#1

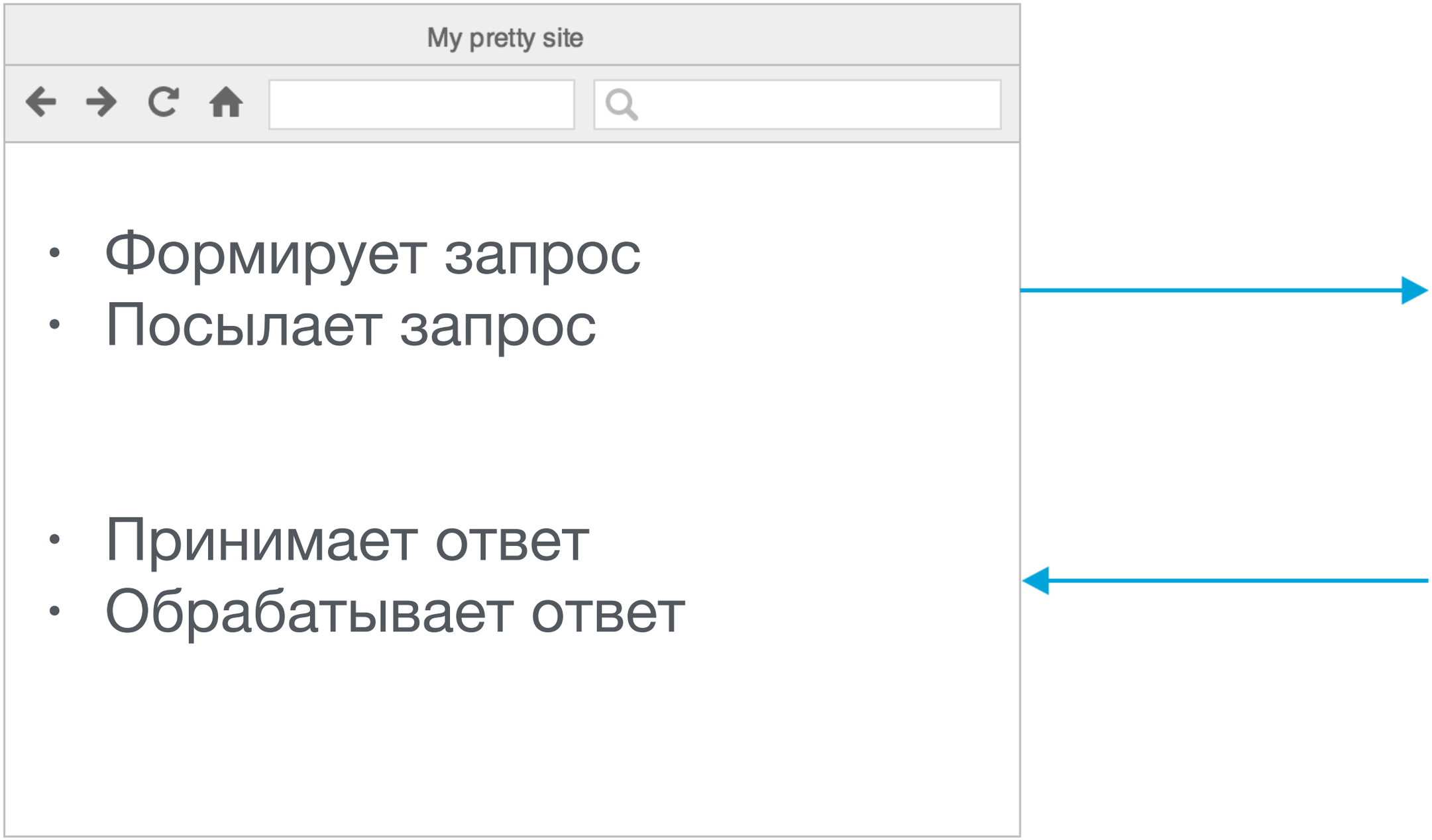
HTML, SVG, CSS

И

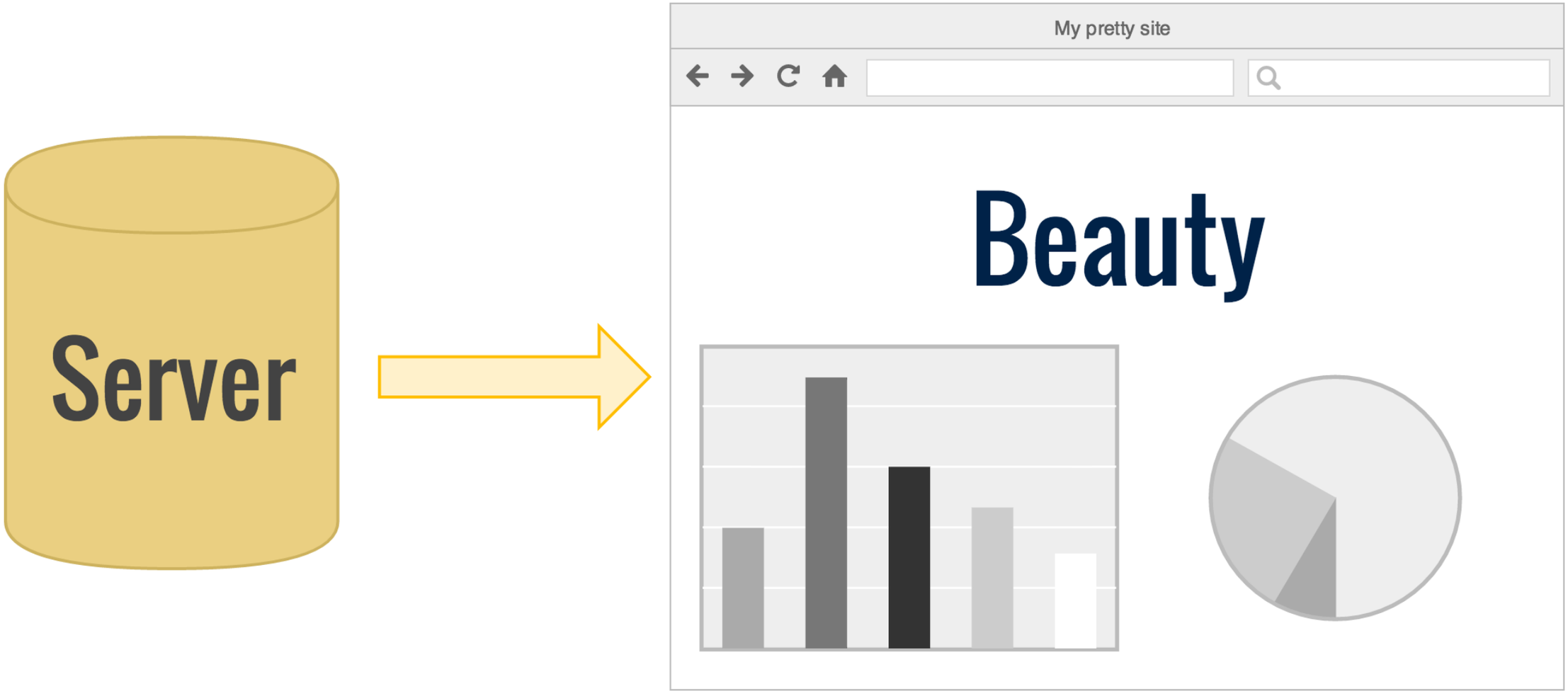
JAVASCRIPT

Представление данных в вебе.

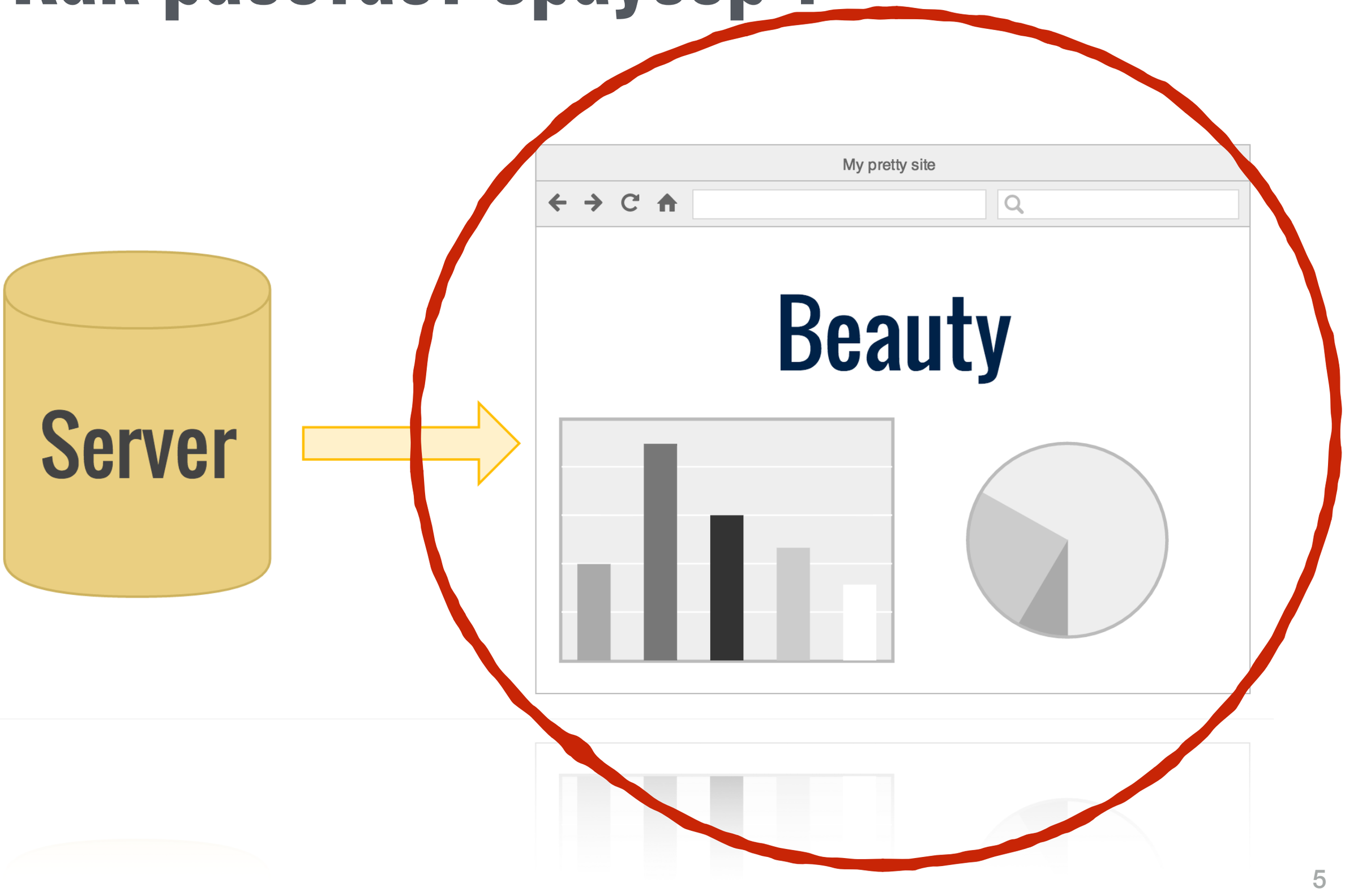
Как работает браузер ?



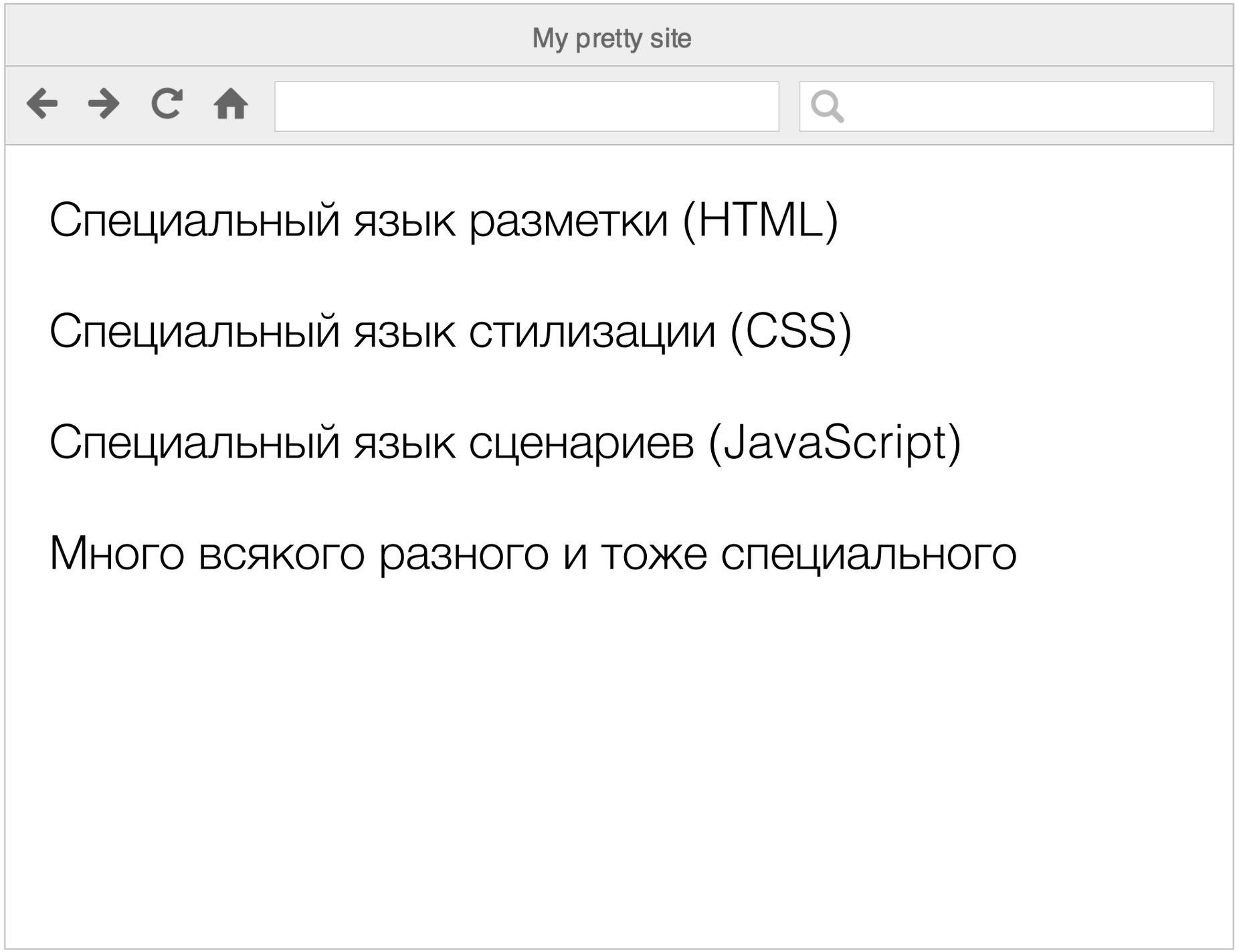
Как работает браузер ?



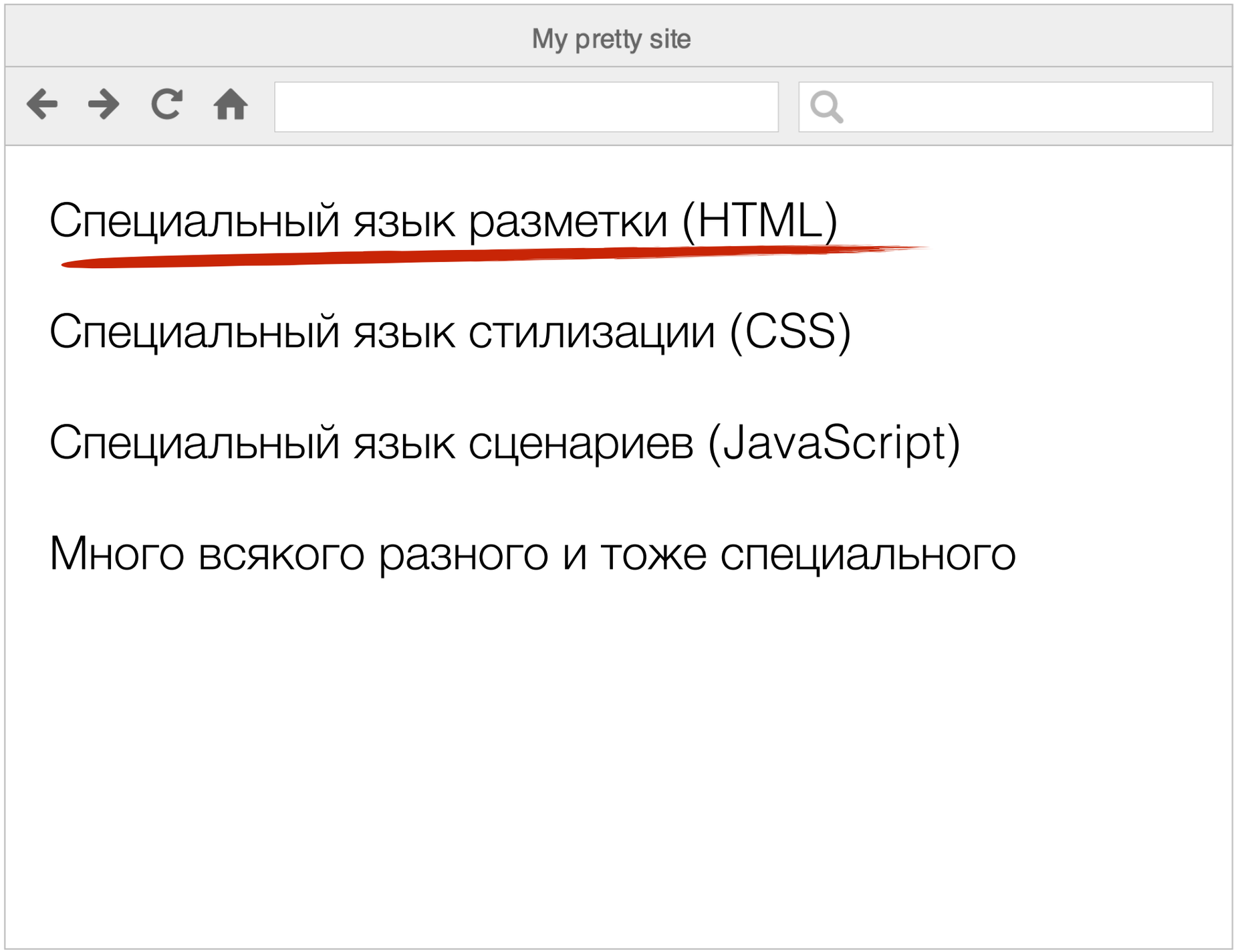
Как работает браузер ?



Как работает браузер ?



Как работает браузер ?



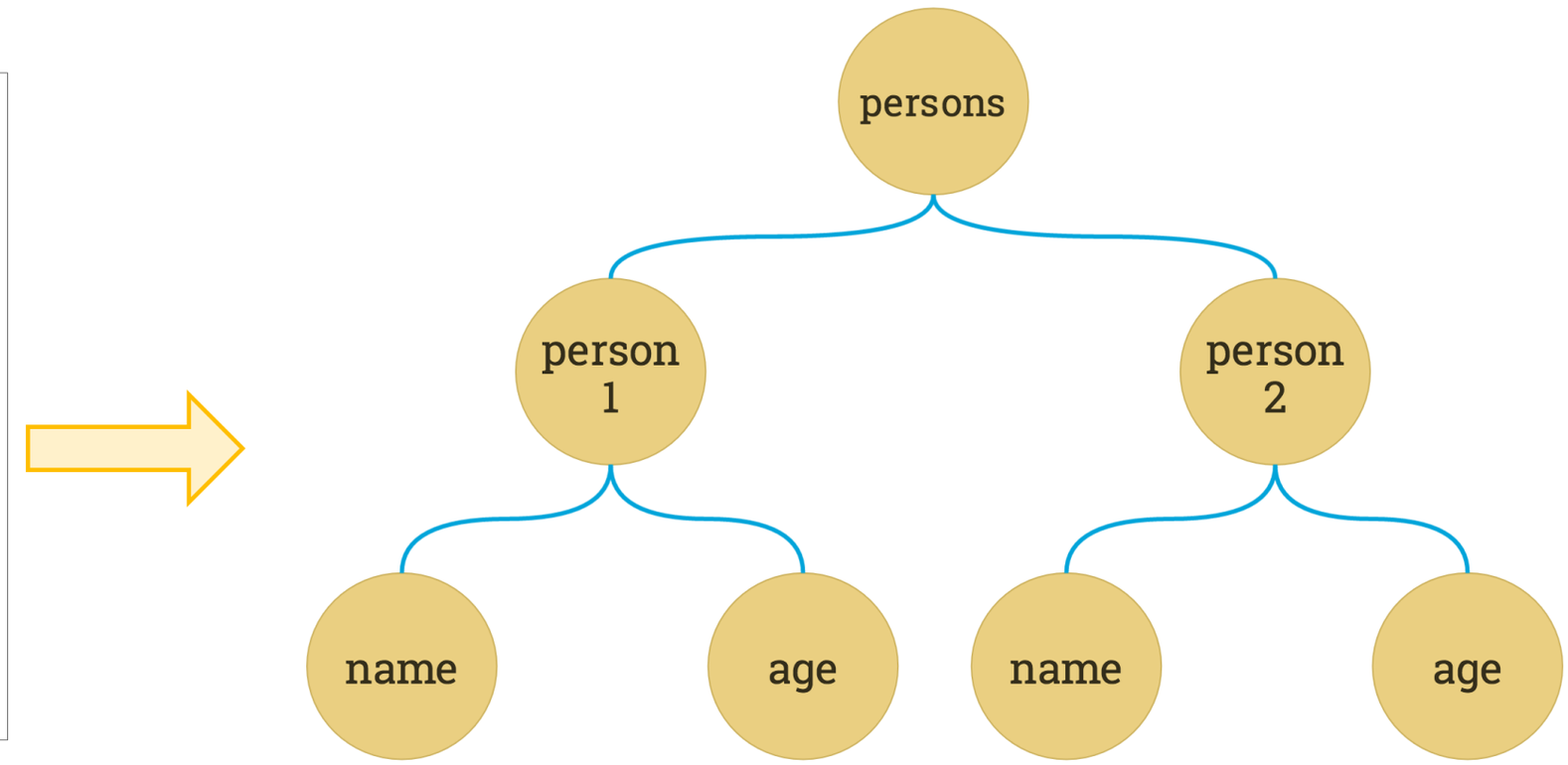
вообще-то

HTML ∈ XML

XML

extensible markup language

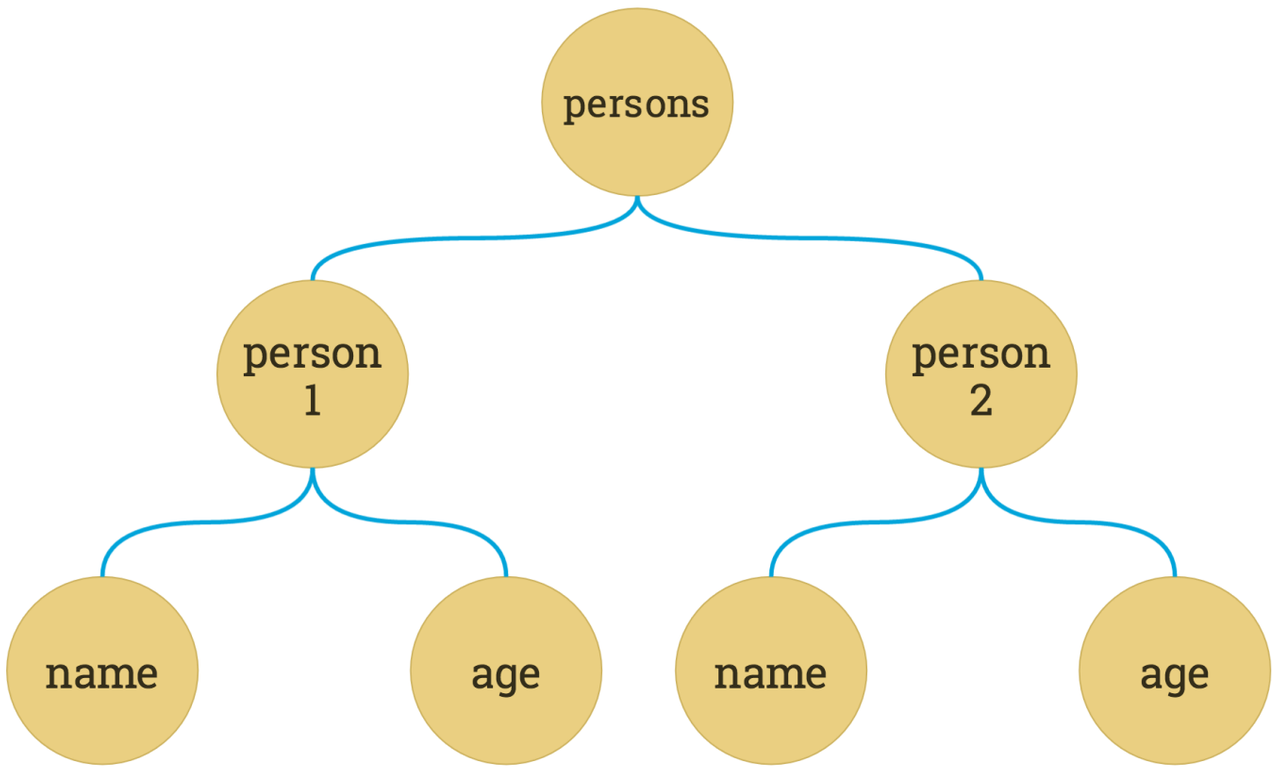
```
<persons>
  <person id="1">
    <name>John</name>
    <age>26</age>
  </person>

  <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>
    <img src=“my-picure.png”>
  </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>
    <img src="my-picure.png">
  </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>
    <img src="my-picure.png">
  </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>
    <img src="my-picure.png">
  </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>
    <img src="my-picure.png" />
  </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>
    <img src=“my-picure.png”>
  </body>
</html>
```

Аттрибут (имя)

HTML

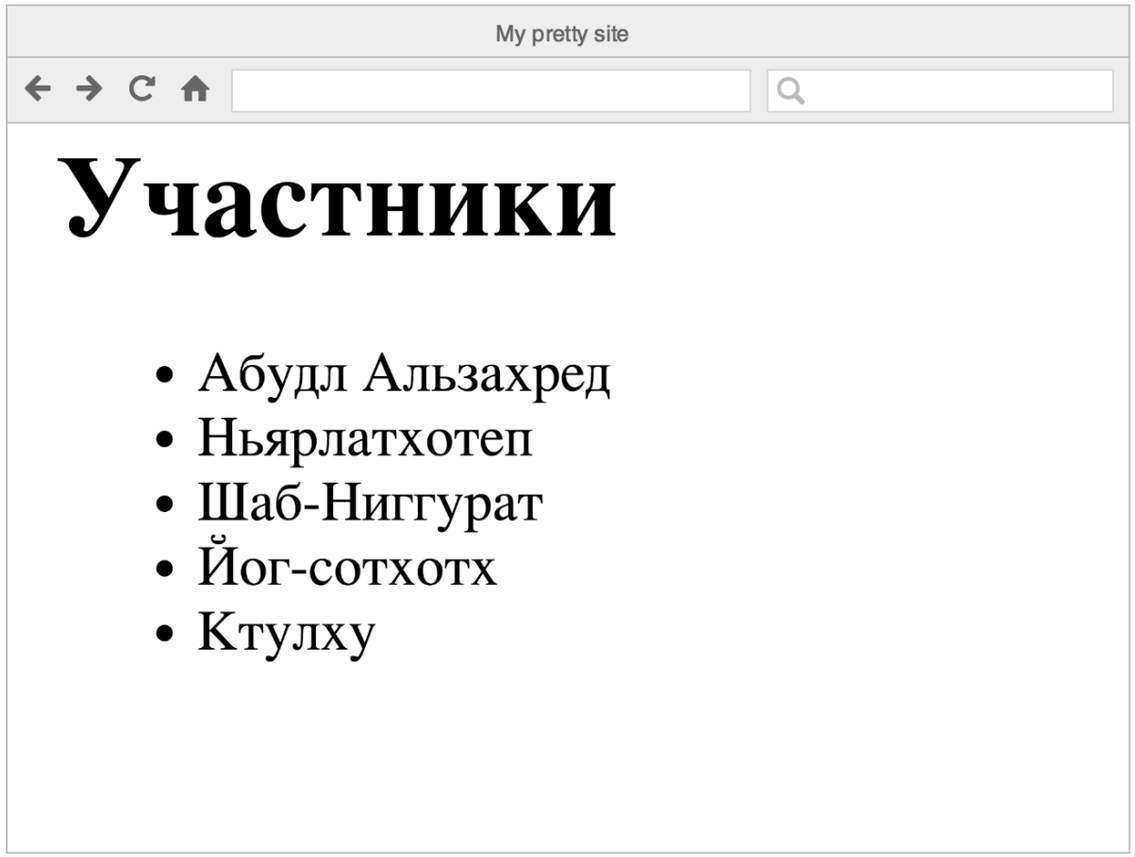
HyperText Markup Language

```
<!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>
    <img src=“my-picure.png”>
  </body>
</html>
```

Аттрибут (значение)

HTML - это иерархия документа

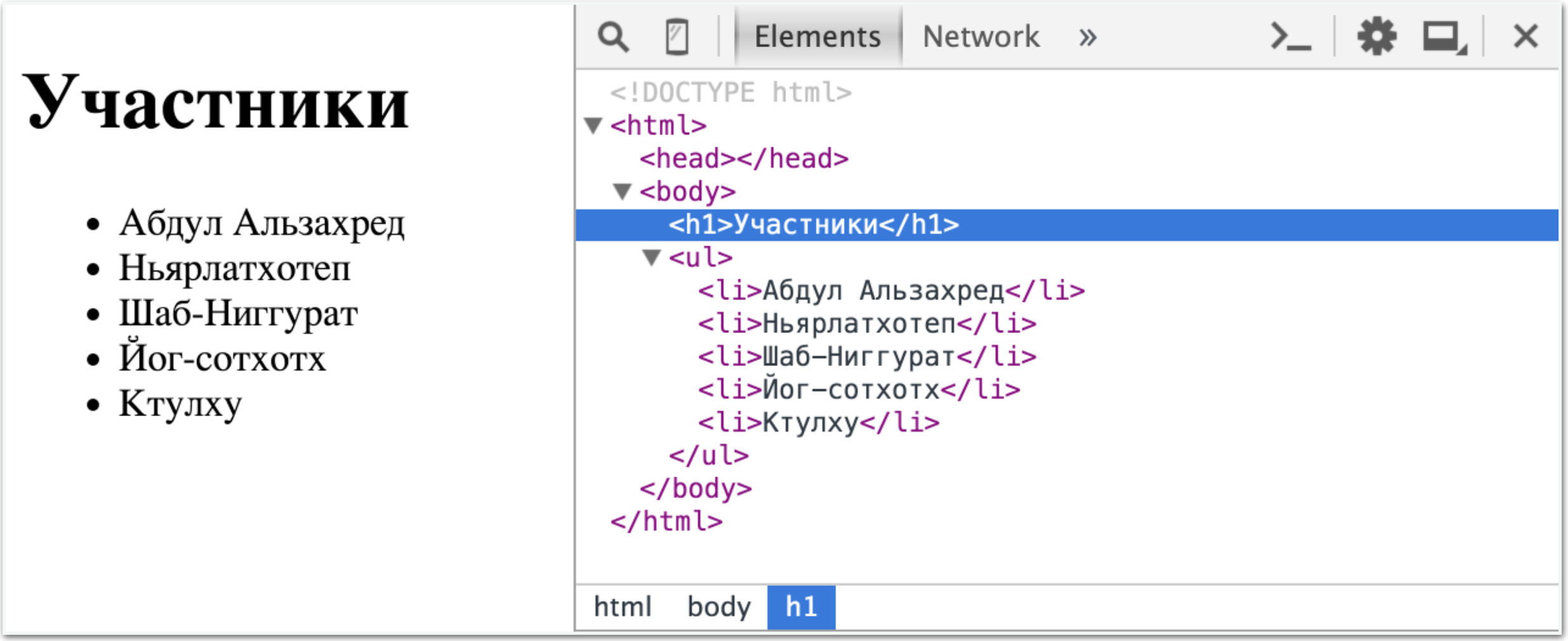
```
<h1>Участники</h1>
<ul>
    <li>Абдул Альзахред</li>
    <li>Ньярлатхотеп</li>
    <li>Шаб-Ниггурат</li>
    <li>Йог-сотхотх</li>
    <li>Ктулху</li>
</ul>
```



Объектная модель документа

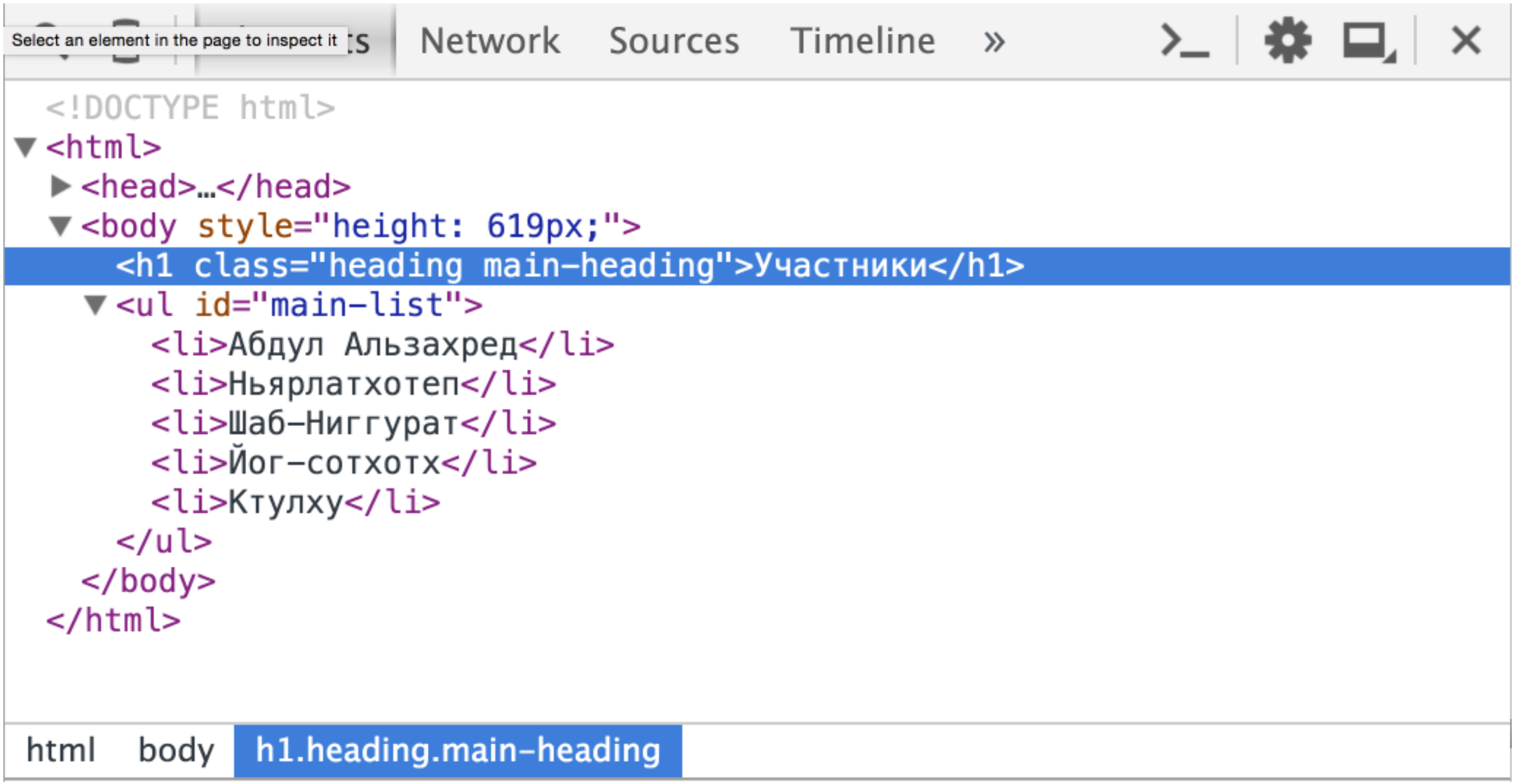
```
01: <!DOCTYPE html>
02: <html>
03:   <head>
04:     <title> Привет, Moscow Coding School! </title>
06:   </head>
07:   <body>
08:     <h1>Участники</h2>
09:     <ul>
10:       <li>Абдул Альзахред</li>
11:       <li>Ньярлатхотеп</li>
12:       <li>Шаб-Ниггурат</li>
13:       <li>Йог-сотхотх</li>
14: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

Объектная модель документа



```
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>Ньярлатхотеп</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>Ньярлатхотеп</li>
12:       <li>Шаб-Ниггурат</li>
13:       <li>Йог-сотхотх</li>
14:       <li>Ктулху</li>
15:     </ul>
16:   </body>
17: </html>
```


Select an element in the page to inspect it
Network
Sources
Timeline
»
<!DOCTYPE html>
<html>
<head>…</head>
<body style="height: 619px;">
<h1 class="heading main-heading">Участники</h1>
<ul id="main-list">
Абдул Альзахред
Ньярлатхотеп
Шаб-Ниггурат
Йог-сотхотх
Ктулху

</body>
</html>
html
body
h1.heading.main-heading

Визуализация

только HTML, только хардкор

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

```
<table>
  <thead>
    <tr>
      <th>Table Heading Cell</th>
      <th>Table Heading Cell</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Table Body Cell</td>
      <td>Table Body Cell</td>
    </tr>
    <tr>
      <td>Table Body Cell</td>
      <td>Table Body Cell</td>
    </tr>
  </tbody>
</table>
```

<table>

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

<thead>

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

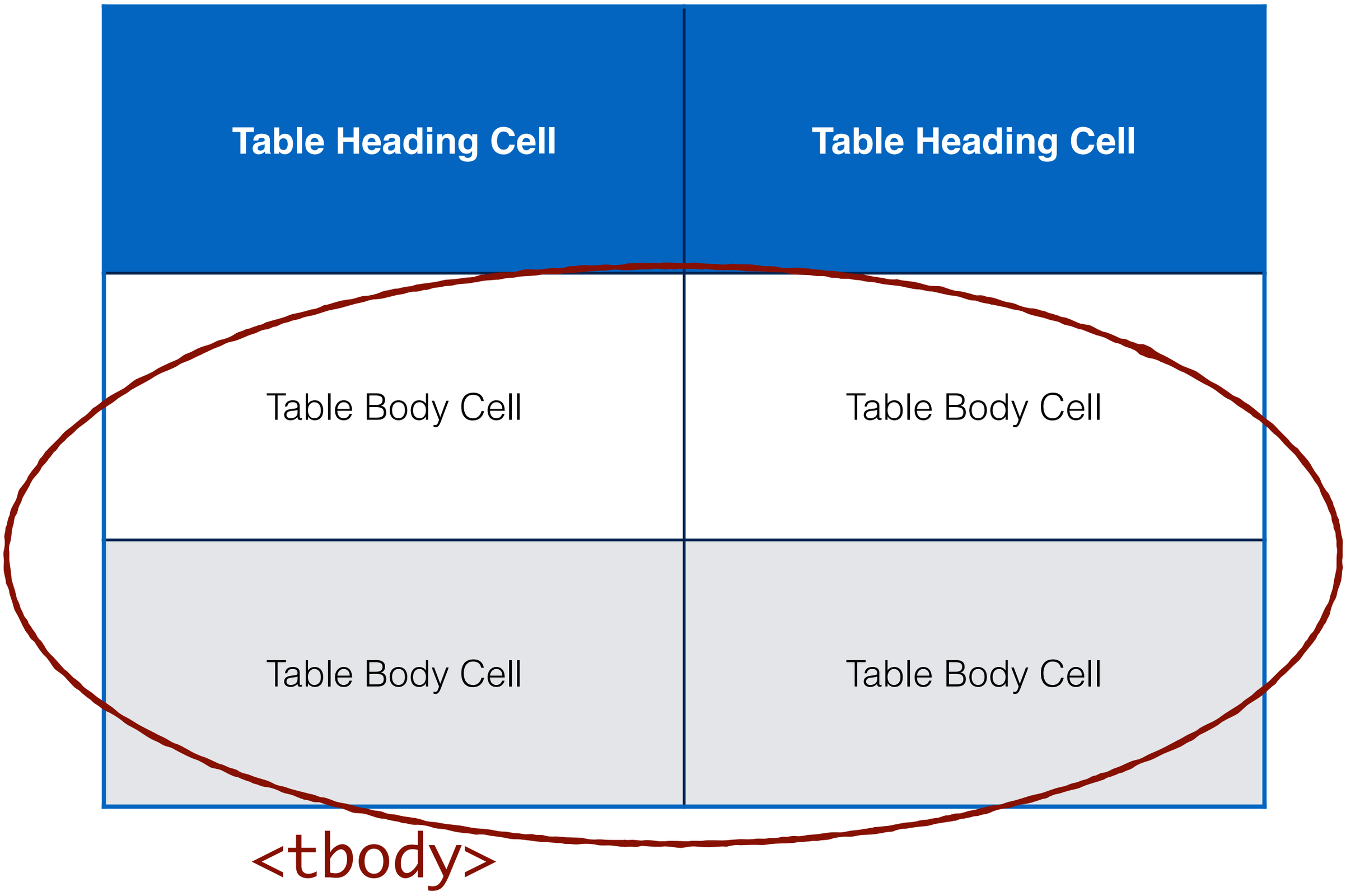


Table Heading Cell
Table Heading Cell
Table Body Cell
Table Body Cell
Table Body Cell
Table Body Cell
<tbody>

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

Table Heading Cell <th>	Table Heading Cell <th>
Table Body Cell <td>	Table Body Cell <td>
Table Body Cell <td>	Table Body Cell <td>

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

```
<table border="1" cellpadding="10">
  <thead>
    <tr>
      <th>Table Heading Cell</th>
      <th>Table Heading Cell</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Table Body Cell</td>
      <td>Table Body Cell</td>
    </tr>
    <tr>
      <td>Table Body Cell</td>
      <td>Table Body Cell</td>
    </tr>
  </tbody>
</table>
```

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

SVG

вообще-то

SVG ∈ XML

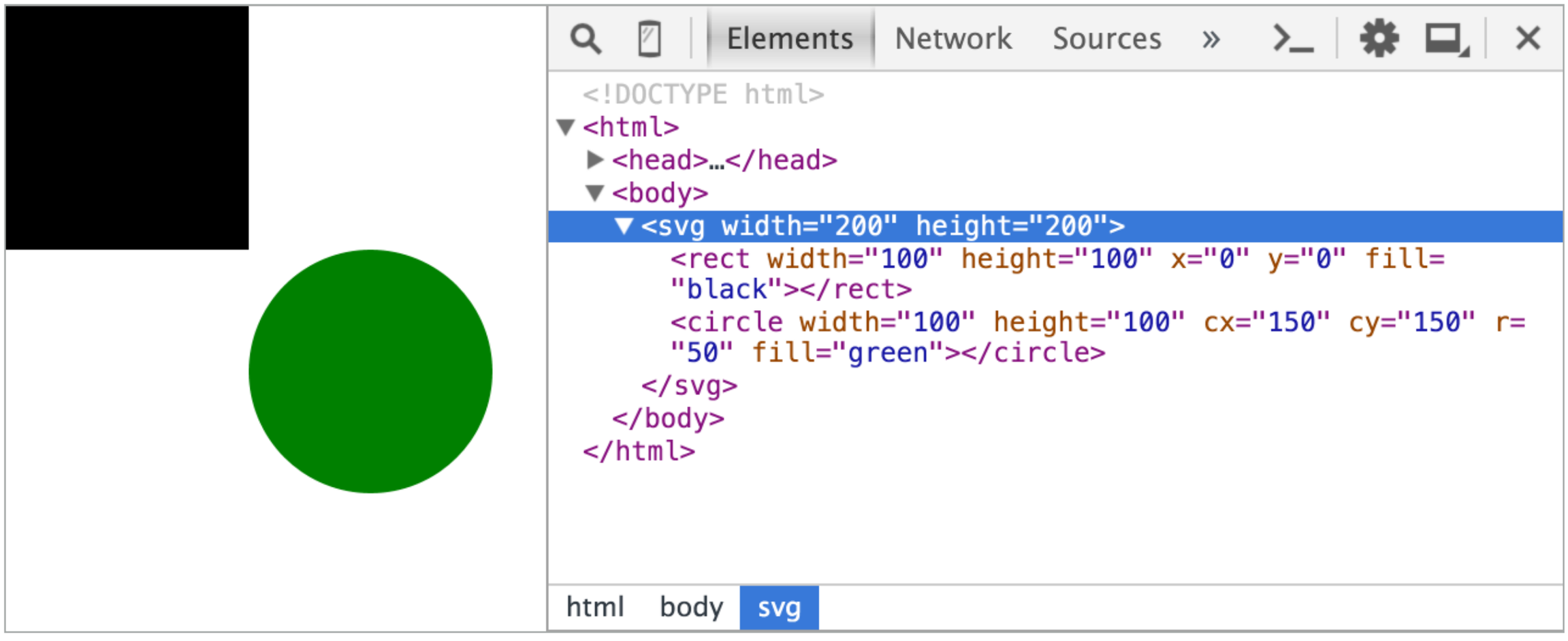
SVG

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>
```

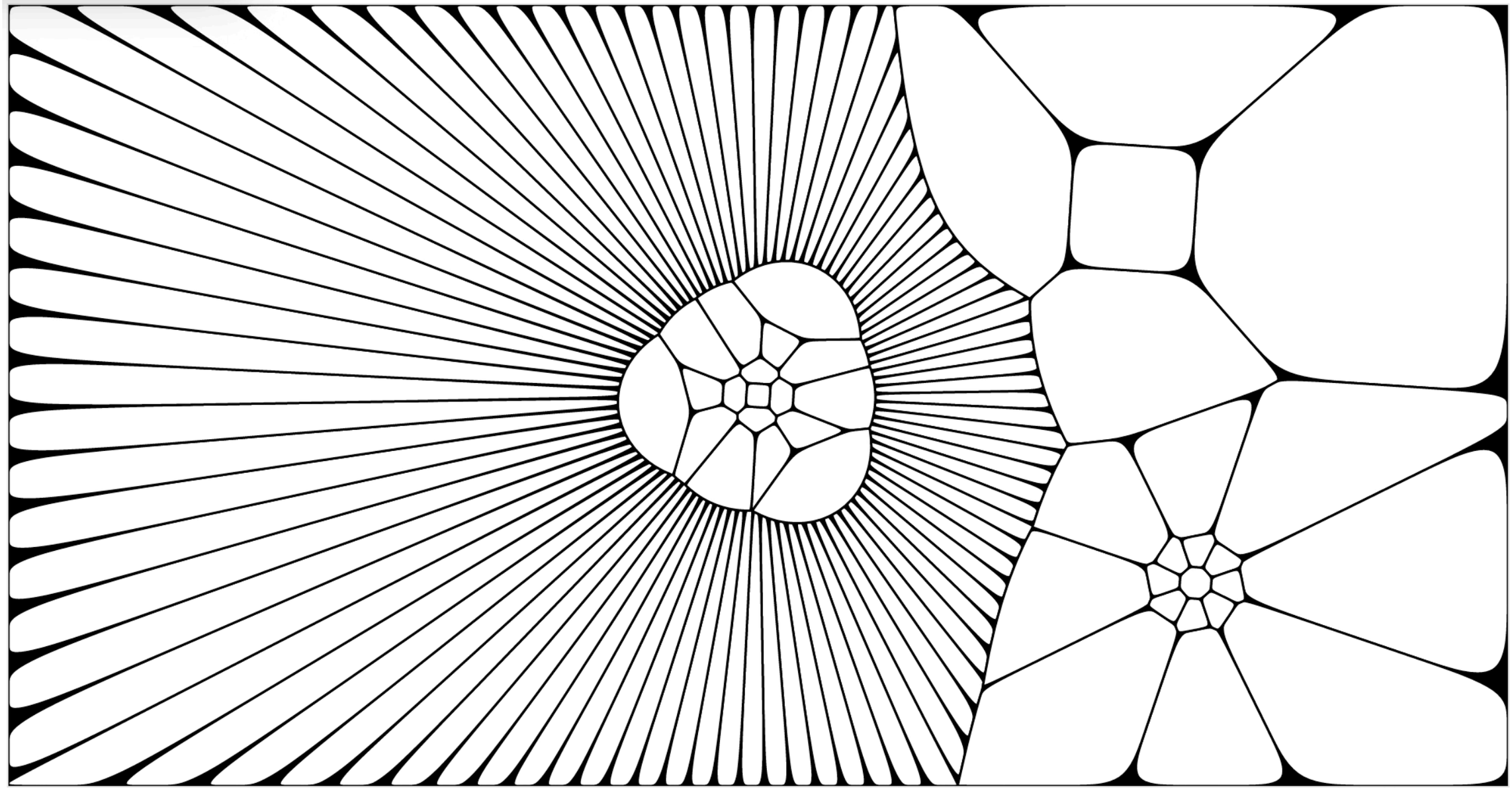
SVG

Scalable Vector Graphic



SVG

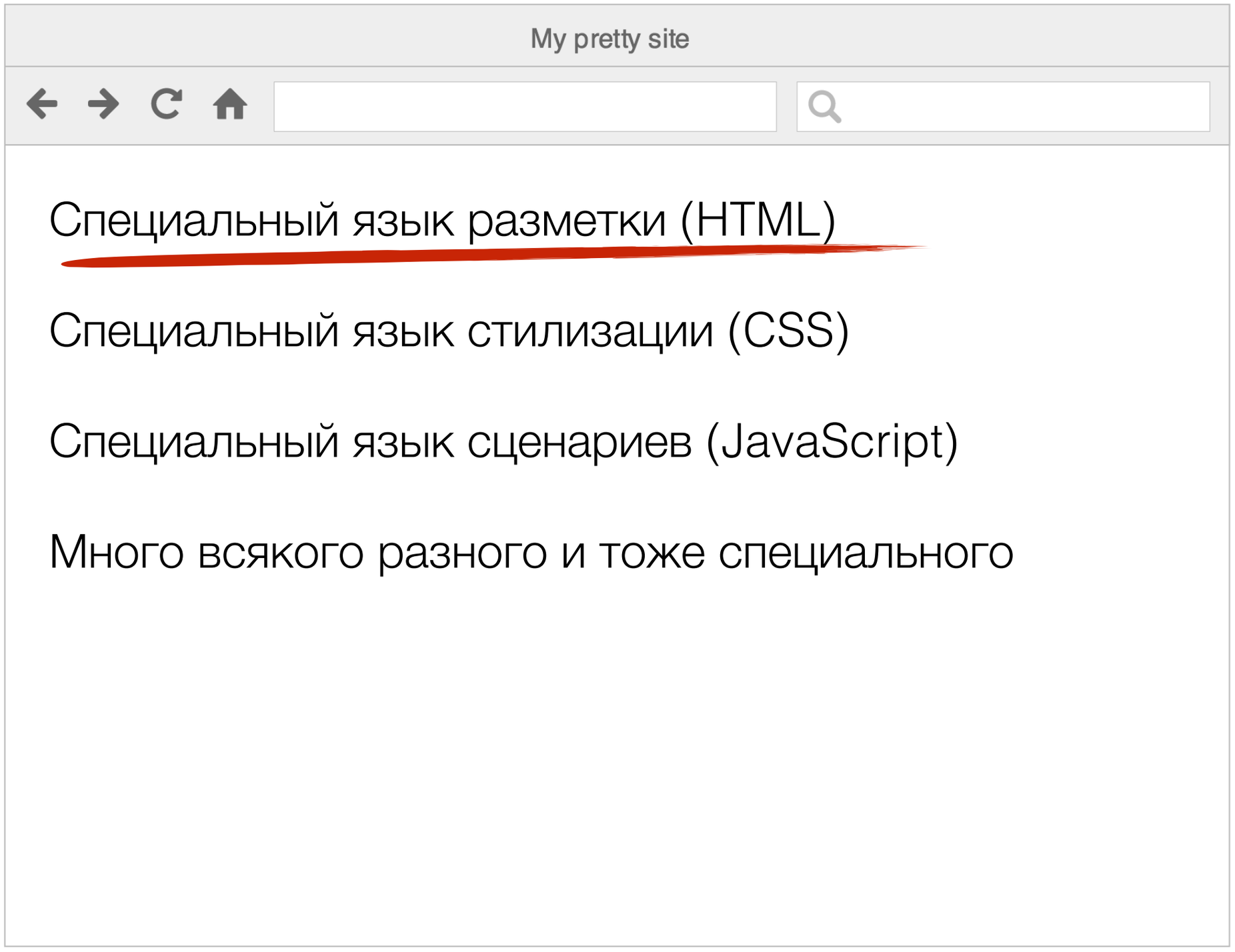
Scalable Vector Graphic



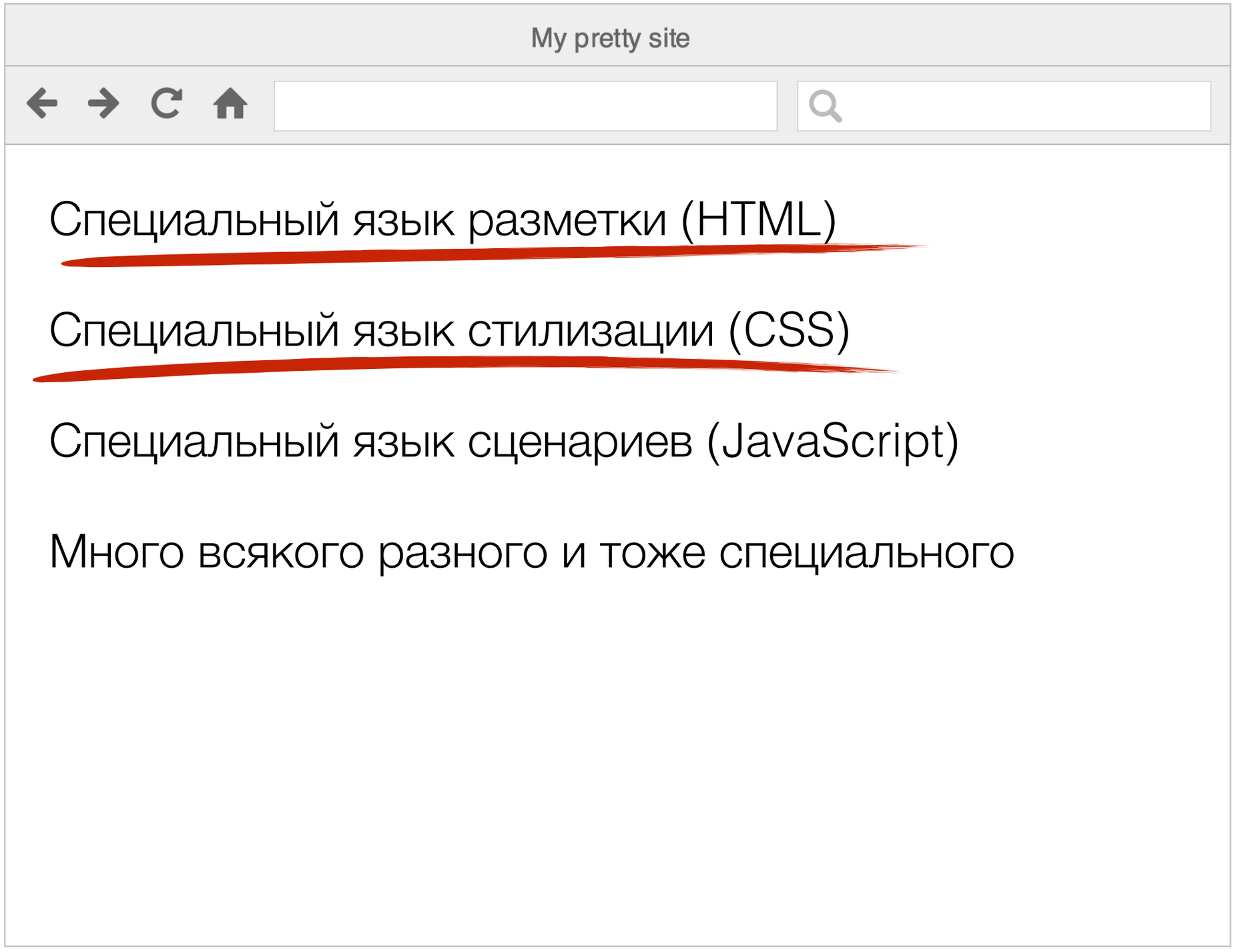
Links

- 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

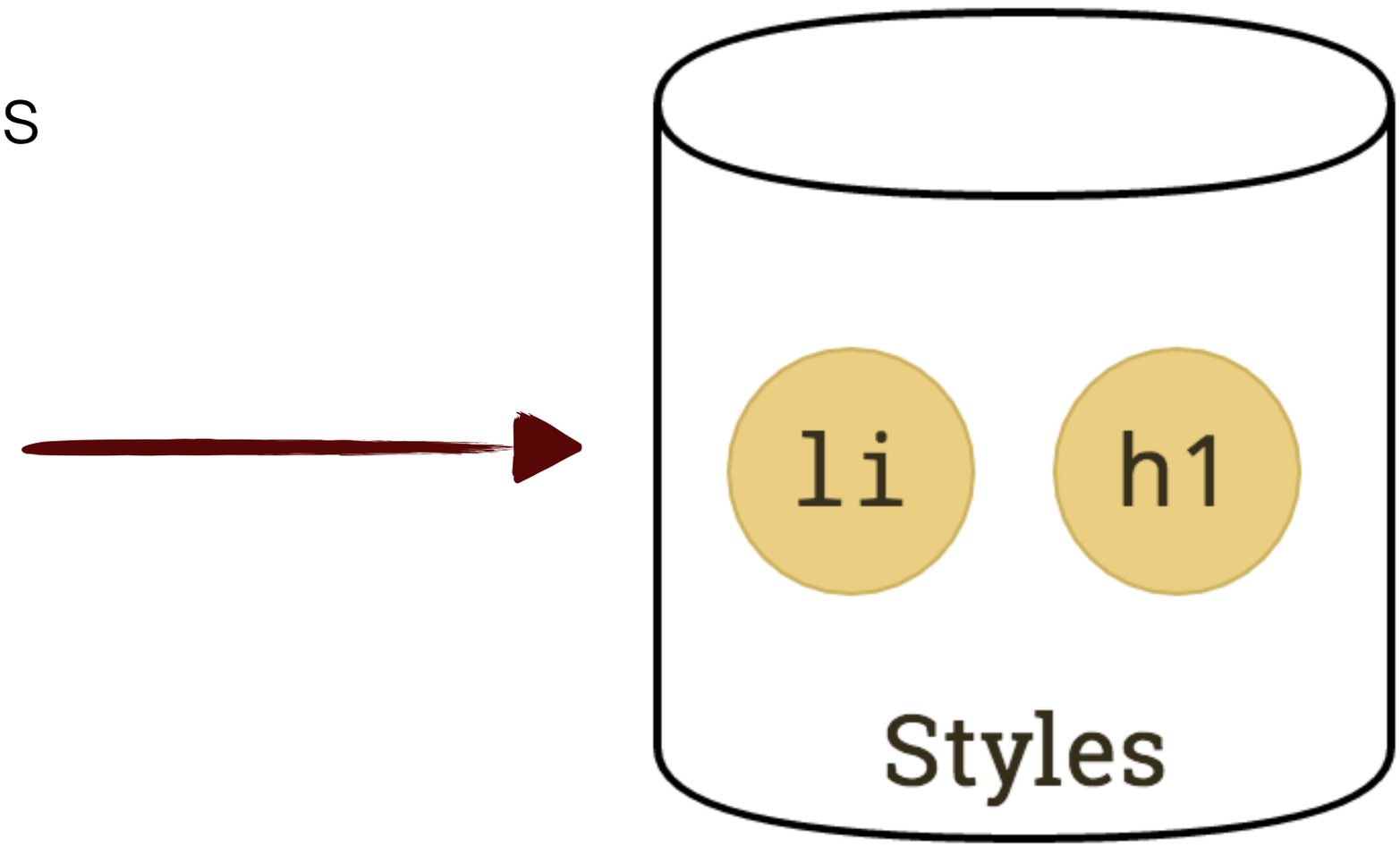
```
01: h1 {
02:     color: #000000;
03     font-size: 18px;
04:     font-weight: bold;
05:     text-decoration: undeline;
06: }
07:
08: li {
09:     color: blue;
10:     font-family: Arial, sans-serif;
11: }
```

CSS

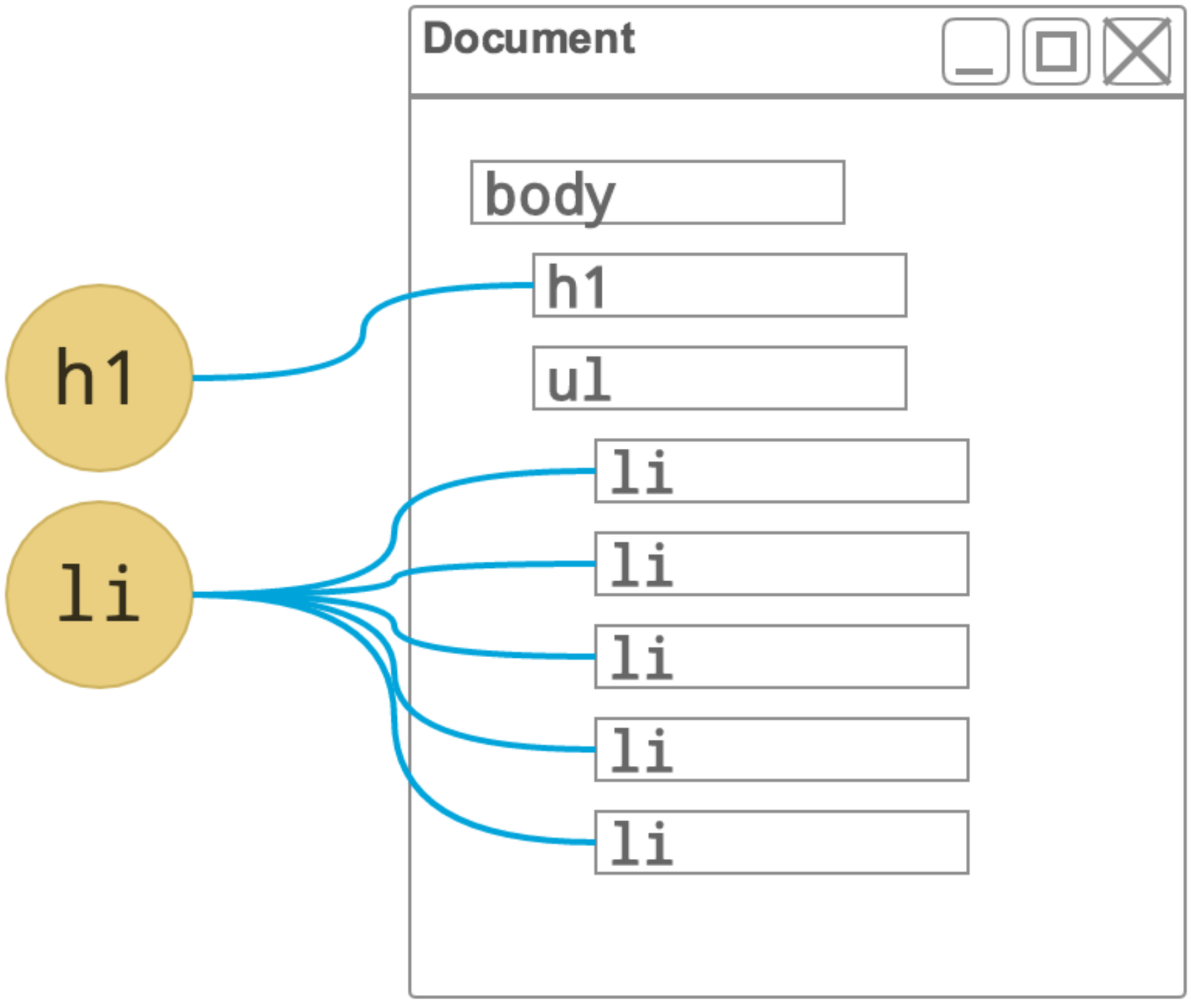
Cascading Style Sheets

styles.css

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

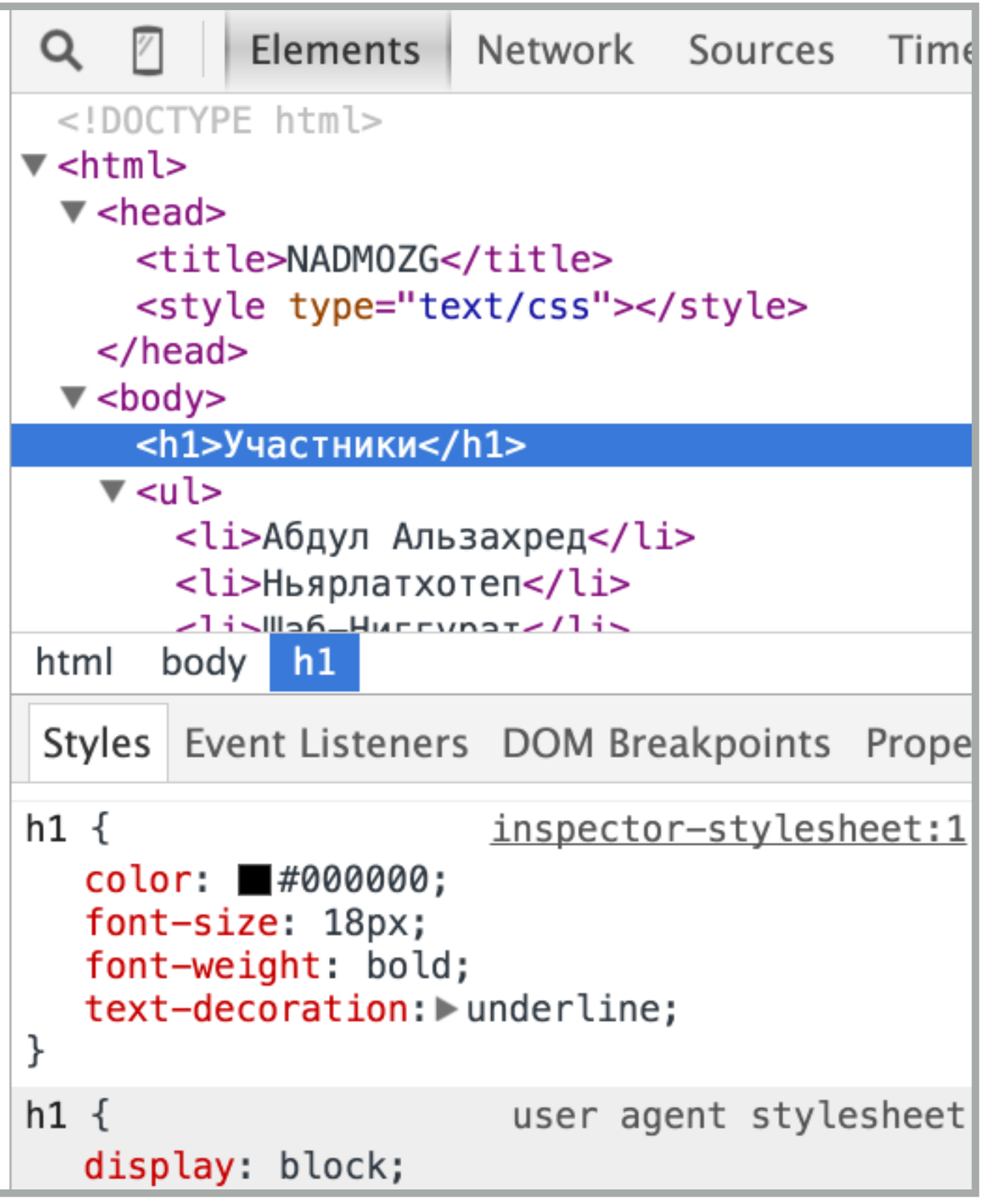


Style matching



Участники

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



Синтаксис

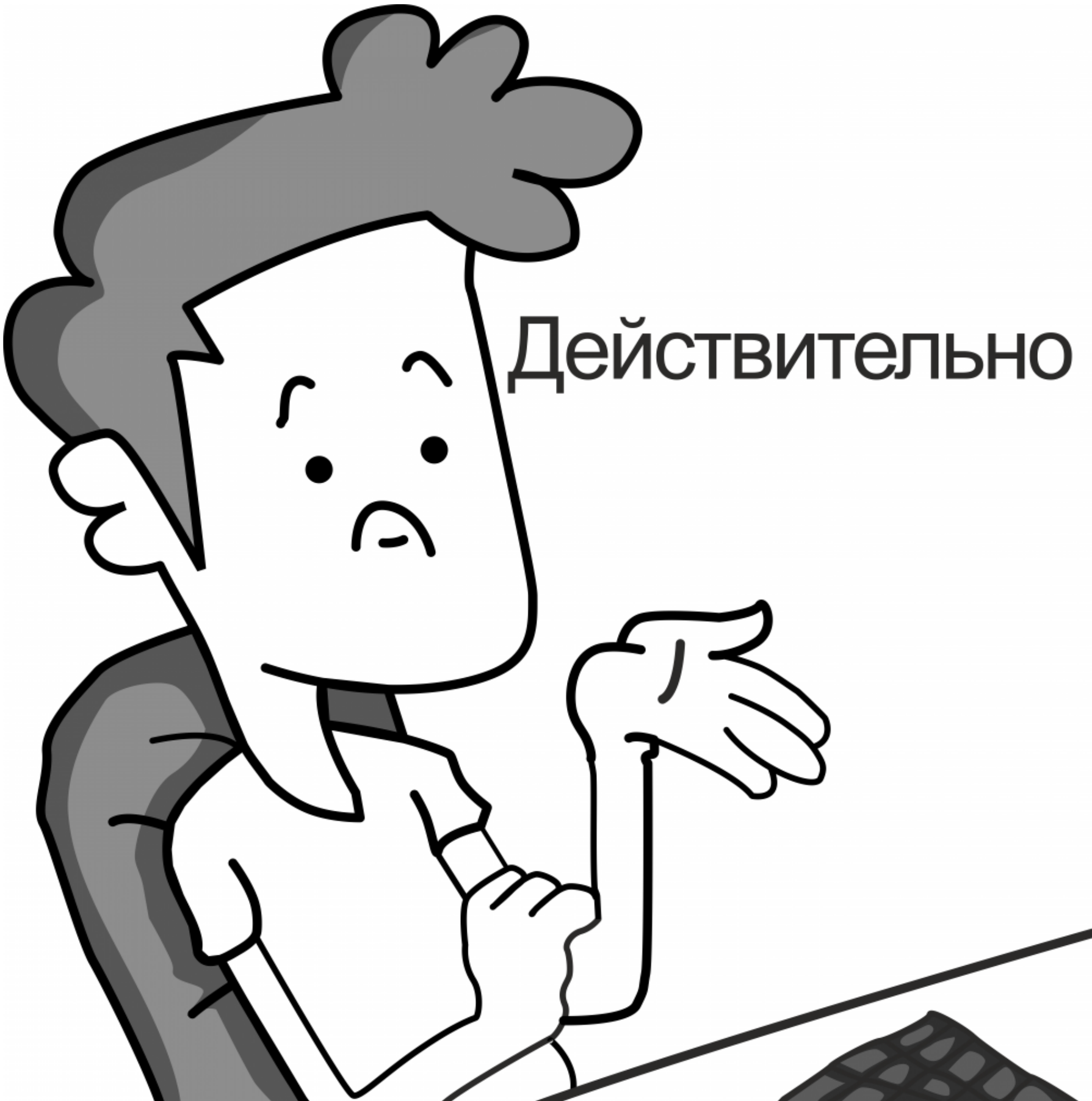
Ничего хитрого

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

Селекторы

Селекторы

селекторы - выбирают


Действительно

```
input#user.classOne.classTwo[type=“text”] {
    …
}
```

```
input#user.classOne.classTwo[type=“text”] {
    …
}
```

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

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

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

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

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

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

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

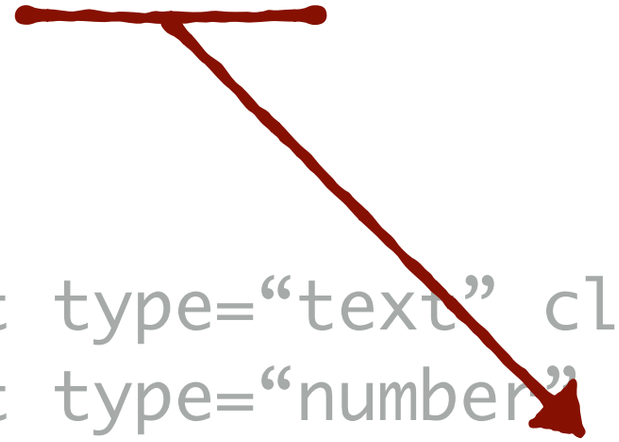
```
input#user.classOne.classTwo[type="text"] {
    …
}
```

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

input#user.classOne.classTwo[type=“text”]

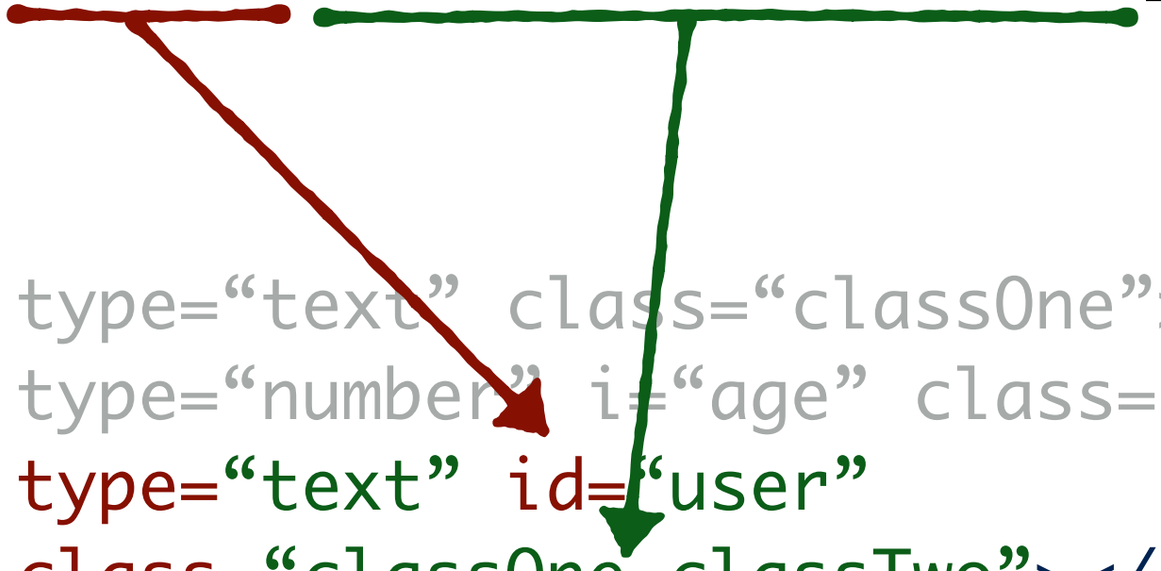
```
<input type=“text” class=“classOne”></input>
<input type=“number” i=“age” class=“classOne”></input>
<input type=“text” id=“user”
       class=“classOne classTwo”></input>
<input type=“text” class=“classOne”></input>
<input type=“text” name=“first-name”></input>
<input class=“classOne” placeholder=“Hello”></input>
```

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



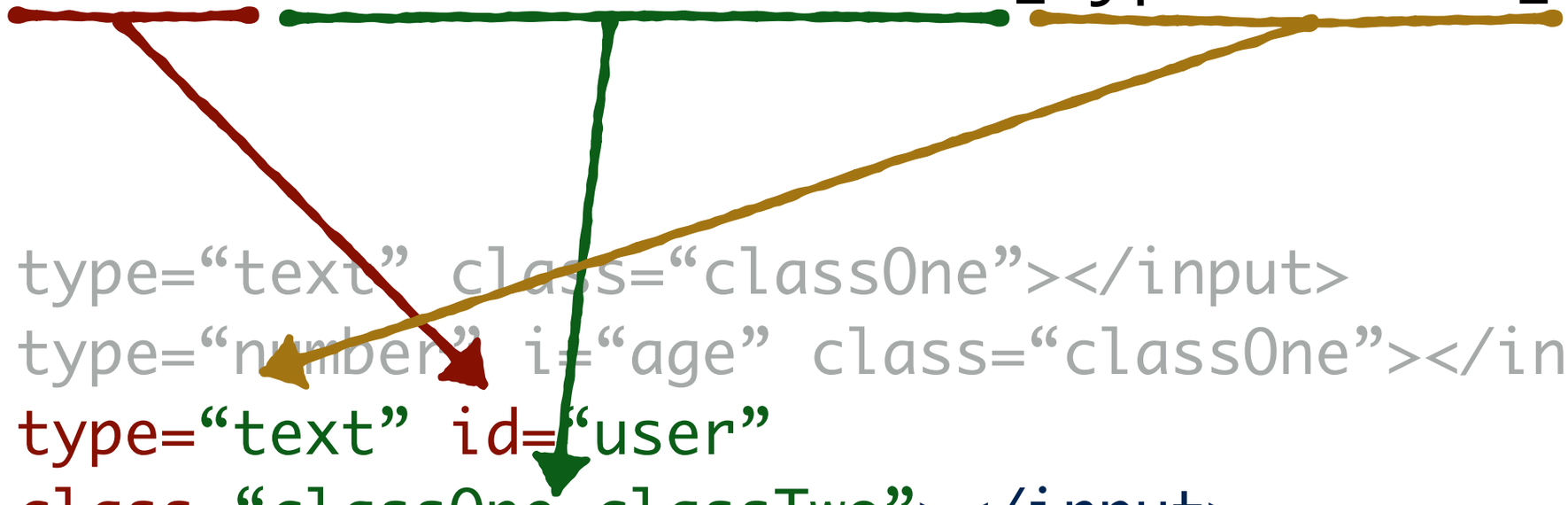
```
01: <input type="text" class="classOne"></input>
02: <input type="number" i="age" class="classOne"></input>
03: <input type="text" id="user"
           class="classOne classTwo"></input>
04: <input type="text" class="classOne"></input>
05: <input type="text" name="first-name"></input>
06: <input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

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



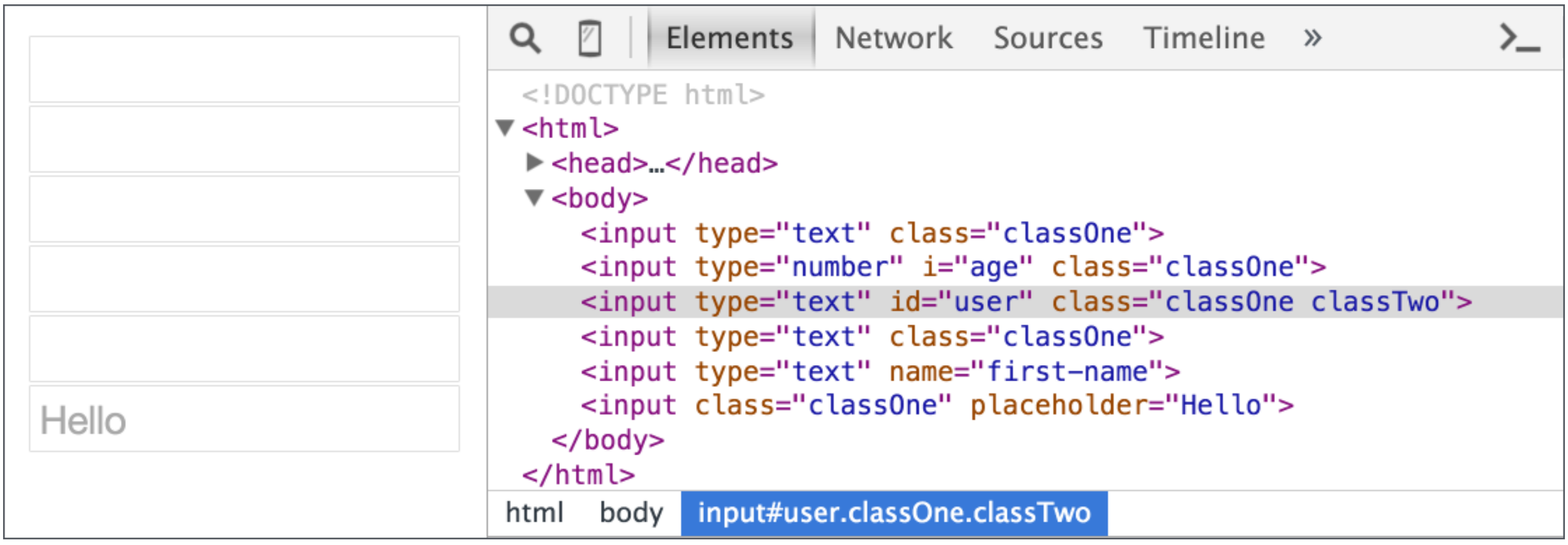
```
01: <input type="text" class="classOne"></input>
02: <input type="number" i="age" class="classOne"></input>
03: <input type="text" id="user"
           class="classOne classTwo"></input>
04: <input type="text" class="classOne"></input>
05: <input type="text" name="first-name"></input>
06: <input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

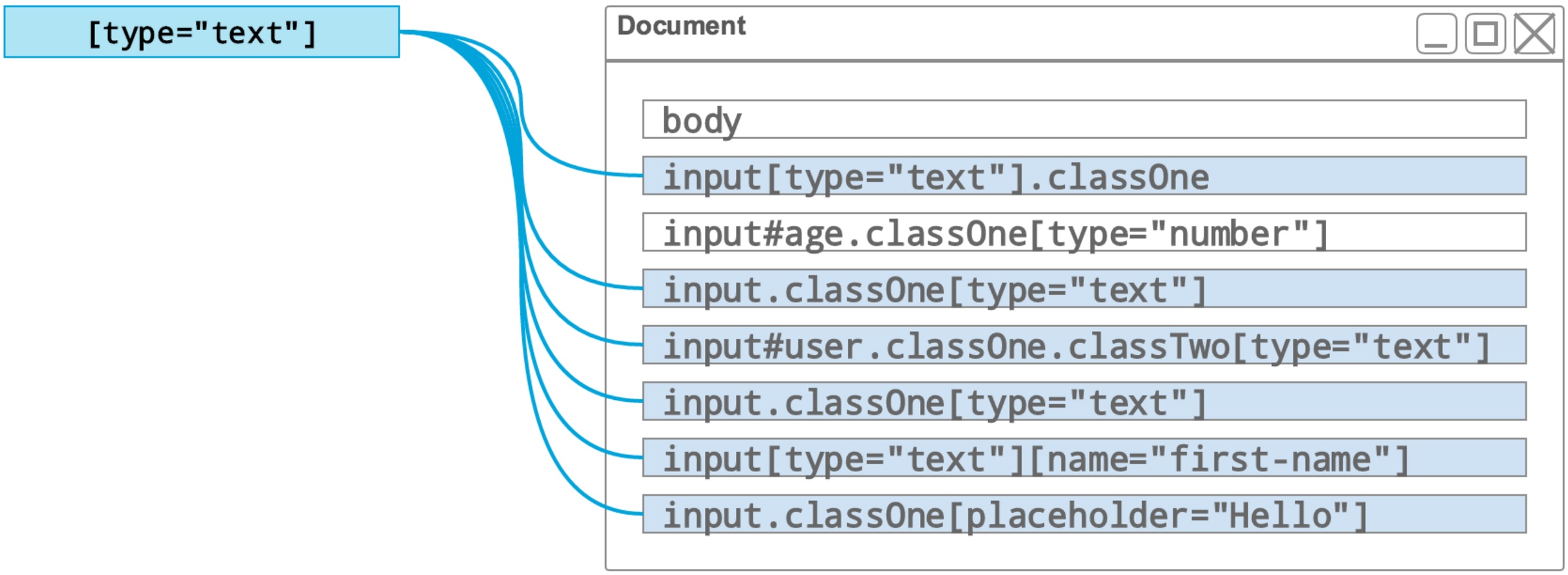
input#user.classOne.classTwo[type="text"]

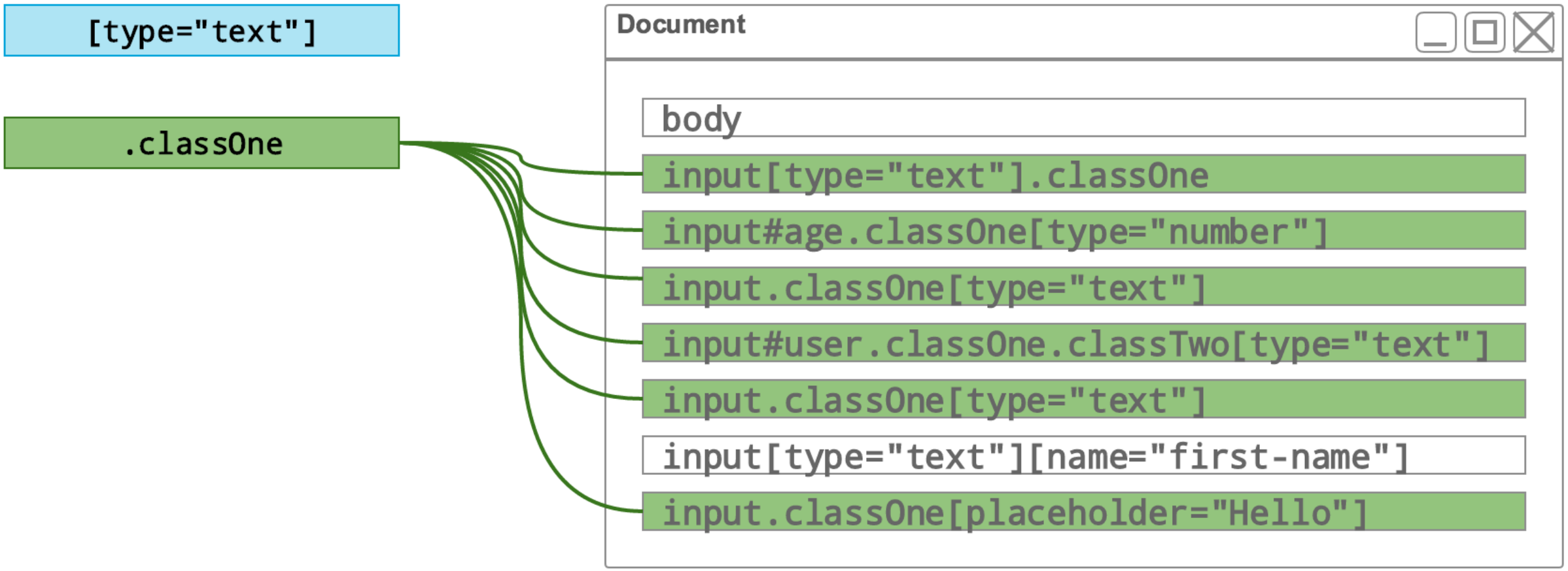


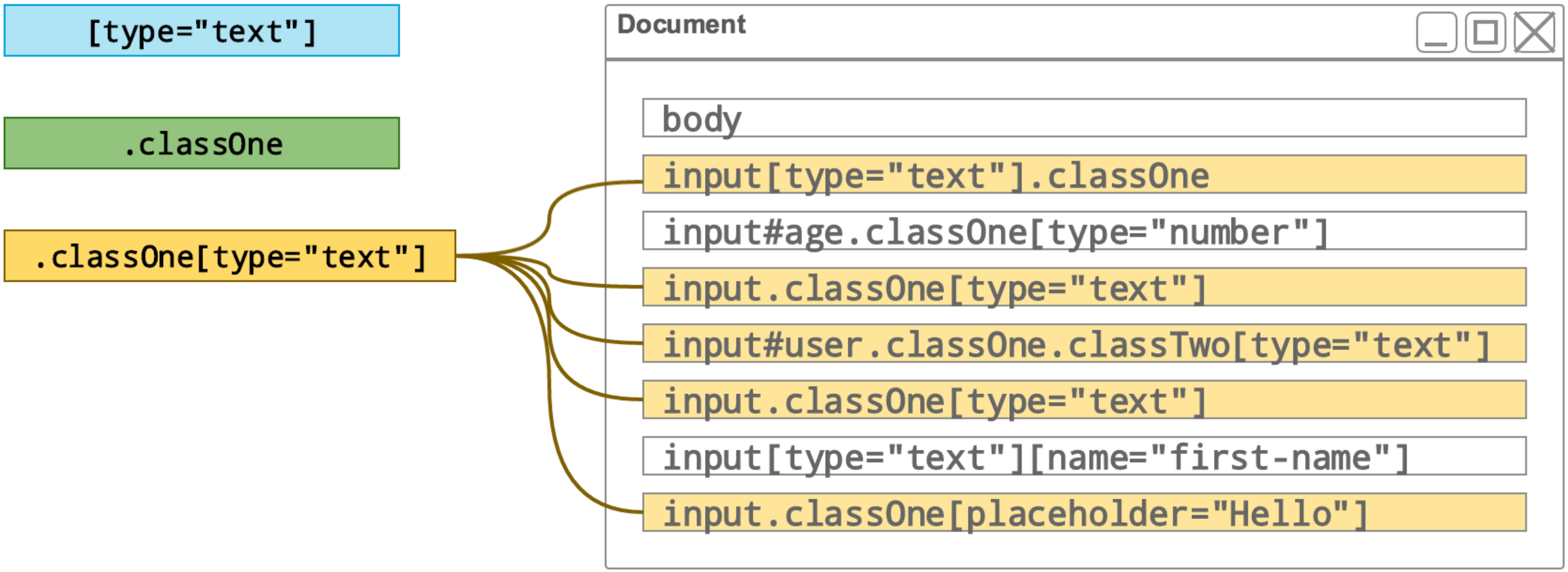
```
01: <input type="text" class="classOne"></input>
02: <input type="number" i="age" class="classOne"></input>
03: <input type="text" id="user"
           class="classOne classTwo"></input>
04: <input type="text" class="classOne"></input>
05: <input type="text" name="first-name"></input>
06: <input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

input#user.classOne.classTwo[type=“text”]




[type="text"]
Document
body
input[type="text"].classOne
input#age.classOne[type="number"]
input.classOne[type="text"]
input#user.classOne.classTwo[type="text"]
input.classOne[type="text"]
input[type="text"][name="first-name"]
input.classOne[placeholder="Hello"]


[type="text"]
.classOne
Document
body
input[type="text"].classOne
input#age.classOne[type="number"]
input.classOne[type="text"]
input#user.classOne.classTwo[type="text"]
input.classOne[type="text"]
input[type="text"][name="first-name"]
input.classOne[placeholder="Hello"]


[type="text"]
.classOne
.classOne[type="text"]
Document
body
input[type="text"].classOne
input#age.classOne[type="number"]
input.classOne[type="text"]
input#user.classOne.classTwo[type="text"]
input.classOne[type="text"]
input[type="text"][name="first-name"]
input.classOne[placeholder="Hello"]

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
```

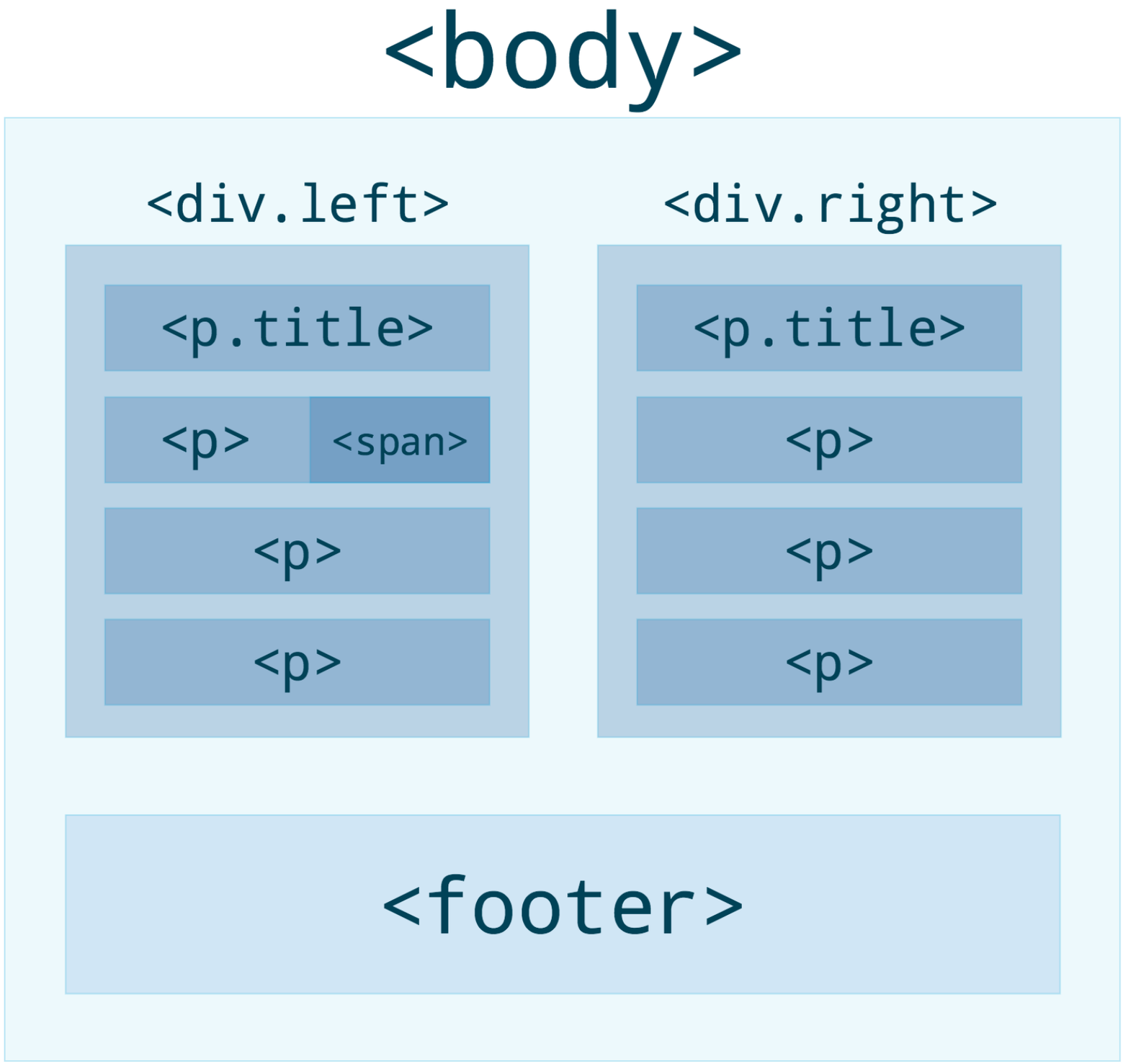
Все элементы <a>, которые находятся внутри элемента <li> с классом active, который находится внутри элемента <ul>, который находится в элемента <nav>, который находится внутри элемента <body>

Каскадирование

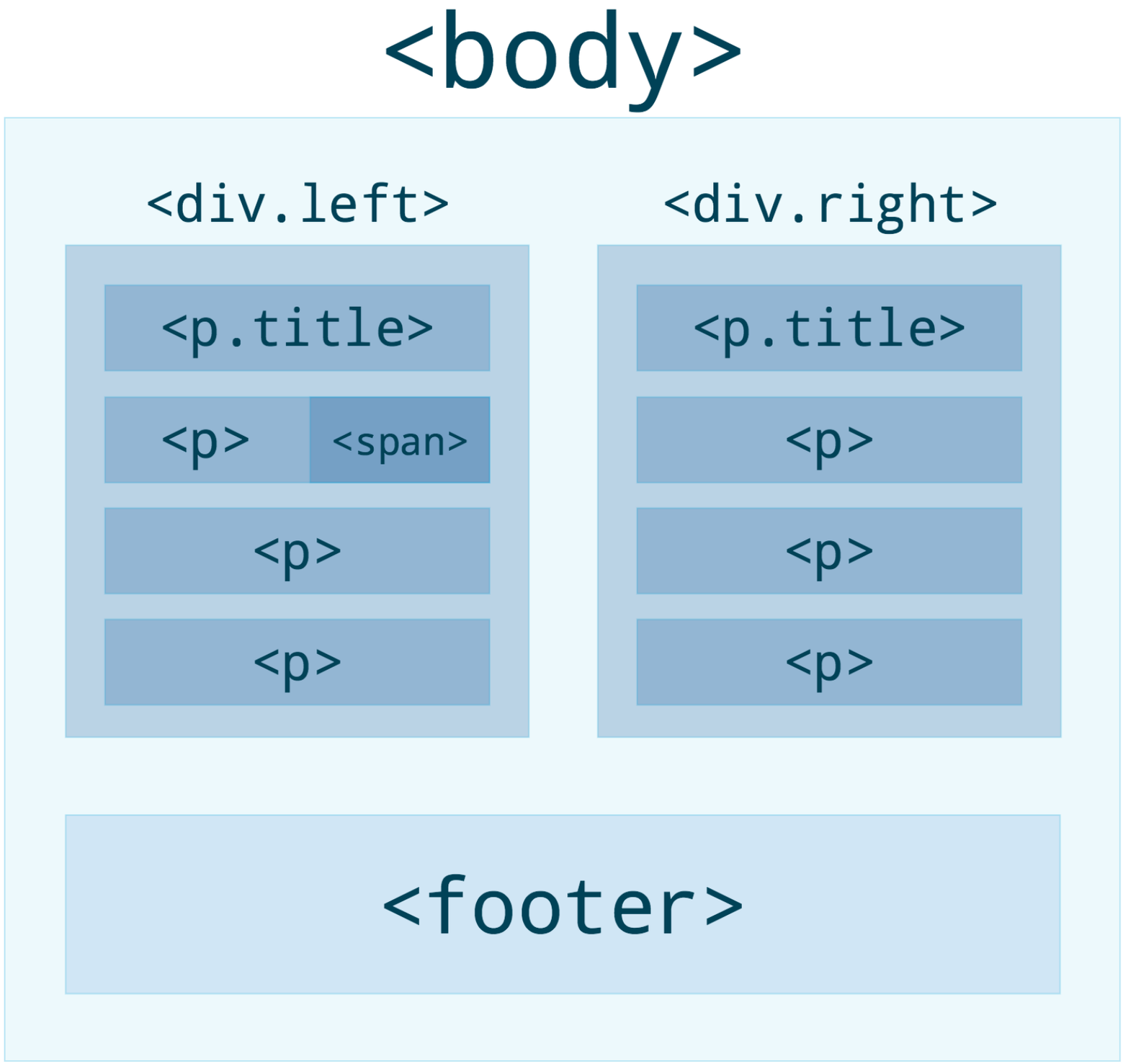
```
body nav ul li.active a
```

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

который построил Джек

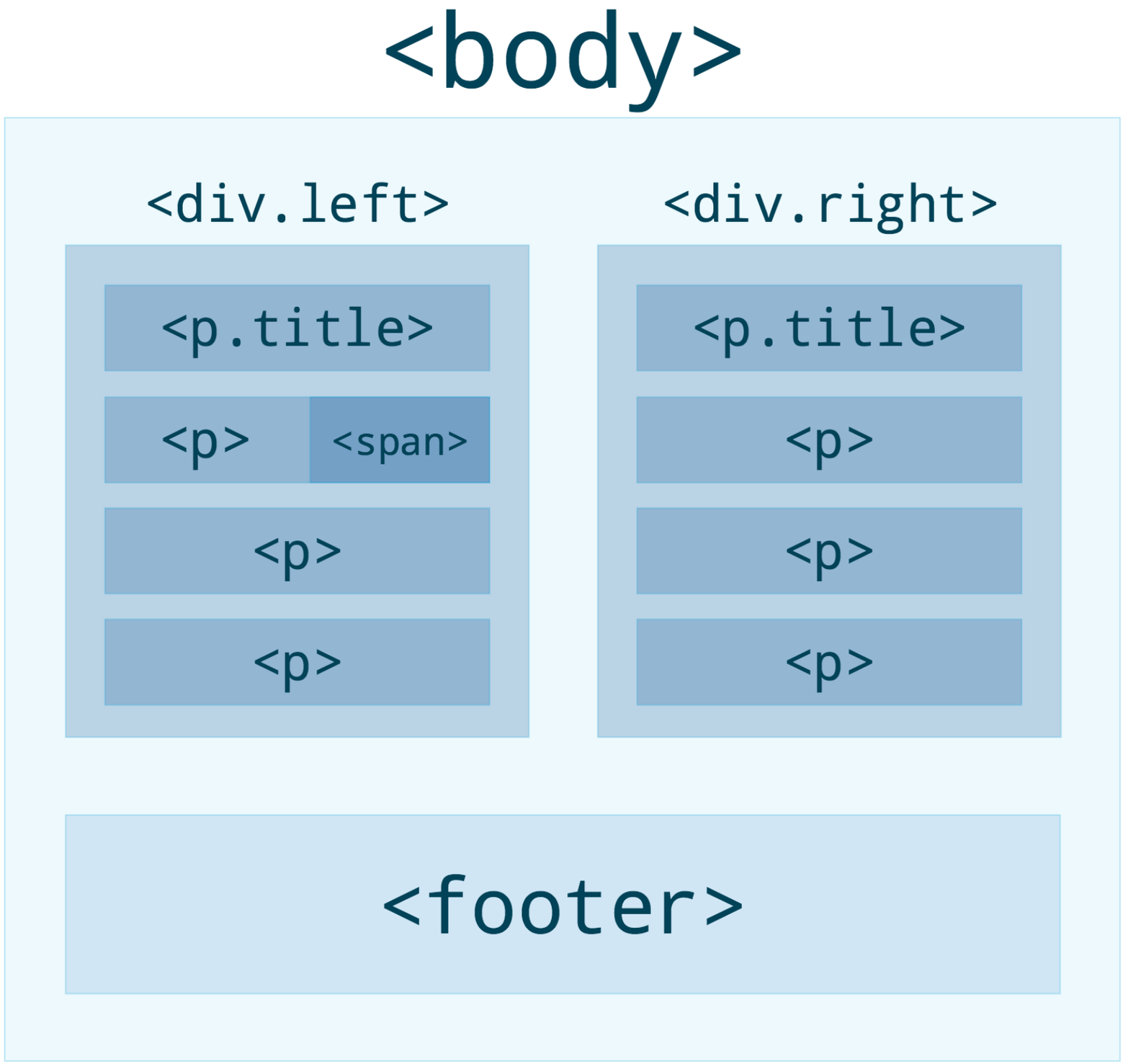

<body>
<div.left>
<p.title>
<p>

<p>
<p>
<div.right>
<p.title>
<p>
<p>
<p>
<footer>

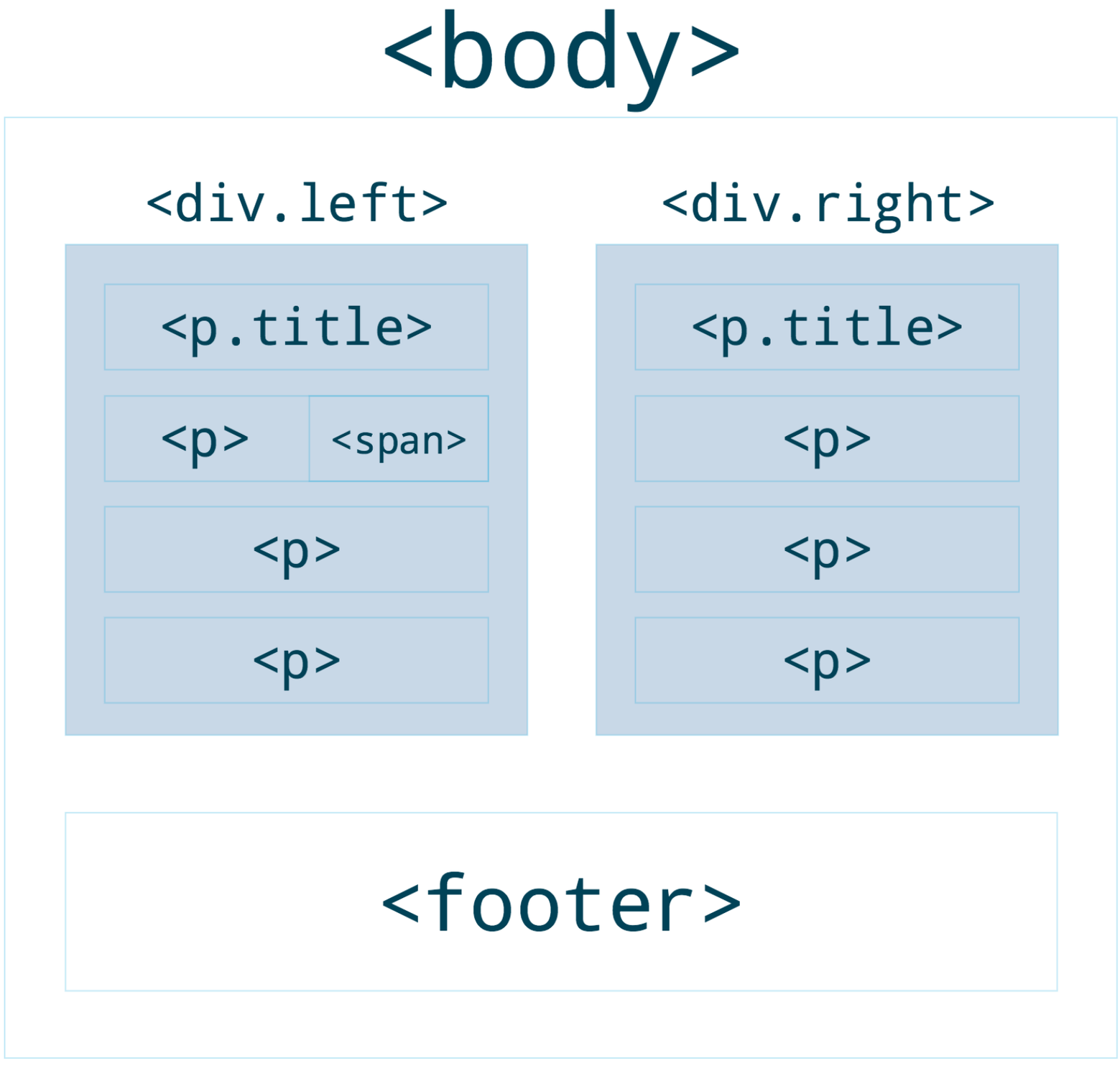

<body>
<div.left>
<p.title>
<p>

<p>
<p>
<div.right>
<p.title>
<p>
<p>
<p>
<footer>

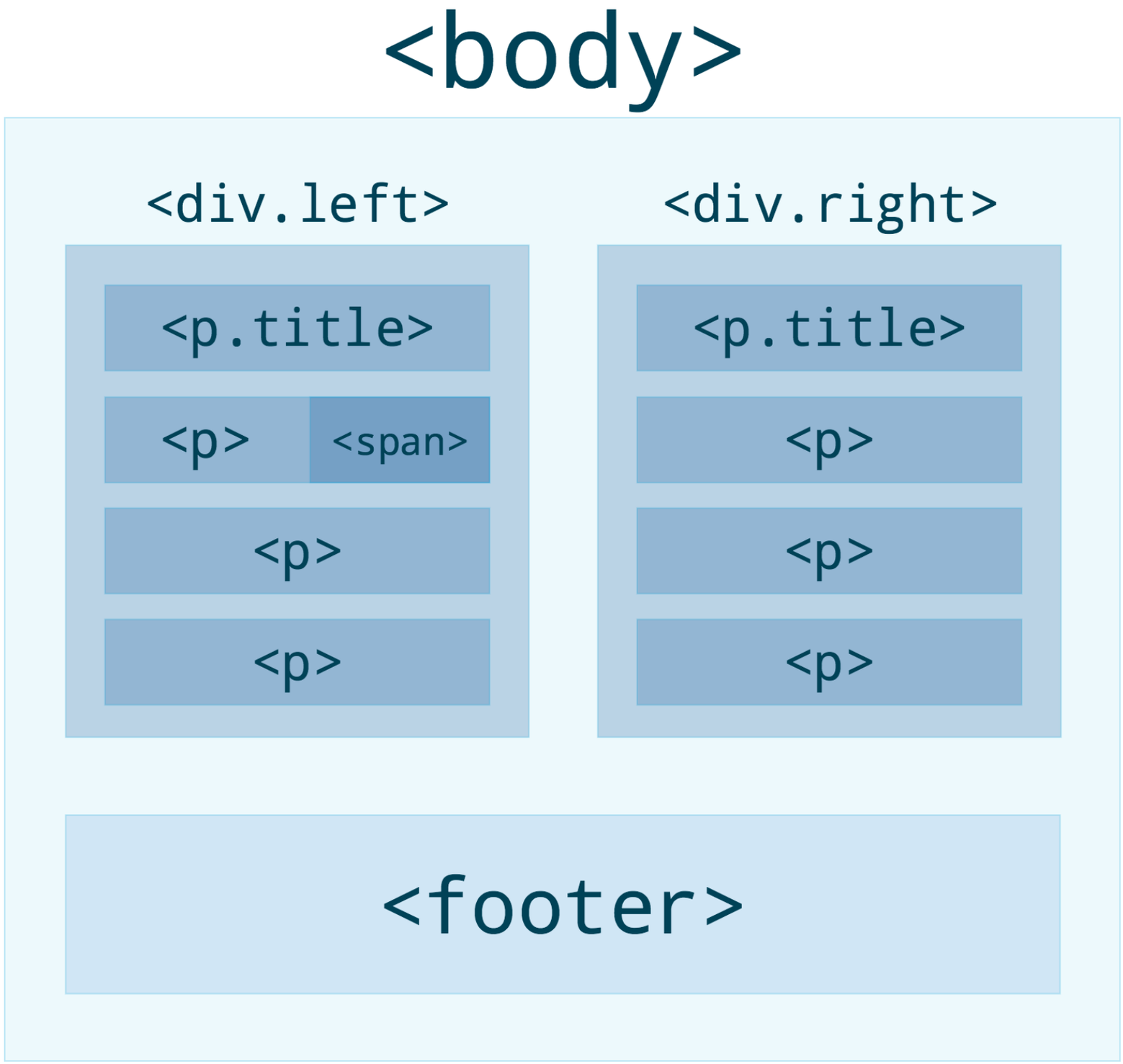
*



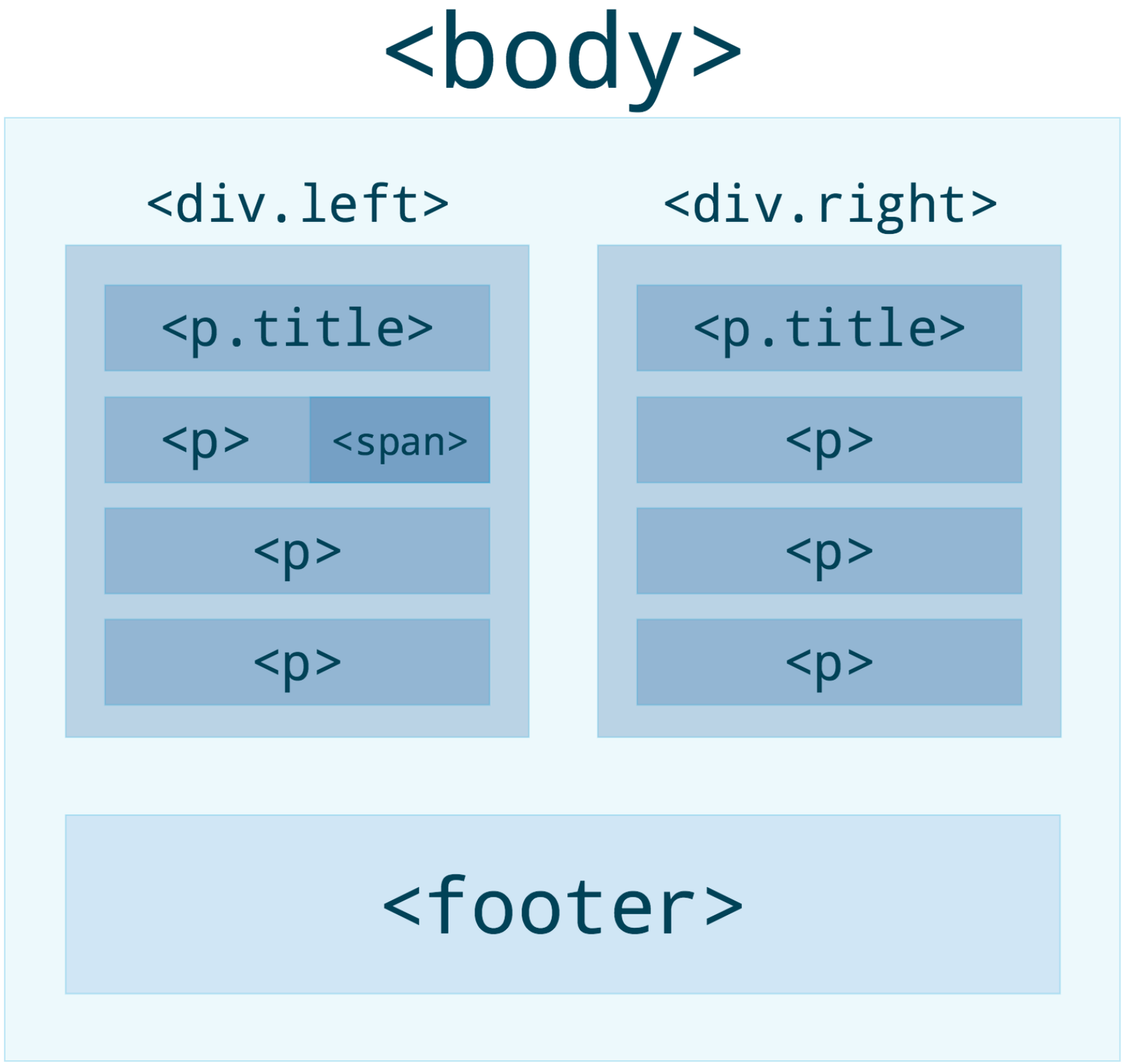
div



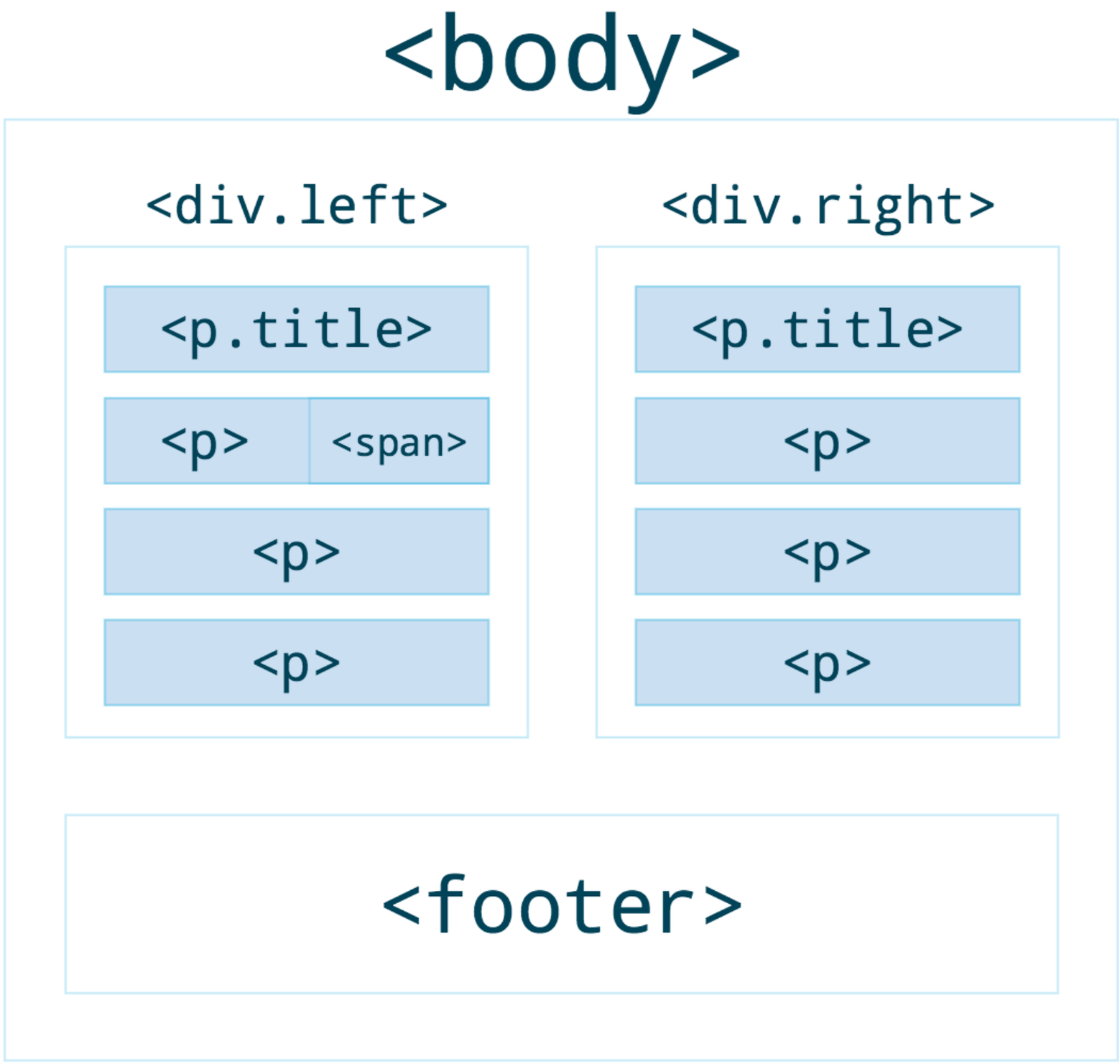
div



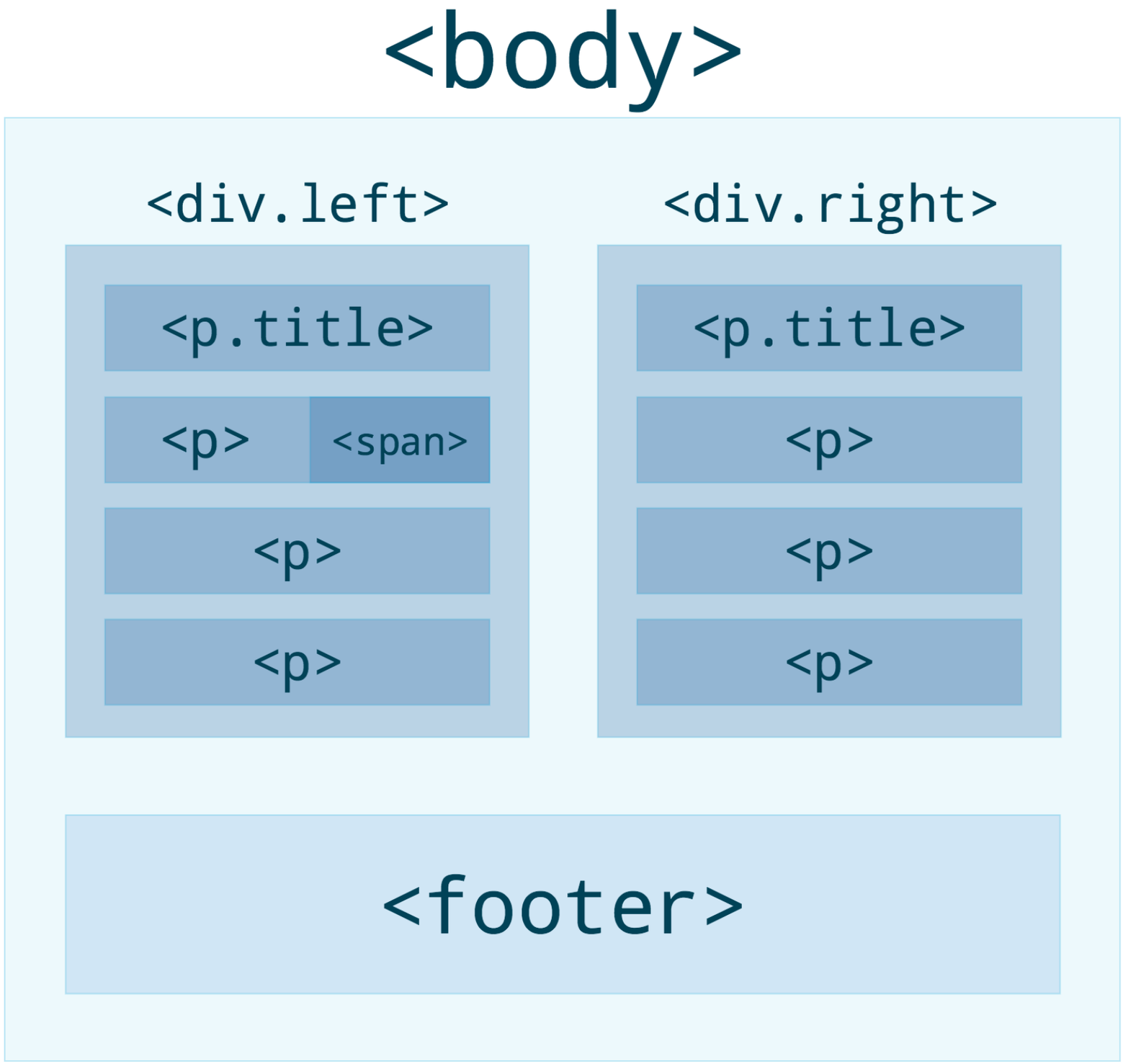
div.left



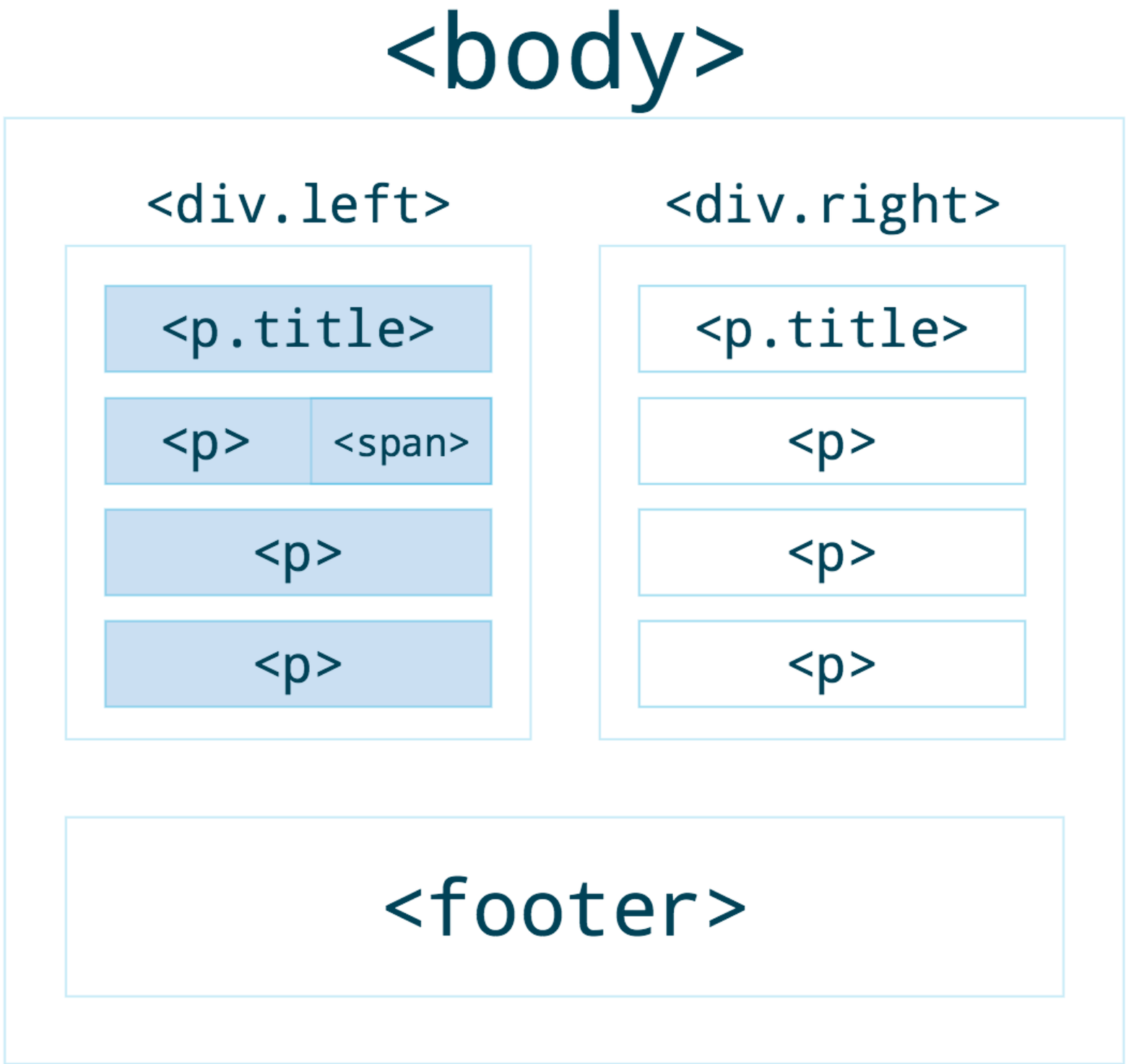
div p



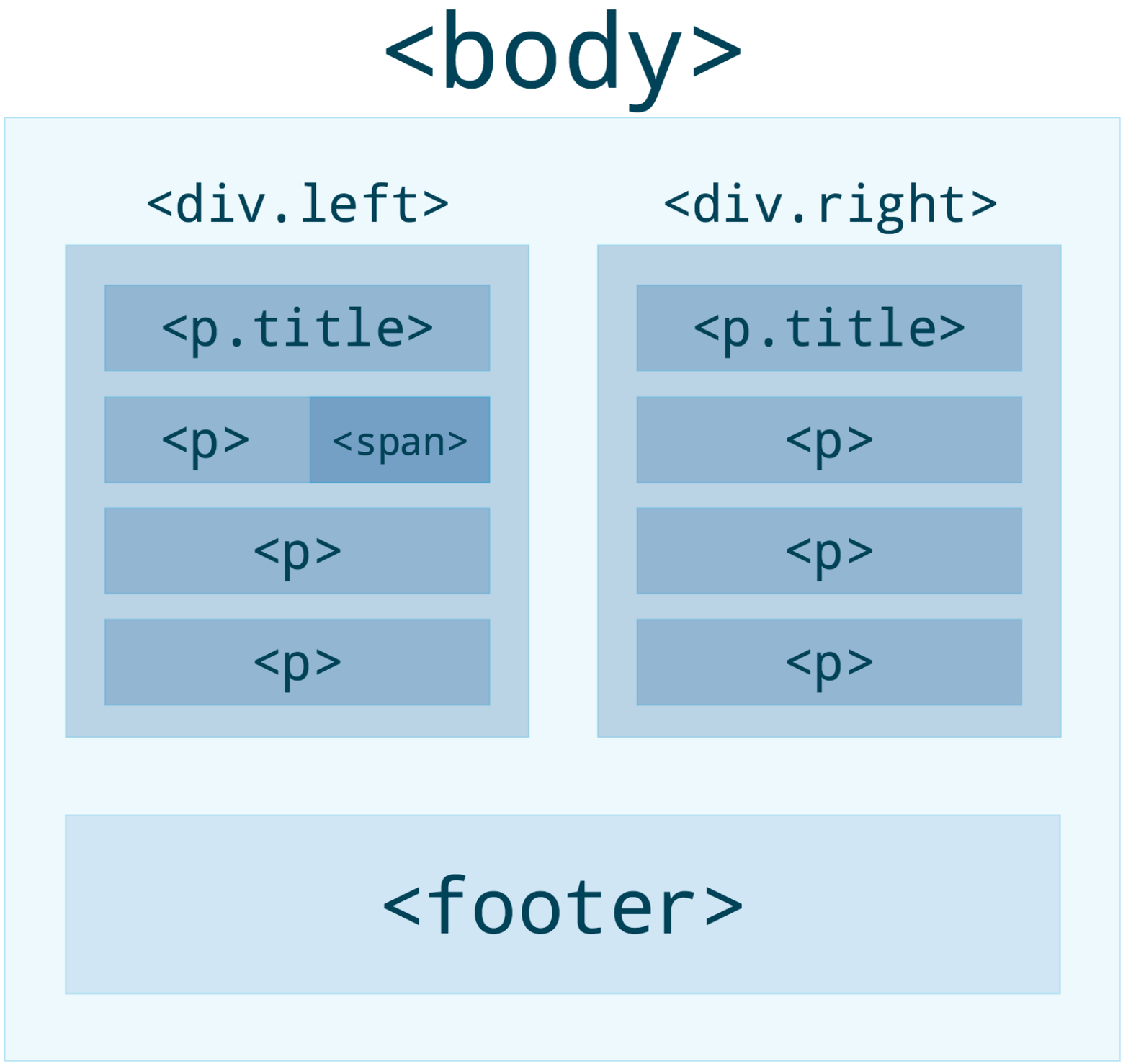
div p



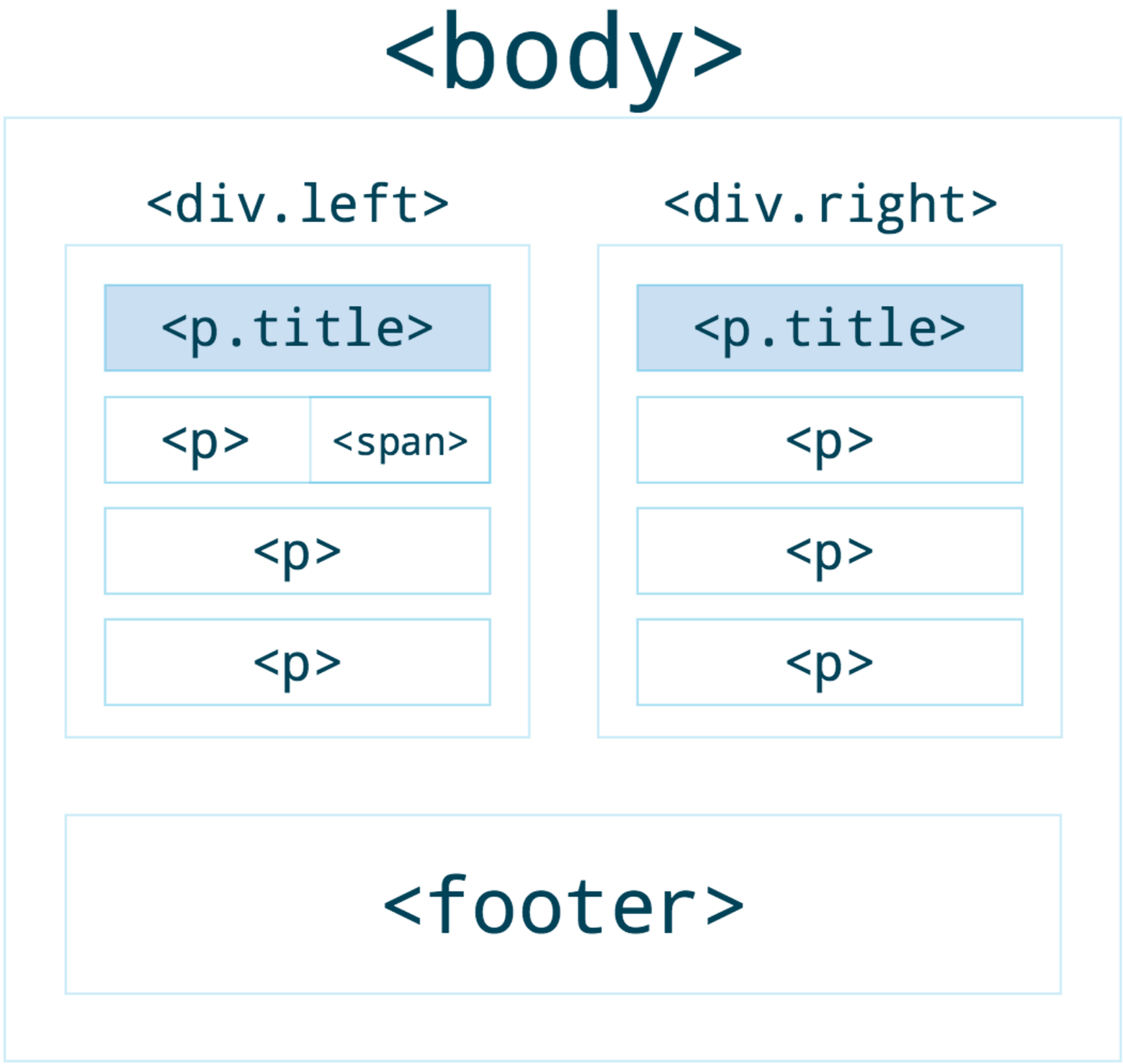
div.left p



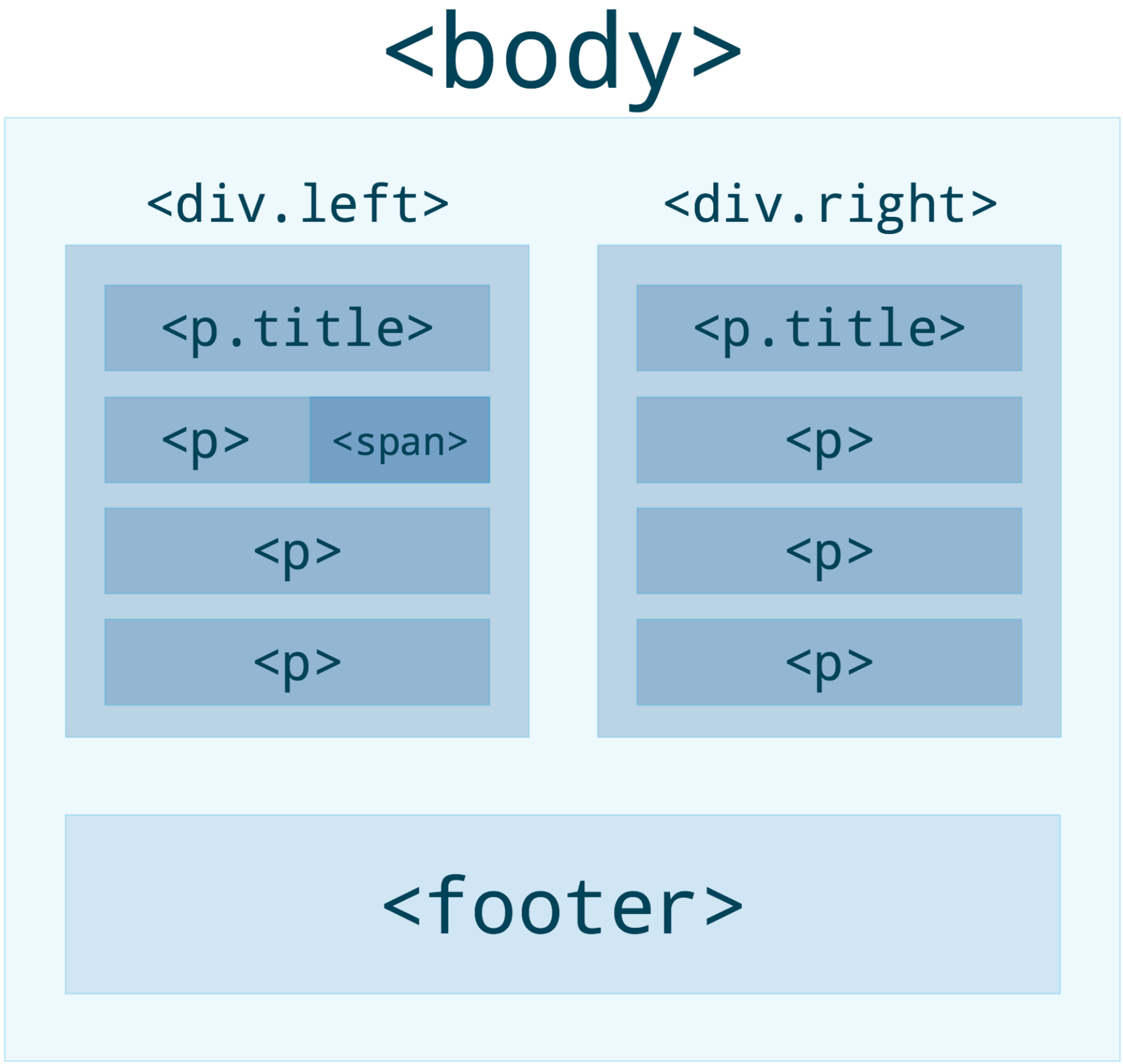
div.left p



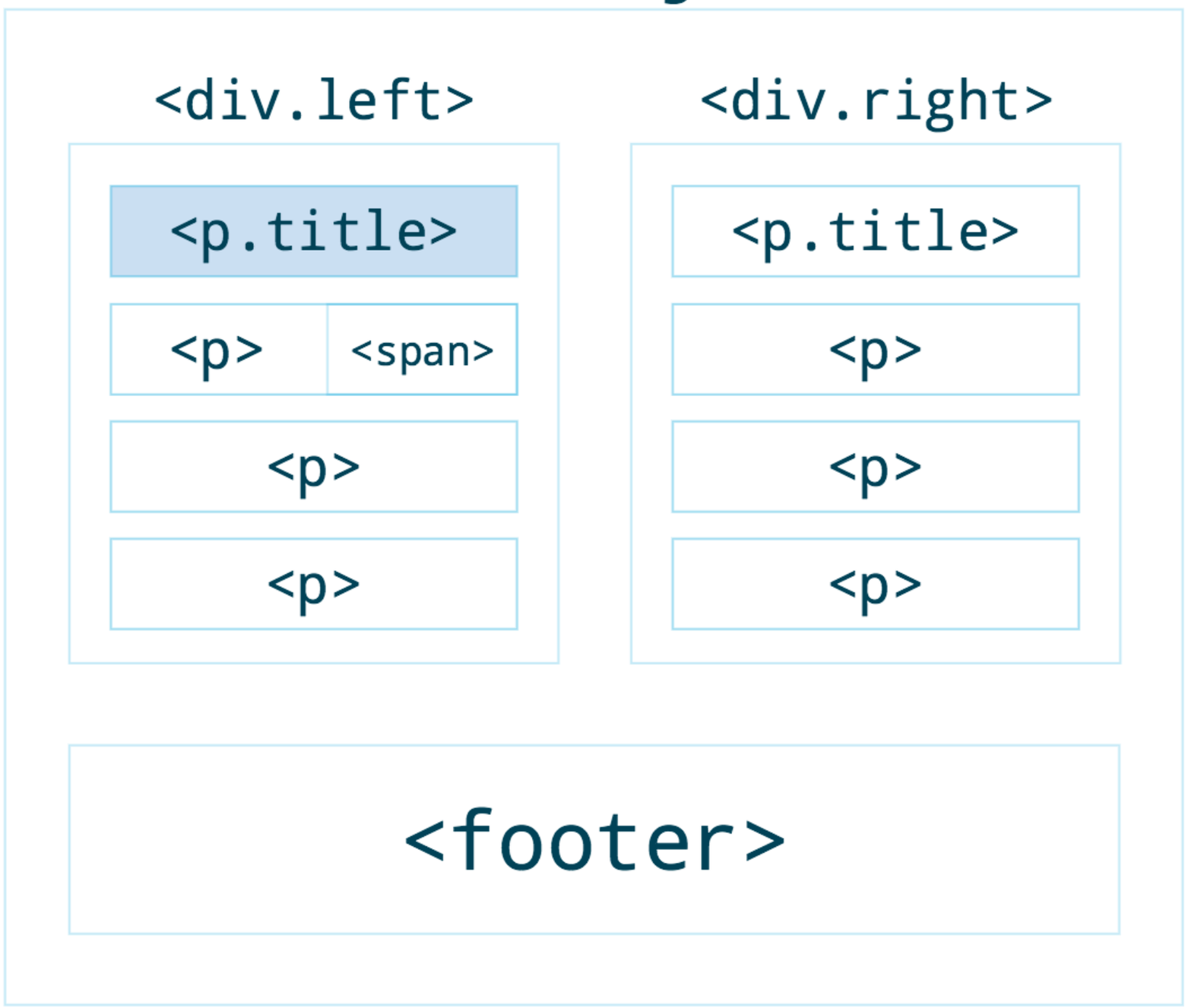
.title



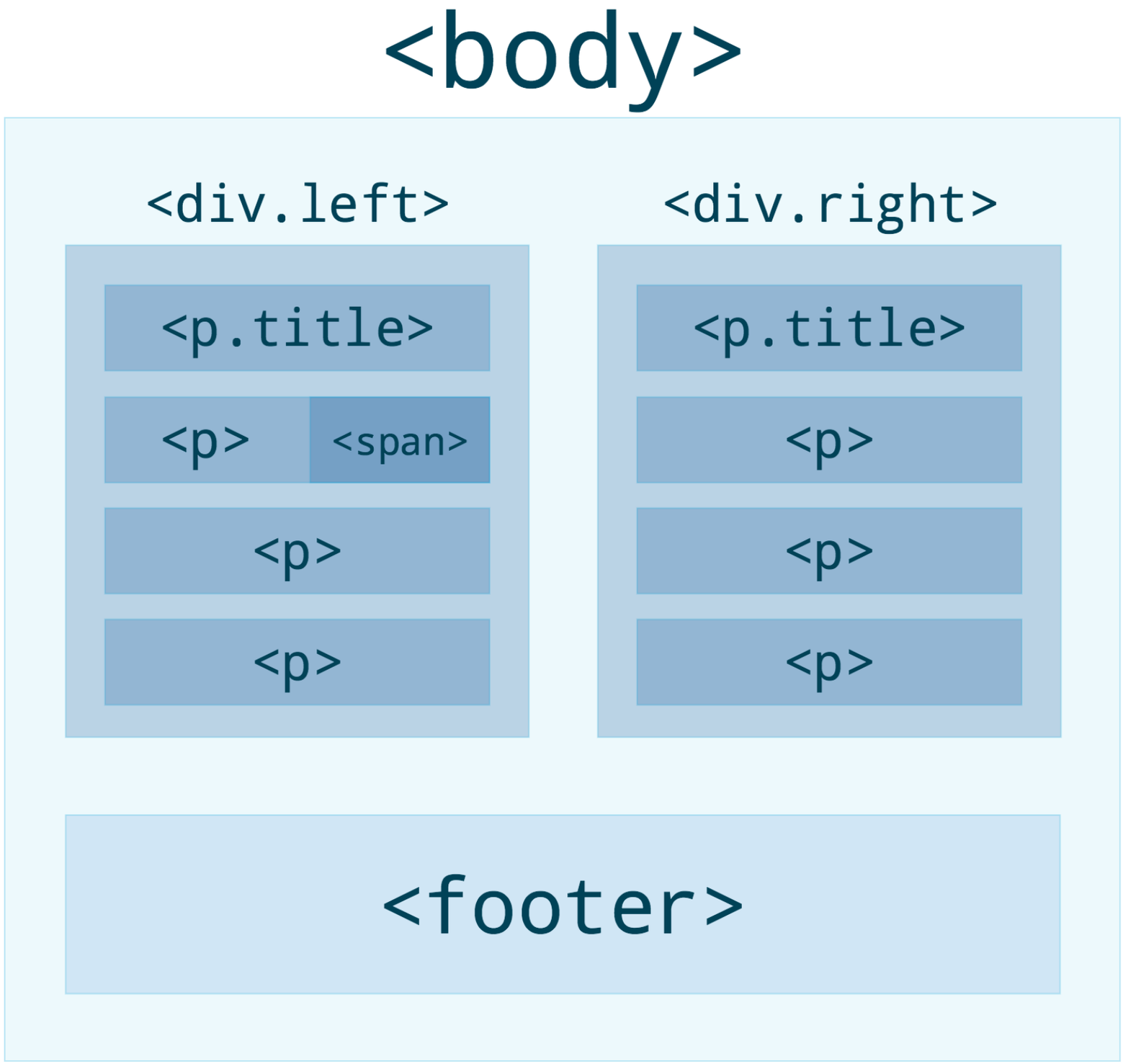
.title



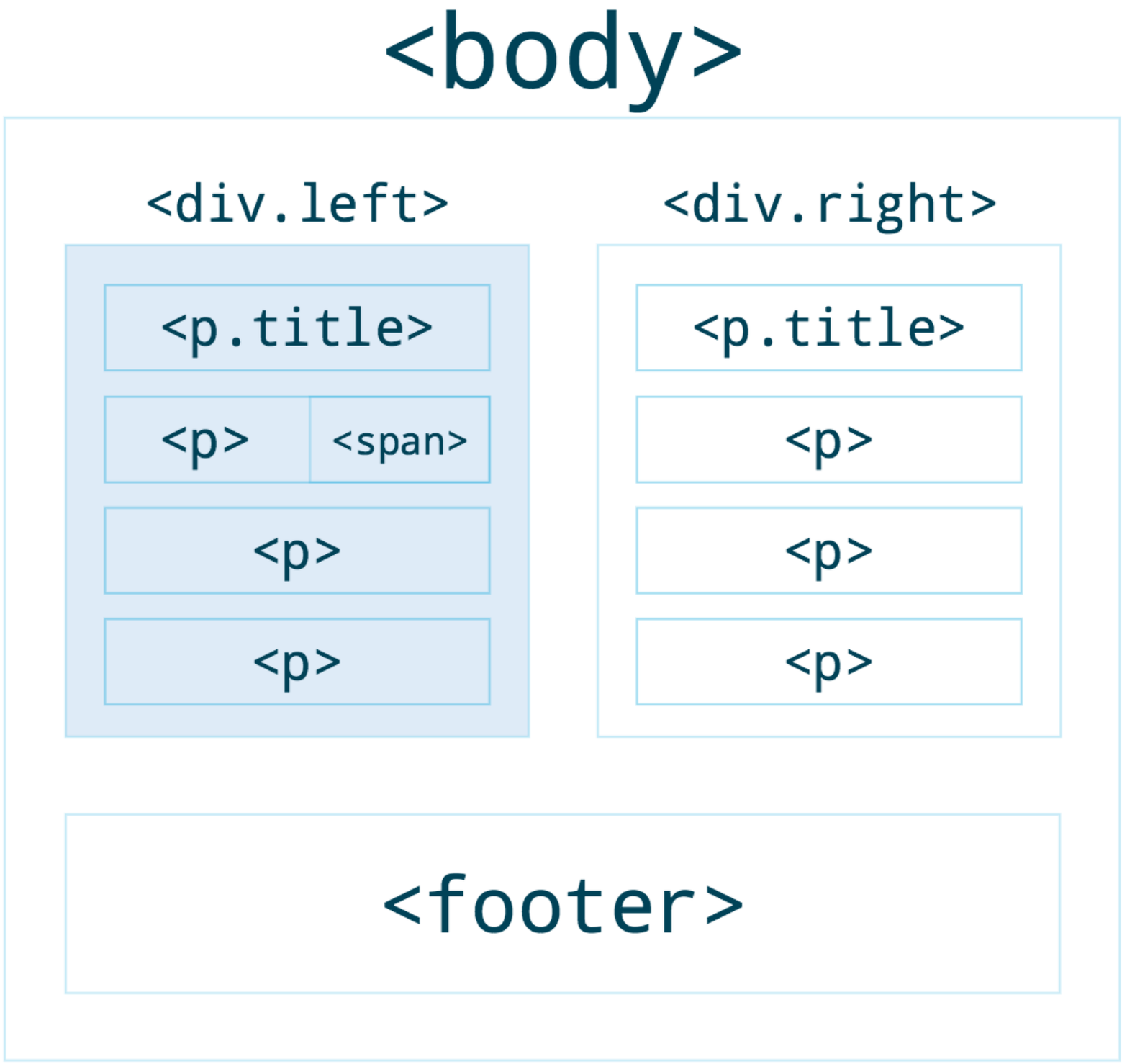
.left .title



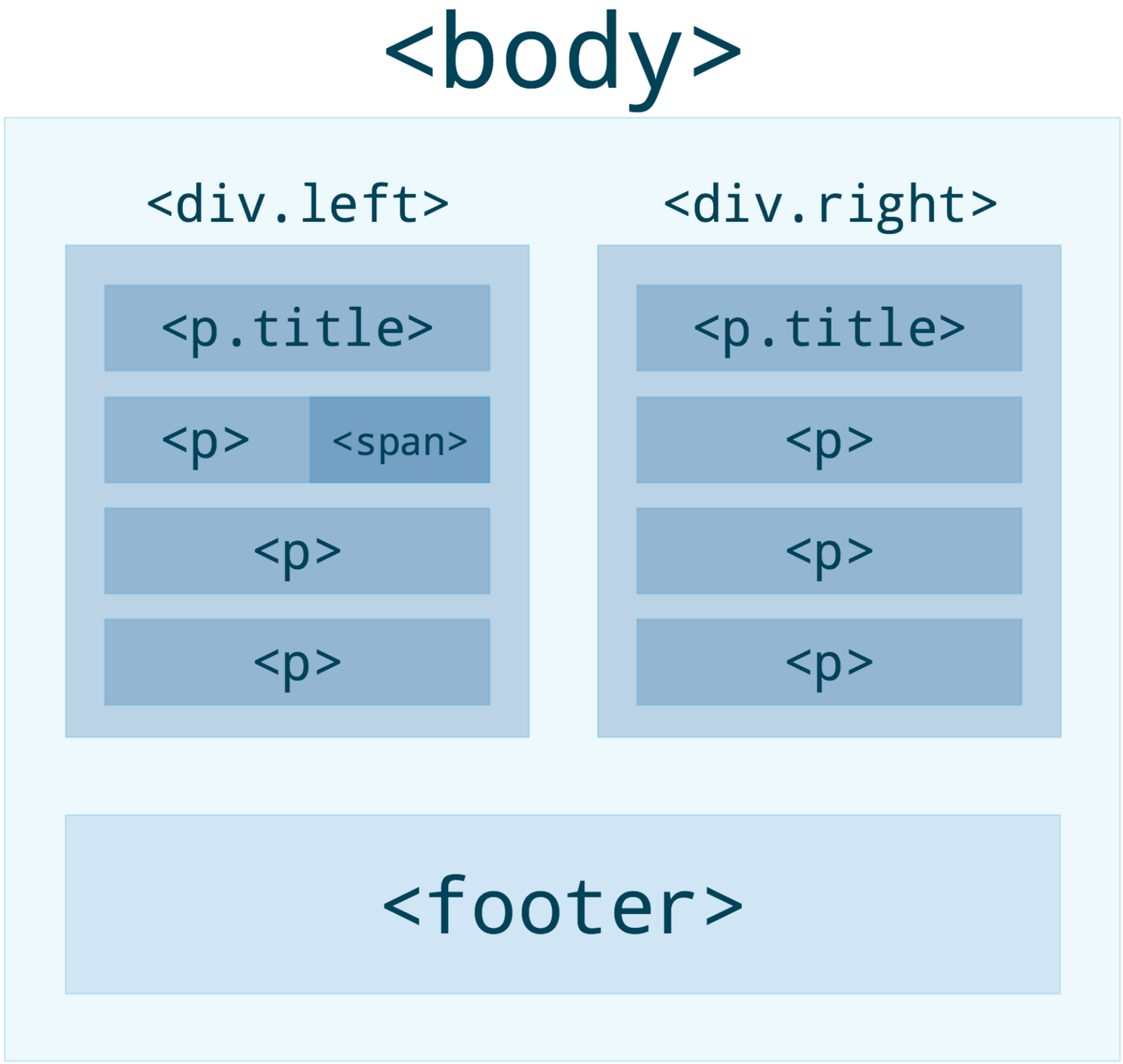
.left .title



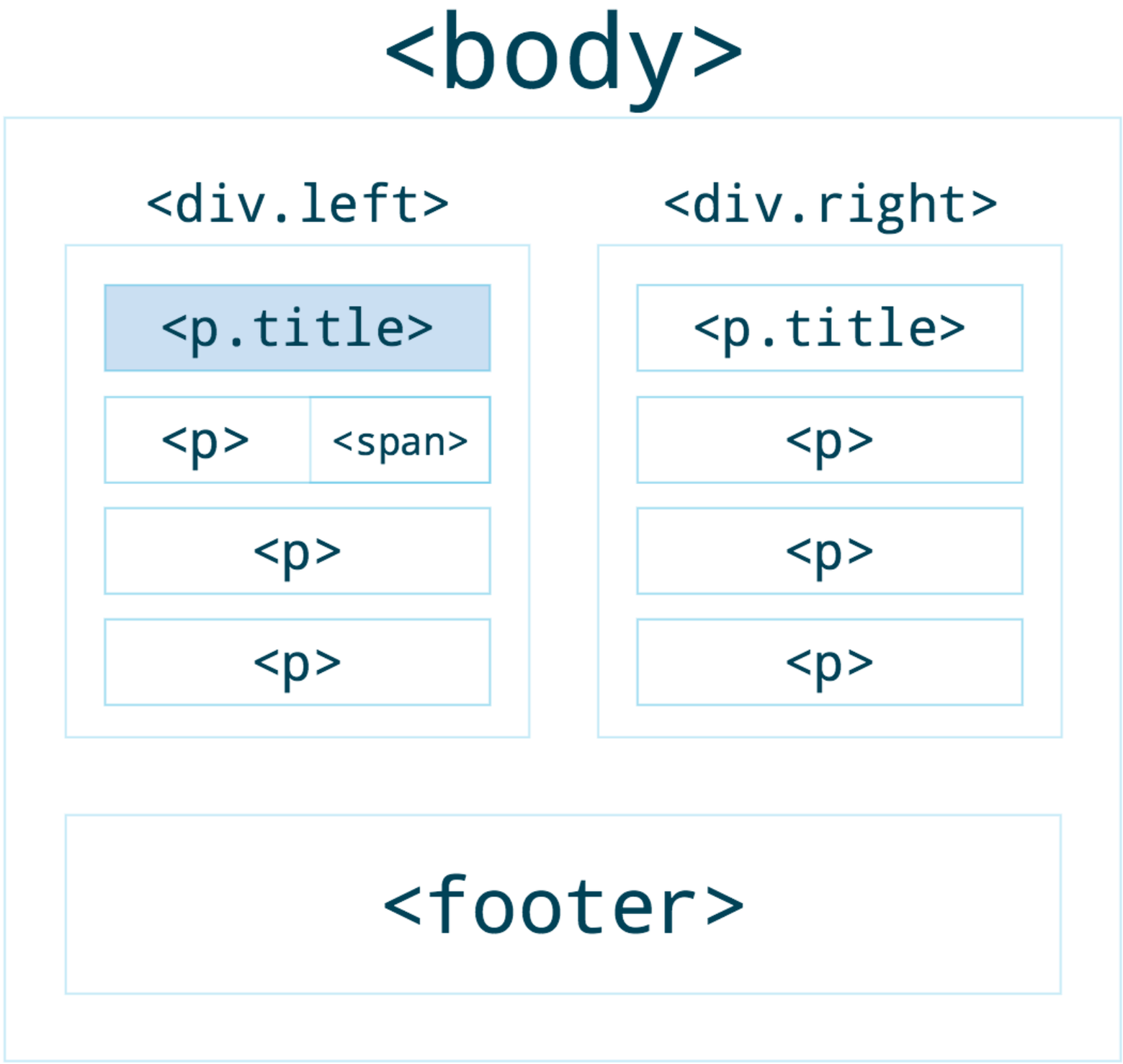
div:first-child



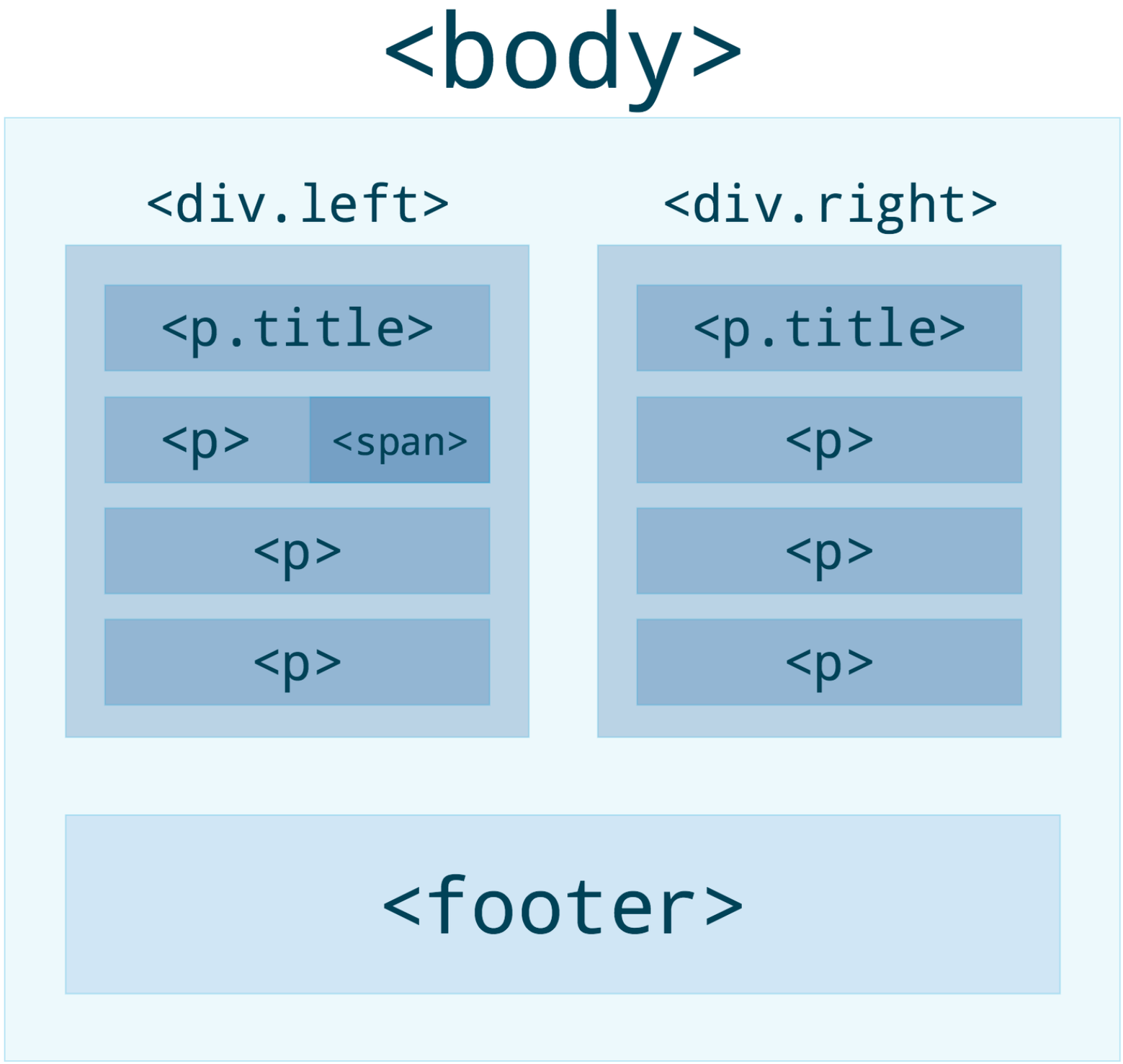
div:first-child



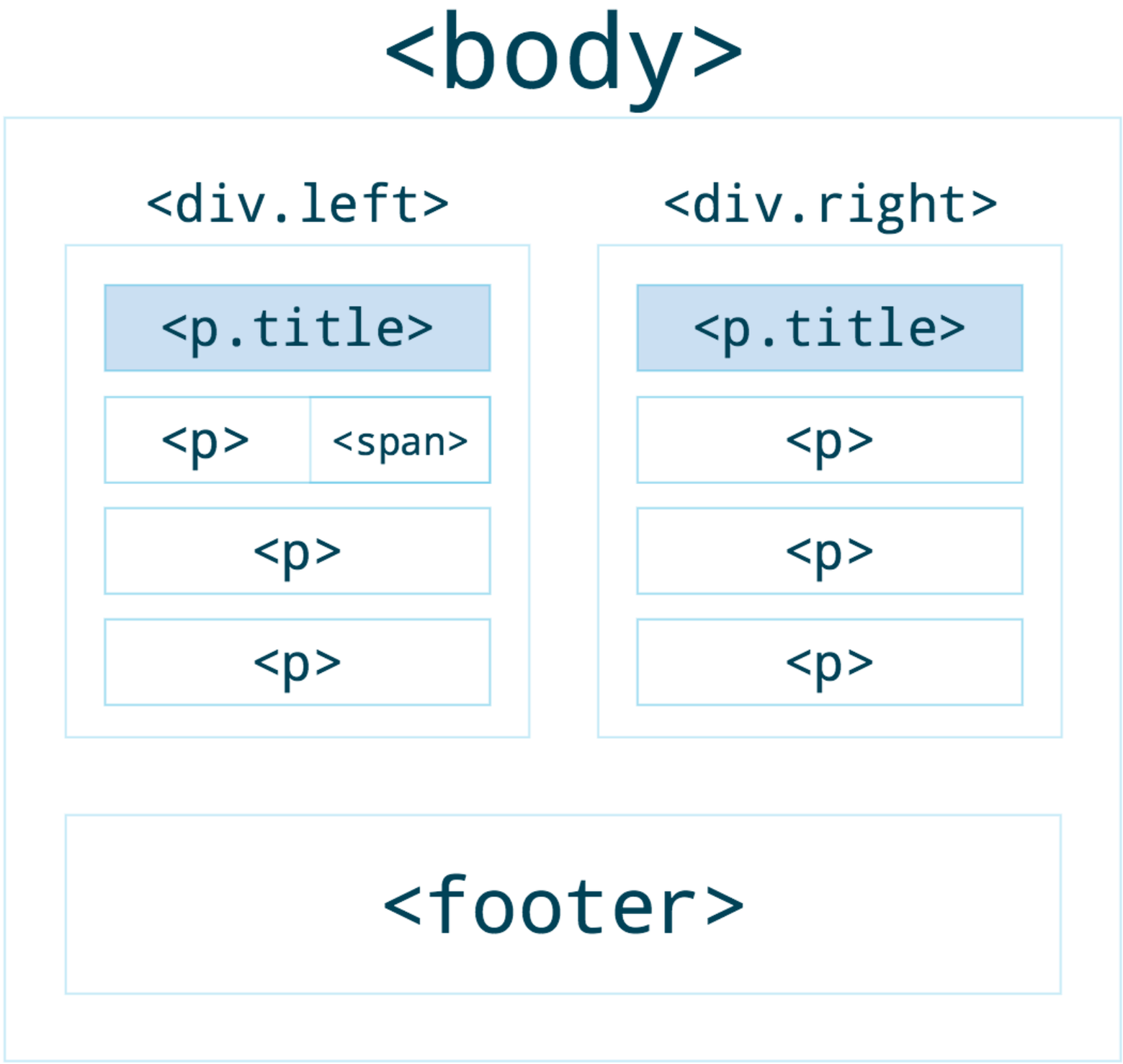
.left p:first-child



.left p:first-child



div p:first-child



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: }
```

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: }
```

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

1. Переопределение

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

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

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

→ Heading

2. Специфичность

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

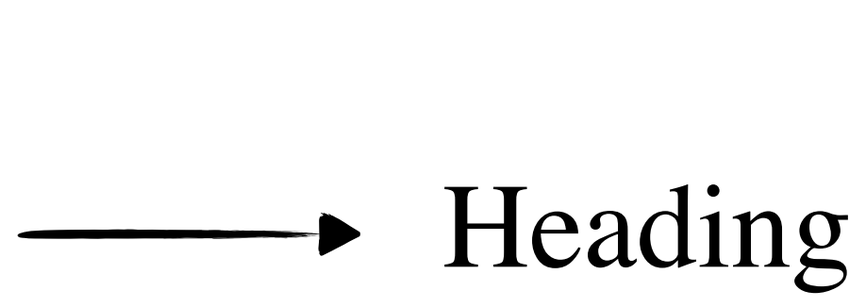
Heading

2. Специфичность

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

…

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



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>
```

this

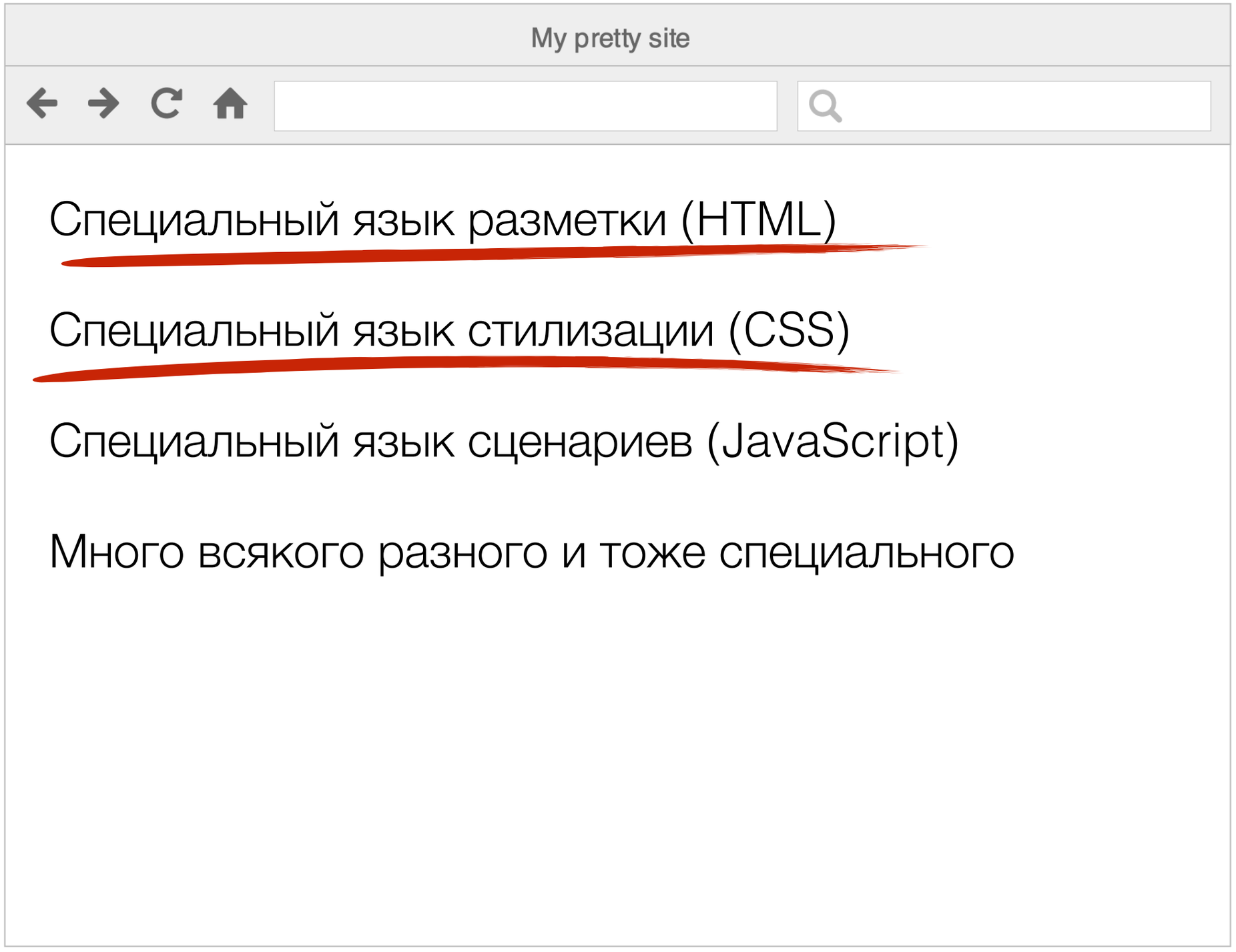
Как всем этим пользоваться

```
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>
```

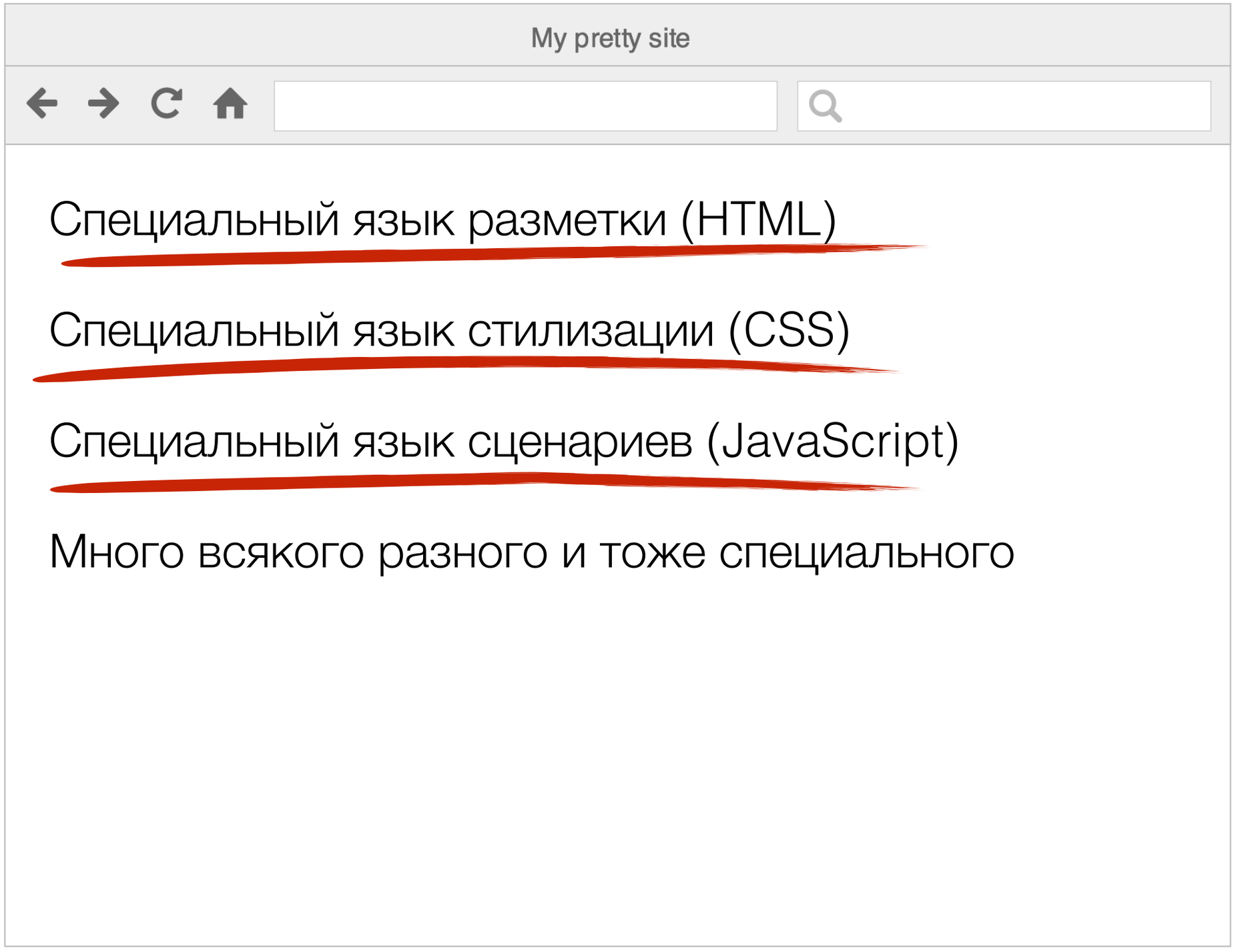
or this

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

Как работает браузер ?

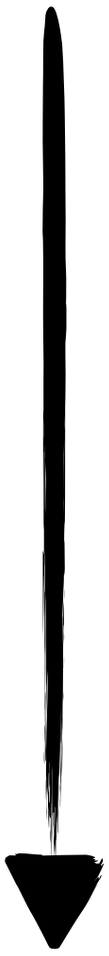


Как работает браузер ?




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;
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);
```

10

20

Интерпретация

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

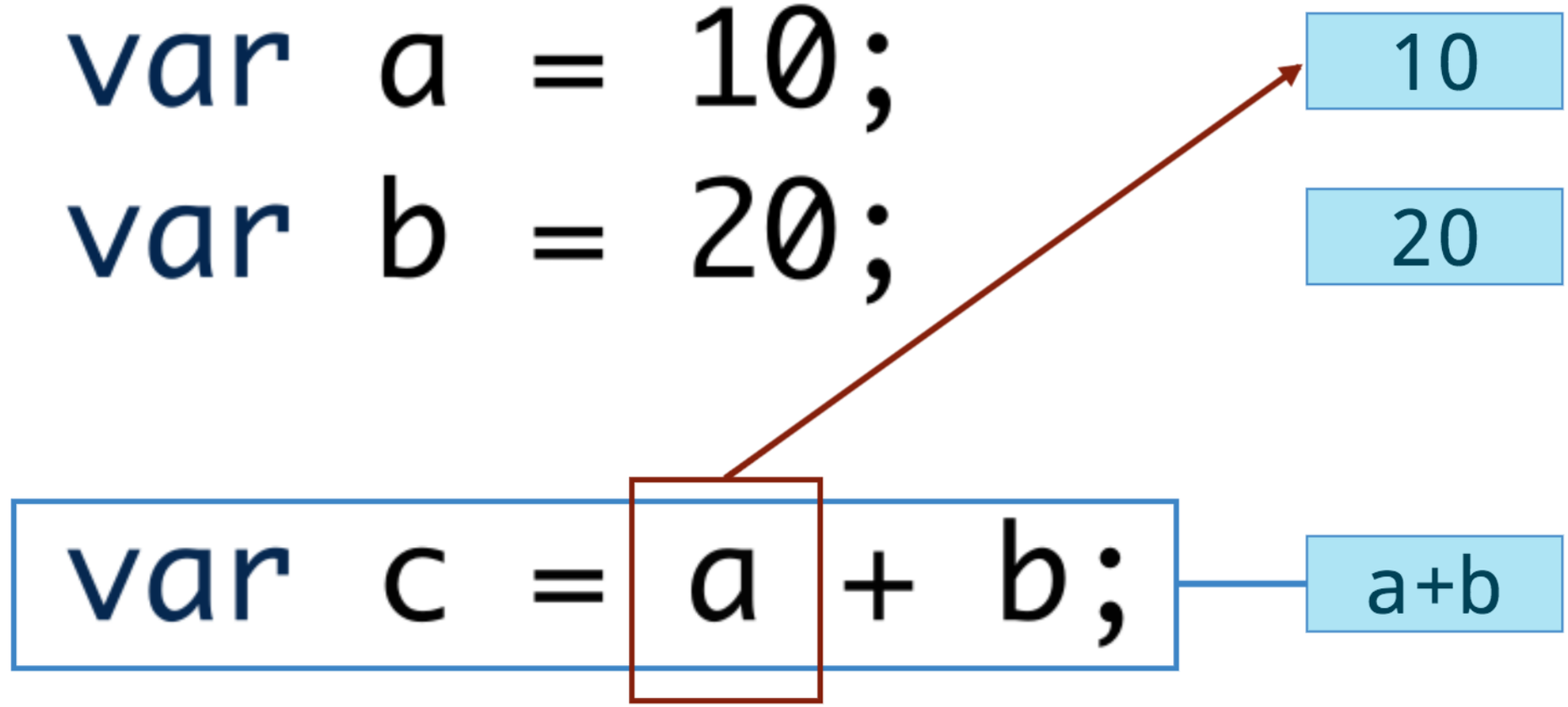
10

20

a+b

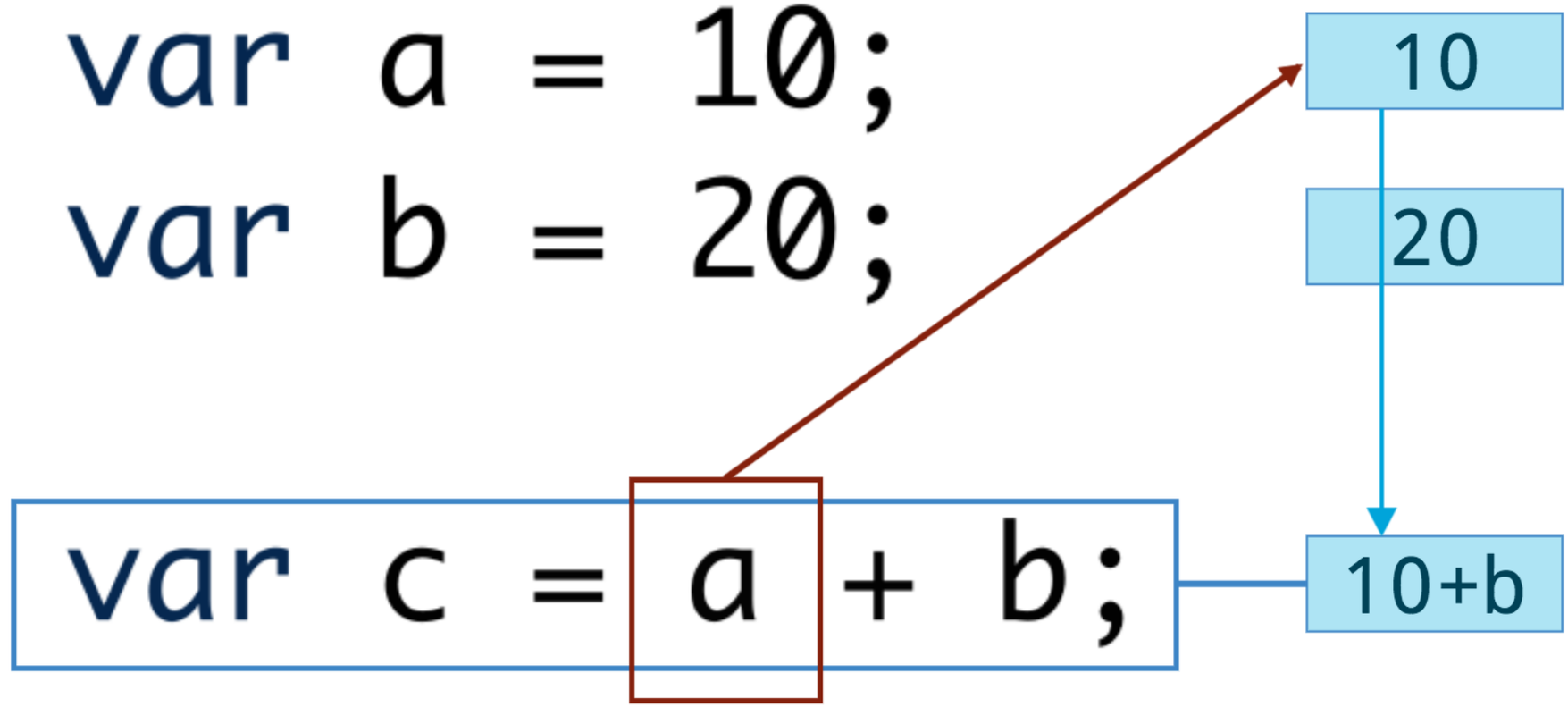
Интерпретация

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



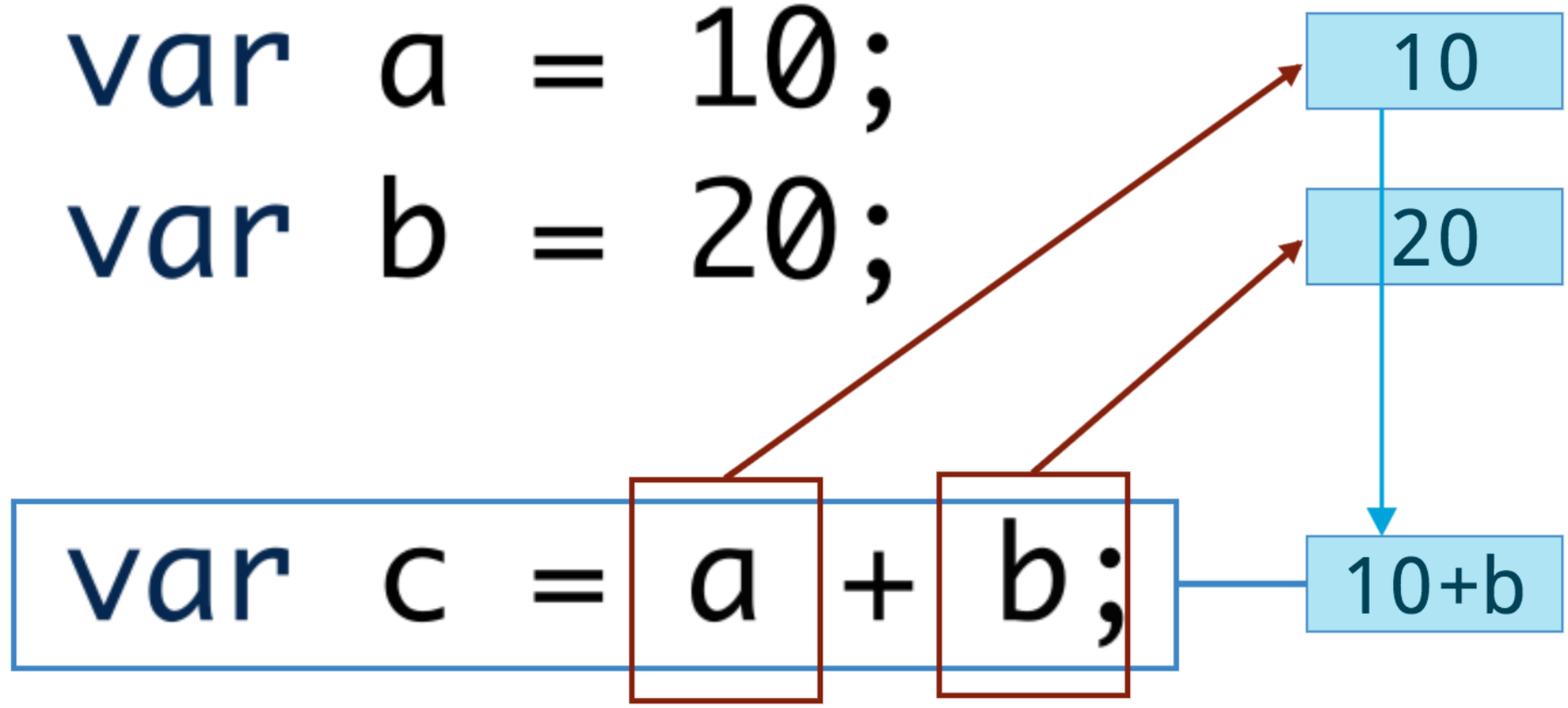
Интерпретация

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



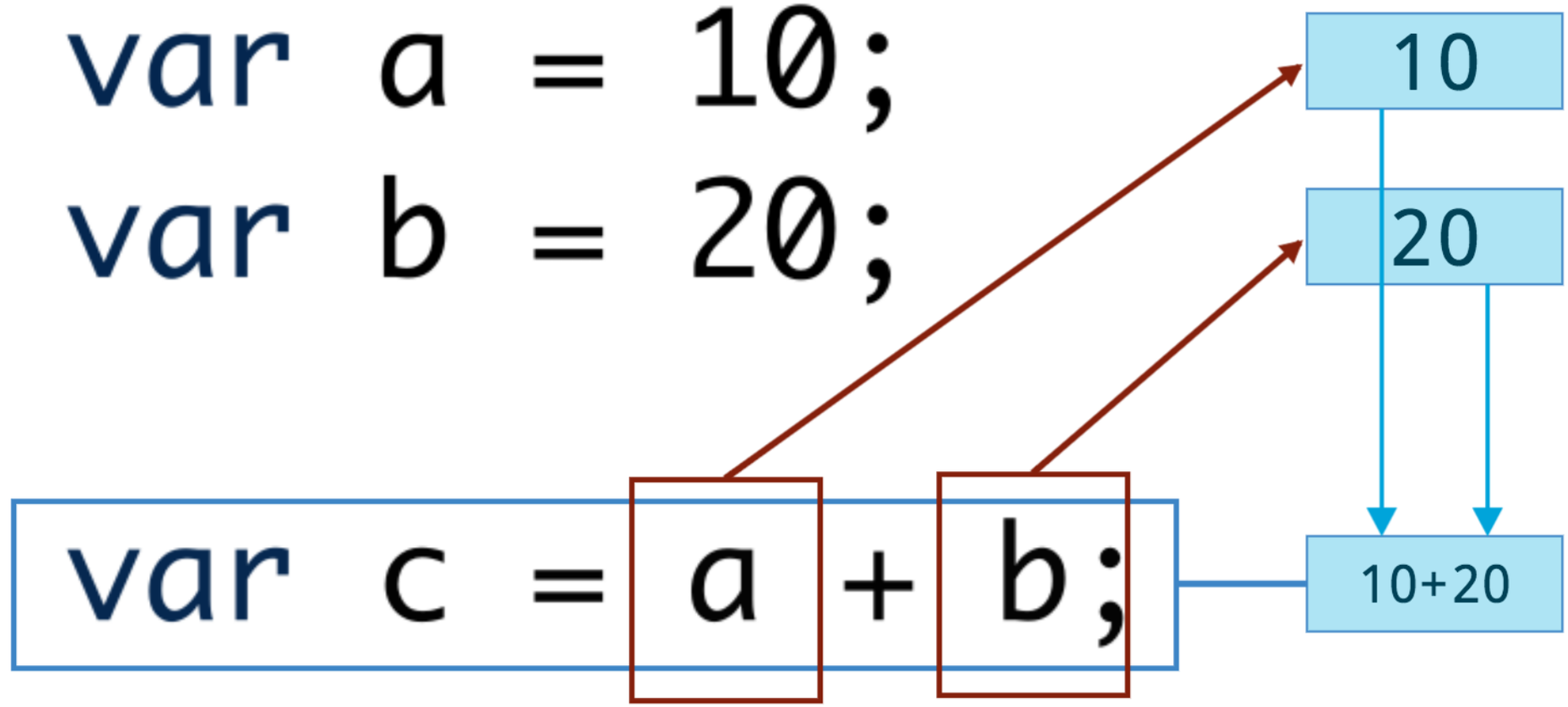
Интерпретация

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



Интерпретация

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



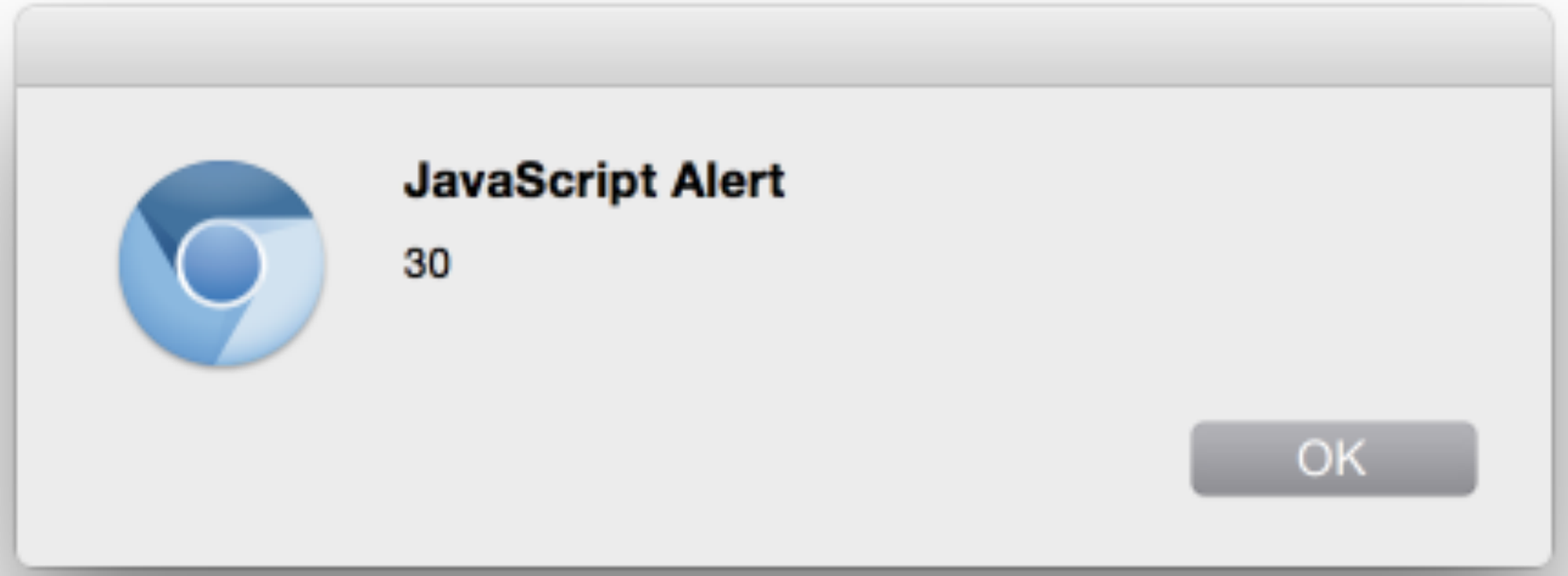
Интерпретация

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

10

20

30


JavaScript Alert
30
OK

Типы данных

Значения	Тип
`10, 010, -10, 12.5`	**number**
`“hello world”, “c”`	**string**
`true, false`	**boolean**
`function () { … }`	**function**
`{x: 10, y: 20}`	**object**
`[10, 20]`	**object**
`undefined`	**undefined**

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 +

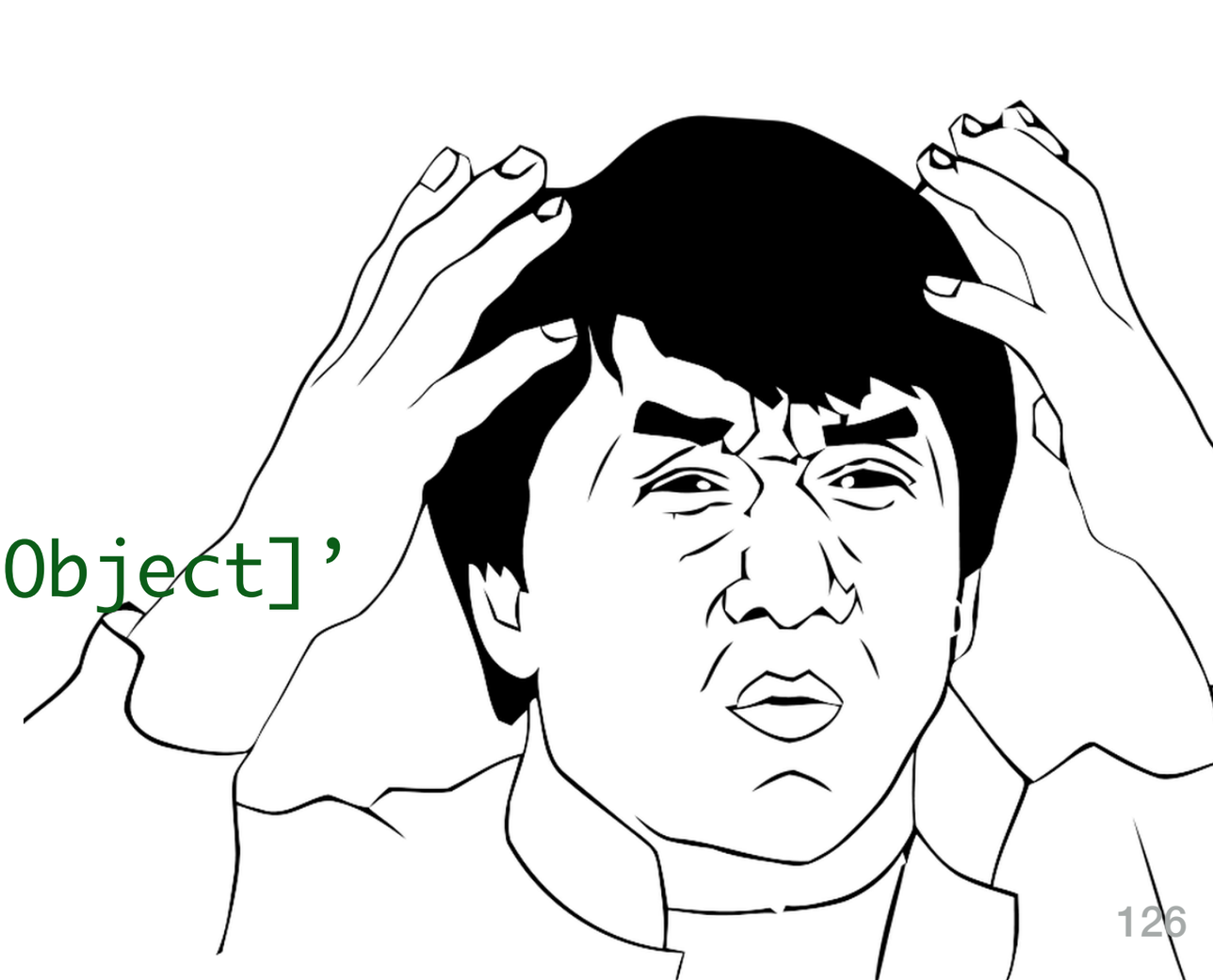
```
01: 5 + 6     -> 11 (число + число)
02:
03: 5 + ‘6’   -> ’56’ (число + строка)
04:
05: +’10’     -> 10
06:
07: {} + {}   -> NaN
08:
09: {} + ‘’   -> 0
10:
11: ‘’ + {}   -> ‘[object Object]’
```

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          - a строго больше b

a >= b         - a больше или ровно b

a < b          - a строго меньше b

a <= b         - a меньше или ровно b

a == b         - a равен b

a === b        - a строго равен b
```

Логические операторы

```
a && b          - a и b

a || b          - a или b

!a              - не a
```

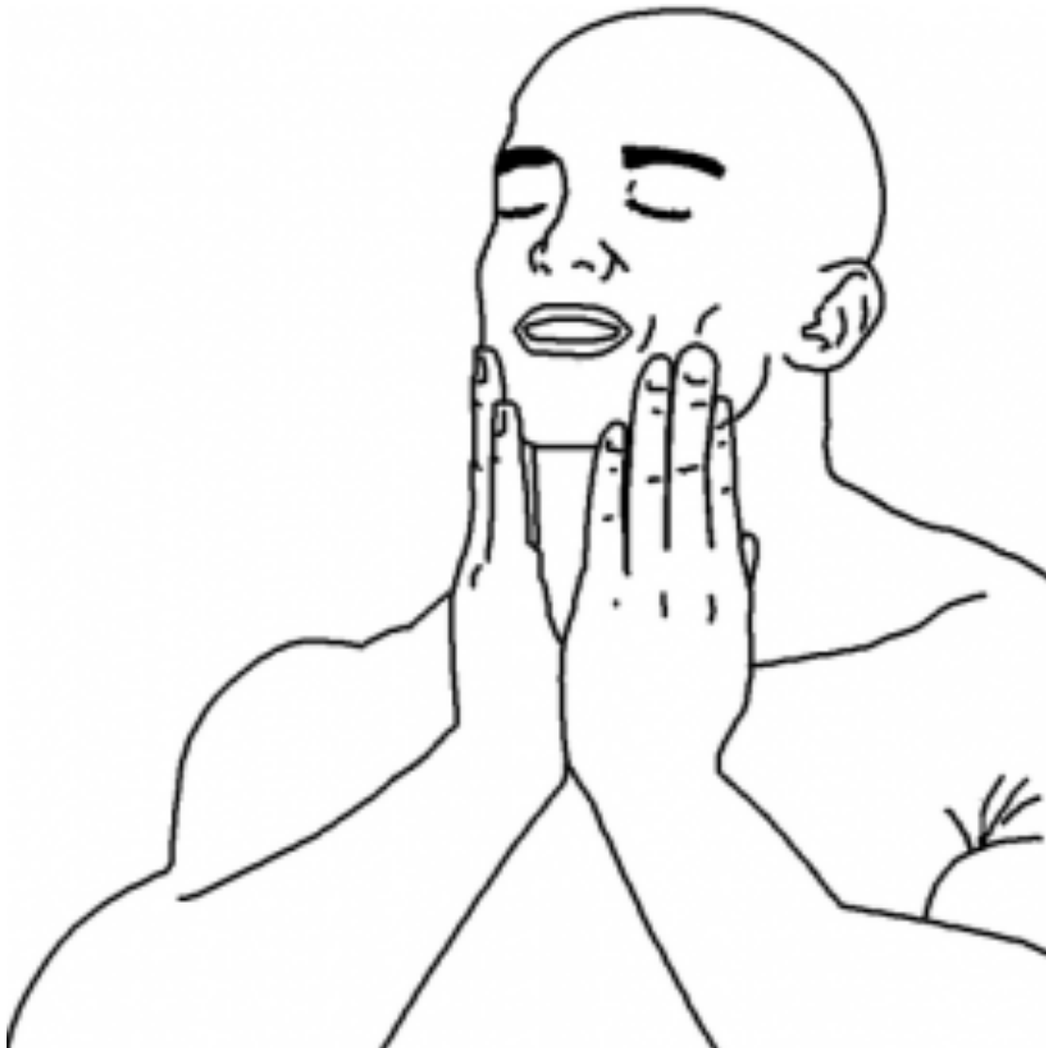
Логические операторы

```
a && b          - a и b

a || b          - a или b

!a              - не a

!a && !b        - не a и не b

a || !b         - a или не b
```

Тернарный оператор

```
a ? b : c
```

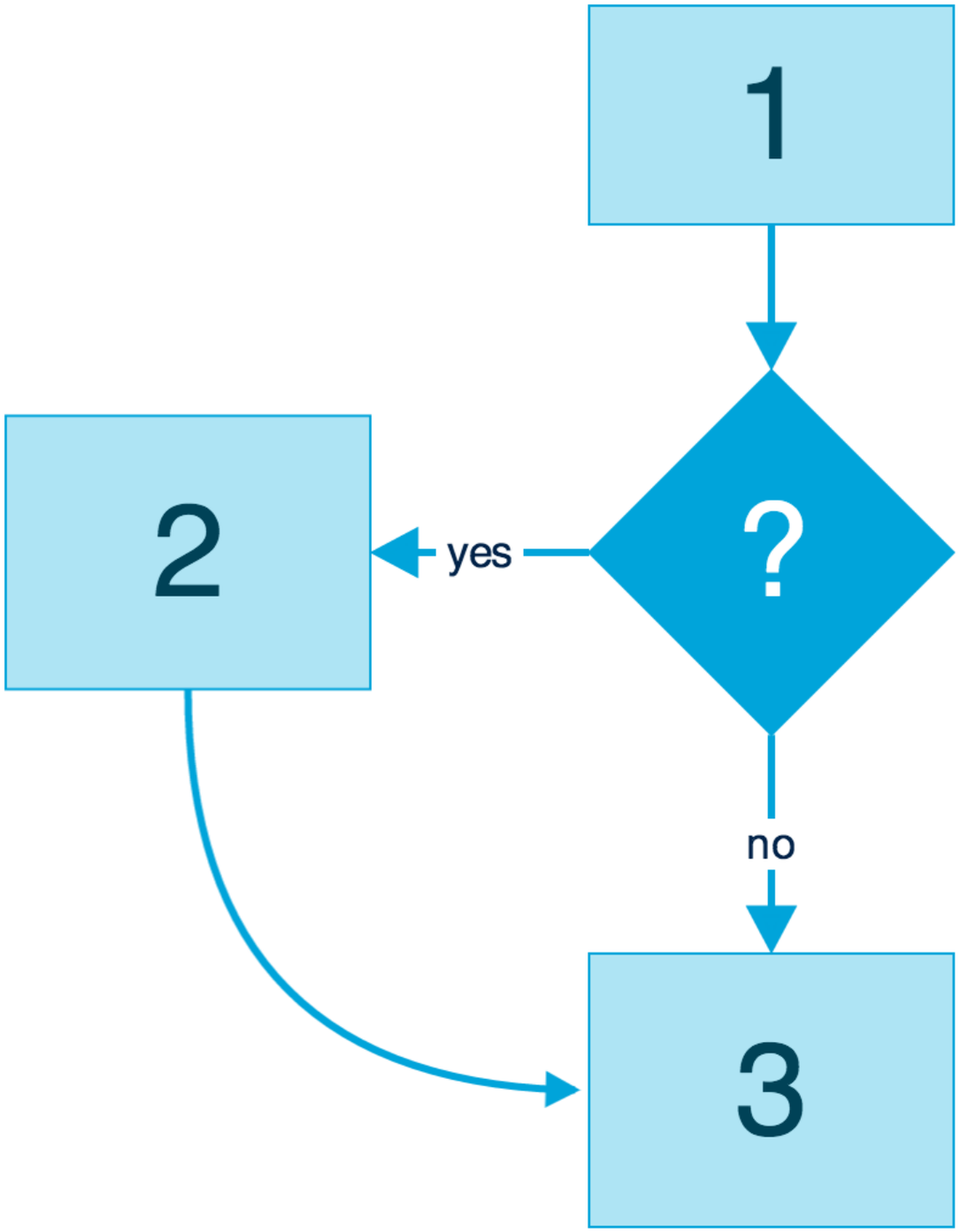
```
if (a) {
  b
} else {
  c
}
```



Управляющие конструкции

- Ветвления
- Циклы

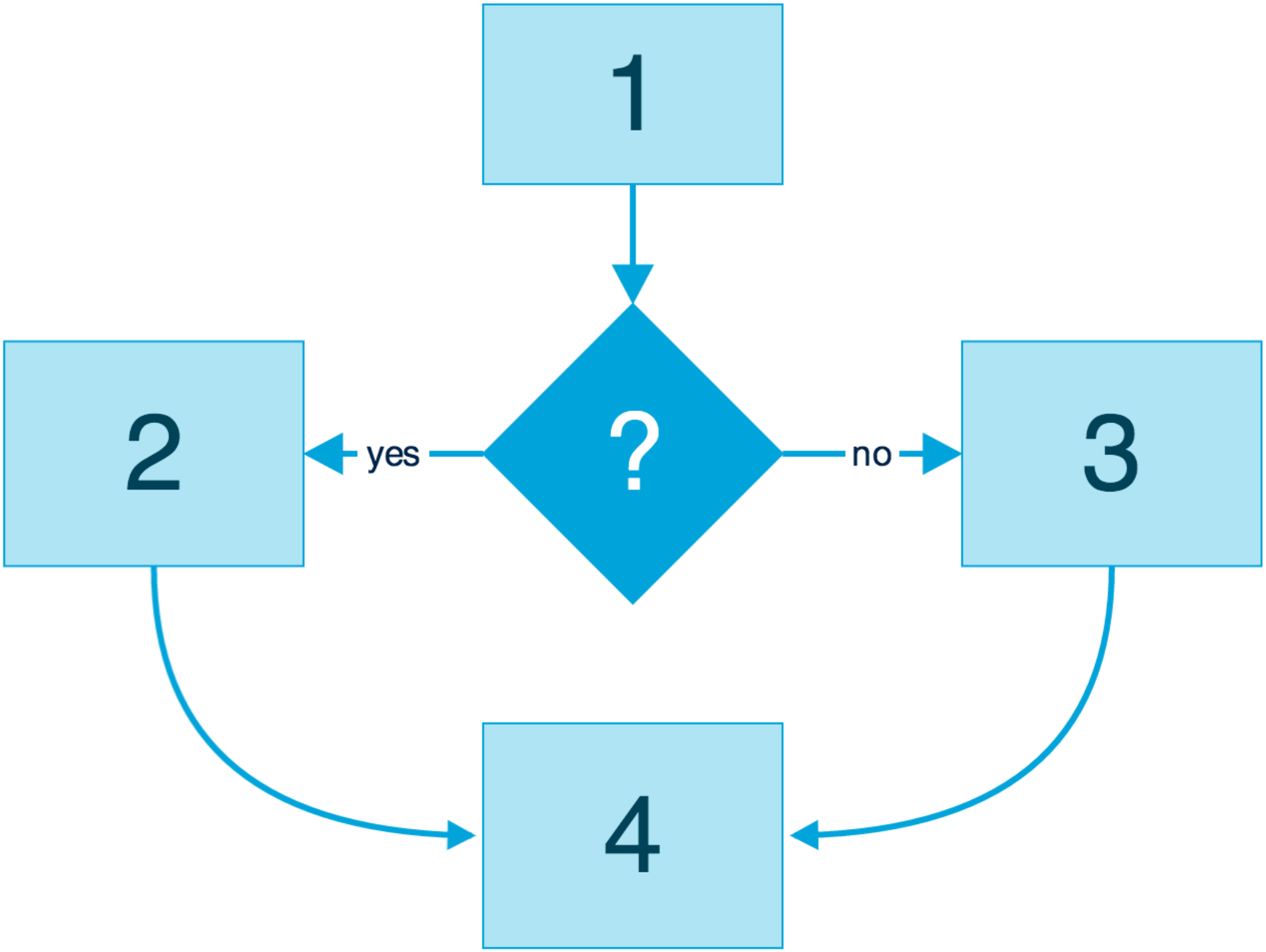
Ветвление



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
```

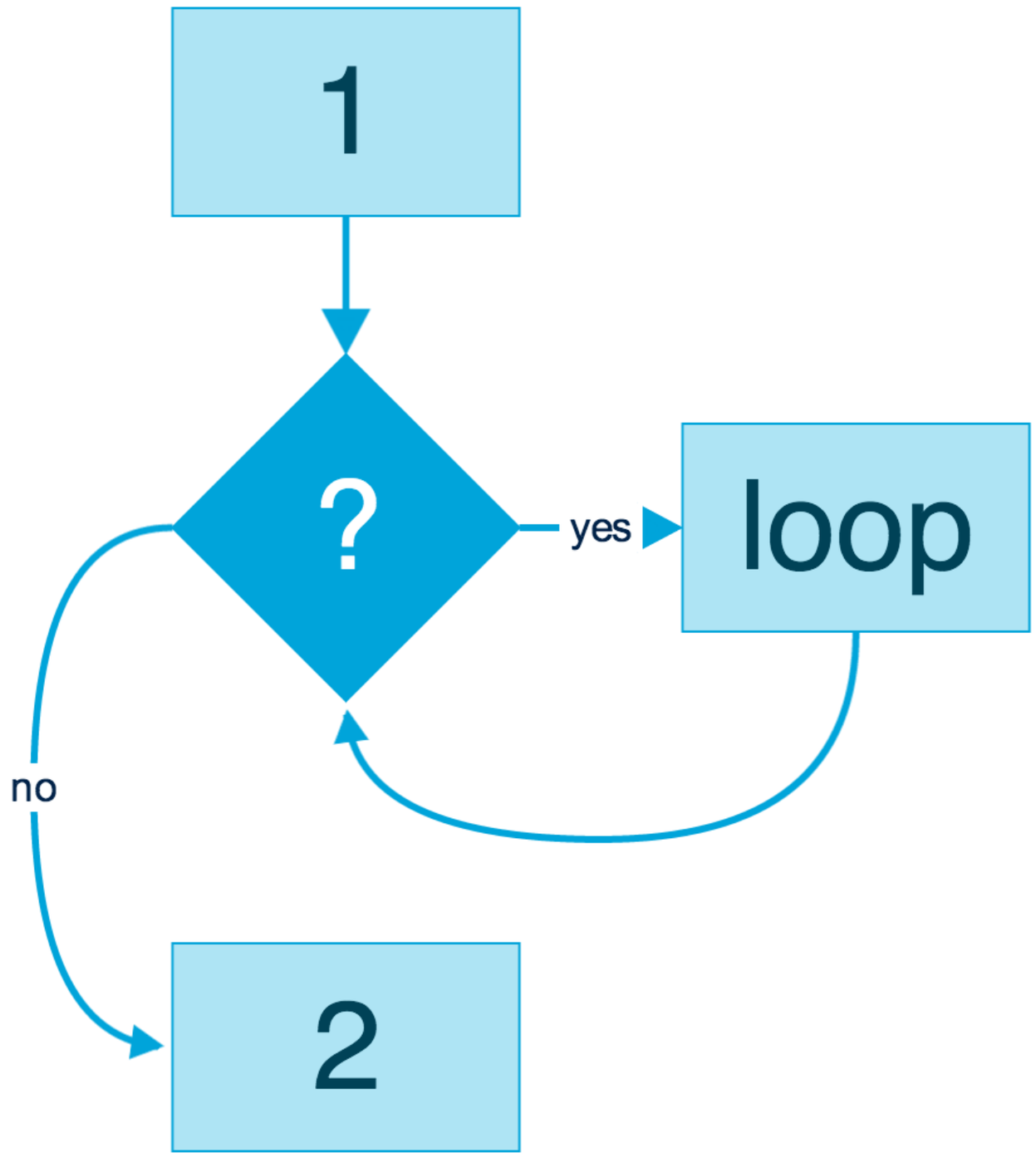
Ветвление

```
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

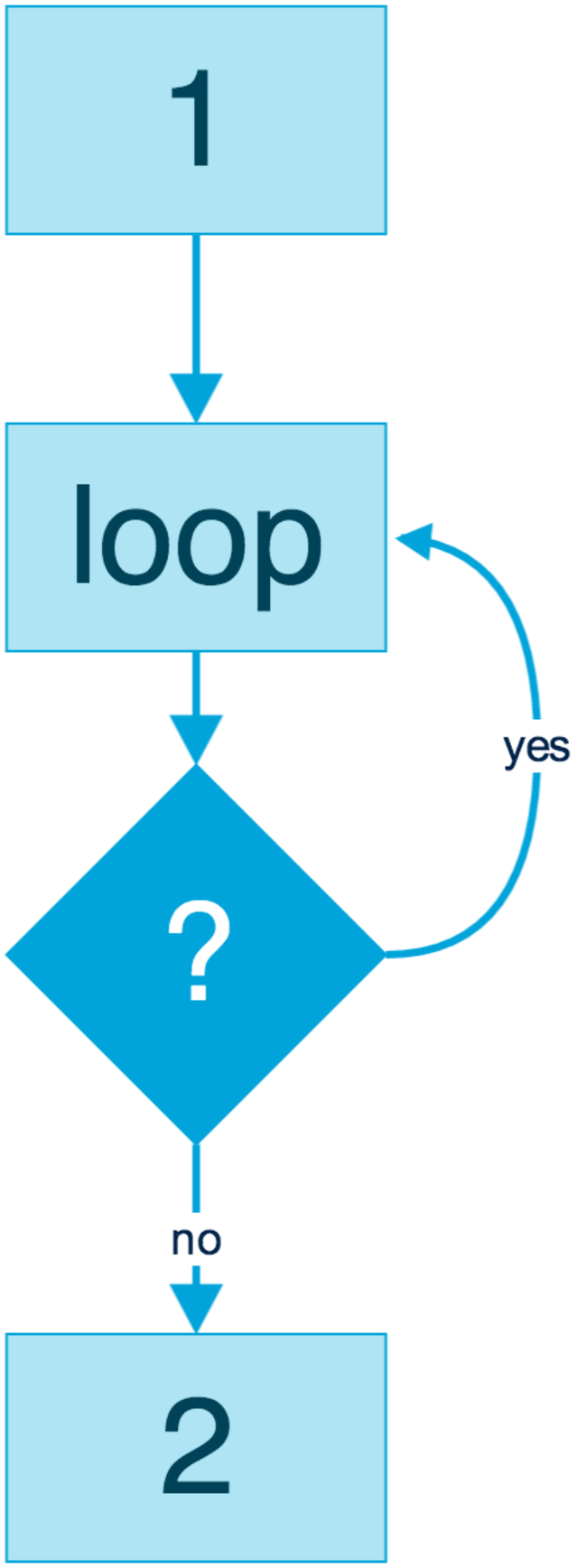
Цикл: while



Цикл: while

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

Цикл: do-while



Цикл: do-while

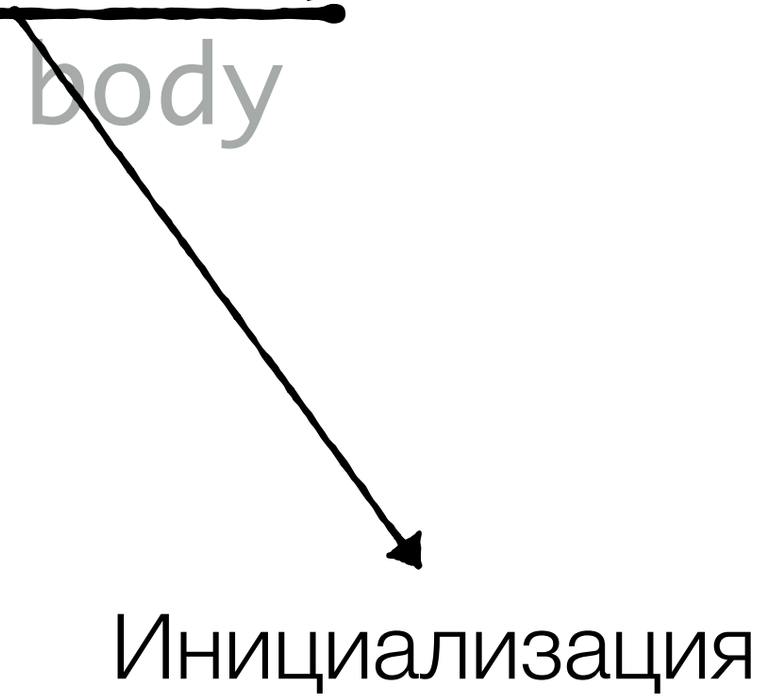
```
01: // 1
02: do {
03:     // loop body
04: } while (condition);
05: // 2
```

Цикл: for

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

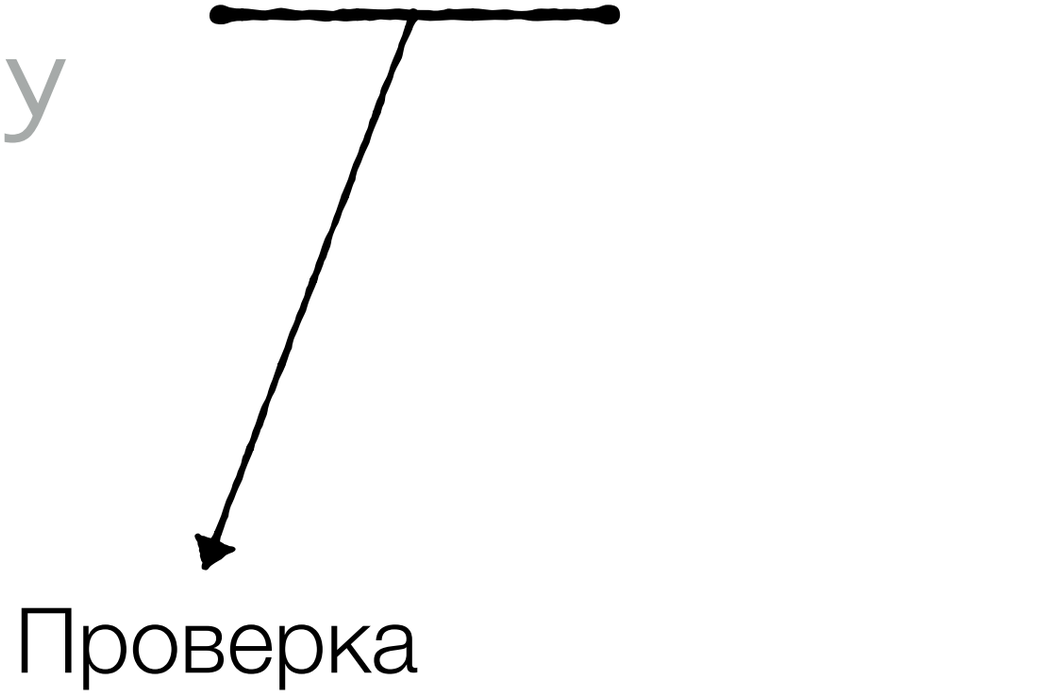
Цикл: for

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



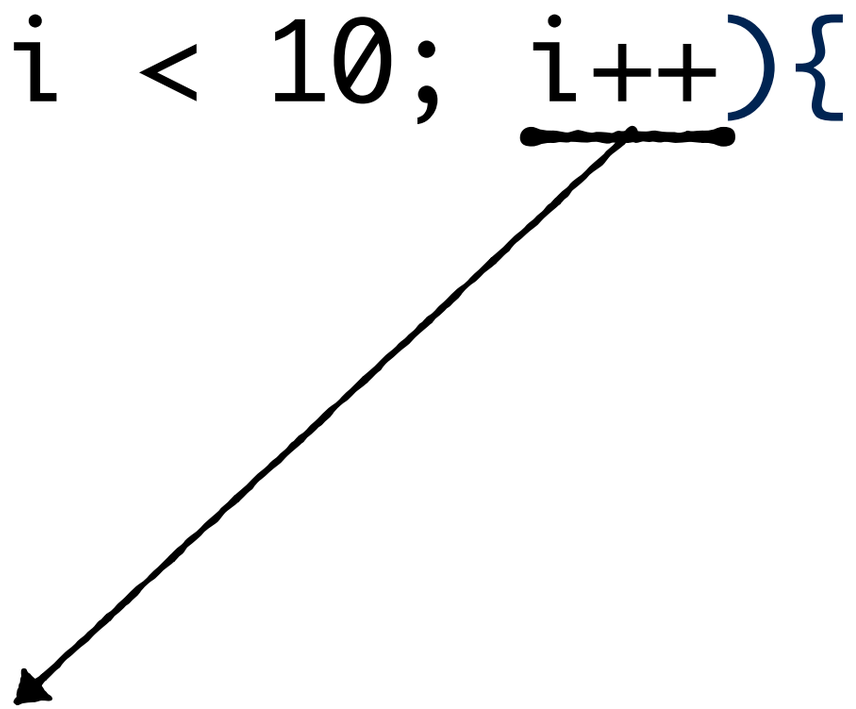
Цикл: for

```
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
```



Цикл: for

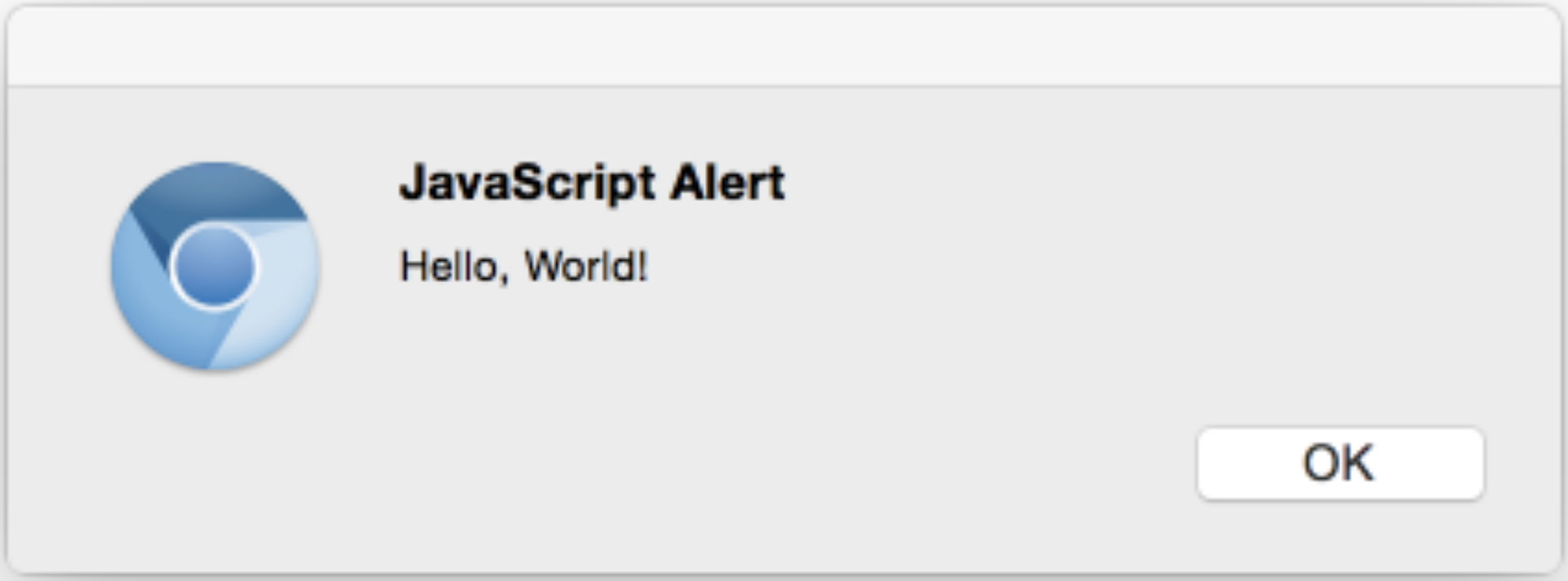
```
01: // 1
02: for (var i = 0; i < 10; i++){
03:     console.log(i);
04: }
05: // 2
```

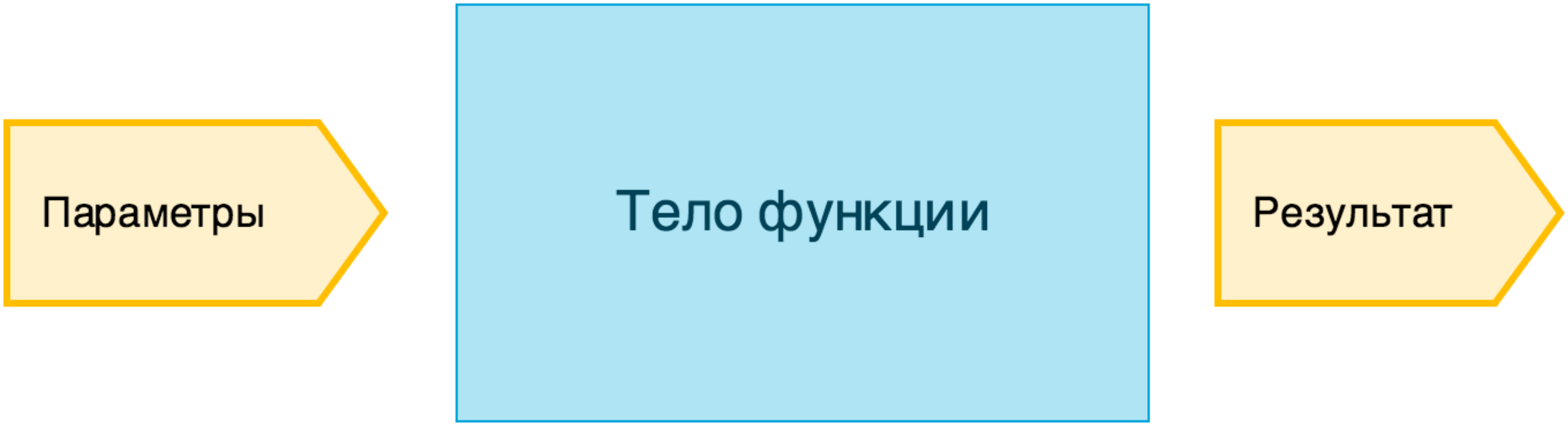
Функции

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

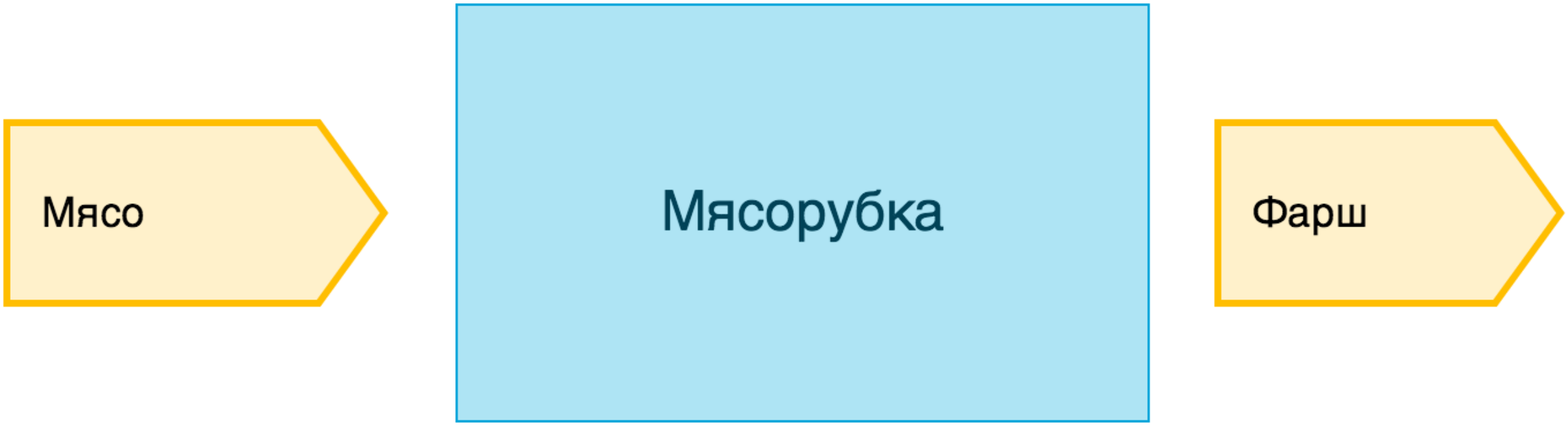
Функции

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


JavaScript Alert
Hello, World!
OK

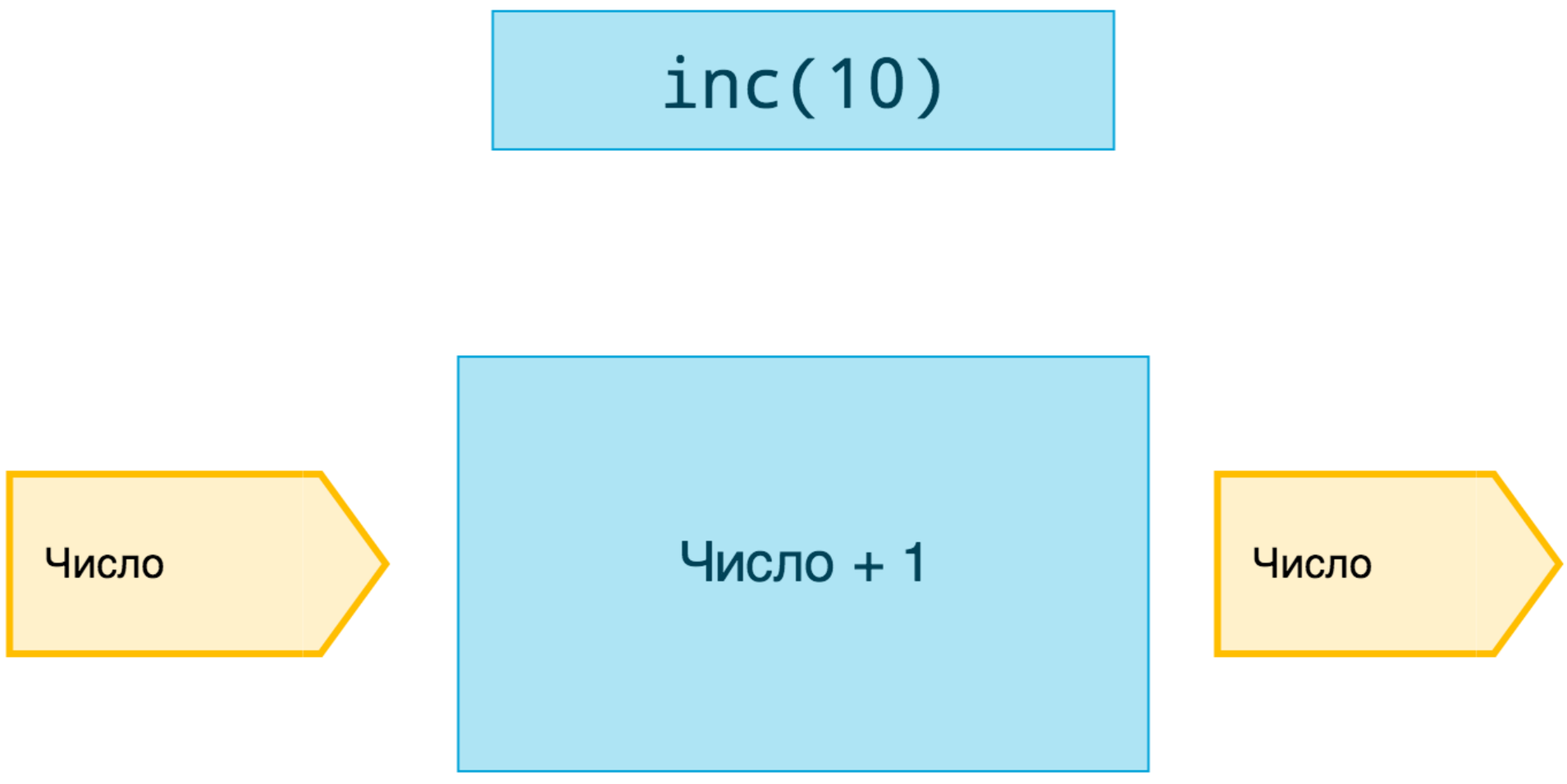

Параметры
Тело функции
Результат

Идеальный пример функции

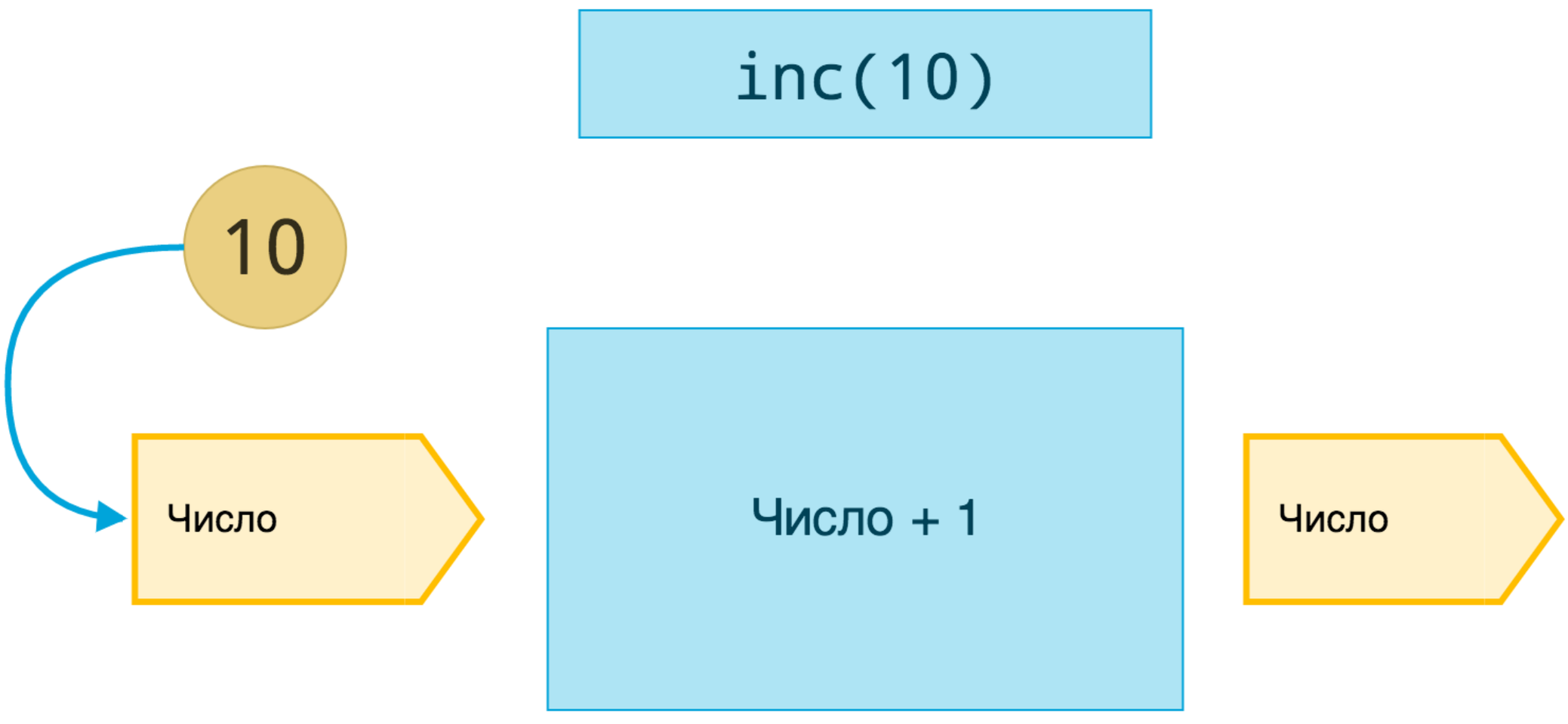


```
01: function inc(x) {
02:     return x + 1;
03: }
04:
05: var a = inc(10);
06:
07: alert(a);
```

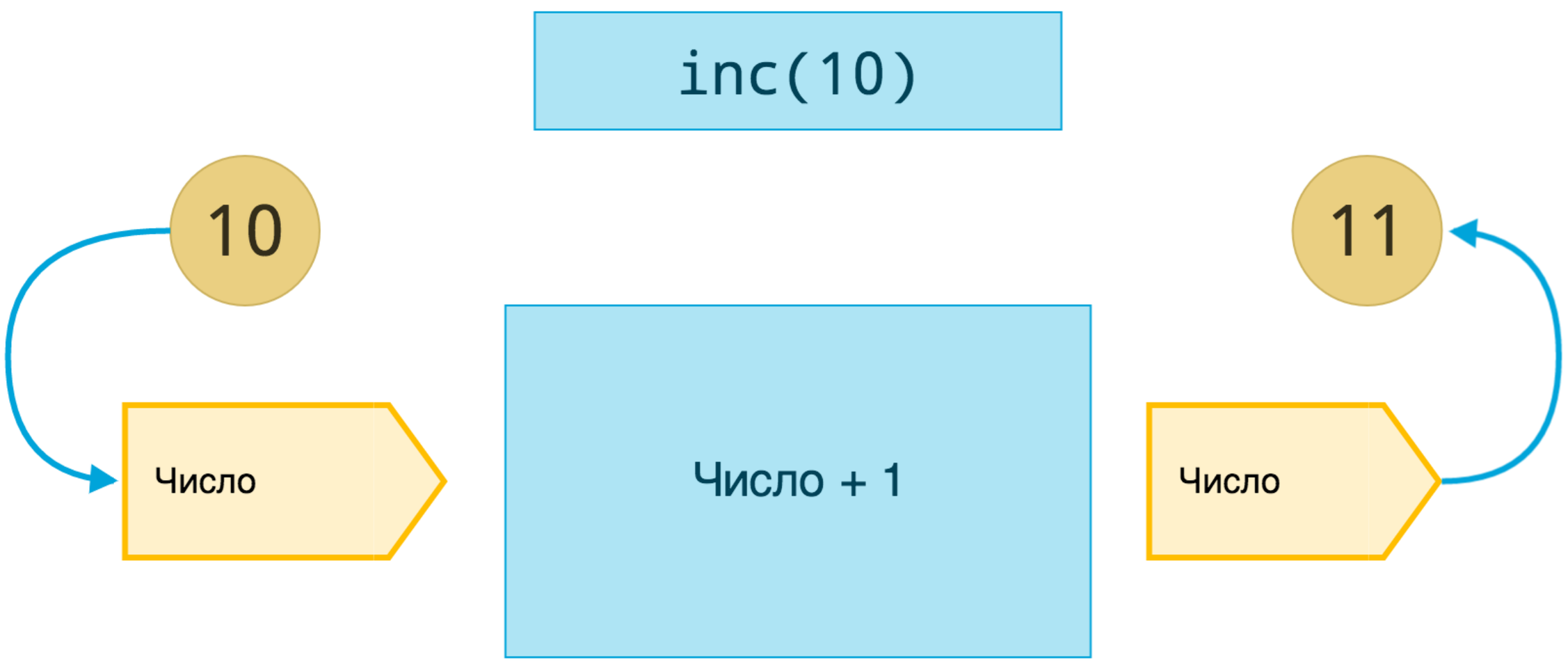
```
01: function inc(x) {
02:     return x + 1;
03: }
04:
05: var a = inc(10);
06:
07: alert(a);
```



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

Объявление локальной переменой

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: }
```

Объявление локальной переменой

Объявление глобальной переменой

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: }
```

глобальная область

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: }
```

глобальная область

локальная область

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: })();
```

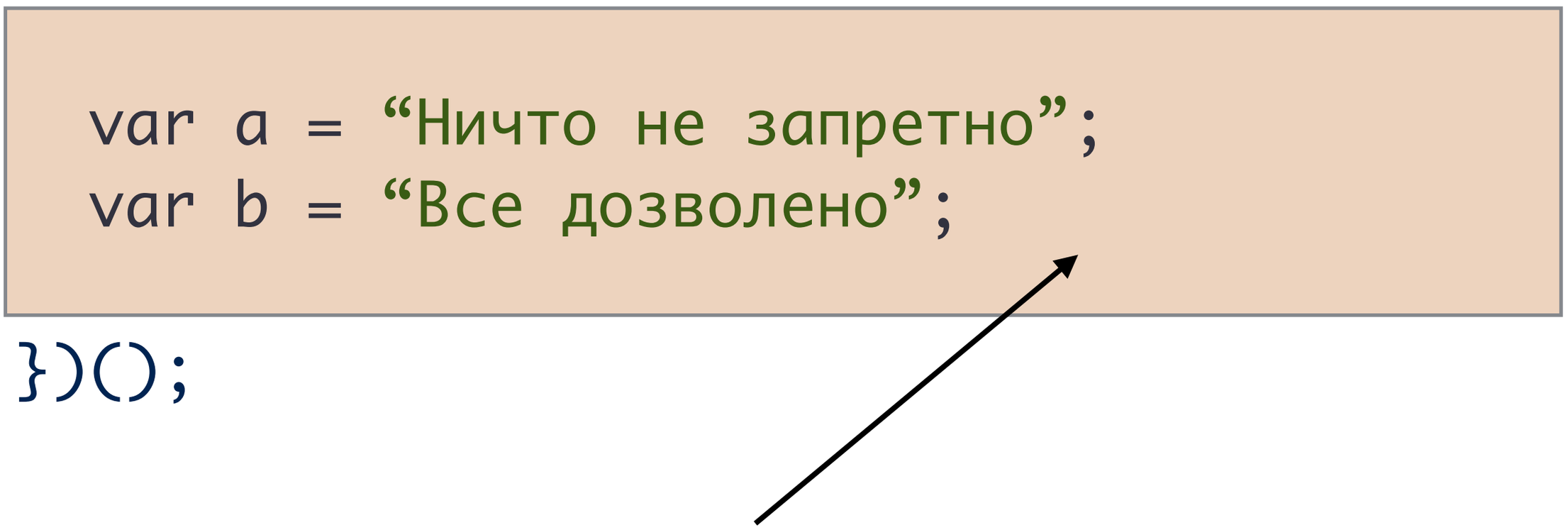
Self-invoking functions

Хитрый кунштюк

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

Хитрый кунштюк

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



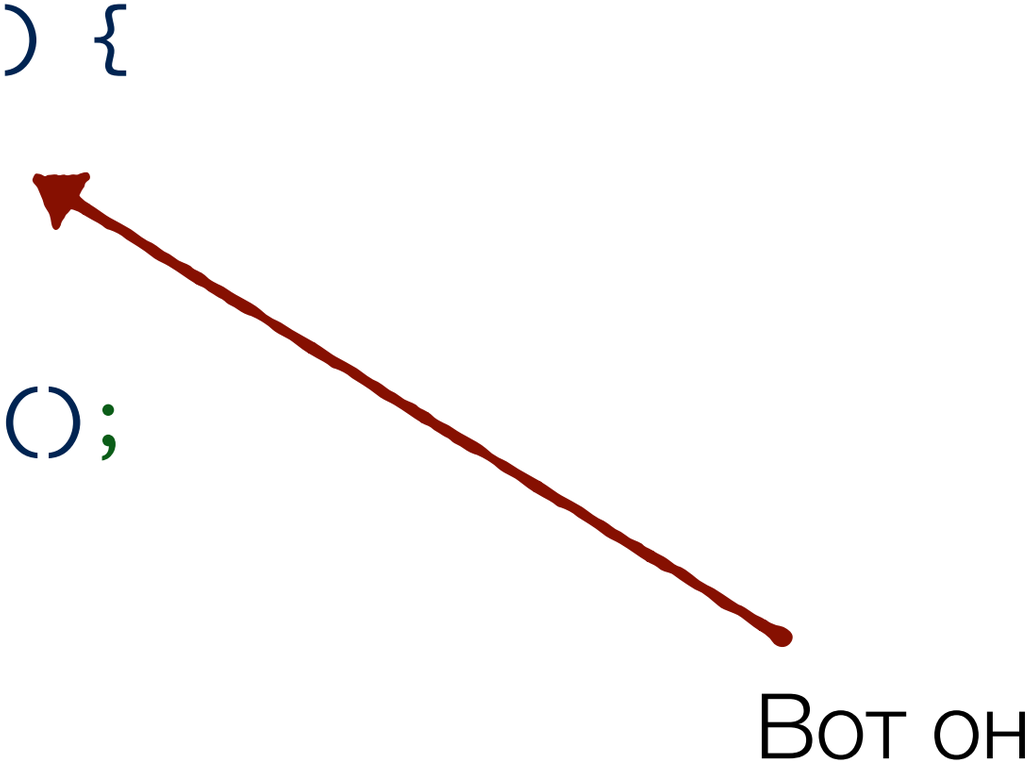
Новый scope. Главное не забывать var.

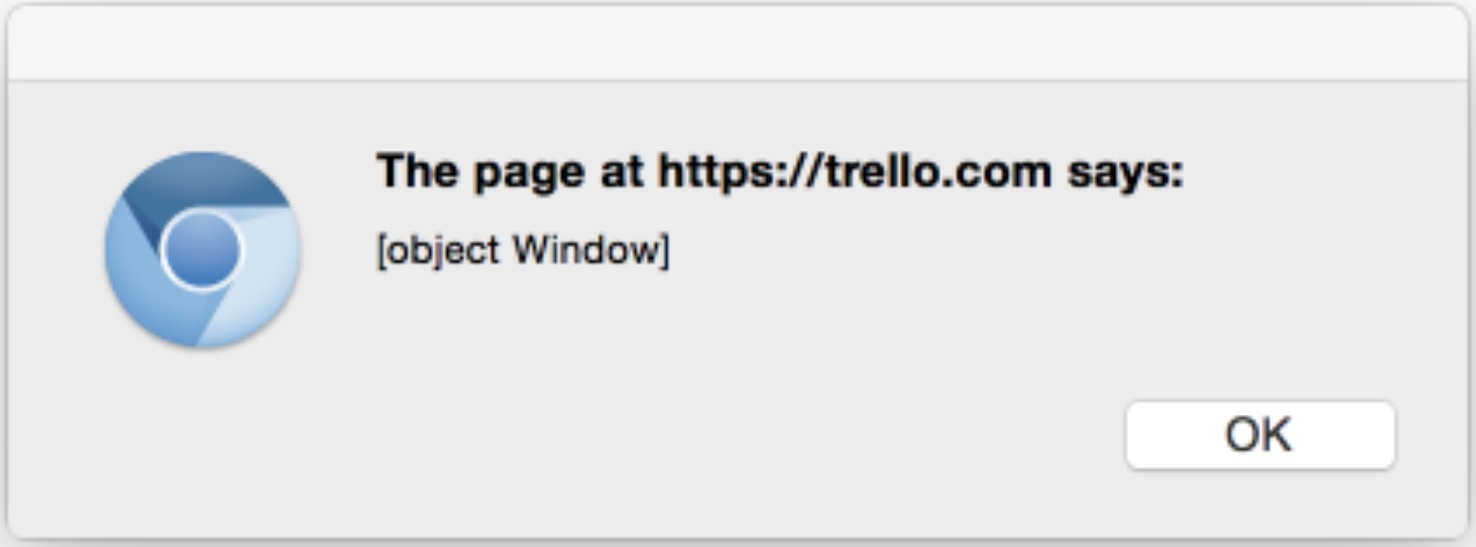
Контекст

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

Контекст

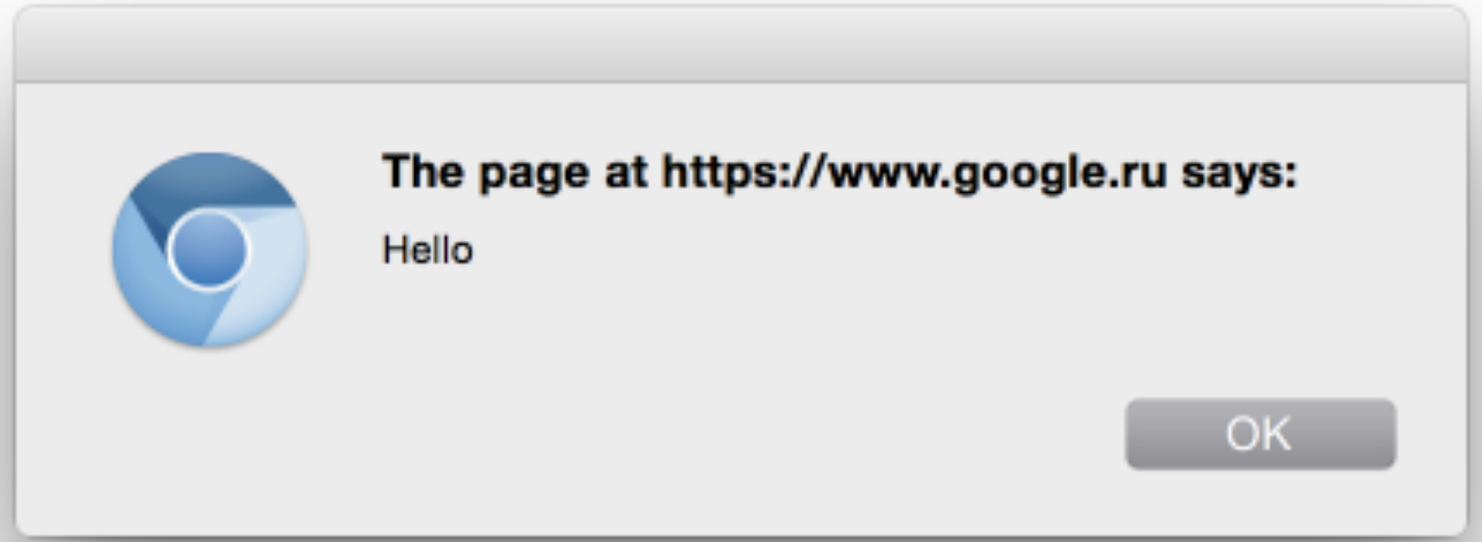
```
01: function context() {
02:     return this;
03: }
04:
05: var ctx = context();
06:
07: alert(ctx);
```




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);
```

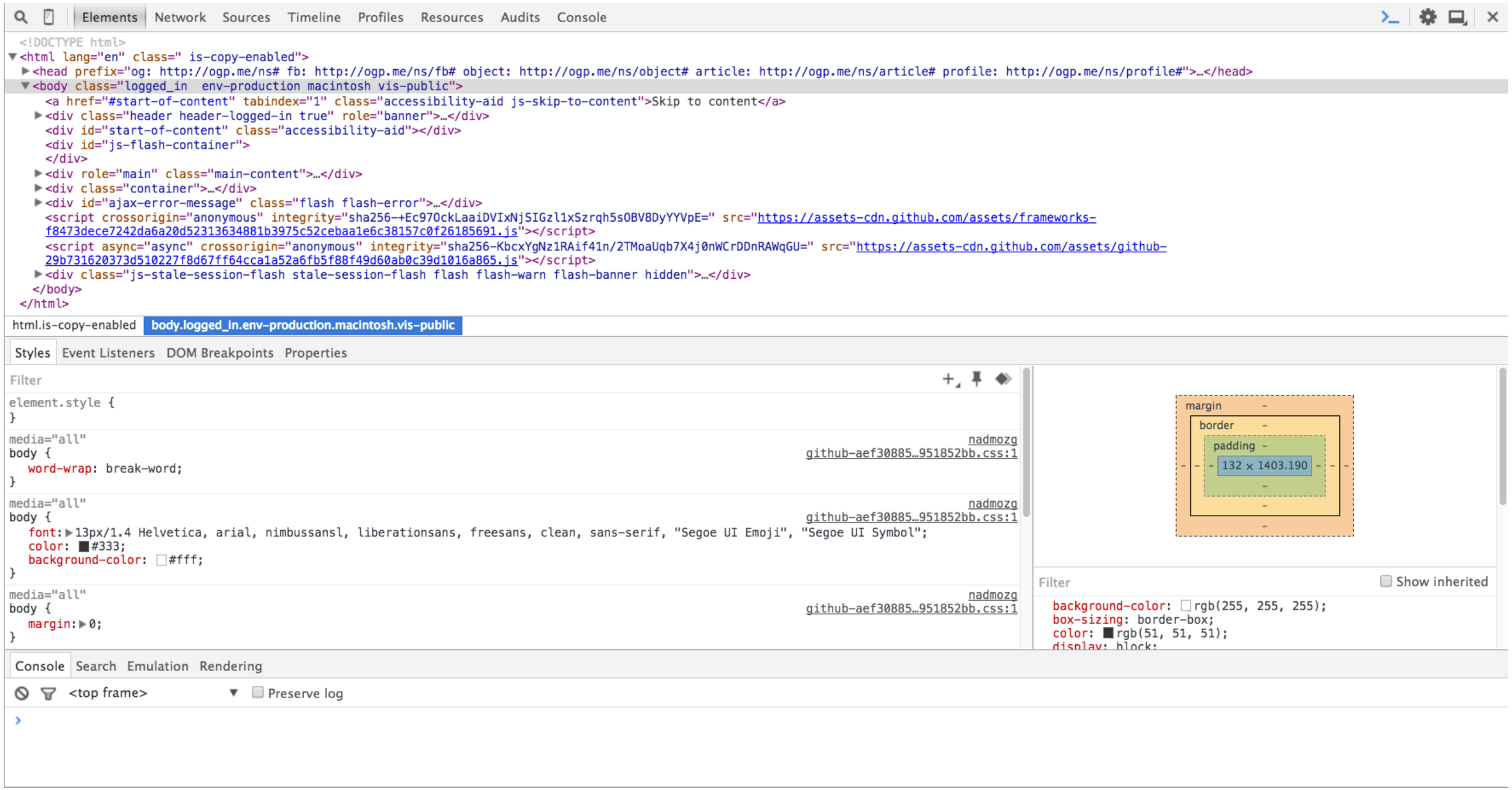
Окружение браузера

- таймеры
- window

Окружение браузера

- таймеры
- window
- document

Dev Tools


Elements Network Sources Timeline Profiles Resources Audits Console
<!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">
Skip to content
…

…
…
…
<script crossorigin="anonymous" integrity="sha256-+Ec97OckLaaiDVIxNjSIGzl1xSzrqh5sOBV8DyYYVpE=" 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>
…
</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;
}
nadmozg
github-aef30885…951852bb.css:1
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;
}
nadmozg
github-aef30885…951852bb.css:1
media="all"
body {
margin: 0;
}
nadmozg
github-aef30885…951852bb.css:1
margin
border
padding
132 × 1403.190
Filter
Show inherited
background-color: rgb(255, 255, 255);
box-sizing: border-box;
color: rgb(51, 51, 51);
Console Search Emulation Rendering
<top frame>
Preserve log

это все

пора за дело