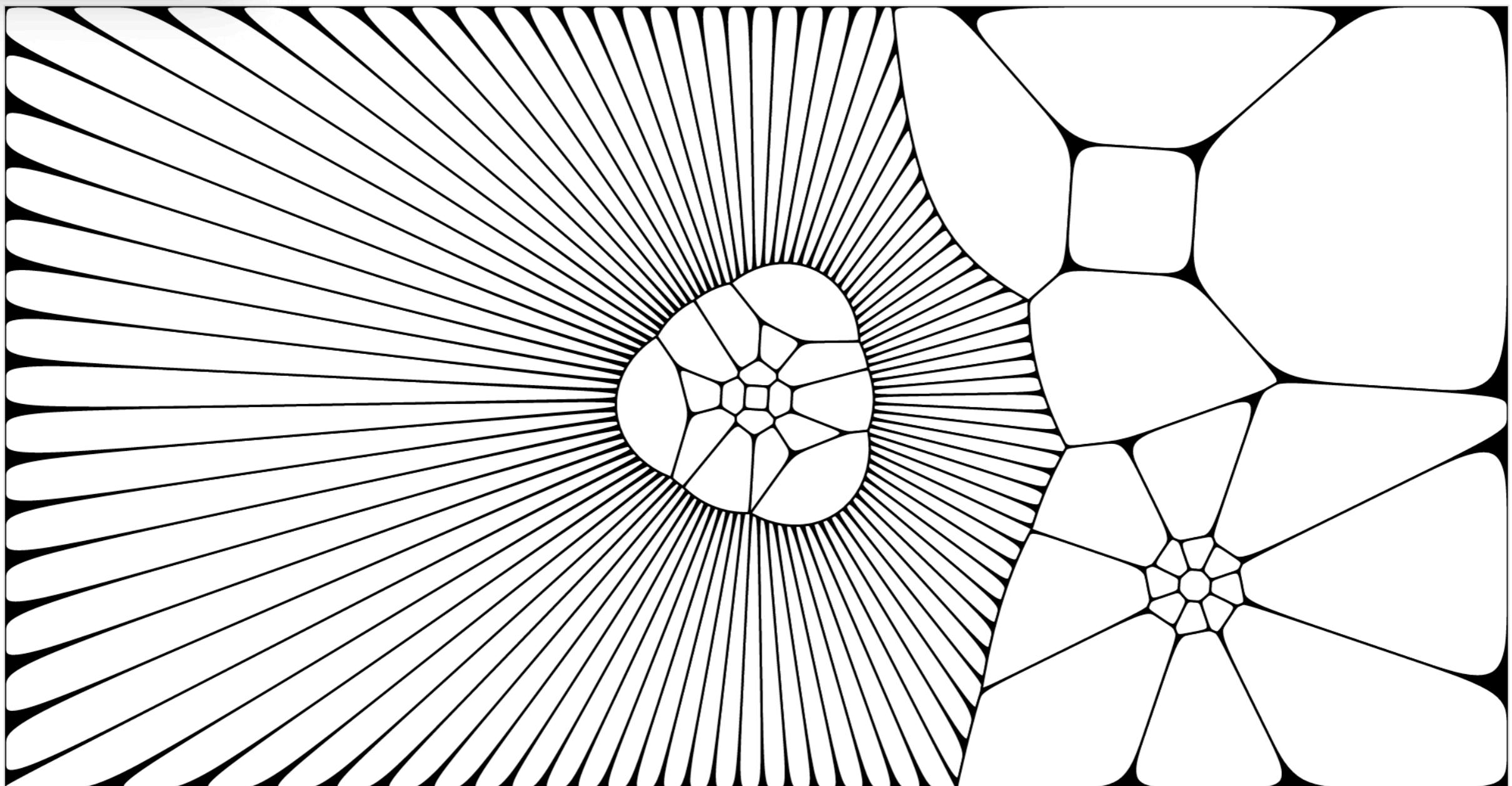


```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <svg width="200" height="200">
      <rect width="100" height="100" x="0" y="0" fill="black"></rect>
      <circle width="100" height="100" cx="150" cy="150" r="50" fill="green"></circle>
    </svg>
  </body>
</html>
```

The screenshot shows a browser window with developer tools open. On the left, there is a black rectangular background with a large green circle centered on it. In the developer tools' Elements tab, the DOM structure is displayed. The root node is an HTML element. It contains a head and body section. The body section contains an SVG element with attributes width="200" and height="200". Inside the SVG, there is a rect element with attributes width="100", height="100", x="0", y="0", and fill="black". There is also a circle element with attributes width="100", height="100", cx="150", cy="150", r="50", and fill="green". The 'Elements' tab is selected in the toolbar at the top. The bottom of the developer tools shows tabs for 'html', 'body', and 'svg', with 'svg' being the active tab.

# SVG

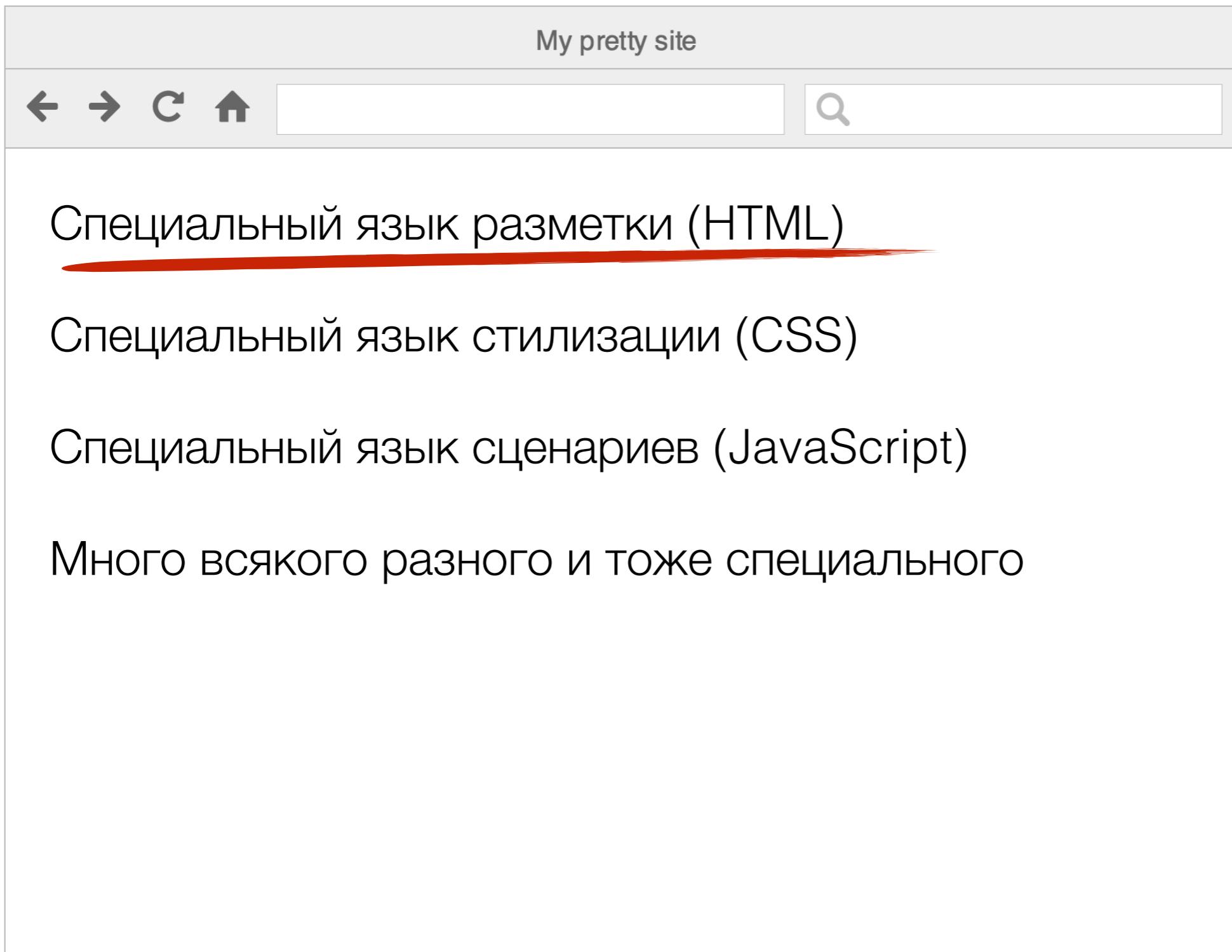
*Scalable Vector Graphic*



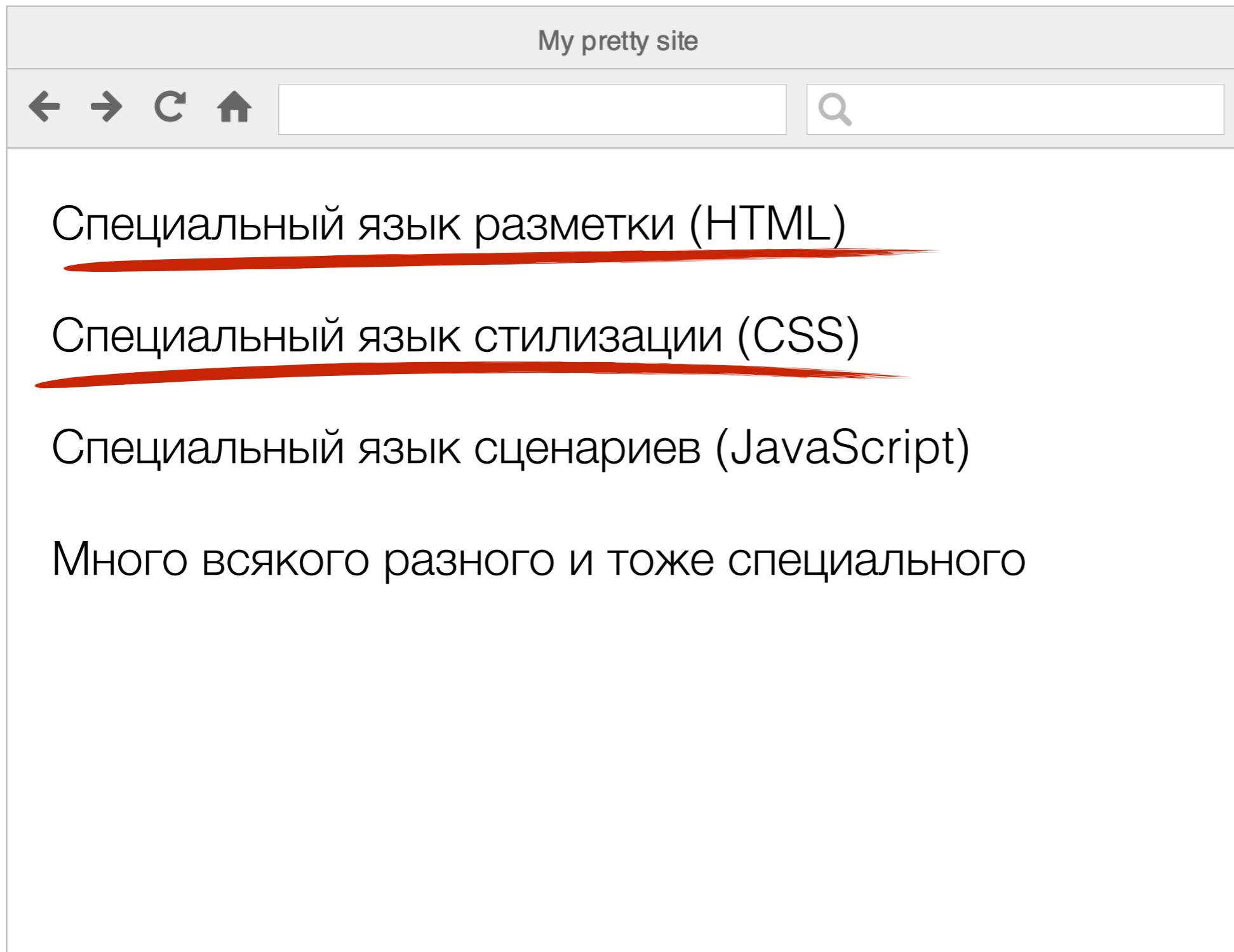
# Links

- [W3C SVG Specification](#)
- [A Primer to Front-end SVG Hacking](#)
- [Mozilla Developers Network \(SVG\)](#)
- [Interactive Data Visualization for the Web](#)

# Как работает браузер ?



# Как работает браузер ?



# CSS

## *Cascading Style Sheets*

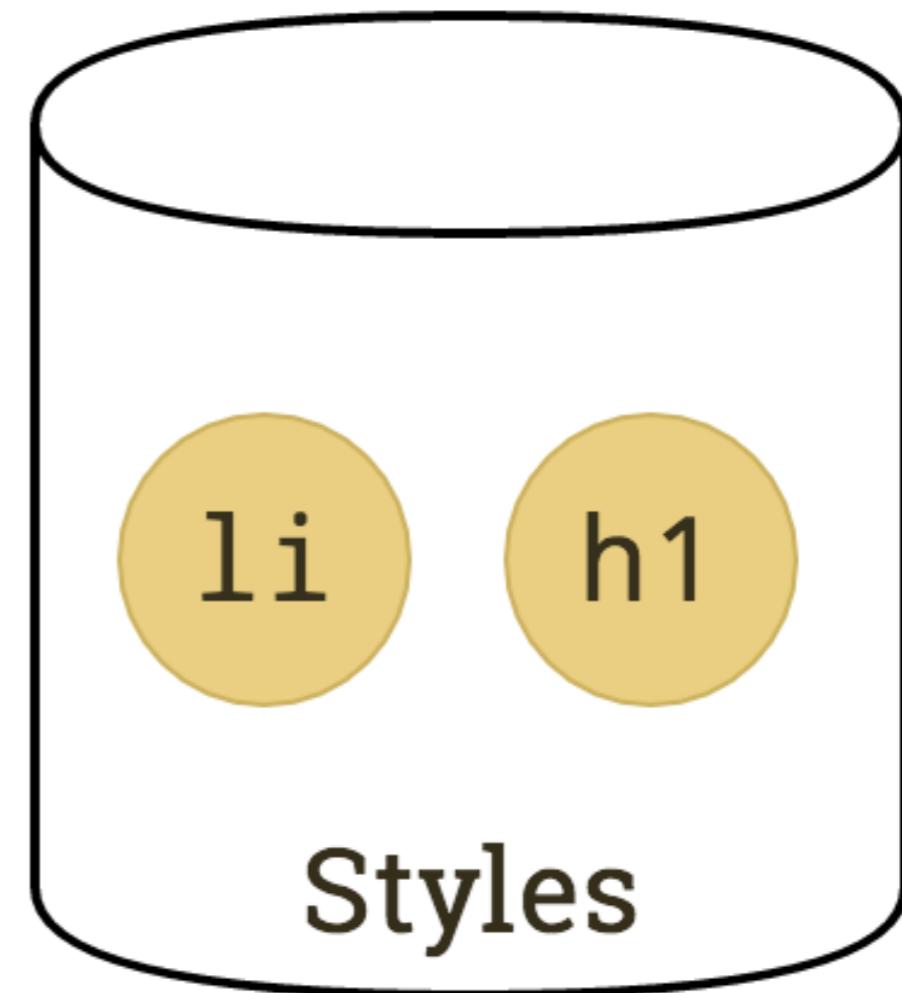
```
01: h1 {  
02:   color: #000000;  
03:   font-size: 18px;  
04:   font-weight: bold;  
05:   text-decoration: underline;  
06: }  
07:  
08: li {  
09:   color: blue;  
10:   font-family: Arial, sans-serif;  
11: }
```

# CSS

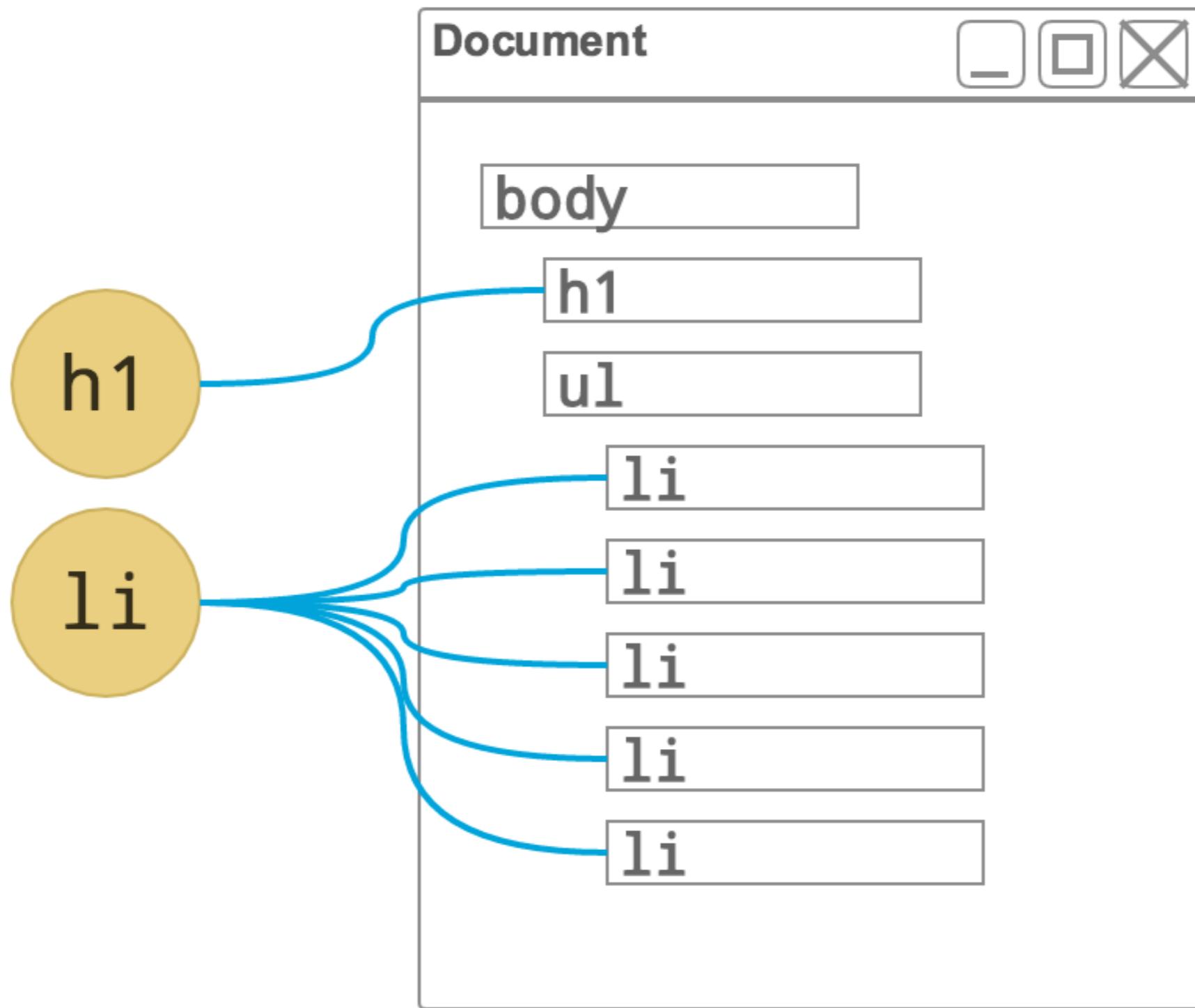
## *Cascading Style Sheets*

styles.css

```
01: h1 {  
02: ...  
03: }  
04:  
05: li {  
06: ...  
07: }
```



# Style matching



# Участники

- Абдул Альзахред
- Нъярлатхотеп
- Шаб-Ниггурат
- Йог-сотхотх
- Ктулху

The screenshot shows the browser's developer tools open to the 'Elements' tab. The DOM tree is visible, with the `<h1>Участники</h1>` element highlighted in blue. Below the tree, there are tabs for Styles, Event Listeners, DOM, Breakpoints, and Properties. Under the Styles tab, two CSS rules for the `h1` selector are listed:

```
h1 { inspector-stylesheet:1
      color: #000000;
      font-size: 18px;
      font-weight: bold;
      text-decoration: underline;
}
h1 { user agent stylesheet
      display: block;
```

# Синтаксис

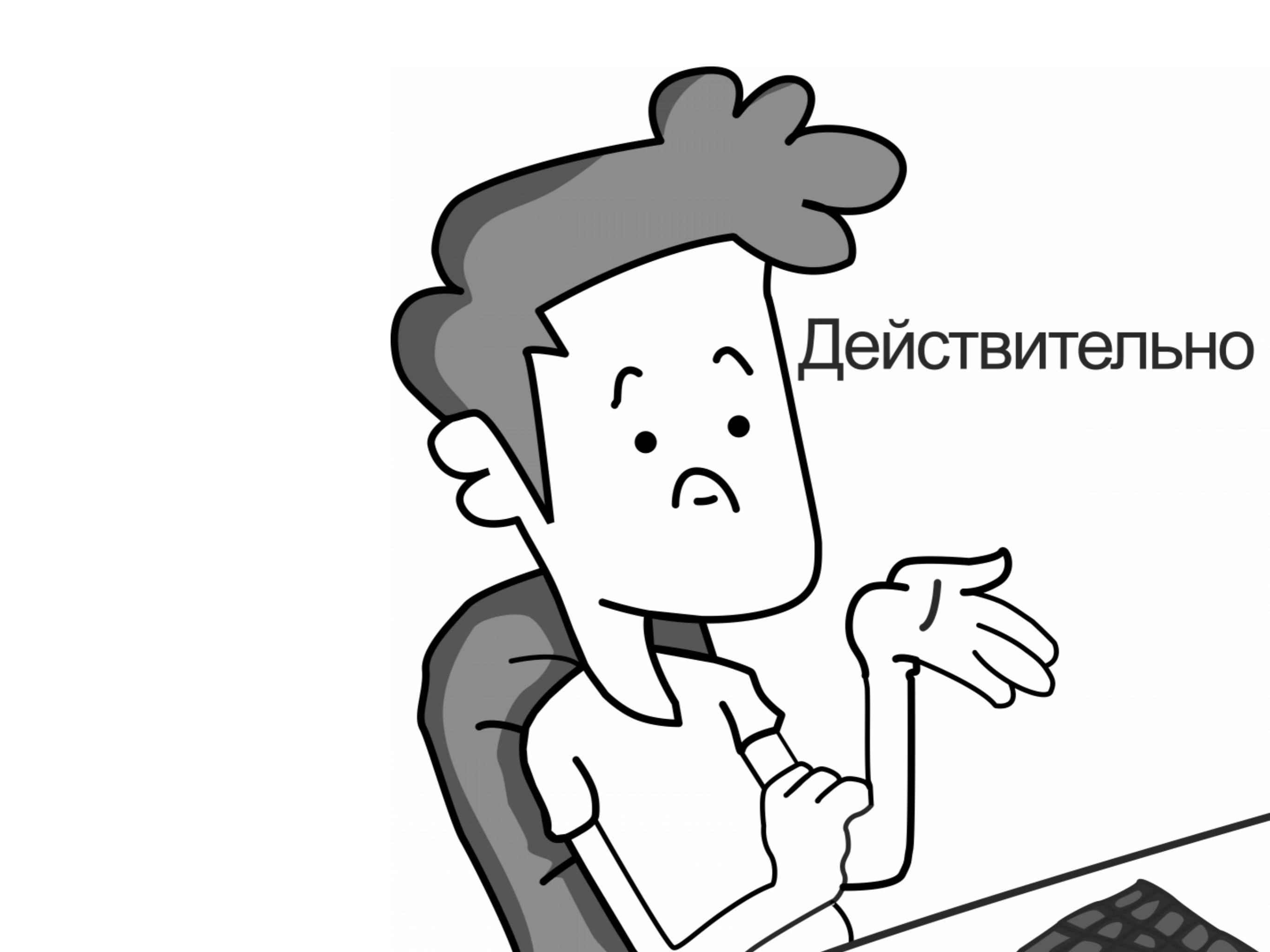
Ничего хитрого

```
01: селектор {  
02:     правило: значение;  
03: }
```

# Селекторы

# Селекторы

селекторы - выбирают

A black and white cartoon illustration of a woman with dark, curly hair tied back with a large bow. She has a shocked or surprised expression, indicated by wide eyes, a small mouth, and a question mark above her head. She is wearing a dark top and a necklace. Her hands are clasped together in front of her. To the right of the character, the word "Действительно" is written in a bold, sans-serif font.

Действительно

```
input#user.classOne.classTwo[type="text"] {  
    ...  
}
```

```
input#user.classOne.classTwo[type="text"] {  
    ...  
}
```

- Тэг - `input`
- Идентификатор - `user`
- Классы - `classOne` и `classTwo`
- Атрибут `type` в значении `text`

```
input#user.classOne.classTwo[type="text"] {  
    ...  
}
```

- Тэг - **input**
- Идентификатор - user
- Классы - classOne и classTwo
- Атрибут type в значении text

```
input#user.classOne.classTwo[type="text"] {  
    ...  
}
```

- Тэг - input
- Идентификатор - user
- Классы - classOne и classTwo
- Атрибут type в значении text

```
input#user.classOne.classTwo[type="text"] {  
    ...  
}
```

- Тэг - input
- Идентификатор - user
- Классы - **classOne** и **classTwo**
- Атрибут type в значении text

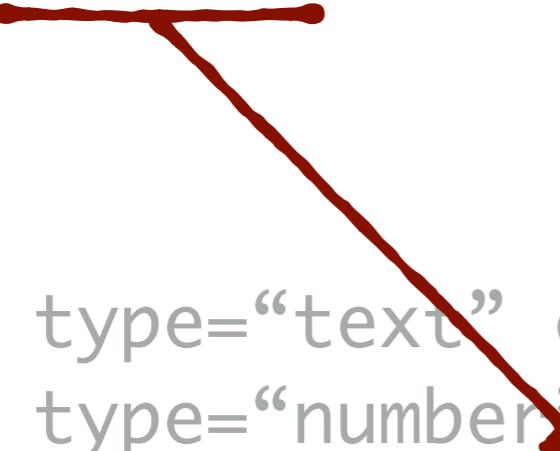
```
input#user.classOne.classTwo[type="text"] {  
    ...  
}
```

- Тэг - `input`
- Идентификатор - `user`
- Классы - `classOne` и `classTwo`
- Атрибут `type` в значении `text`

input#user.classOne.classTwo[type="text"]

```
01: <input type="text" class="classOne"></input>
02: <input type="number" i="age" class="classOne"></input>
03: <input type="text" id="user"
         class="classOne classTwo"></input>
04: <input type="text" class="classOne"></input>
05: <input type="text" name="first-name"></input>
06: <input class="classOne" placeholder="Hello"></input>
```

`input#user.classOne.classTwo[type="text"]`



```
01: <input type="text" class="classOne"></input>
02: <input type="number" i="age" class="classOne"></input>
03: <input type="text" id="user"
         class="classOne classTwo"></input>
04: <input type="text" class="classOne"></input>
05: <input type="text" name="first-name"></input>
06: <input class="classOne" placeholder="Hello"></input>
```

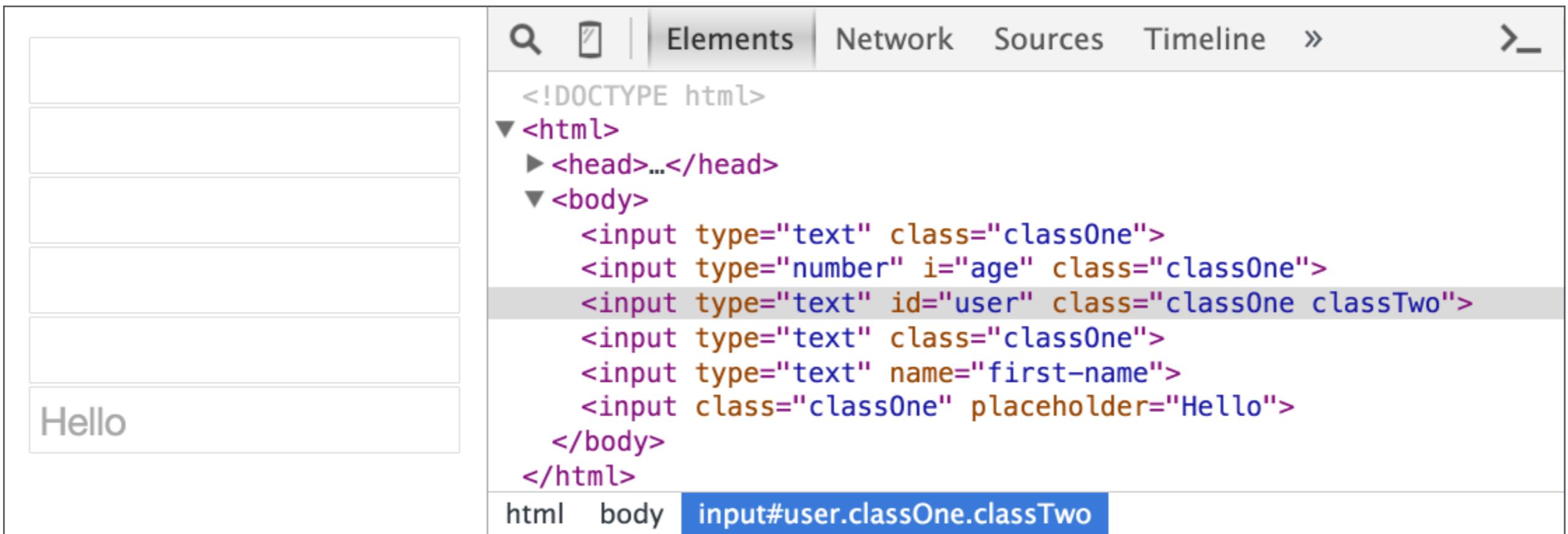
`input#user.classOne.classTwo[type="text"]`

```
01: <input type="text" class="classOne"></input>
02: <input type="number" id="age" class="classOne"></input>
03: <input type="text" id="user"
      class="classOne classTwo"></input>
04: <input type="text" class="classOne"></input>
05: <input type="text" name="first-name"></input>
06: <input class="classOne" placeholder="Hello"></input>
```

`input#user.classOne.classTwo[type="text"]`

```
01: <input type="text" class="classOne"></input>
02: <input type="number" id="age" class="classOne"></input>
03: <input type="text" id="user"
      class="classOne classTwo"></input>
04: <input type="text" class="classOne"></input>
05: <input type="text" name="first-name"></input>
06: <input class="classOne" placeholder="Hello"></input>
```

```
input#user.classOne.classTwo[type="text"]
```

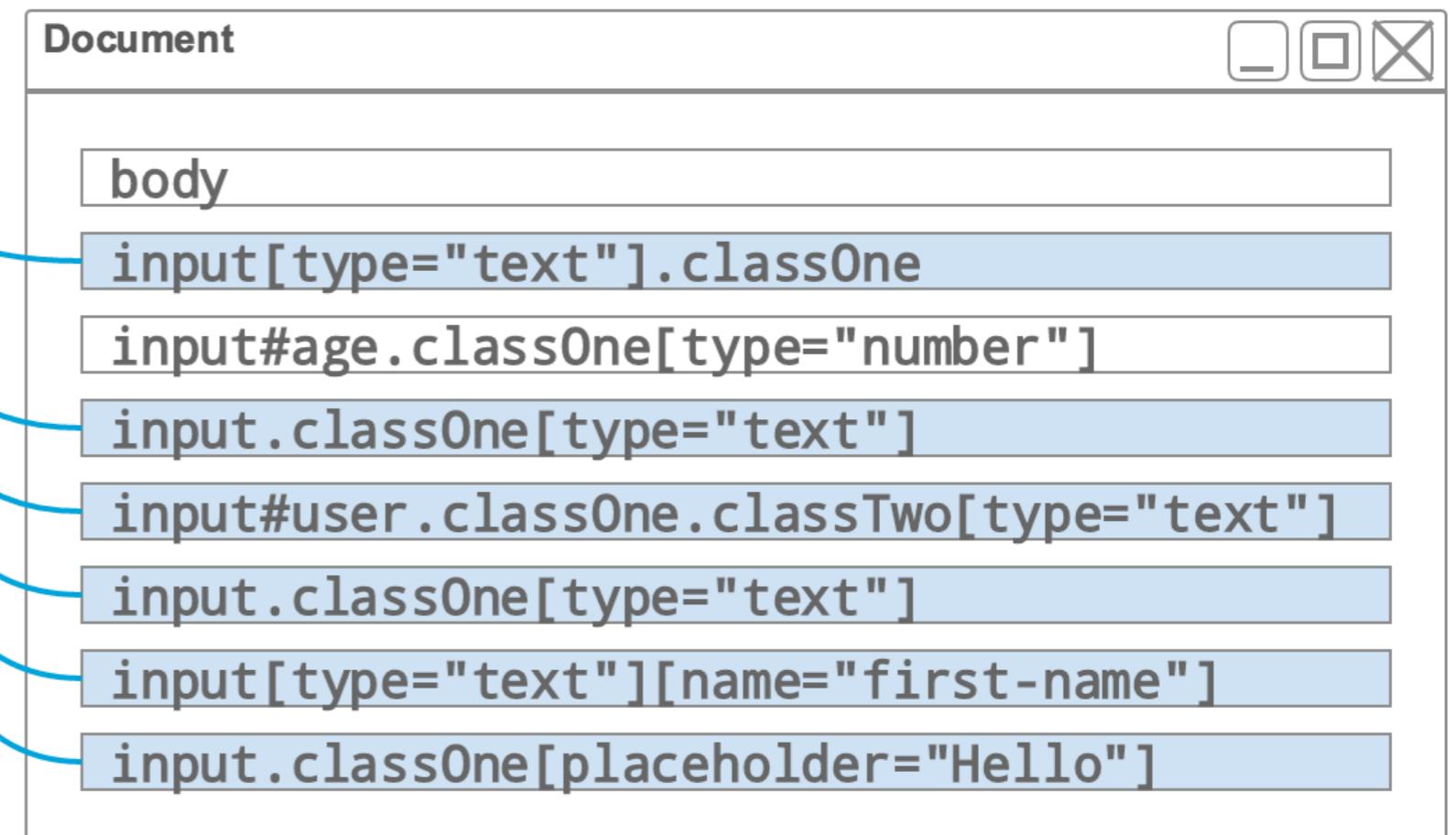


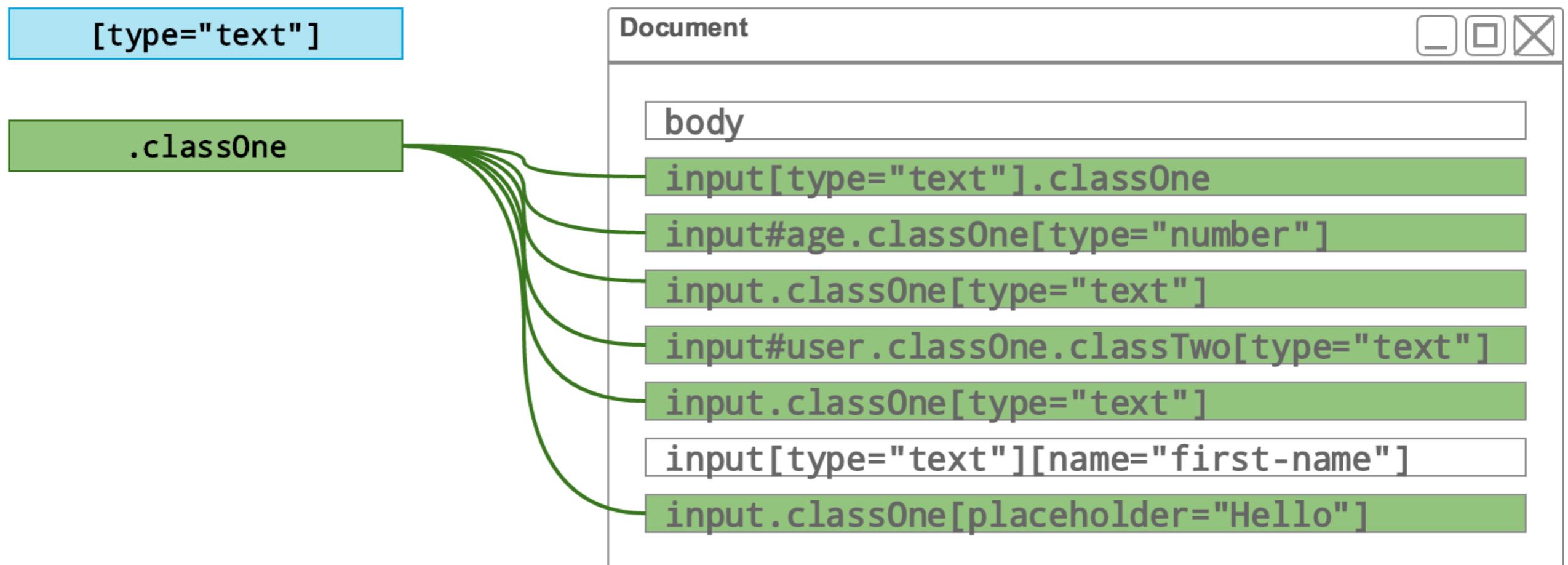
The screenshot shows a browser's developer tools with the "Elements" tab selected. On the left, there is a visual representation of the page containing several input fields. One input field in the middle has the value "Hello". On the right, the DOM tree is displayed:

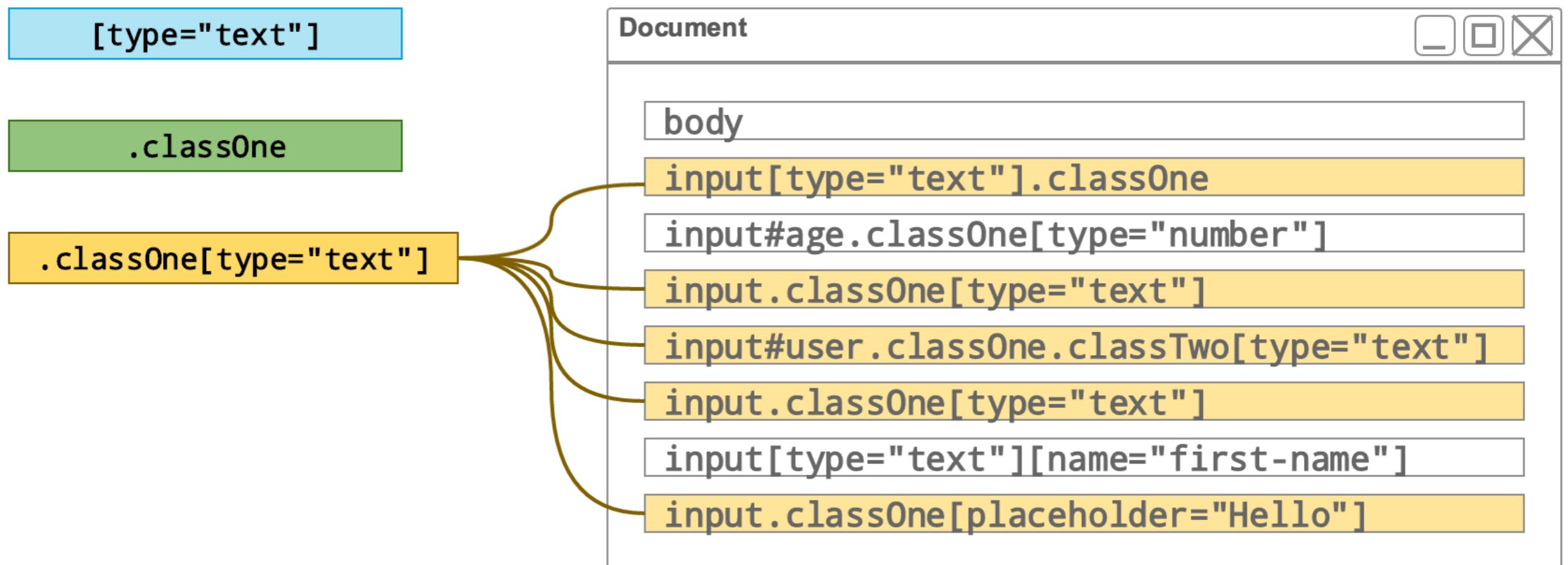
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <input type="text" class="classOne">
    <input type="number" i="age" class="classOne">
    <input type="text" id="user" class="classOne classTwo"> // This node is highlighted with a gray background
    <input type="text" class="classOne">
    <input type="text" name="first-name">
    <input class="classOne" placeholder="Hello">
  </body>
</html>
```

The bottom status bar shows the path: html > body > input#user.classOne.classTwo.

[type="text"]







# Каскадирование

```
body nav ul li.active a
```

# Каскадирование

body nav ul li.active a



# Каскадирование

body nav ul li.active a



Все элементы <a>

# Каскадирование

body nav ul li.active a



Все элементы `<a>`, которые находятся внутри элемента `<li>` с классом `active`

# Каскадирование

```
body nav ul li.active a  
          ←
```

Все элементы `<a>`, которые находятся внутри элемента `<li>` с классом `active`, который находится внутри элемента `<ul>`, который находится в элементе `<nav>`, который находится внутри элемента `<body>`

# Каскадирование

body nav ul li.active a



Все элементы `<a>`, которые находятся внутри элемента `<li>` с классом `active`, который находится внутри элемента `<ul>`, который находится в элементе `<nav>`, который находится внутри элемента `<body>`,

который построил Джек

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

\*

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div.left

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div p

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div p

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div.left p

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div.left p

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

.title

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

.title

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

.left .title

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

.left .title

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div:first-child

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div:first-child

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

.left p:first-child

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

.left p:first-child

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div p:first-child

# <body>

<div.left>

<p.title>

<p> <span>

<p>

<p>

<div.right>

<p.title>

<p>

<p>

<p>

<footer>

div p:first-child

# Правила

вообще-то это важно

# 1. Переопределение

```
01: h1 {  
02:   color: #000000;  
03:   font-size: 18px;  
04:   font-weight: bold;  
05: }
```



Heading

# 1. Переопределение

```
01: h1 {  
02:   color: #000000;  
03:   font-size: 18px;  
04:   font-weight: bold;  
05: }
```



Heading

```
01: h1 {  
02:   color: blue;  
03: }
```

# 1. Переопределение

```
01: h1 {  
02:   color: #000000;  
03:   font-size: 18px;  
04:   font-weight: bold;  
05: }
```



→ Heading

```
01: h1 {  
02:   color: blue;  
03: }
```

# 1. Переопределение

```
01: h1 {  
02:   color: #000000;  
03:   font-size: 18px;  
04:   font-weight: bold;  
05: }
```

A diagram illustrating CSS inheritance. A red arrow points from the 'color' declaration in the first code block to the word 'Heading' on the right. Another red arrow points from the 'color' declaration in the third code block to the same 'Heading' text.

→ Heading

```
01: h1 {  
02:   color: blue !important;  
03: }
```

```
01: h1 {  
02:   color: red;  
03: }
```

## 2. Специфичность

```
01: #h1Id {  
02:   color: #000000;  
03: }
```



Heading

# 2. Специфичность

```
01: #h1Id {  
02:   color: #000000;  
03: }
```

...

```
01: h1 {  
02:   color: blue;  
03: }
```



Heading

# 2. Специфичность

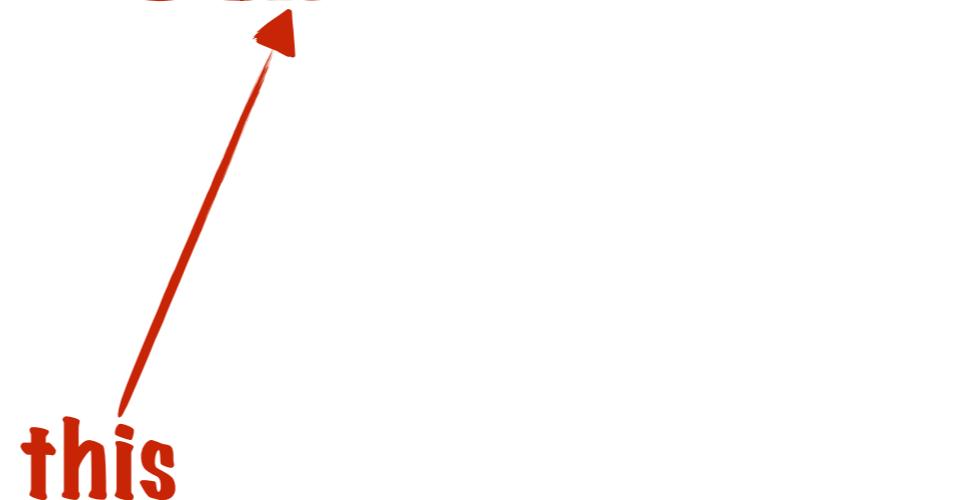
селектор	A	B	C	Специфичность
*	0	0	0	0
div	0	0	1	1
div:first	0	0	2	2
div span	0	0	2	2
div.head span	0	1	2	12
.head.title	0	2	0	20
#nav	1	0	0	100
#nav div#title	2	0	1	201

# Как всем этим пользоваться

```
01: <!DOCTYPE html>
02: <html>
03:   <head>
04:     <title> Привет, Moscow Coding School! </title>
05:     <link rel="stylesheet" href="cssfile.css">
06:   </head>
07:   <body>
08:     ...
09:   </body>
10: </html>
```

# Как всем этим пользоваться

```
01: <!DOCTYPE html>
02: <html>
03:   <head>
04:     <title> Привет, Moscow Coding School! </title>
05:     <link rel="stylesheet" href="cssfile.css">
06:   </head>
07:   <body>
08:     ...
09:   </body>
10: </html>
```



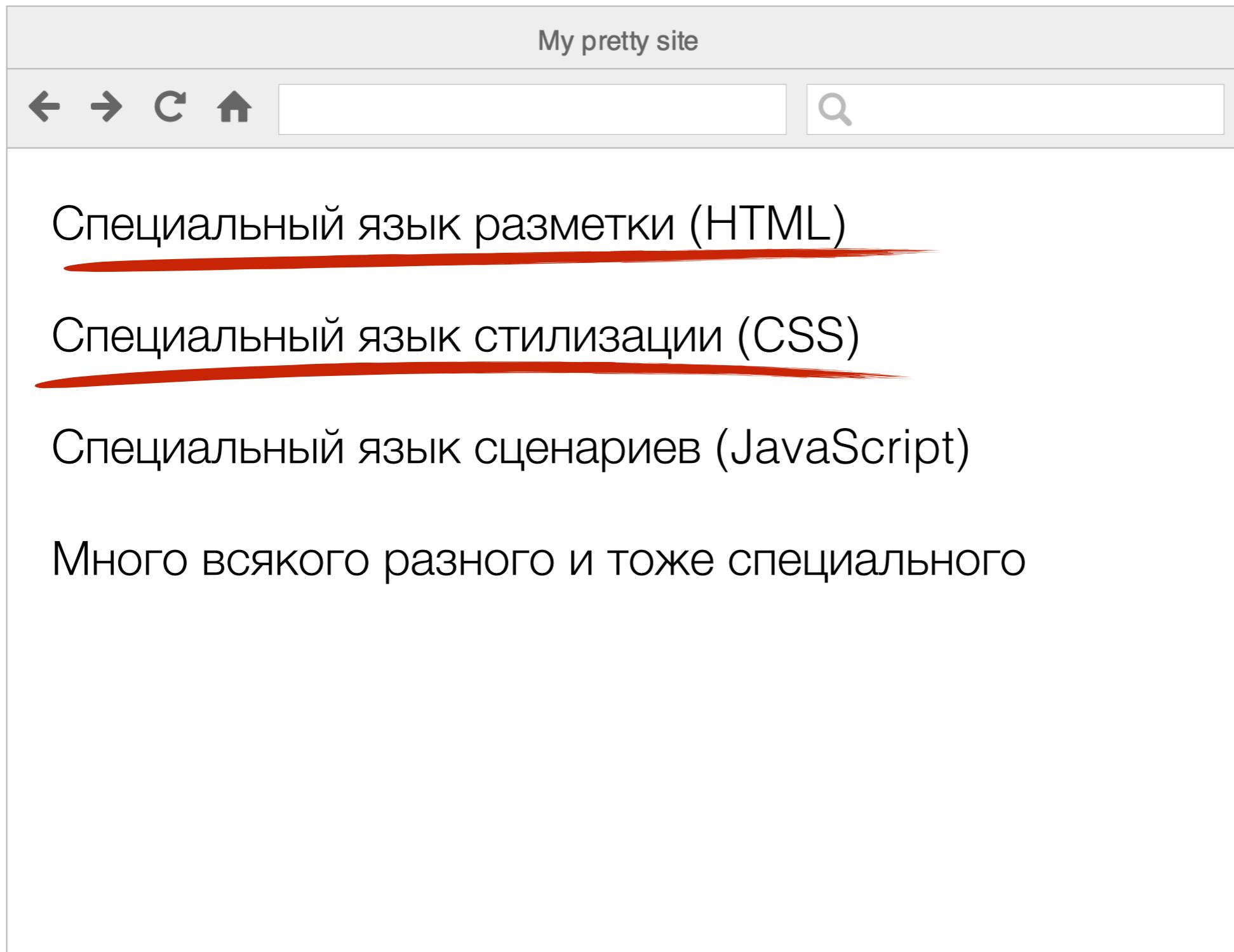
# Как всем этим пользоваться

```
01: <!DOCTYPE html>
02: <html>
03:   <head>
04:     <title>Привет, Moscow Coding School! </title>
05:     <style>
06:       h1 {
07:         color: black;
08:       }
09:     </style>
10:   </head>
11:   <body>
12:     ...
13:   </body>
14: </html>
```

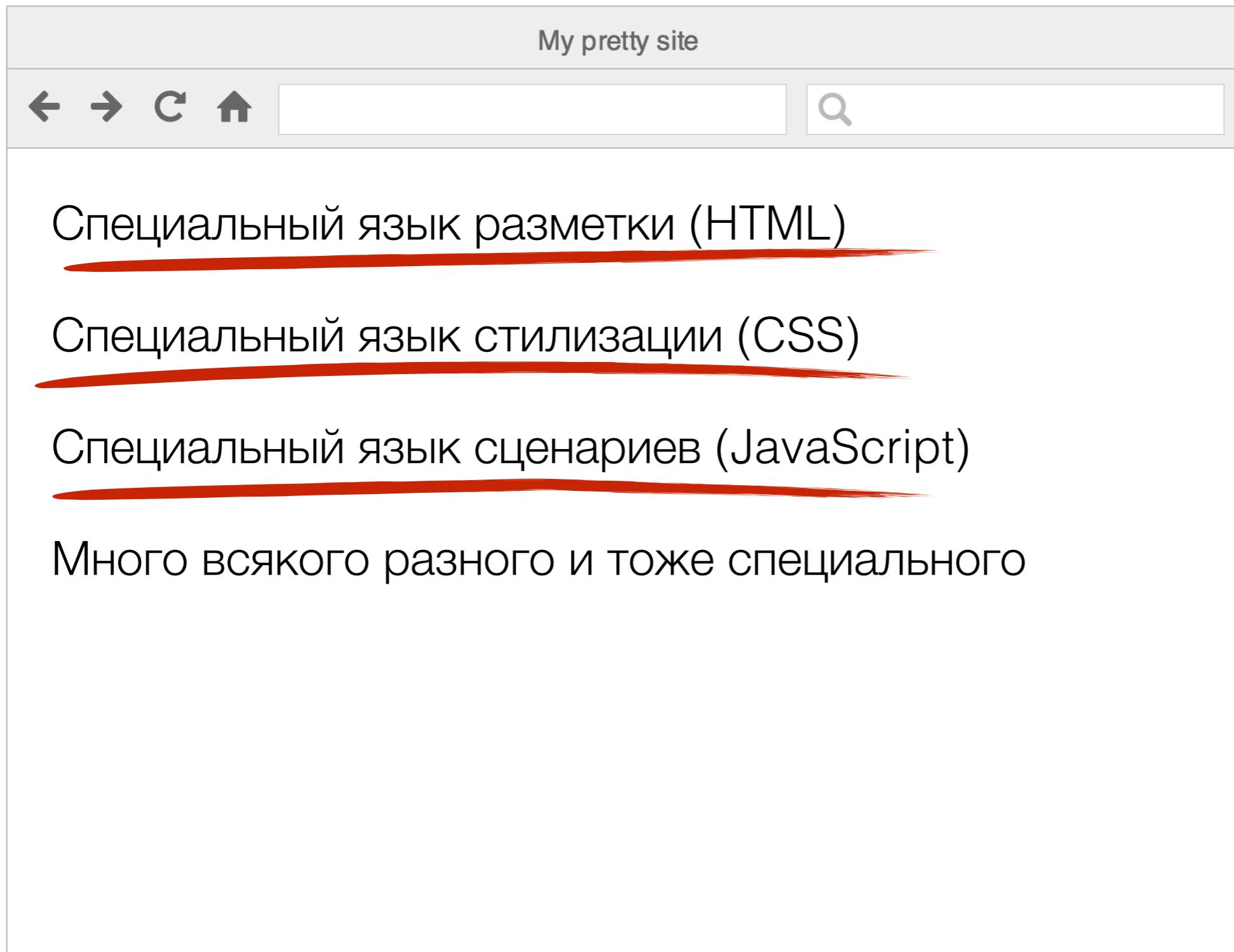
or this

<b>Table Heading Cell</b>	<b>Table Heading Cell</b>
Table Body Cell	Table Body Cell
Table Body Cell	Table Body Cell

# Как работает браузер ?



# Как работает браузер ?



JS

```
01: var a = 10;  
02: var b = 20;  
03:  
04: var c = a + b;  
05:  
06: alert(c);
```



```
01: var a = 10;  
02: var b = 20;  
03:  
04: var c = a + b;  
05:  
06: alert(c);
```

# Интерпретация

```
01: var a = 10;  
02: var b = 20;  
03:  
04: var c = a + b;  
05:  
06: alert(c);
```

# Интерпретация

```
01: var a = 10;           10
02: var b = 20;
03:
04: var c = a + b;
05:
06: alert(c);
```

# Интерпретация

```
01: var a = 10;           10
02: var b = 20;           20
03:
04: var c = a + b;
05:
06: alert(c);
```

# Интерпретация

```
01: var a = 10;           10
02: var b = 20;           20
03:
04: var c = a + b;        a+b
05:
06: alert(c);
```

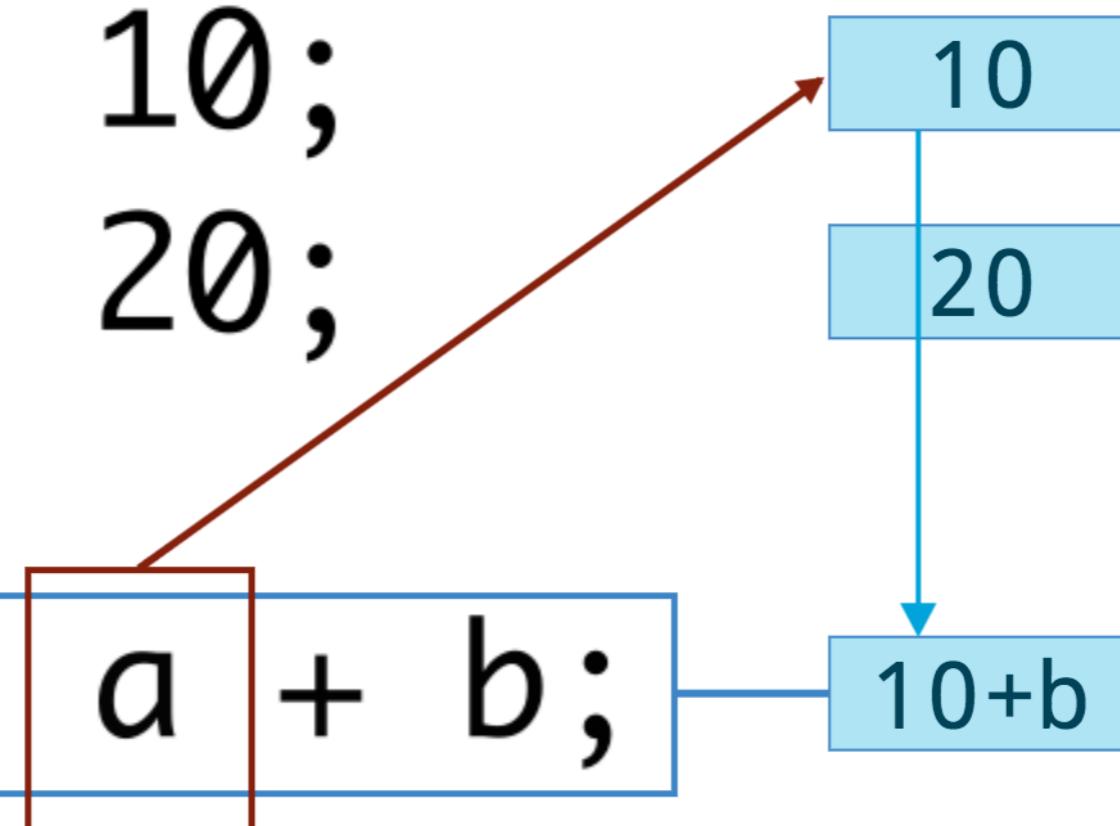
# Интерпретация

```
01: var a = 10;  
02: var b = 20;  
03:  
04: var c = a + b;  
05:  
06: alert(c);
```

The diagram illustrates the state of variables during the execution of the code. Variable 'a' is bound to the value 10, and the expression 'a+b' is bound to the value 'a+b'. The variable 'b' is bound to the value 20.

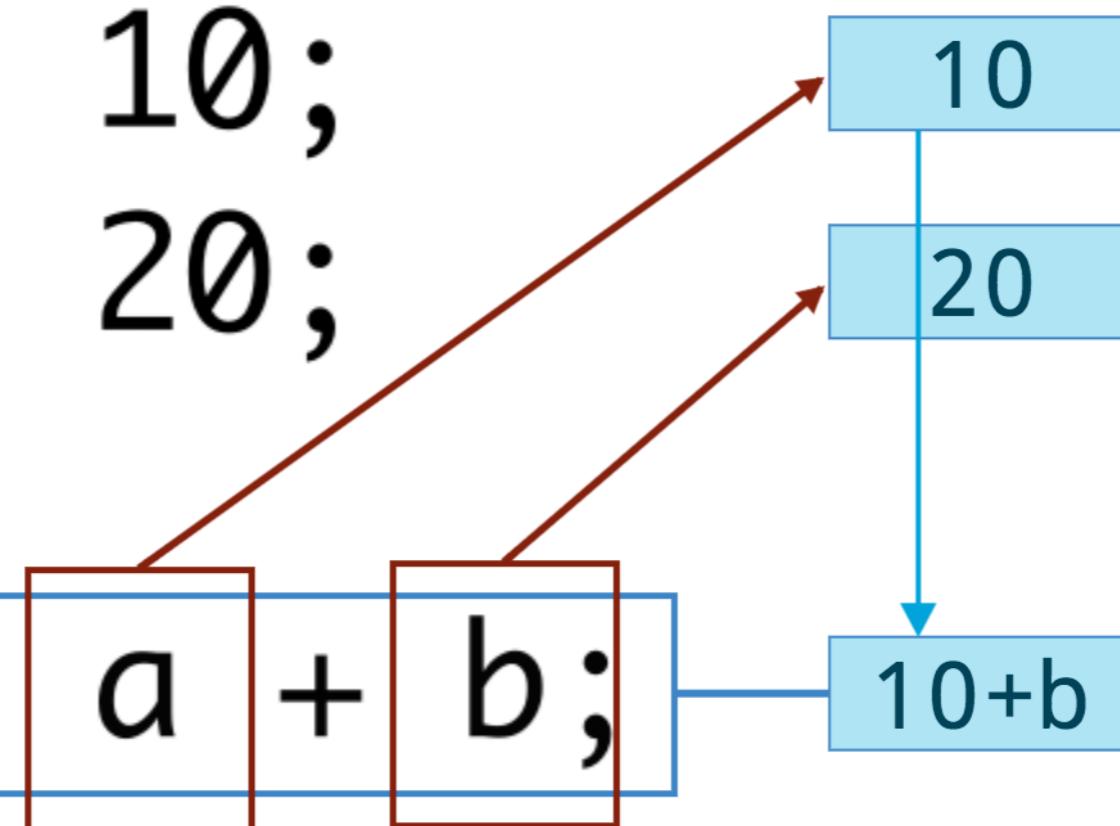
# Интерпретация

```
01: var a = 10;  
02: var b = 20;  
03:  
04: var c = a + b;  
05:  
06: alert(c);
```



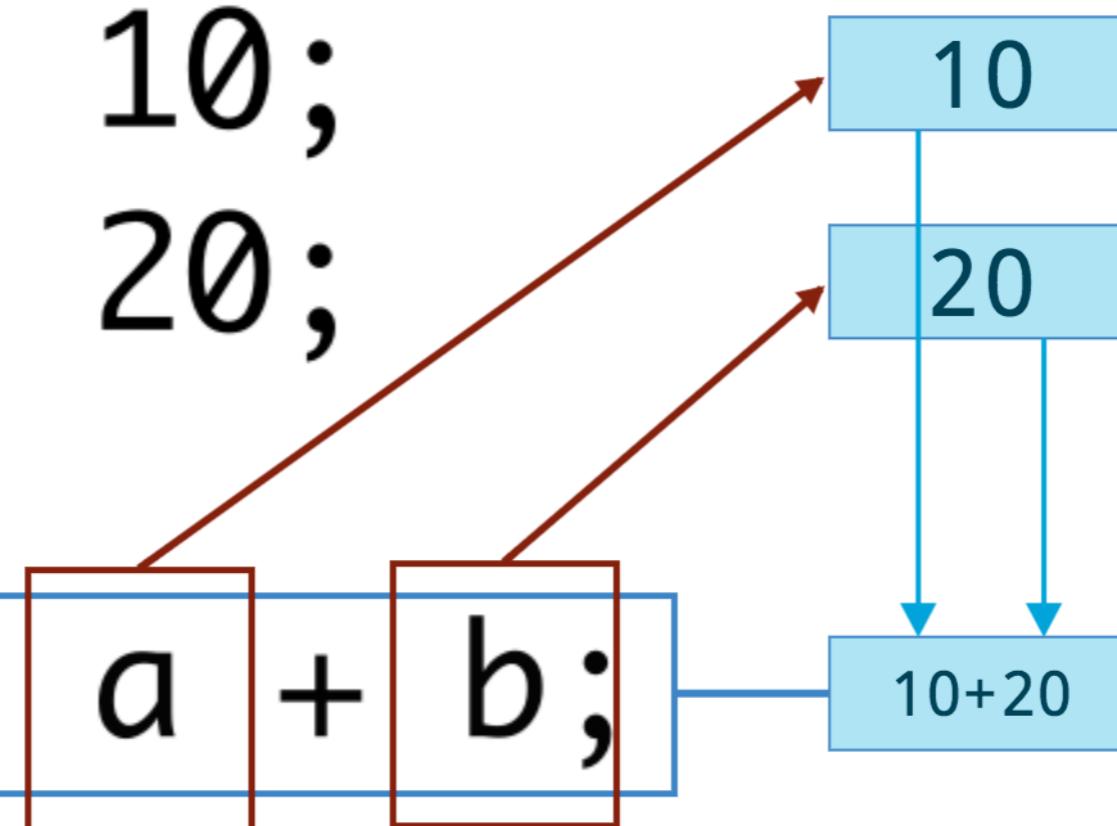
# Интерпретация

```
01: var a = 10;  
02: var b = 20;  
03:  
04: var c = a + b;  
05:  
06: alert(c);
```



# Интерпретация

```
01: var a = 10;  
02: var b = 20;  
03:  
04: var c = a + b;  
05:  
06: alert(c);
```



# Интерпретация

```
01: var a = 10;           10
02: var b = 20;           20
03:
04: var c = a + b;        30
05:
06: alert(c);
```



## JavaScript Alert

30

OK

# Типы данных

Значения	Тип
10, 010, -10, 12.5	<b>number</b>
“hello world”, “с”	<b>string</b>
true, false	<b>boolean</b>
function () { ... }	<b>function</b>
{x: 10, y: 20}	<b>object</b>
[10, 20]	<b>object</b>
undefined	<b>undefined</b>

# Выражения

```
01: // comment  
02:  
03: 666;  
04:  
05: 1 + 1;  
06:  
07: var x;  
08:  
09: var x = 10;  
10:  
11: var x = 10 + 20;
```

# Операторы

# Оператор +

01: 5 + 6 → 11 (число + число)

02:

03: 5 + '6' → '56' (число + строка)

# Оператор +

01: 5 + 6      -> 11 (число + число)

02:

03: 5 + '6'    -> '56' (число + строка)

04:

05: +'10'      -> 10

# Оператор +

01: 5 + 6 → 11 (число + число)

02:

03: 5 + '6' → '56' (число + строка)

04:

05: +'10' → 10

06:

07: {} + {} → NaN

# Оператор +

01: 5 + 6 → 11 (число + число)

02:

03: 5 + '6' → '56' (число + строка)

04:

05: +'10' → 10

06:

07: {} + {} → NaN

08:

09: {} + '' → 0

# Оператор +

01: 5 + 6      -> 11 (число + число)  
02:  
03: 5 + '6'    -> '56' (число + строка)  
04:  
05: +'10'      -> 10  
06:  
07: {} + {}    -> NaN  
08:  
09: {} + ''    -> 0  
10:  
11: '' + {}   -> '[object Object]'

# Оператор +

01: 5 + 6 -> 11 (число + число)  
02:  
03: 5 + '6' -> '56' (число + строка)  
04:  
05: +'10' -> 10  
06:  
07: {} + {} -> NaN  
08:  
09: {} + '' -> 0  
10:  
11: '' + {} -> '[object Object]'  
12:  
13: true + 1 -> 2

# Оператор +

01: 5 + 6 → 11 (число + число)

02:

03: 5 + '6' → '56' (число + строка)

04:

05: +'10' → 10

06:

07: {} + {} → NaN

08:

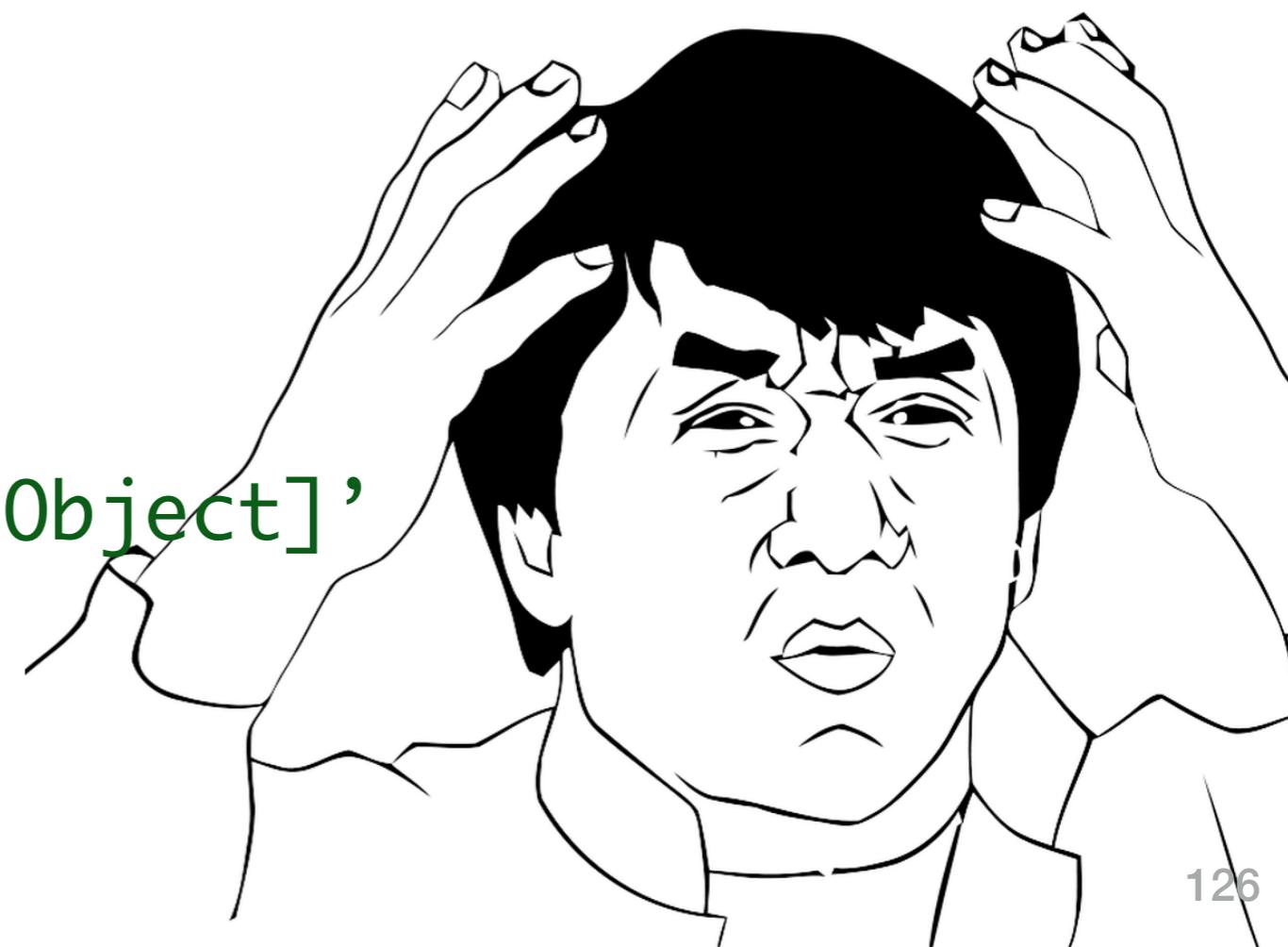
09: {} + '' → 0

10:

11: '' + {} → '[object Object]'

12:

13: true + 1 → 2



# Операторы -, \*, /

Тоже самое, что +, только приводят к числу

# Операторы сравнения

$a > b$  - а строго больше b

$a \geq b$  - а больше или ровно b

$a < b$  - а строго меньше b

$a \leq b$  - а меньше или ровно b

$a == b$  - а равен b

$a === b$  - а строго равен b

# Логические операторы

`a && b` - а и b

`a || b` - а или b

`!a` - не a

# Логические операторы

`a && b` - а и b

`a || b` - а или b

`!a` - не a

`!a && !b` - не a и не b

`a || !b` - а или не b

# Тернарный оператор

a ? b : c

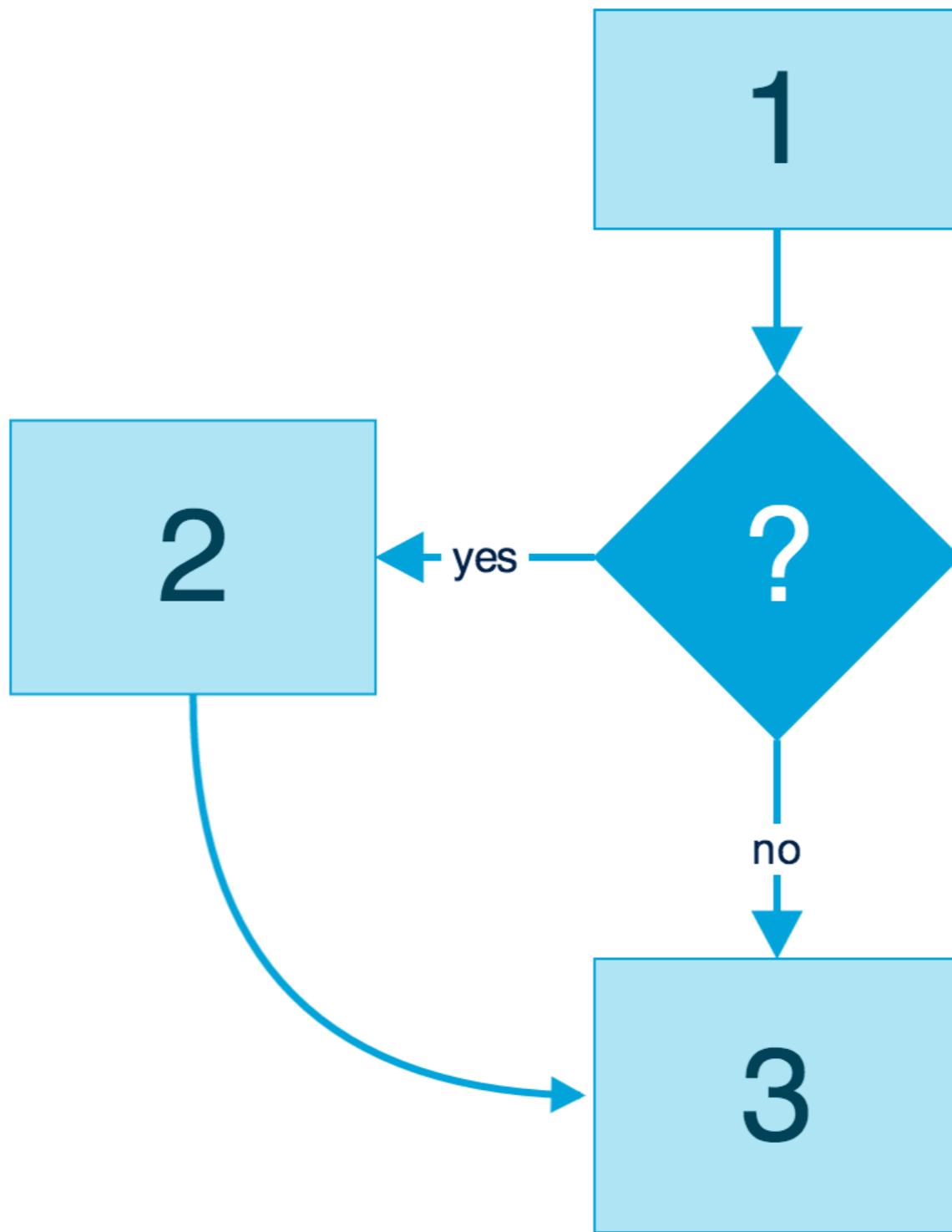
```
if (a) {  
    b  
} else {  
    c  
}
```



# Управляющие конструкции

- Ветвления
- Циклы

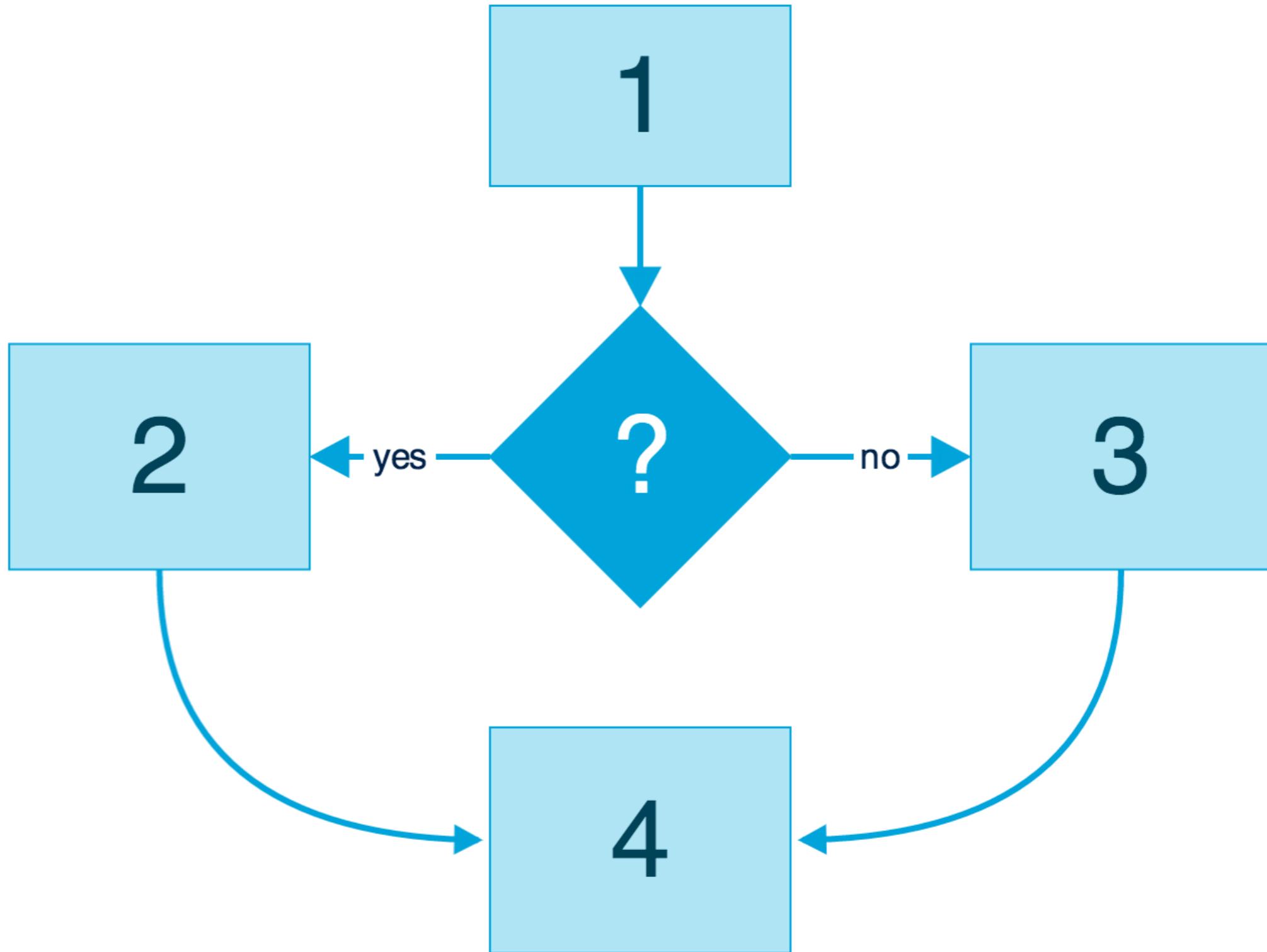
# Ветвление



# Ветвление

```
01: // 1
02: if (condition) {
03:     // 2
04: }
05: // 3
```

# Ветвление



# Ветвление

```
01: // 1
02: if (condition) {
03:     // 2
04: } else {
05:     // 3
06: }
07: // 4
```

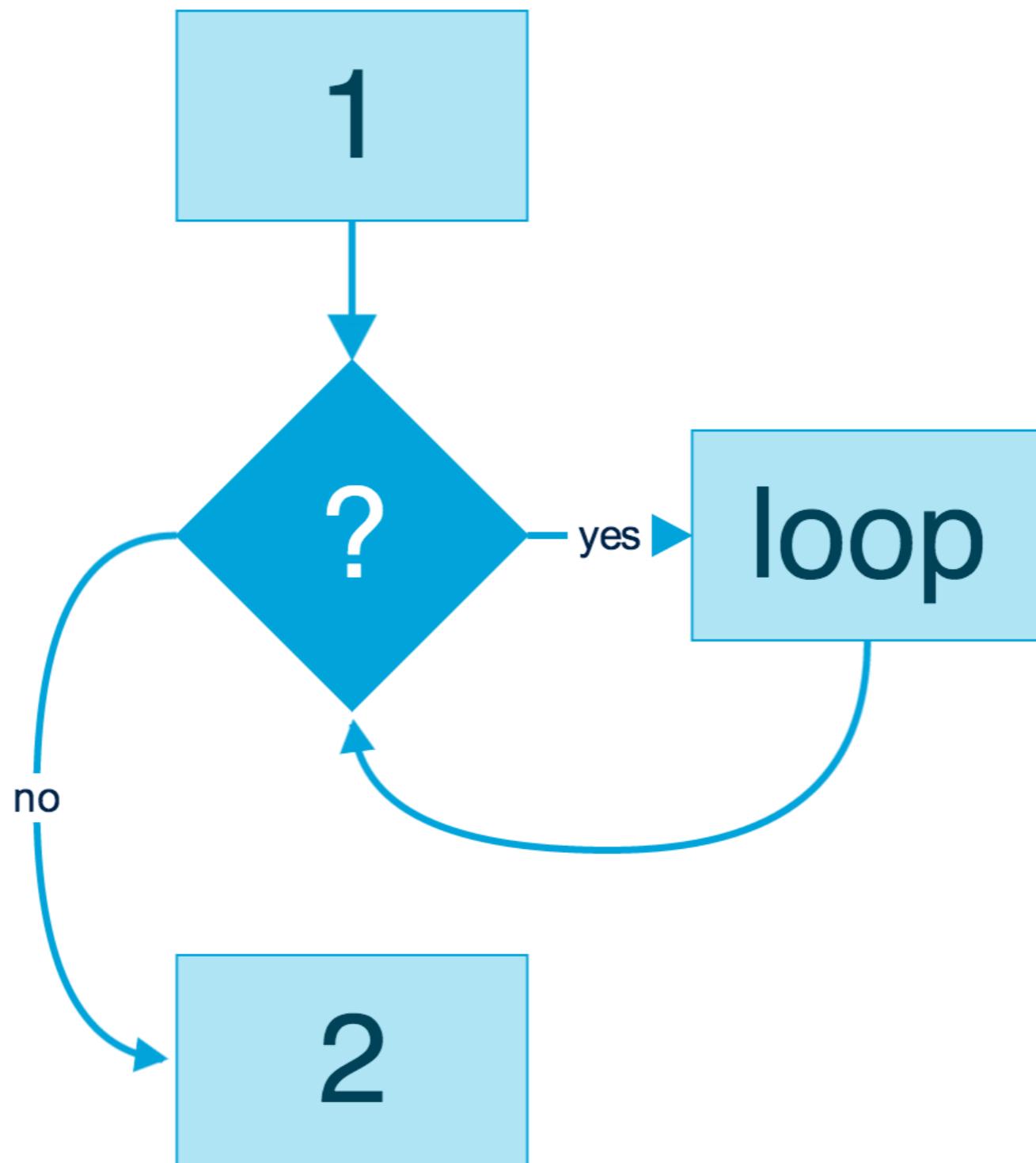
# Ветвление

```
01: // 1
02: if (condition) {
03:     // 2
04: } else
05: if (condition2) {
06:     // 3
07: } else {
08:     // 4
09: }
10: // 5
```

# Ветвление: switch

проще на словах, поверьте

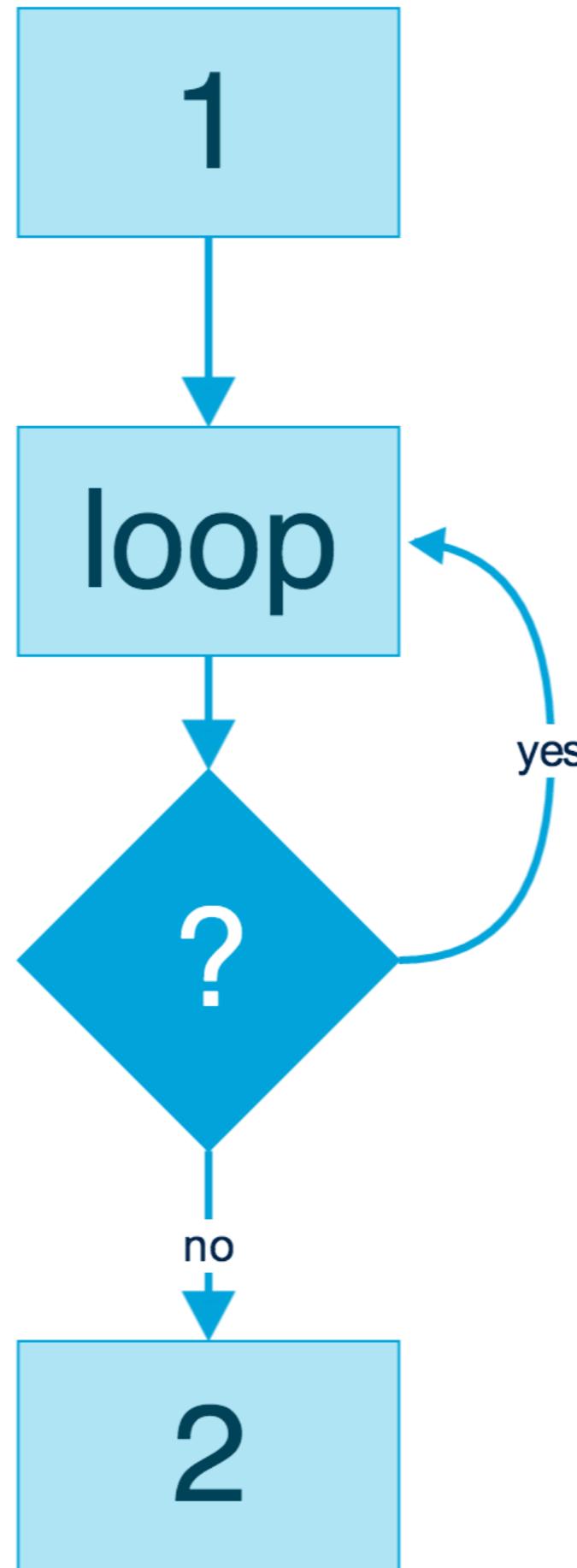
# Цикл: while



# Цикл: while

```
01: // 1
02: while (condition) {
03:     // loop body
04: }
05: // 2
```

# Цикл: do-while



# Цикл: do-while

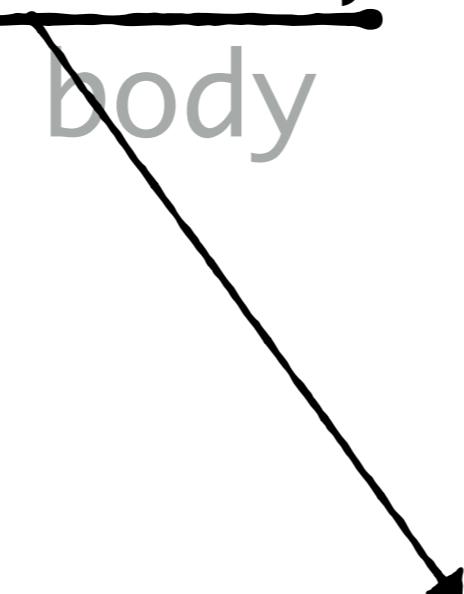
```
01: // 1
02: do {
03:     // loop body
04: } while (condition);
05: // 2
```

# Цикл: for

```
01: // 1
02: for (var i = 0; i < 10; i++){
03:     // loop body
04: }
05: // 2
```

# Цикл: for

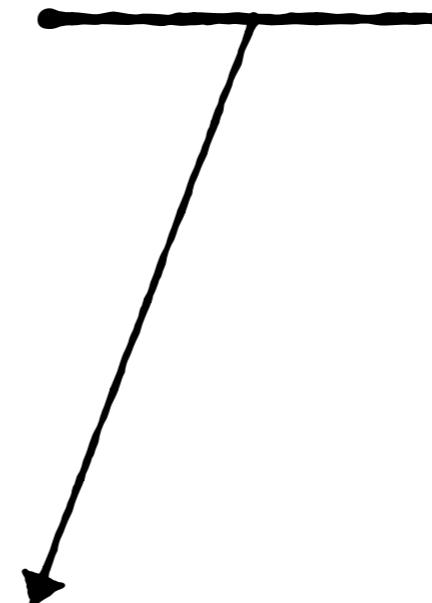
```
01: // 1  
02: for (var i = 0; i < 10; i++) {  
03:     // loop body  
04: }  
05: // 2
```



Инициализация

# Цикл: for

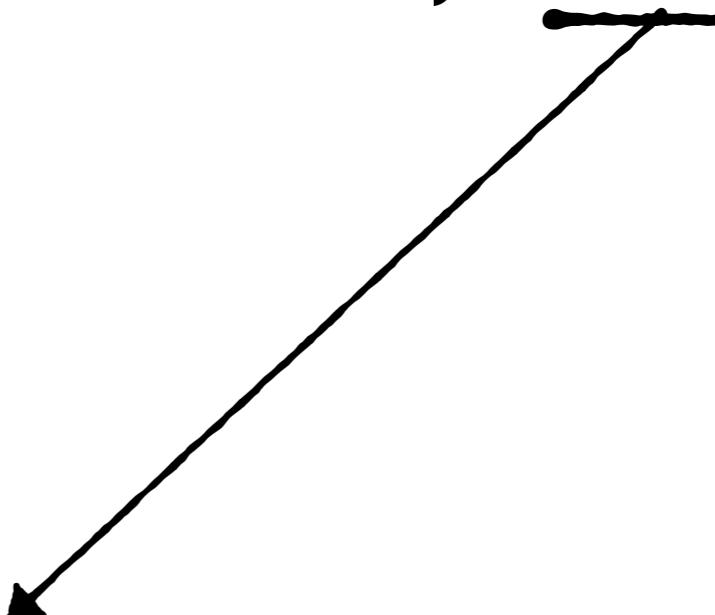
```
01: // 1  
02: for (var i = 0; i < 10; i++){  
03:     // loop body  
04: }  
05: // 2
```



Проверка

# Цикл: for

```
01: // 1  
02: for (var i = 0; i < 10; i++){  
03:     // loop body  
04: }  
05: // 2
```



Обновление

# Цикл: for

```
01: // 1
02: for (var i = 0; i < 10; i++){
03:   console.log(i);
04: }
05: // 2
```

# ФУНКЦИИ

```
01: function greet(name) {  
02:     return 'Hello, ' + name;  
03: }  
04:  
05: var message = greet('World!');  
06:  
07: alert(message);
```

# ФУНКЦИИ

```
01: function greet(name) {  
02:     return 'Hello, ' + name;  
03: }  
04:  
05: var message = greet('World!');  
06:  
07: alert(message);
```



## JavaScript Alert

Hello, World!

OK

Параметры

Тело функции

Результат

# Идеальный пример функции

Мясо

Мясорубка

Фарш

```
01: function inc(x) {  
02:     return x + 1;  
03: }  
04:  
05: var a = inc(10);  
06:  
07: alert(a);
```

```
01: function inc(x) {  
02:     return x + 1;  
03: }  
04:  
05: var a = inc(10);  
06:  
07: alert(a);
```

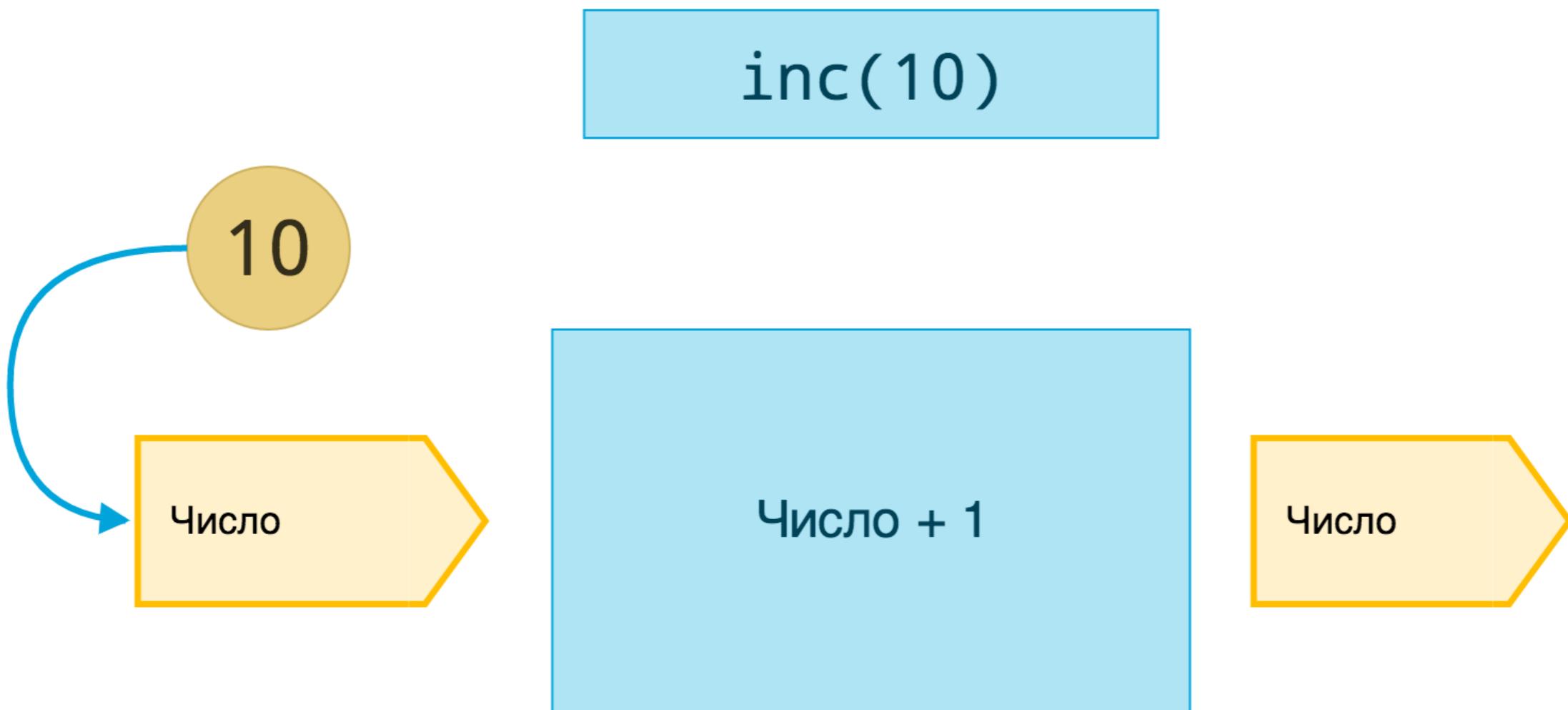
inc(10)

Число

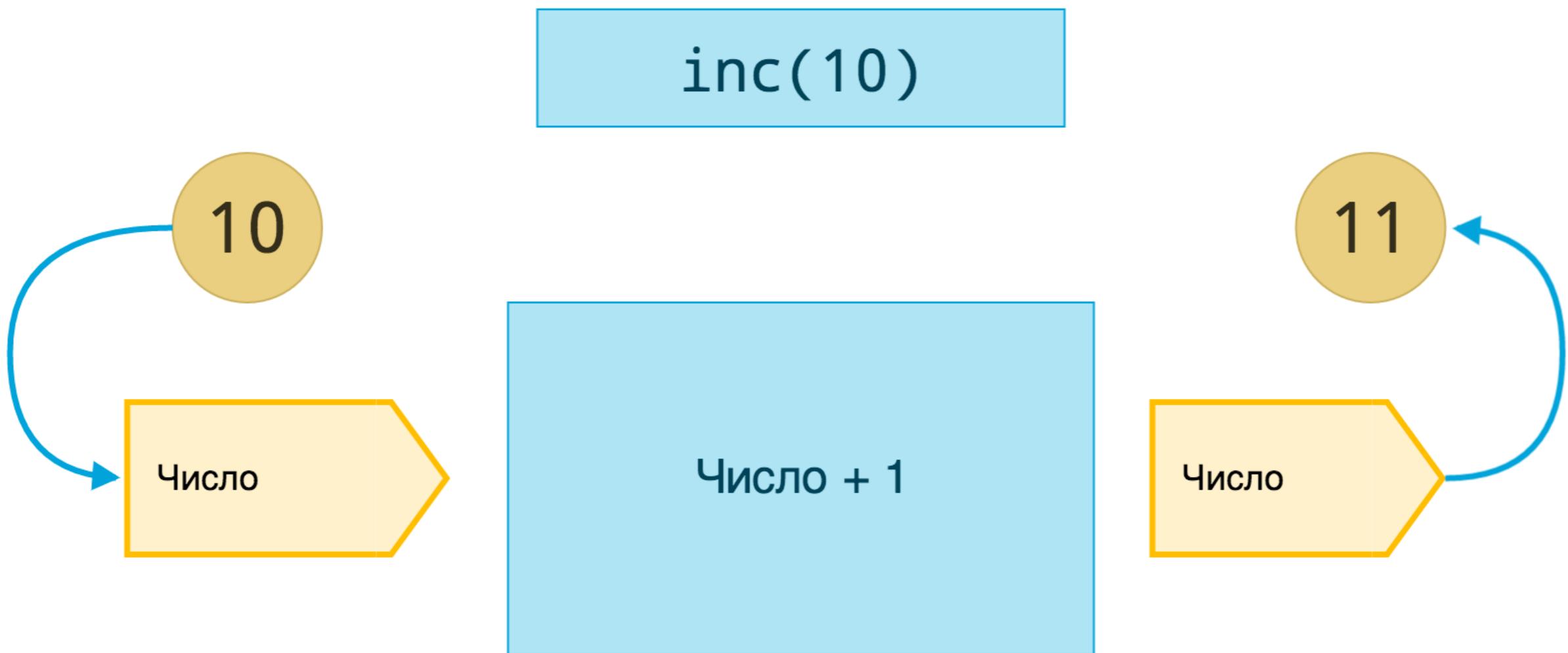
Число + 1

Число

```
01: function inc(x) {  
02:     return x + 1;  
03: }  
04:  
05: var a = inc(10);  
06:  
07: alert(a);
```



```
01: function inc(x) {  
02:     return x + 1;  
03: }  
04:  
05: var a = inc(10);  
06:  
07: alert(a);
```



# Функции - значения

```
01: var greet = function (name) {  
02:     return 'Hello, ' + name;  
03: }  
04:  
05: alert(greet('World!'));
```

# Функции - значения

```
01: var add = function (x, y) {  
02:     return x + y;  
03: }  
04:  
05: var add1 = function (x) {  
06:     return add(x, 1);  
07: }  
08:  
09: alert(add1(10));
```

# Функции - значения

```
01: var addN = function (x) {  
02:     return function(y){  
03:         return x + y;  
04:     };  
05: }  
06:  
07: var add1 = addN(1);  
08:  
09: alert(add1(10));
```

# Область видимости

```
01: var a = 20;  
02:  
03: b = 30;  
04:  
05: function name(){  
06:     var a = 30;  
07: }  
08:  
09: if(true){  
10:     var a = 30;  
11: }
```

# Область видимости

```
01: var a = 20; ← Объявление локальной
02:
03: b = 30;
04:
05: function name(){
06:     var a = 30;
07: }
08:
09: if(true){
10:     var a = 30;
11: }
```

# Область видимости

```
01: var a = 20; ← Объявление локальной
02:
03: b = 30; ← Объявление глобальной
04:
05: function name(){
06:     var a = 30;
07: }
08:
09: if(true){
10:     var a = 30;
11: }
```

# Область видимости

```
01: var a = 20;                                     глобальная область
02:
03: b = 30;
04:
05: function name(){
06:   var a = 30;
07: }
08:
09: if(true){
10:   var a = 30;
11: }
```

# Область видимости

```
01: var a = 20;                                     глобальная область
02:
03: b = 30;
04:
05: function name(){
06:   var a = 30;                                 локальная область
07: }
08:
09: if(true){
10:   var a = 30;
11: }
```

# Область видимости

В JavaScript только функции создают новую область видимости

# Замыкания

```
01: var cache = function (){  
02:     var store = {};  
03:  
04:     return function (key, value){  
05:         if (value){  
06:             return store[key] = value;  
07:         } else {  
08:             return store[key]  
09:         }  
10:     }  
11: };
```

# Замыкания

```
01: var cache = function (){  
02:     var store = {};  
03:  
04:     return function (key, value){  
05:         if (value){  
06:             return store[key] = value;  
07:         } else {  
08:             return store[key]  
09:         }  
10:     }  
11: };
```

# Замыкания

```
01: var cache = function (){  
02:     var store = {};  
03:  
04:     return function (key, value){  
05:         if (value){  
06:             return store[key] = value;  
07:         } else {  
08:             return store[key]  
09:         }  
10:     }  
11: };
```

# Замыкания

```
01: var cache = function () {
02:   var store = {};
03:
04:   return function (key, value){
05:     if (value){
06:       return store[key] = value;
07:     } else {
08:       return store[key]
09:     }
10:   }
11: };
```

сохраняется ссылка  
на store, но сама  
переменная store  
недоступна

# Замыкания

В JavaScript - единственный способ создать приватные переменные (те, к которым нет доступа из вне) — это замыкания.

# Хитрый кунштюк

```
01: var cache = function (){  
02:     var store = {};  
03:  
04:     return function (key, value){  
05:         if (value){  
06:             return store[key] = value;  
07:         } else {  
08:             return store[key]  
09:         }  
10:     }  
11: }
```

# Хитрый кунштюк

```
01: var cache = (function (){  
02:     var store = {};  
03:  
04:         return function (key, value){  
05:             if (value){  
06:                 return store[key] = value;  
07:             } else {  
08:                 return store[key]  
09:             }  
10:         }  
11: })();
```

Self-invoking functions

# Хитрый кунштюк

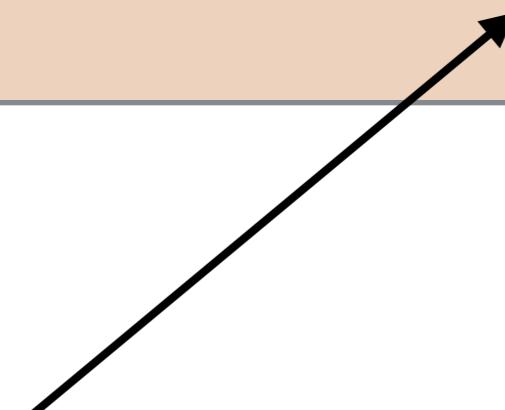
```
01: (function (){  
03  
03:     var a = “Ничто не запретно”;  
04:     var b = “Все дозволено”;  
05  
06: })();
```

# Хитрый кунштюк

```
01: (function () {
```

```
03:     var a = "Ничто не запретно";  
04:     var b = "Все дозволено";
```

```
05: })();
```



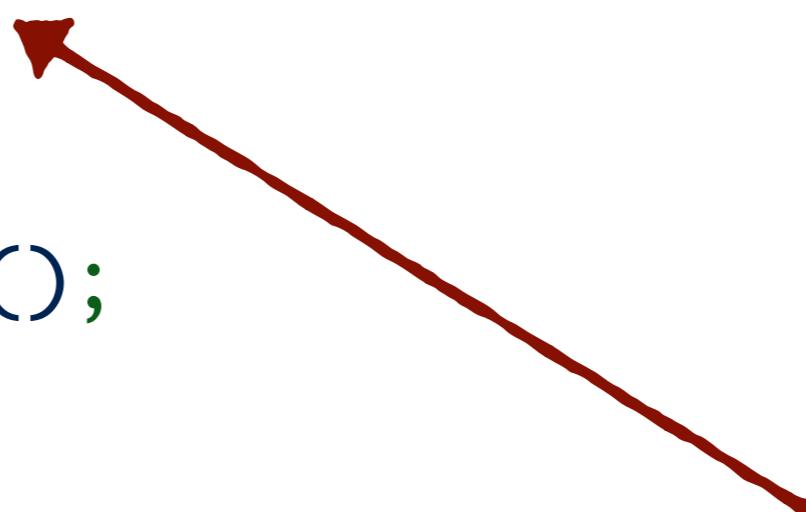
Новый scope. Главное не забывать var.

# Контекст

```
01: function context() {  
02:     return this;  
03: }  
04:  
05: var ctx = context();  
06:  
07: alert(ctx);
```

# Контекст

```
01: function context() {  
02:   return this;  
03: }  
04:  
05: var ctx = context();  
06:  
07: alert(ctx);
```



BOT OH



**The page at <https://trello.com> says:**

[object Window]

OK

# Контекст

```
01: function context() {  
02:     return this;  
03: }  
04:  
05: var ctx = context.bind("Hello")();  
06:  
07: alert(ctx);
```



**The page at <https://www.google.ru> says:**

Hello

OK

# Контекст

- bind - связывание контекста
- call или apply - вызов в определенном контексте

# Окружение браузера

- таймеры

# Окружение браузера

- таймеры

```
01: setInterval(function(){  
02:     console.log('Tick!');  
03: }, 1000);  
04:  
05: setTimeout(function(){  
06:     console.log('Опоздал на 5 секунд');  
07: }, 5000);
```

# Окружение браузера

- таймеры
- window

# Окружение браузера

- таймеры
- window
- document

# Dev Tools

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. The top navigation bar includes 'Elements', 'Network', 'Sources', 'Timeline', 'Profiles', 'Resources', 'Audits', and 'Console'. The main area displays the DOM tree for the current page, with the root node being the body element. The body element has several classes applied: 'logged\_in', 'env-production', 'macintosh', and 'vis-public'. A tooltip for the 'logged\_in' class shows its definition from 'github-aef30885...951852bb.css:1' by nadmozg, which includes a padding of 132x1403.190. The bottom section of the DevTools shows the 'Styles' panel for the body element, listing various CSS properties like font, color, and background-color. The 'Properties' tab is also visible in the styles panel.

<!DOCTYPE html>

▼ <html lang="en" class=" is-copy-enabled">

► <head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb# object: http://ogp.me/ns/object# article: http://ogp.me/ns/article# profile: http://ogp.me/ns/profile#">...</head>

▼ <body class="logged\_in env-production macintosh vis-public">

  <a href="#start-of-content" tabindex="1" class="accessibility-aid js-skip-to-content">Skip to content</a>

  ► <div class="header header-logged-in true" role="banner">...</div>

  <div id="start-of-content" class="accessibility-aid"></div>

  <div id="js-flash-container"></div>

  ► <div role="main" class="main-content">...</div>

  ► <div class="container">...</div>

  ► <div id="ajax-error-message" class="flash flash-error">...</div>

    <script crossorigin="anonymous" integrity="sha256-+Ec970ckLaaIDVIxNjSIGzl1xSrzqh5s0BV8DyYYVpE=" src="https://assets-cdn.github.com/assets/frameworks-f8473dece7242da6a20d52313634881b3975c52cebaa1e6c38157c0f26185691.js"></script>

    <script async="async" crossorigin="anonymous" integrity="sha256-KbcxYgNz1RAif41n/2TMoaUqb7X4j0nWCrDDnRAWqGU=" src="https://assets-cdn.github.com/assets/github-29b731620373d510227f8d67ff64cca1a52a6fb5f88f49d60ab0c39d1016a865.js"></script>

  ► <div class="js-stale-session-flash stale-session-flash flash flash-warn flash-banner hidden">...</div>

  </body>

</html>

html.is-copy-enabled body.logged\_in.env-production.macintosh.vis-public

Styles Event Listeners DOM Breakpoints Properties

Filter

element.style {

}

media="all"

body {

  word-wrap: break-word;

}

media="all"

body {

  font: 13px/1.4 Helvetica, arial, nimbussansl, liberationsans, freesans, clean, sans-serif, "Segoe UI Emoji", "Segoe UI Symbol";

  color: #333;

  background-color: #fff;

}

media="all"

body {

  margin: 0;

}

github-aef30885...951852bb.css:1 nadmozg

github-aef30885...951852bb.css:1 nadmozg

github-aef30885...951852bb.css:1 nadmozg

Console Search Emulation Rendering

<top frame> ▾ Preserve log

это все  
пора за дело