



Игорь Алексеенко
училка в HTML Academy



Многие современные проблемы
фронтендеров были решены
ещё в 80-х 😎📼💾☎️⚙️🚀

Просто мы об этом не знаем 👶👶👶



**То, что мы разрабатываем,
скорее, уже не сайты, а RIA**

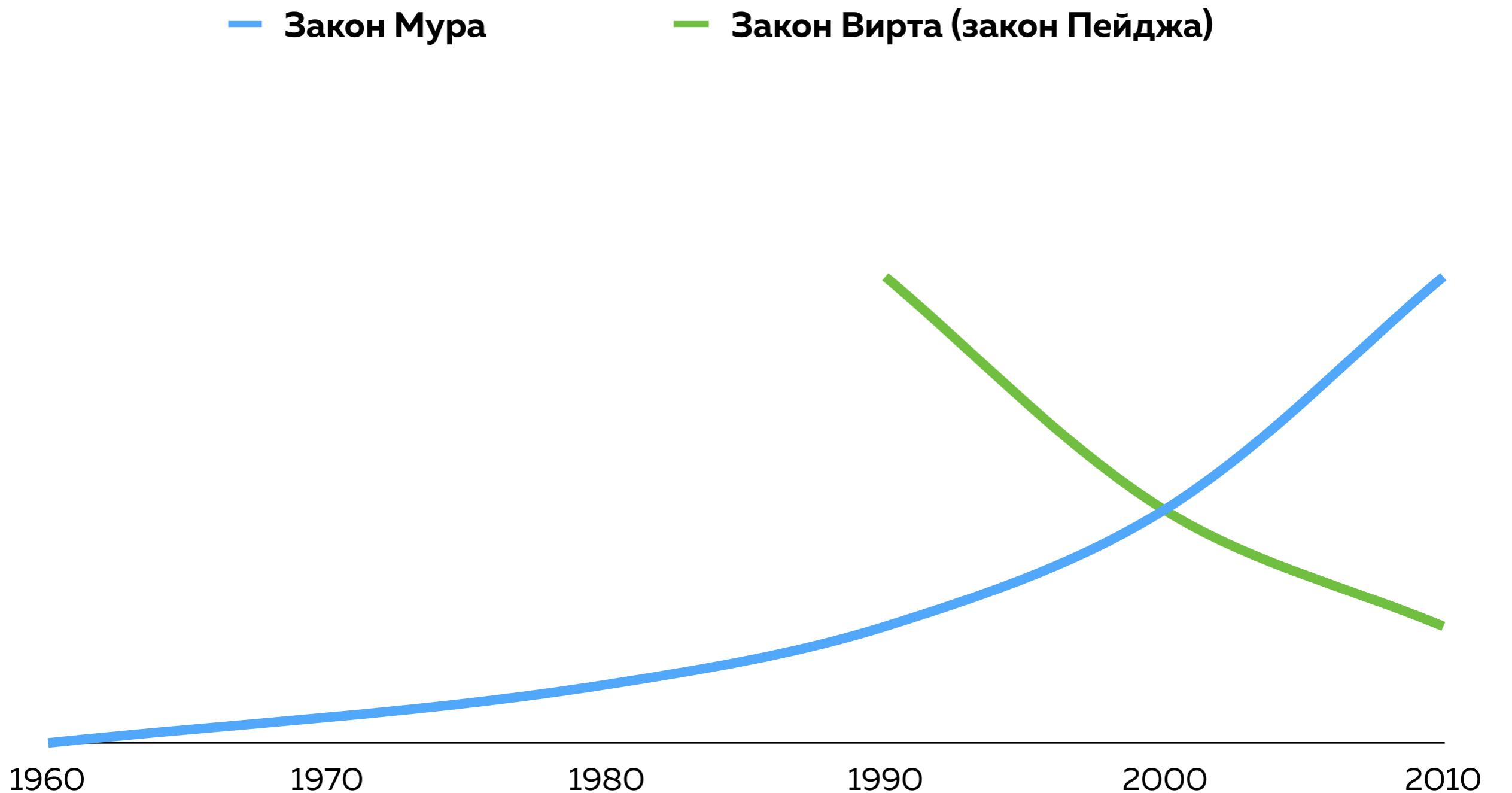


RIA –

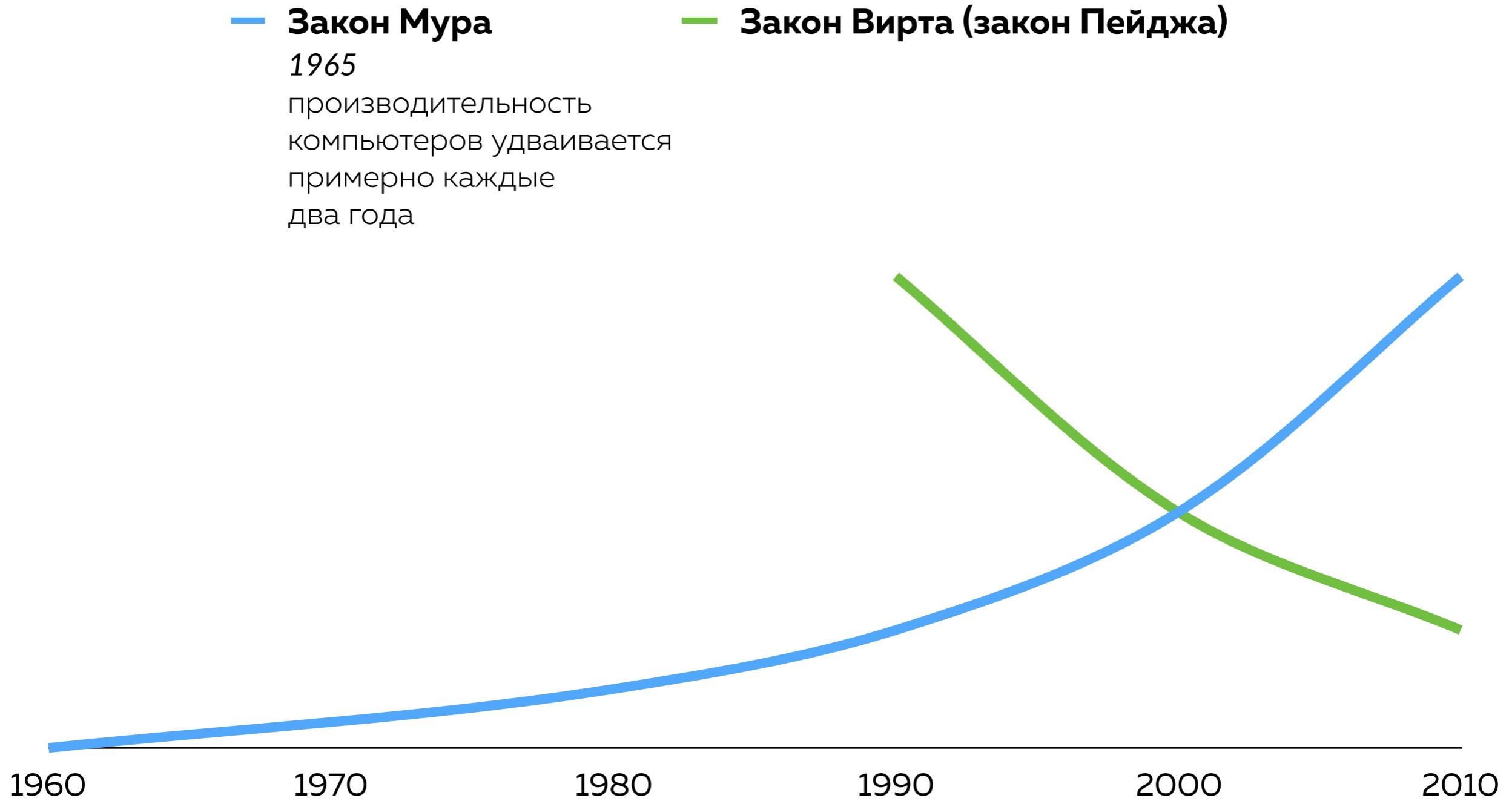
(исландск. *Rich Internet Application*, насыщенное интернет-приложение), иногда IIA – Installable Internet Application – сайт, обладающий свойствами настольного приложения



Законы развития ПО



Законы развития ПО



Законы развития ПО

— Закон Мура

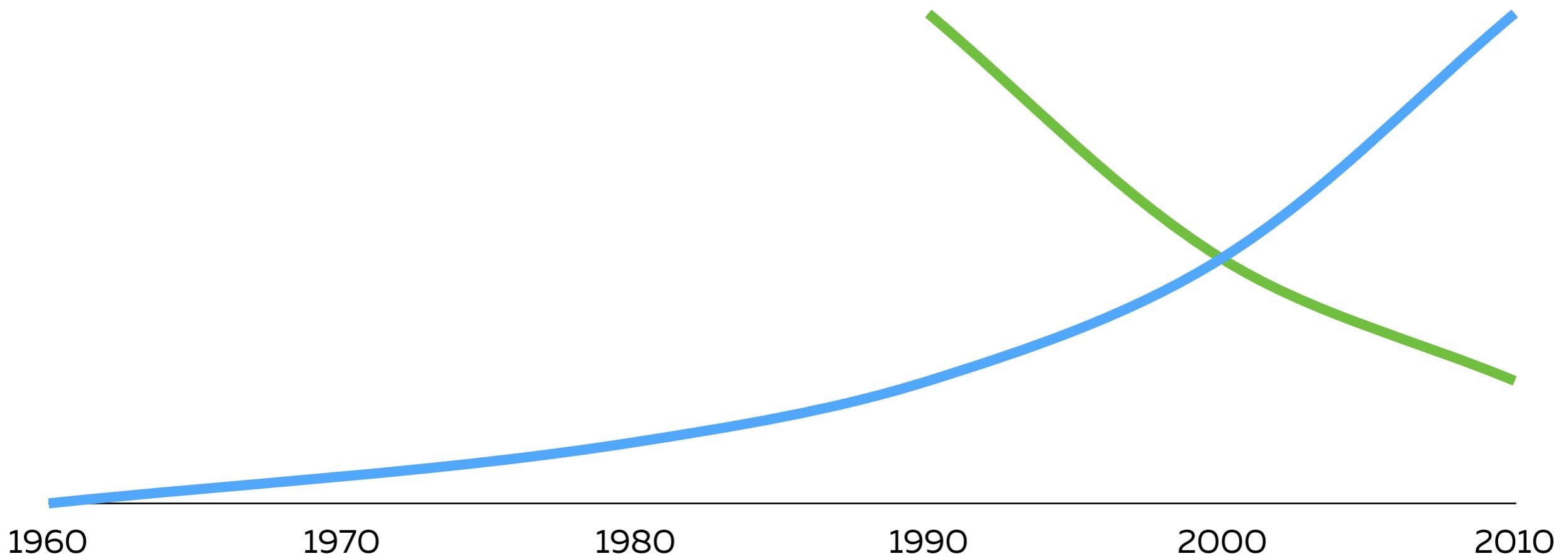
1965

производительность
компьютеров удваивается
примерно каждые
два года

— Закон Вирта (закон Пейджа)

~1995

медлительность программ
возрастает быстрее чем
производительность
компьютеров



Законы развития ПО

— Закон Мура

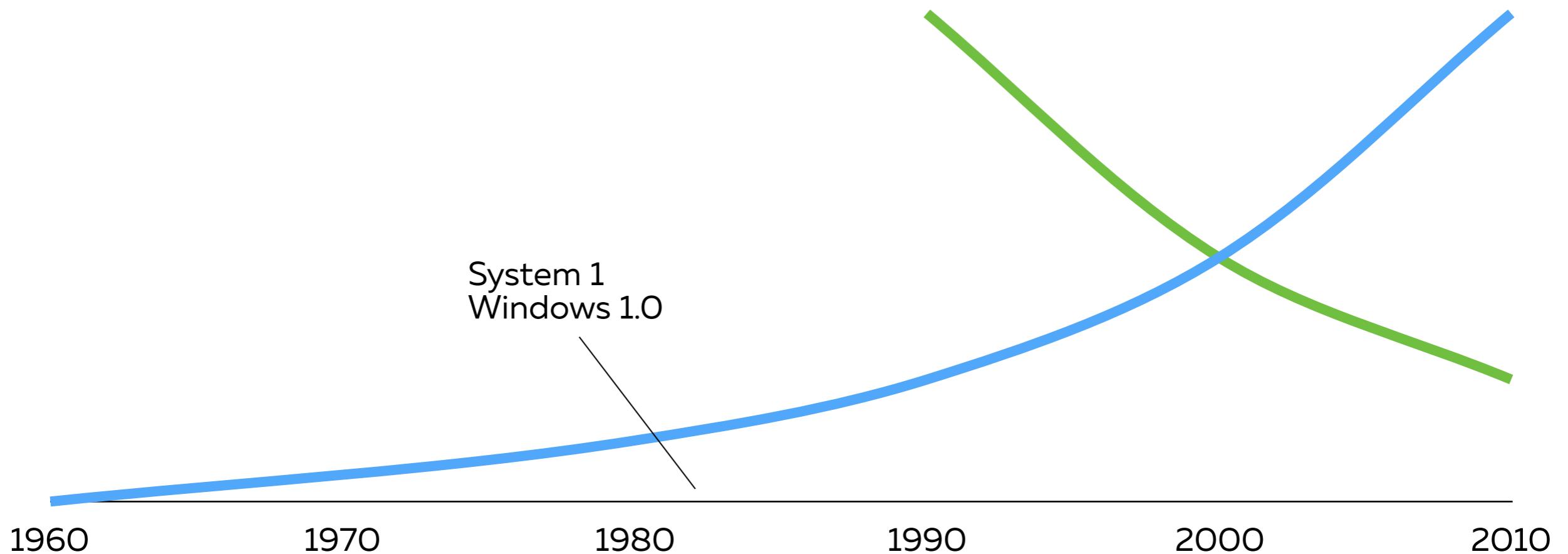
1965

производительность
компьютеров удваивается
примерно каждые
два года

— Закон Вирта (закон Пейджа)

~1995

медлительность программ
возрастает быстрее чем
производительность
компьютеров



Законы развития ПО

— Закон Мура

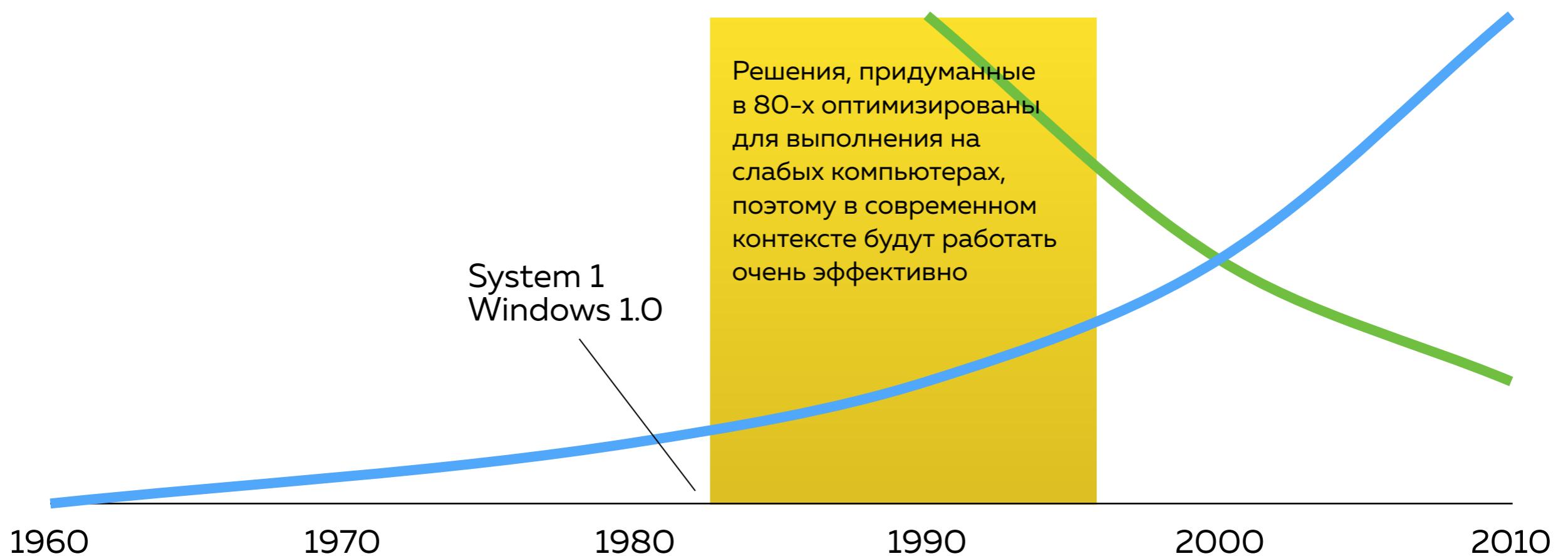
1965

производительность
компьютеров удваивается
примерно каждые
два года

— Закон Вирта (закон Пейджа)

~1995

медлительность программ
возрастает быстрее чем
производительность
компьютеров



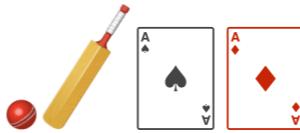
Компонента курильщика



Интерактивная демонстрация



Битовые карты .



Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



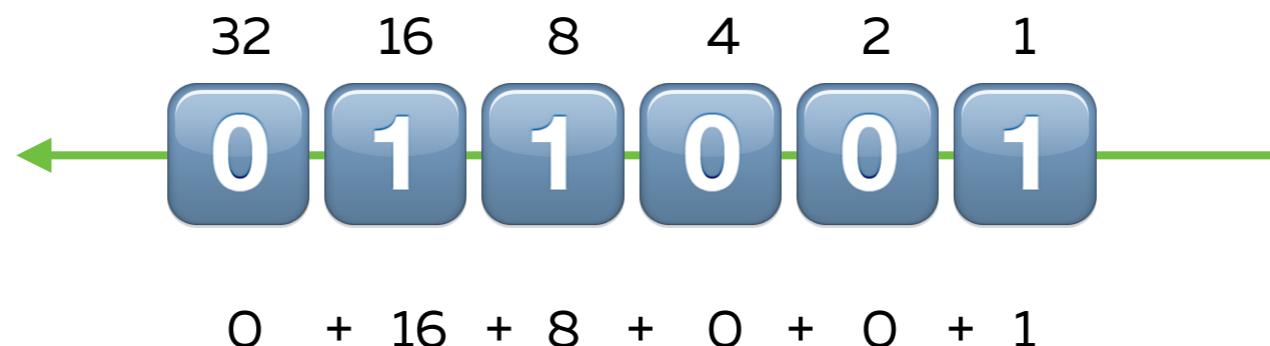
Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



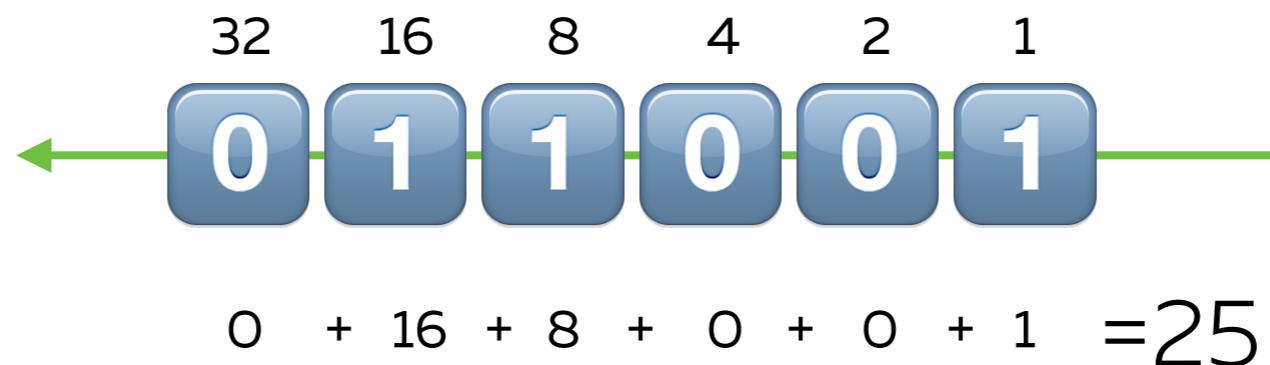
Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



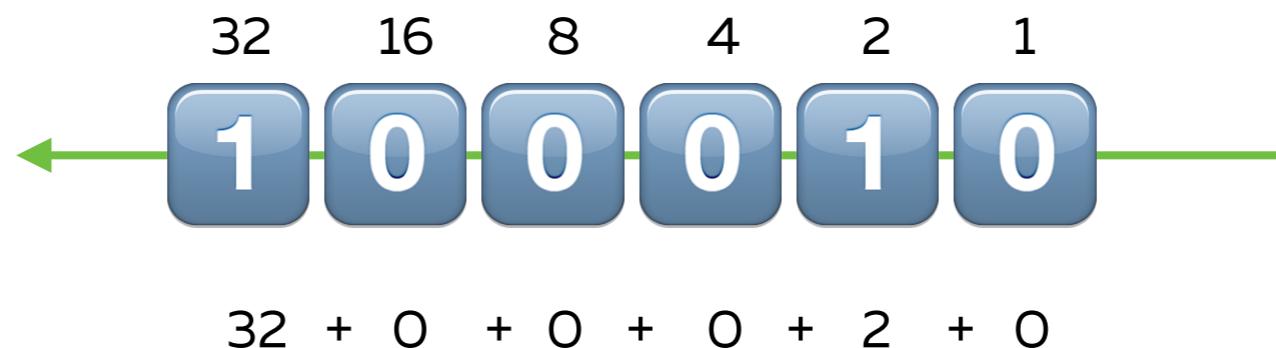
Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



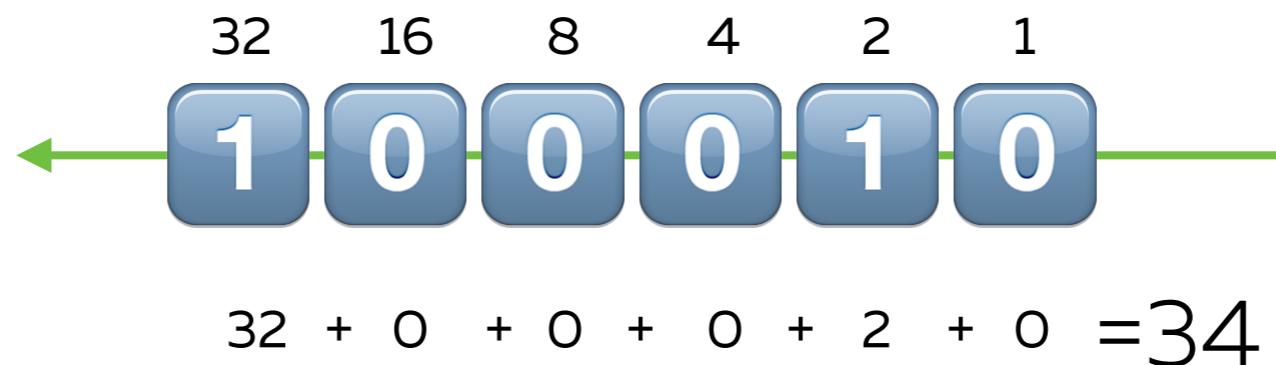
Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



Двоичная запись

В памяти компьютера все числа хранятся в двоичном виде – как последовательность нулей и единиц. Разряды читаются справа налево. Нумерация разрядов начинается с нуля. Единица в разряде означает что итоговое число содержит двойку в степени номера разряда



Битовые карты –

(яп. – *bitmap*) они же битовые массивы (кор. – *bitset*, *bitarray*) последовательности бит, которые кодируют не степени двойки, а любую другую информацию.
Хранятся в виде обычных чисел



Битовые карты

Битовой картой называется последовательность бит (нулей и единиц). В битовых картах может храниться не только число, но и любой другой набор данных



Битовые карты

Битовой картой называется последовательность бит (нулей и единиц). В битовых картах может храниться не только число, но и любой другой набор данных

- IP-адрес

192.168.1.1 / 255.255.255.0



Битовые карты

Битовой картой называется последовательность бит (нулей и единиц). В битовых картах может храниться не только число, но и любой другой набор данных

- IP-адрес

192.168.1.1 / 255.255.255.0

- цвет

#FACE8D



Битовые карты

Битовой картой называется последовательность бит (нулей и единиц). В битовых картах может храниться не только число, но и любой другой набор данных

- IP-адрес

192.168.1.1 / 255.255.255.0

- цвет

#FACE8D

- права пользователя

chmod -R 777 .



Битовые карты

Битовой картой называется последовательность бит (нулей и единиц). В битовых картах может храниться не только число, но и любой другой набор данных

- IP-адрес

192.168.1.1 / 255.255.255.0

- цвет

#FACE8D

- права пользователя

chmod -R 777 .

- карта уровня в игре или пиксели изображения



Битовые карты

Битовой картой называется последовательность бит (нулей и единиц). В битовых картах может храниться не только число, но и любой другой набор данных

- IP-адрес

192.168.1.1 / 255.255.255.0

- цвет

#FACE8D

- права пользователя

chmod -R 777 .

- карта уровня в игре или пиксели изображения

- состояние UX-компоненты



Компонента здорового человека



Интерактивная демонстрация. Кодирование состояний
в битах



Битовые операции –

логические операции над цепочками битов, в которых
биты выступают как значения true или false



Побитовое «ИЛИ»

Складывает переданные значения и сохраняет включенные биты обоих операторов.

Идеально подходит для сложения нескольких состояний

010

OR 100



Побитовое «ИЛИ»

Складывает переданные значения и сохраняет включенные биты обоих операторов.

Идеально подходит для сложения нескольких состояний

$$\begin{array}{r} 010 \\ \text{OR } \underline{100} \\ 0 \end{array}$$



Побитовое «ИЛИ»

Складывает переданные значения и сохраняет включенные биты обоих операторов.

Идеально подходит для сложения нескольких состояний

$$\begin{array}{r} 010 \\ \text{OR } \underline{100} \\ 10 \end{array}$$



Побитовое «ИЛИ»

Складывает переданные значения и сохраняет включенные биты обоих операторов.

Идеально подходит для сложения нескольких состояний

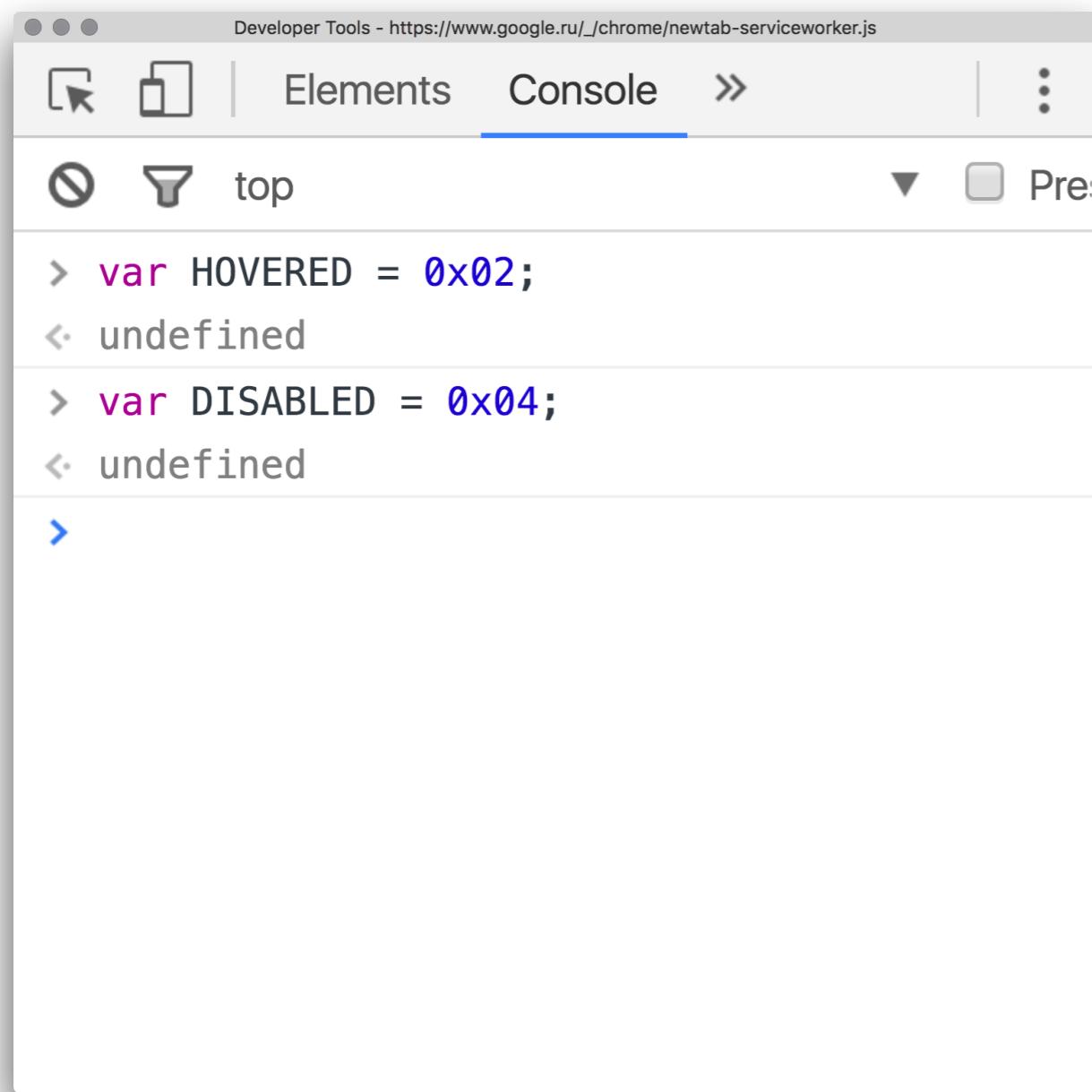
$$\begin{array}{r} 010 \\ \text{OR } \underline{100} \\ 110 \end{array}$$



Побитовое «ИЛИ»

Складывает переданные значения и сохраняет включенные биты обоих операторов.

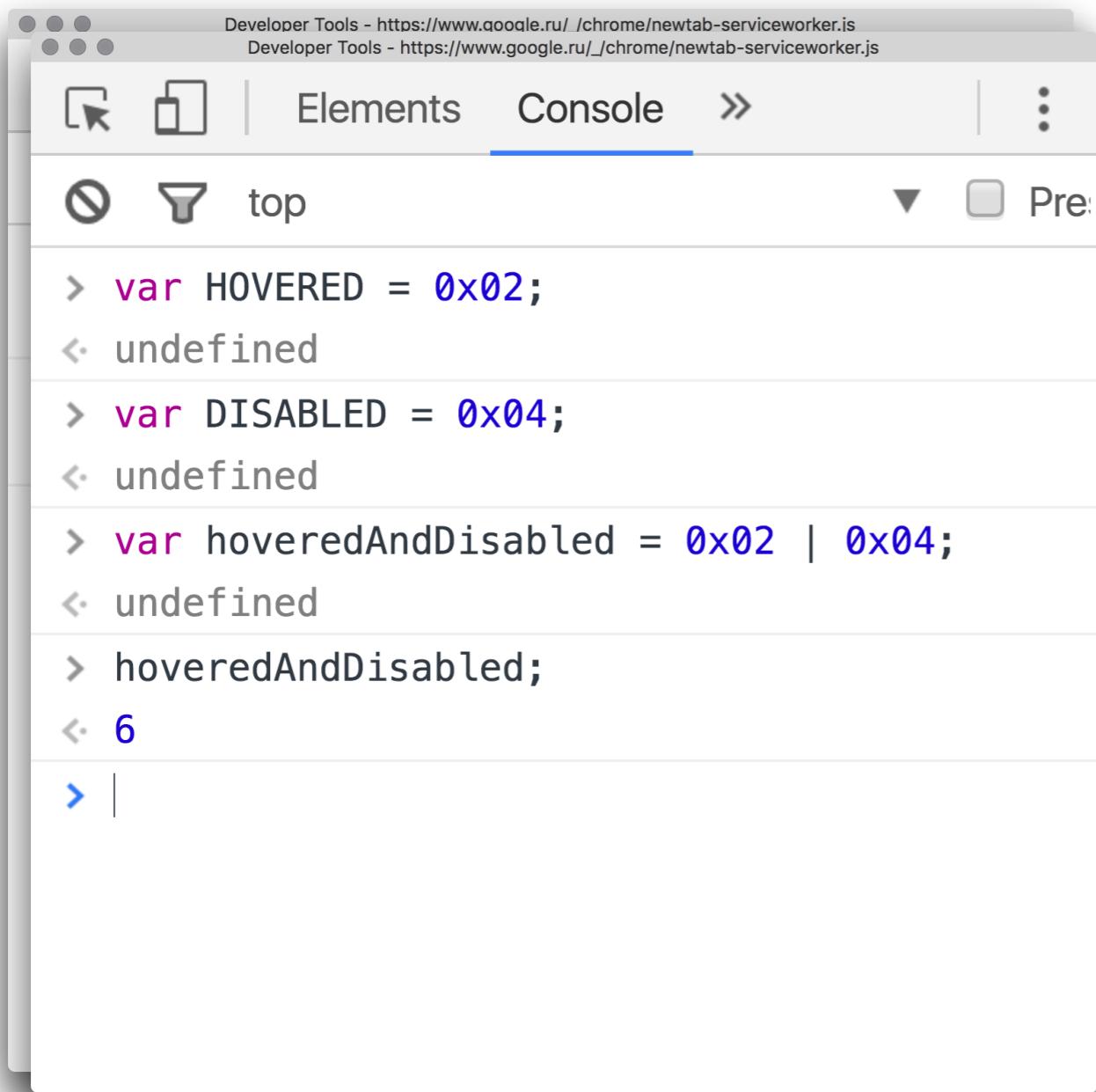
Идеально подходит для сложения нескольких состояний



Побитовое «ИЛИ»

Складывает переданные значения и сохраняет включенные биты обоих операторов.

Идеально подходит для сложения нескольких состояний



Developer Tools - https://www.google.ru/_/chrome/newtab-serviceworker.js
Developer Tools - https://www.google.ru/_/chrome/newtab-serviceworker.js

Elements **Console** »

top

```
> var HOVERED = 0x02;
< undefined
> var DISABLED = 0x04;
< undefined
> var hoveredAndDisabled = 0x02 | 0x04;
< undefined
> hoveredAndDisabled;
< 6
> |
```

$$\begin{array}{r} 010 \\ \text{OR} \quad 100 \\ \hline 110 \end{array}$$



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния

$$\begin{array}{r} 1010 \\ \text{AND } 1100 \\ \hline \end{array}$$



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния

$$\begin{array}{r} 1010 \\ \text{AND } 1100 \\ \hline 0 \end{array}$$



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния

$$\begin{array}{r} 1010 \\ \text{AND } 1100 \\ \hline 00 \end{array}$$



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния

$$\begin{array}{r} 1010 \\ \text{AND } 1100 \\ \hline 000 \end{array}$$



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния

$$\begin{array}{r} 1010 \\ \text{AND } 1100 \\ \hline 1000 \end{array}$$



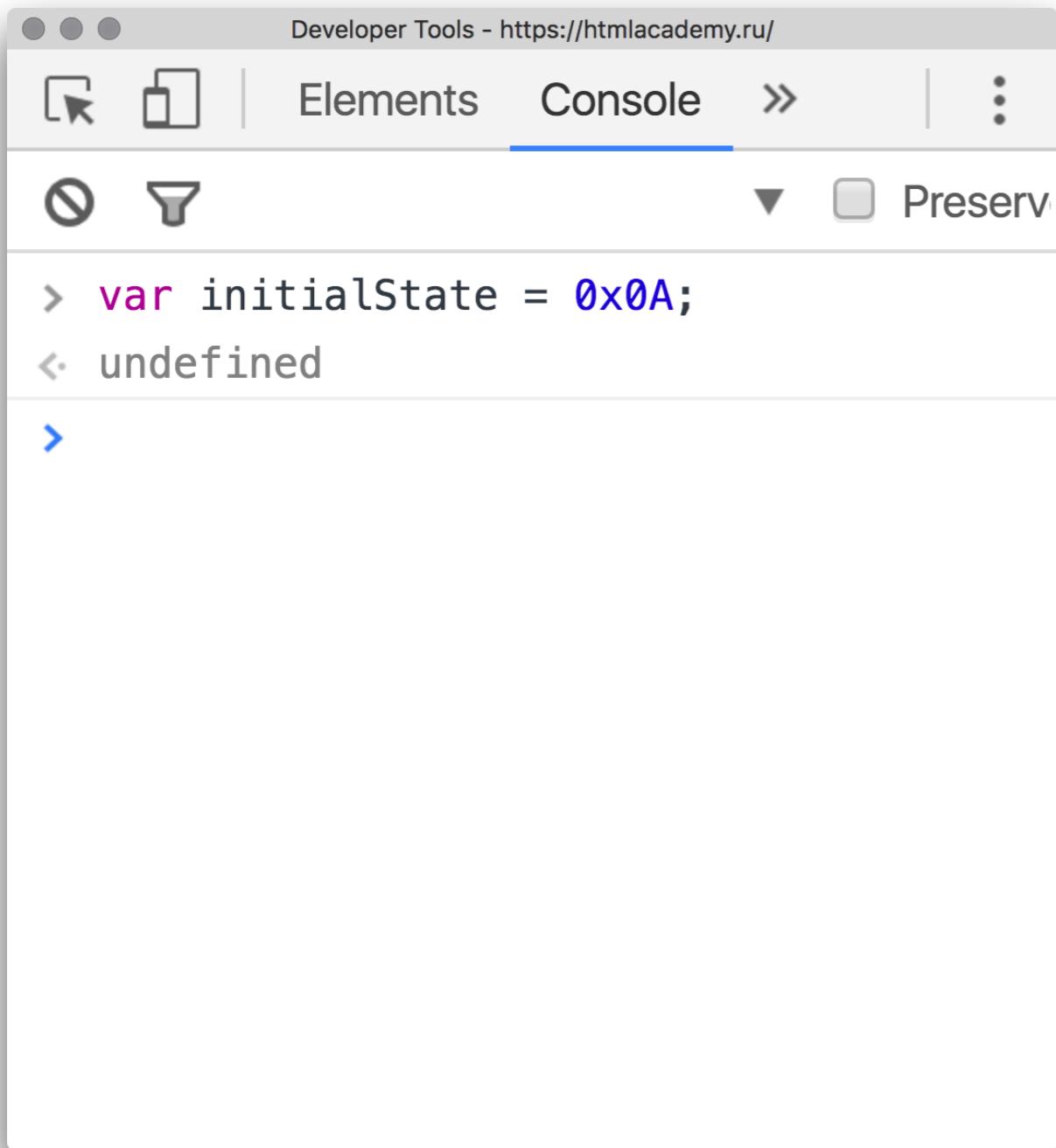
Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



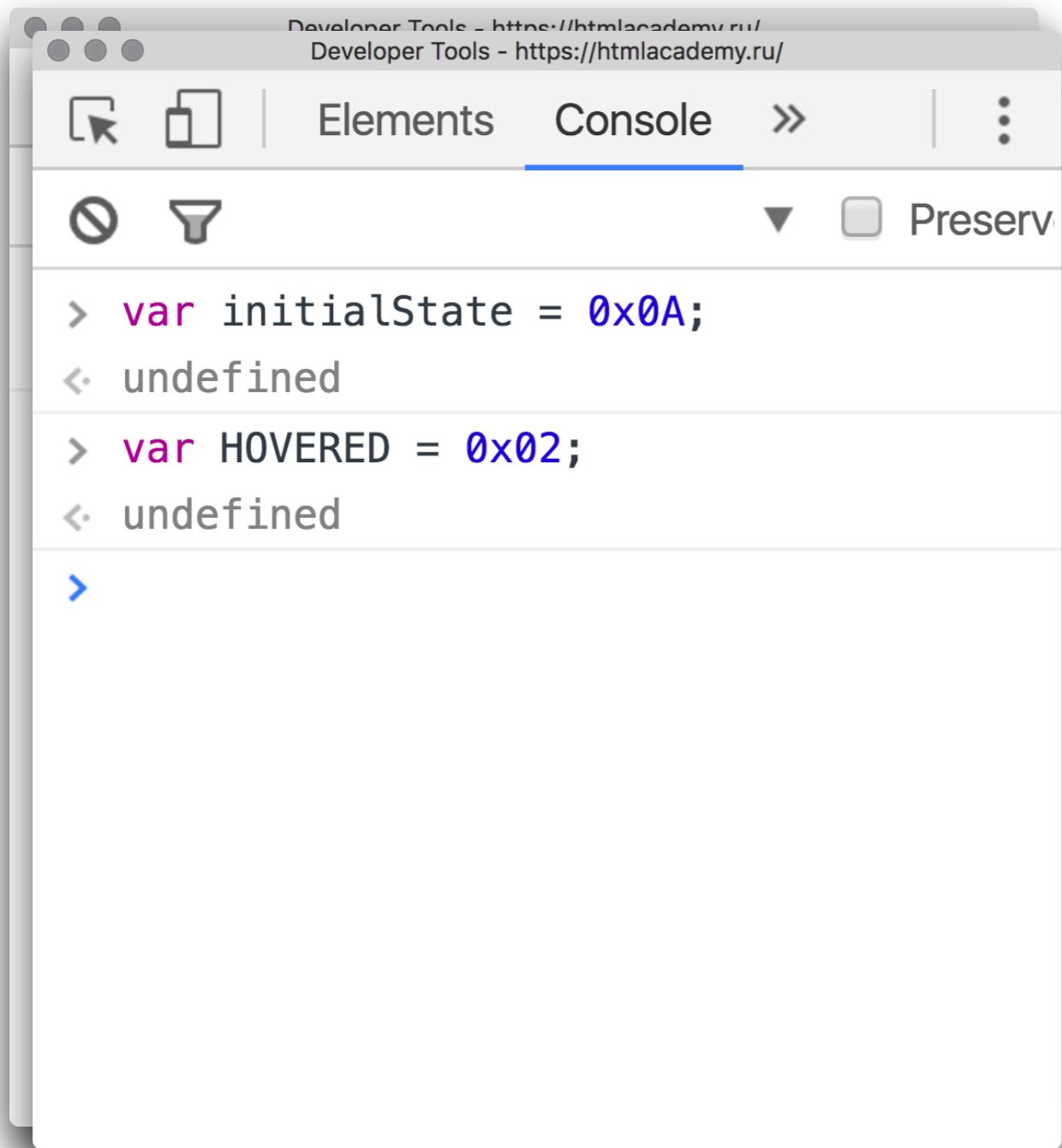
The screenshot shows the Chrome Developer Tools interface with the 'Console' tab selected. The console output window displays the following code:

```
> var initialState = 0x0A;  
< undefined  
>
```



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



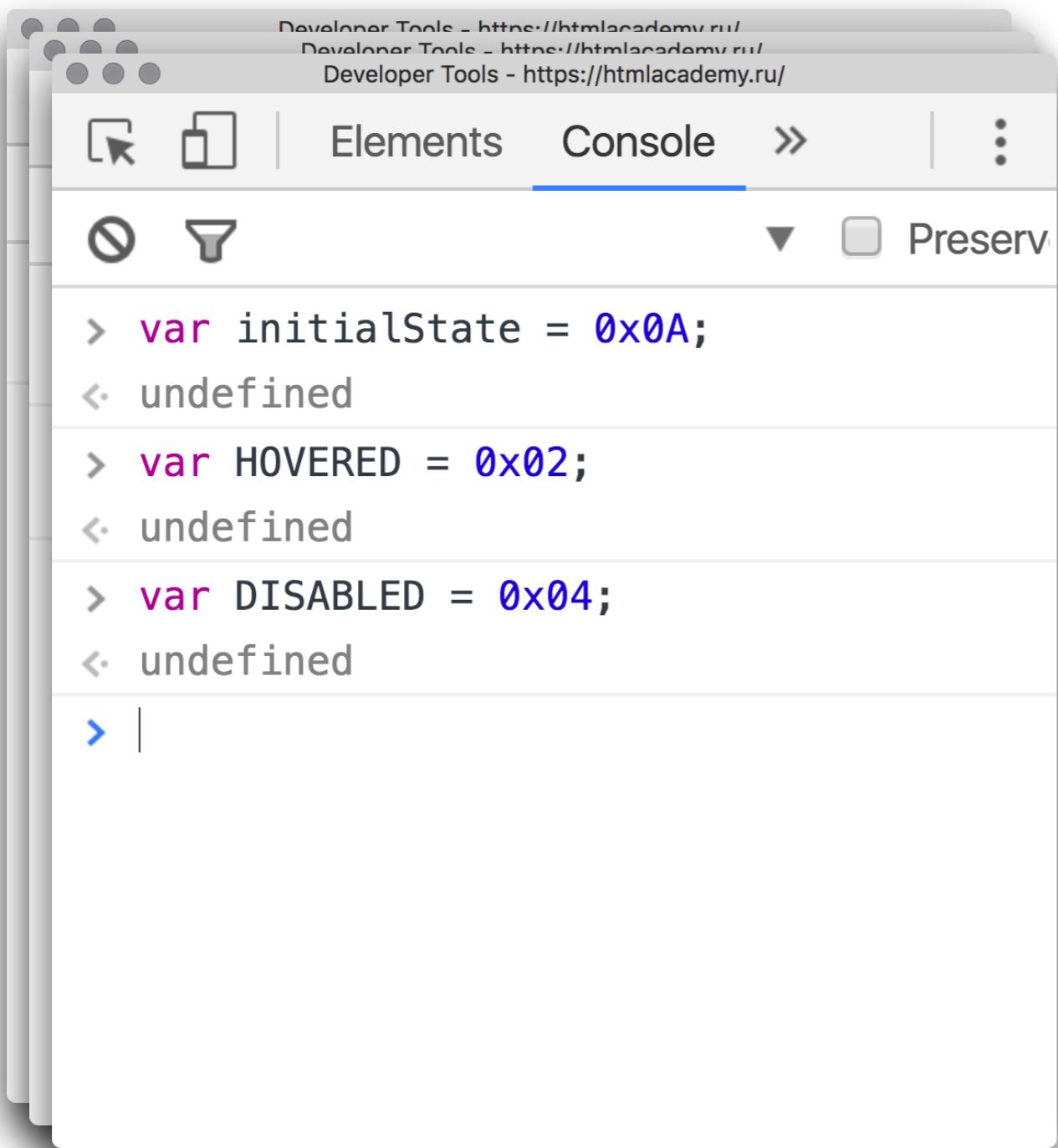
The screenshot shows a browser's developer tools console window. The title bar reads "Developer Tools - https://htmlacademy.ru/" and "Developer Tools - https://htmlacademy.ru/". Below the title bar, there are tabs for "Elements" and "Console", with "Console" being the active tab. The console area contains the following code:

```
> var initialState = 0x0A;  
< undefined  
> var HOVERED = 0x02;  
< undefined  
>
```



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



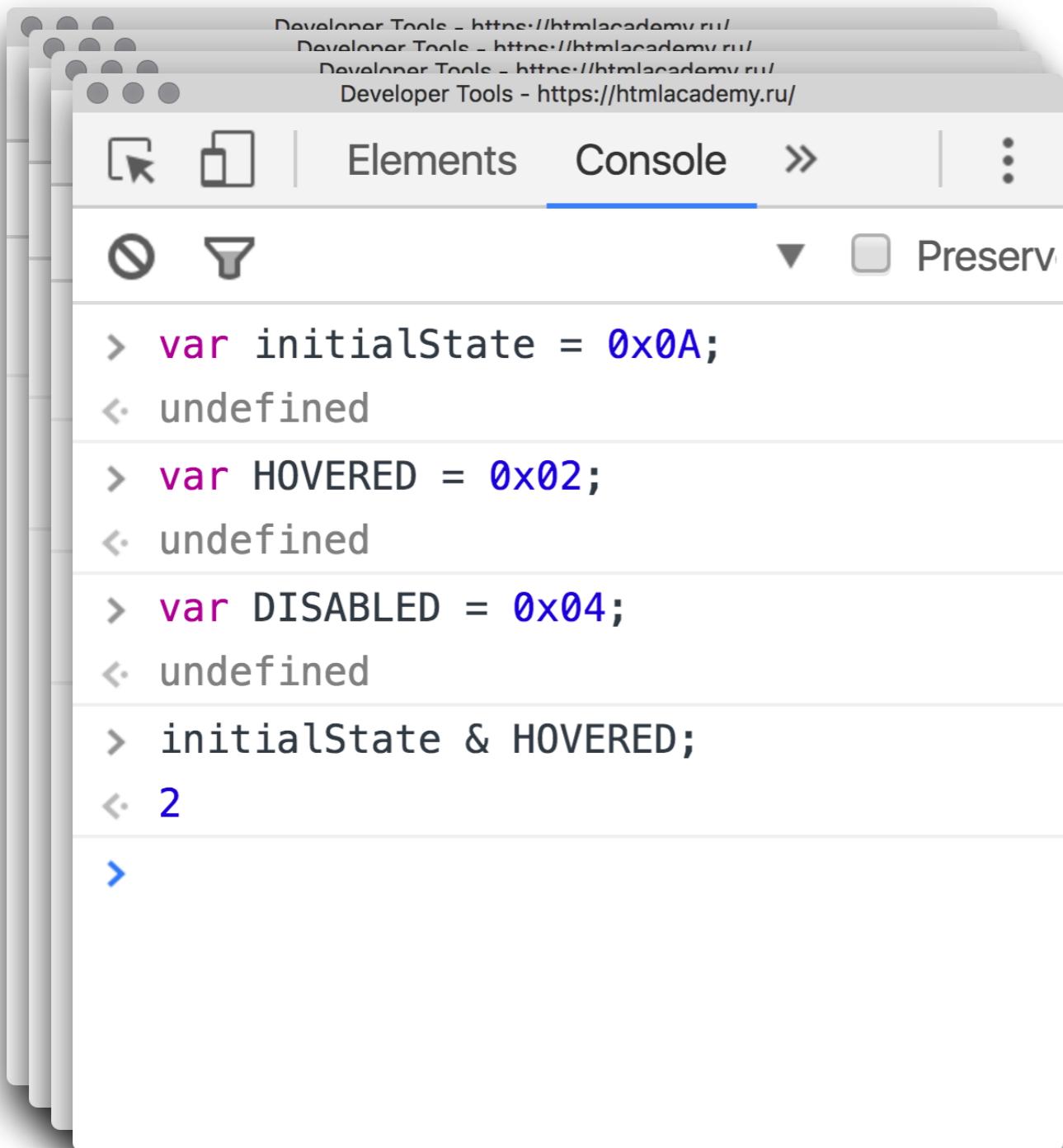
The screenshot shows the Chrome Developer Tools interface with the 'Console' tab selected. The console window displays the following code:

```
> var initialState = 0x0A;  
← undefined  
> var HOVERED = 0x02;  
← undefined  
> var DISABLED = 0x04;  
← undefined  
> |
```



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



The screenshot shows the Google Chrome Developer Tools Console tab selected. The code entered is:

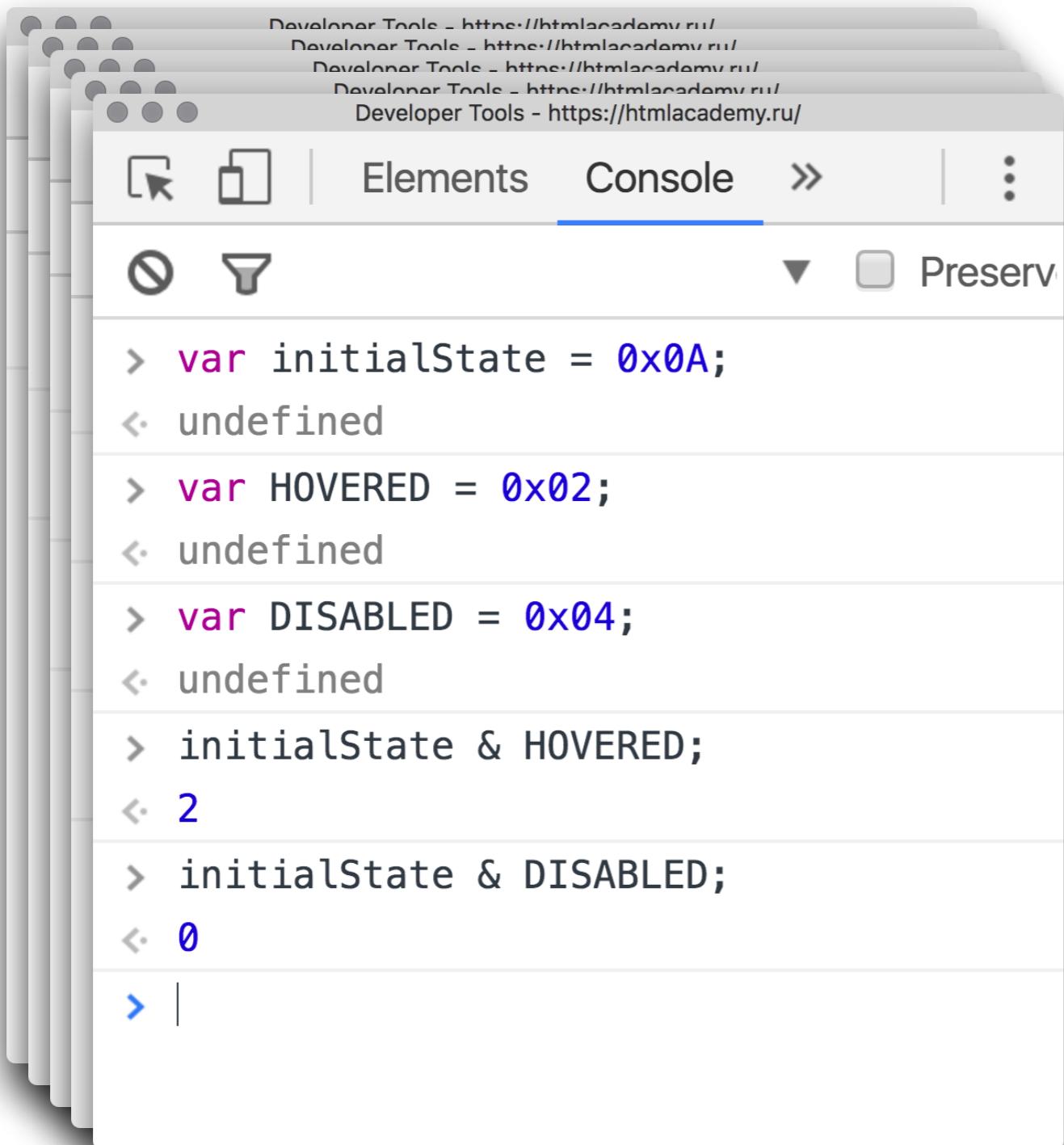
```
> var initialState = 0x0A;  
< undefined  
> var HOVERED = 0x02;  
< undefined  
> var DISABLED = 0x04;  
< undefined  
> initialState & HOVERED;  
< 2  
>
```

$$\begin{array}{r} 1010 \\ \text{AND } 0010 \\ = 0010 \end{array}$$



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



The screenshot shows the Chrome Developer Tools interface with multiple tabs labeled "Developer Tools - https://htmlacademy.ru/". The "Console" tab is active, displaying the following JavaScript code:

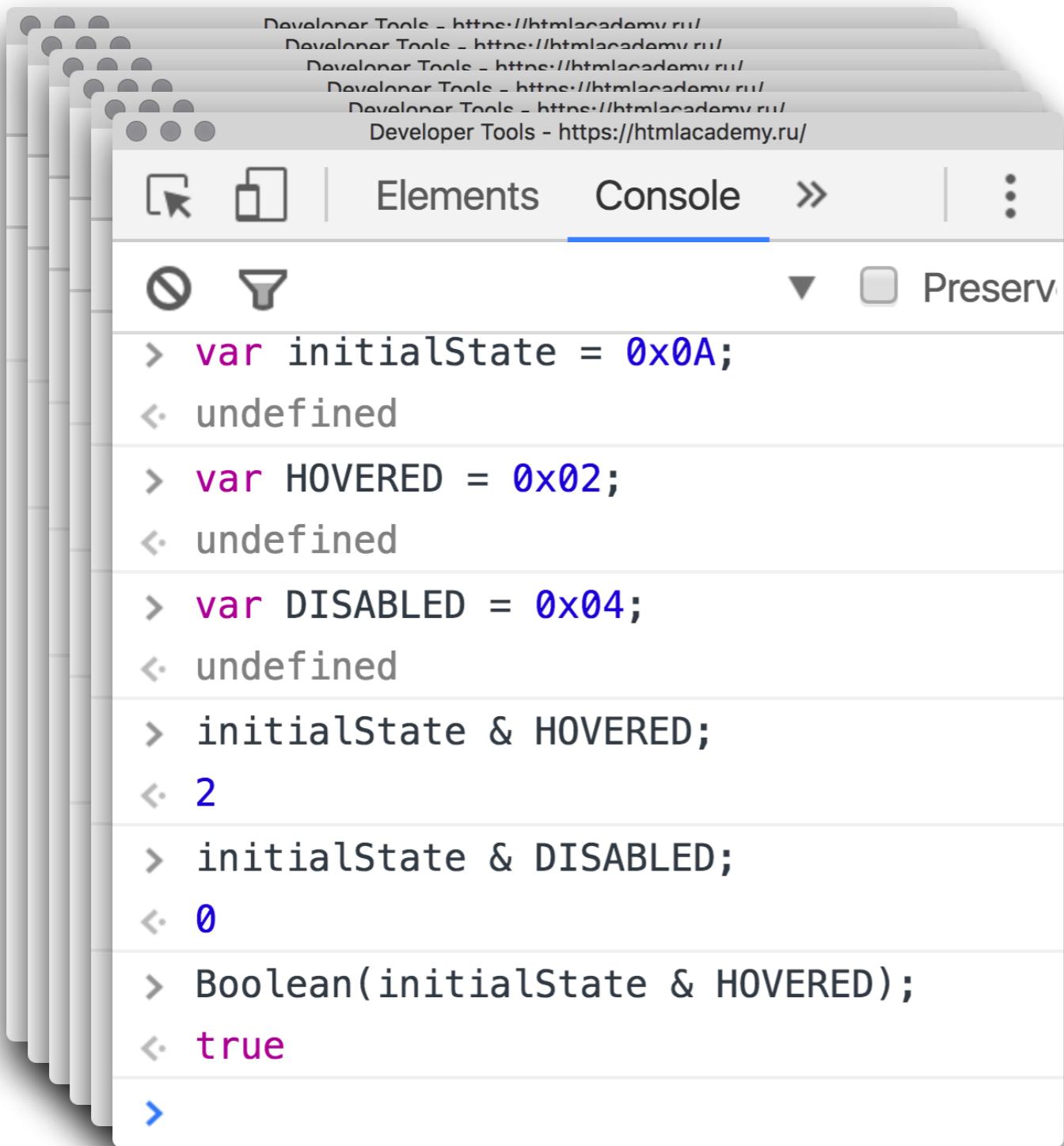
```
> var initialState = 0x0A;  
← undefined  
> var HOVERED = 0x02;  
← undefined  
> var DISABLED = 0x04;  
← undefined  
> initialState & HOVERED;  
← 2  
> initialState & DISABLED;  
← 0  
> |
```

$$\begin{array}{r} 1010 \\ \text{AND} \quad 0100 \\ \hline = \quad 0000 \end{array}$$



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



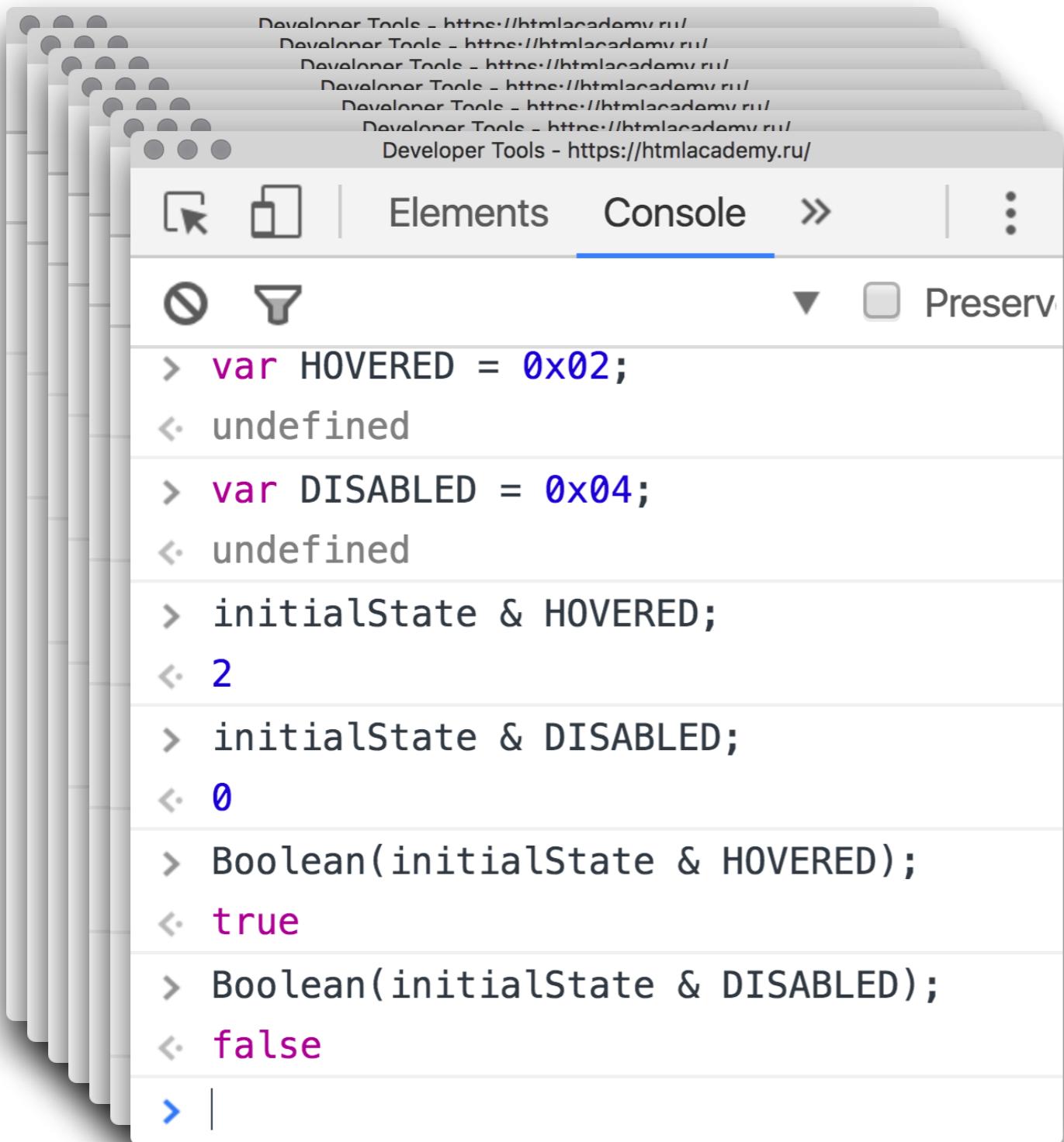
The screenshot shows the Developer Tools console in the Chrome browser. The tabs at the top are 'Elements' and 'Console'. The 'Console' tab is active, indicated by a blue underline. The console output is as follows:

```
> var initialState = 0x0A;
< undefined
> var HOVERED = 0x02;
< undefined
> var DISABLED = 0x04;
< undefined
> initialState & HOVERED;
< 2
> initialState & DISABLED;
< 0
> Boolean(initialState & HOVERED);
< true
>
```



Побитовое «И»

Сохраняет только те биты, которые были включены в обоих операндах. Идеально подходит для проверки состояния



The screenshot shows the Chrome Developer Tools interface with multiple tabs labeled "Developer Tools - https://htmlacademy.ru/". The "Console" tab is active, displaying the following code and results:

```
> var HOVERED = 0x02;
< undefined
> var DISABLED = 0x04;
< undefined
> initialState & HOVERED;
< 2
> initialState & DISABLED;
< 0
> Boolean(initialState & HOVERED);
< true
> Boolean(initialState & DISABLED);
< false
> |
```



Компонента здорового человека



Интерактивная демонстрация. Продолжение



Декларативный стиль

Мы описываем не последовательность шагов
(императивный стиль), а реакцию на изменение
состояния

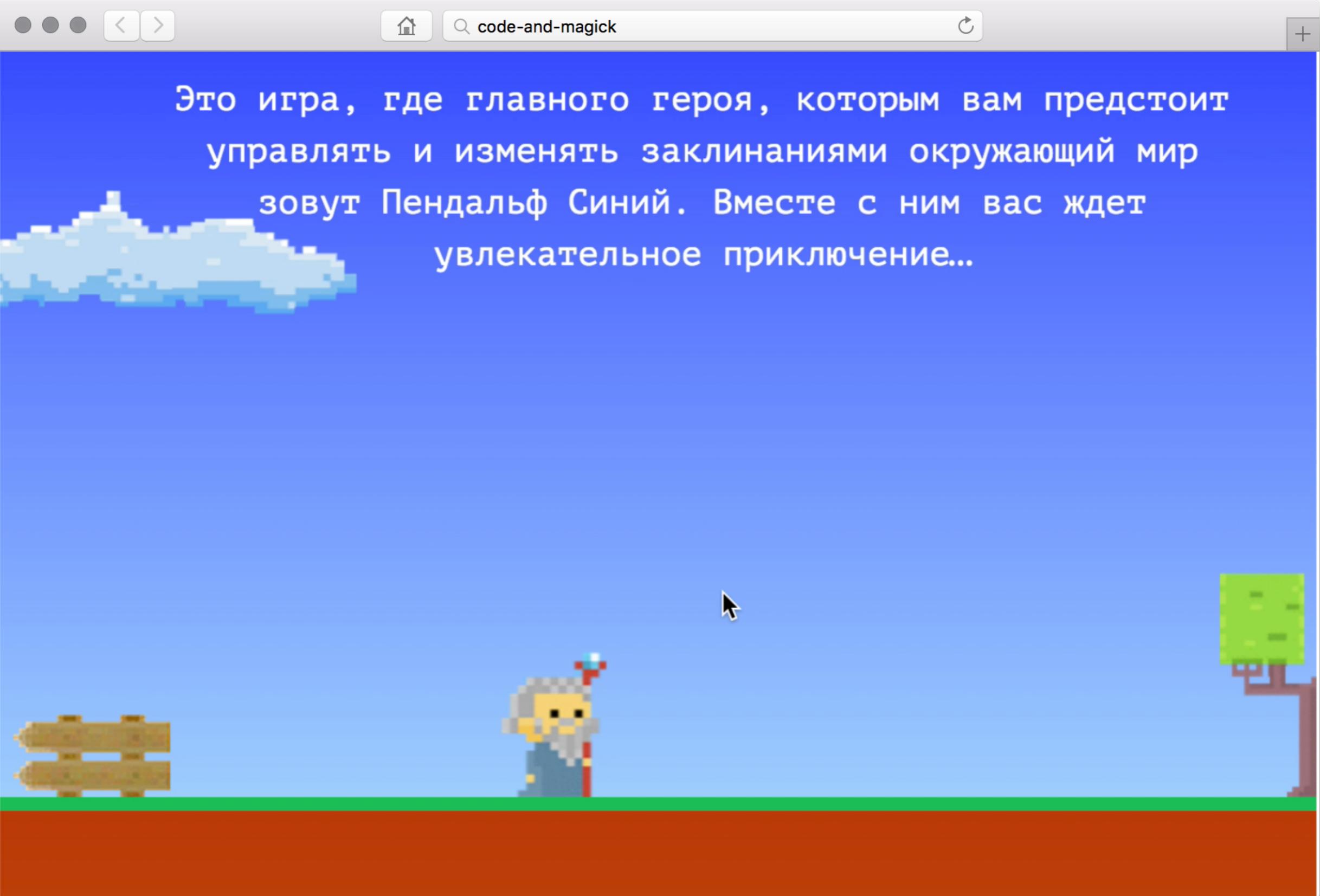


Полученный код содержит всего пять (одну) управляемых строк. Остальной код – статические словари. Вне зависимости от размеров словарей (количество состояний) размер управляющего кода меняться не будет



Использование состояний для описания поведения





Описание поведения

В качестве значений в словарях можно использовать не только обычные значения, но и функции. В этом случае можно описывать поведение каждого состояния независимо



Описание поведения

В качестве значений в словарях можно использовать не только обычные значения, но и функции. В этом случае можно описывать поведение каждого состояния независимо

```
var Direction = {  
    RIGHT: 0x01  
};
```

```
var DirectionBehaviour = {  
    0x01: function(character) {  
        return Object.assign({  
            left: character.left + character.hSpeed  
        }, character);  
    }  
};
```



Описание поведения

В качестве значений в словарях можно использовать не только обычные значения, но и функции. В этом случае можно описывать поведение каждого состояния независимо

```
var Direction = {  
    RIGHT: 0x01,  
    TOP: 0x02  
};
```

```
var DirectionBehaviour = {  
    0x01: function(character) {},  
    0x02: function(character) {}  
};
```



Даже Redux!



Даже Redux!

Можно использовать битовые маски как параметр туре для управляющих объектов action. Так, за одно обращение к хранилищу можно делать несколько операций с данными



Даже Redux!

Можно использовать битовые маски как параметр type для управляющих объектов action. Так, за одно обращение к хранилищу можно делать несколько операций с данными

```
const ActionType = {  
  UPDATE_1: 0x01,  
  UPDATE_2: 0x02  
};  
  
store.dispatch({  
  type: ActionType.UPDATE_1 | ActionType.UPDATE_2  
});
```



И кроме этого есть ещё



И кроме этого есть ещё

- **исключающие или XOR, ^**
используется для переключения состояния (*toggle*)



И кроме этого есть ещё

- **исключающее или XOR, ^**
используется для переключения состояния (*toggle*)
- **побитовый сдвиг влево <<**



И кроме этого есть ещё

- **исключающее или XOR, ^**
используется для переключения состояния (*toggle*)
- **побитовый сдвиг влево <<**
- **побитовый сдвиг вправо >>**



И кроме этого есть ещё

- **исключающее или XOR, ^**
используется для переключения состояния (*toggle*)
- **побитовый сдвиг влево <<**
- **побитовый сдвиг вправо >>**
- **сдвиг вправо с заполнением нулями >>>**



И кроме этого есть ещё

- **исключающее или XOR, ^**
используется для переключения состояния (*toggle*)
- **побитовый сдвиг влево <<**
- **побитовый сдвиг вправо >>**
- **сдвиг вправо с заполнением нулями >>>**
- **битовые маски**
когда одно состояние хранится в нескольких соседних битах используются маски, которые показывают единицами, в какие именно биты состояние записано (*IP-адрес и маска подсети*)



Зависимости в форме





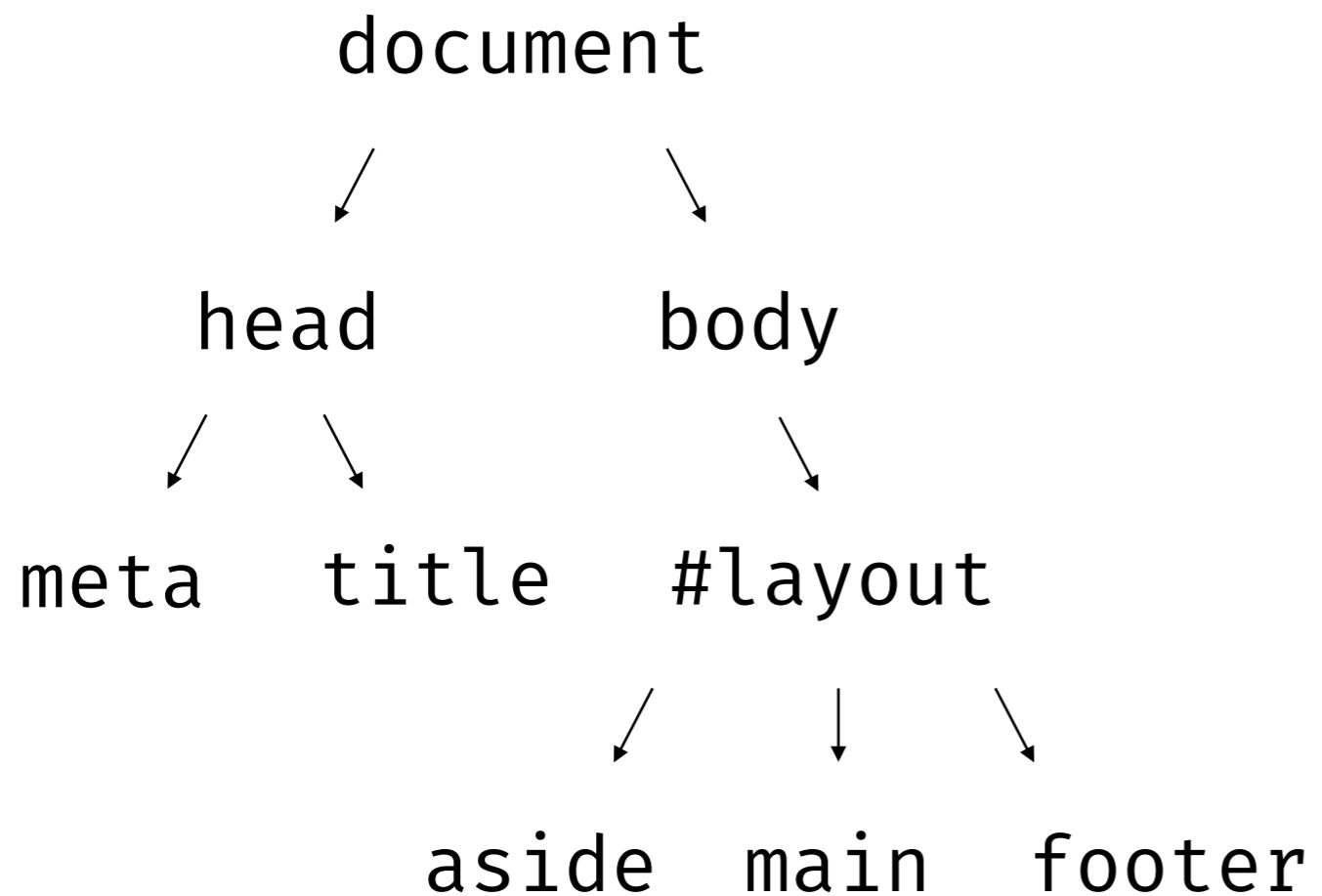
Деревья





Деревья

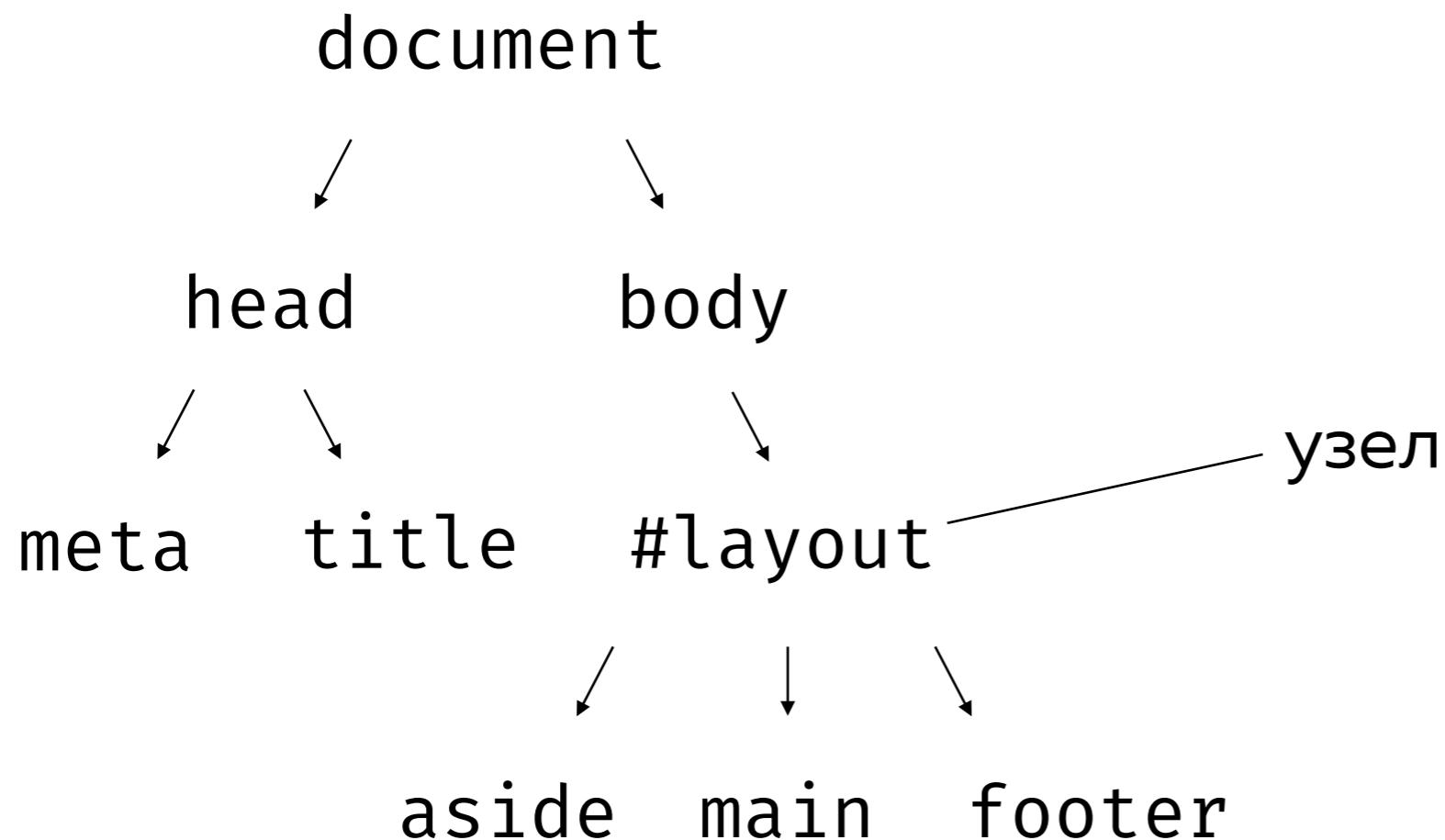
Связанная структура данных, состоящая из вложенных друг в друга повторяющихся узлов. Самый наглядный пример, для фронтенд-разработчика – DOM-дерево





Деревья

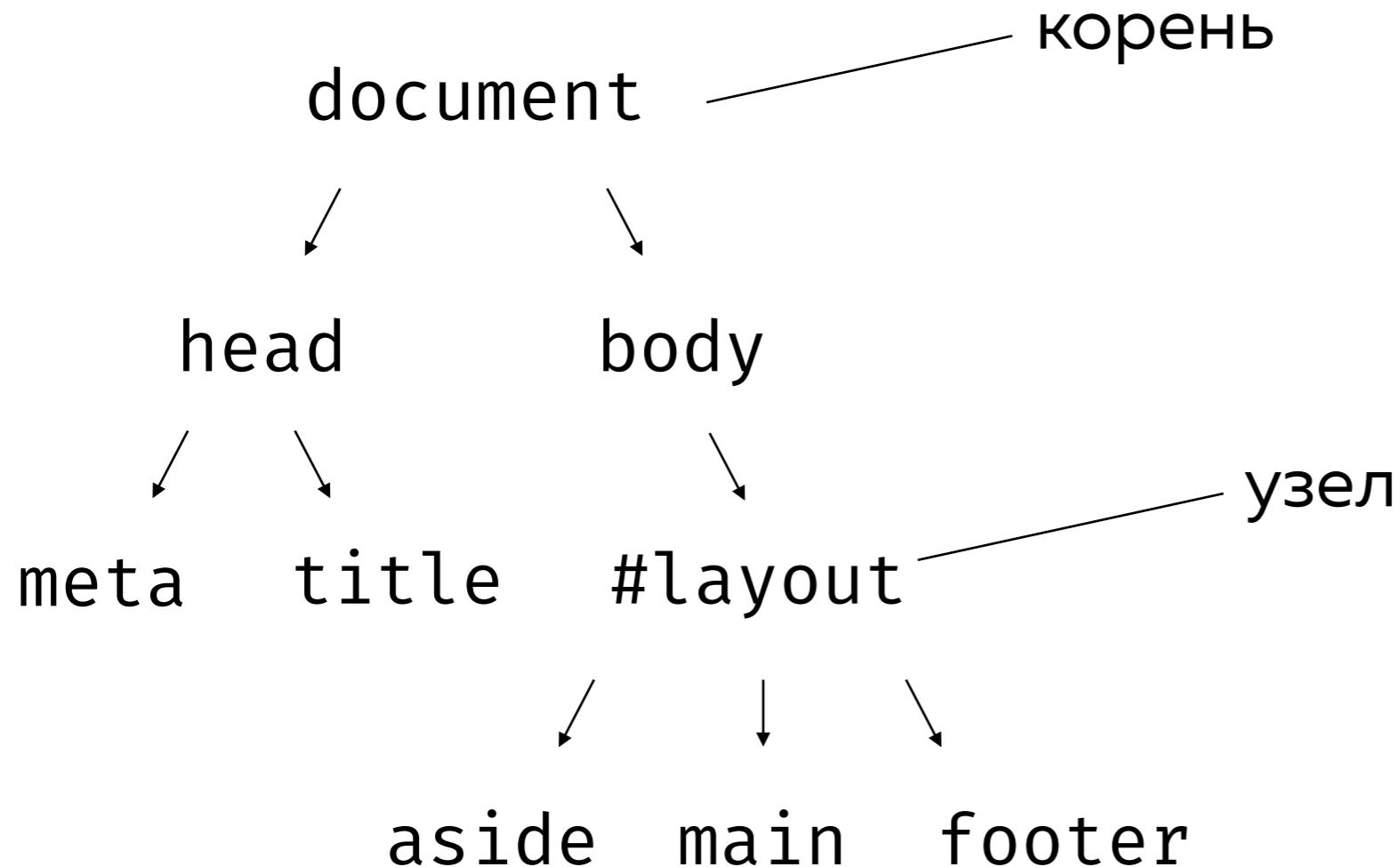
Связанная структура данных, состоящая из вложенных друг в друга повторяющихся узлов. Самый наглядный пример, для фронтенд-разработчика – DOM-дерево





Деревья

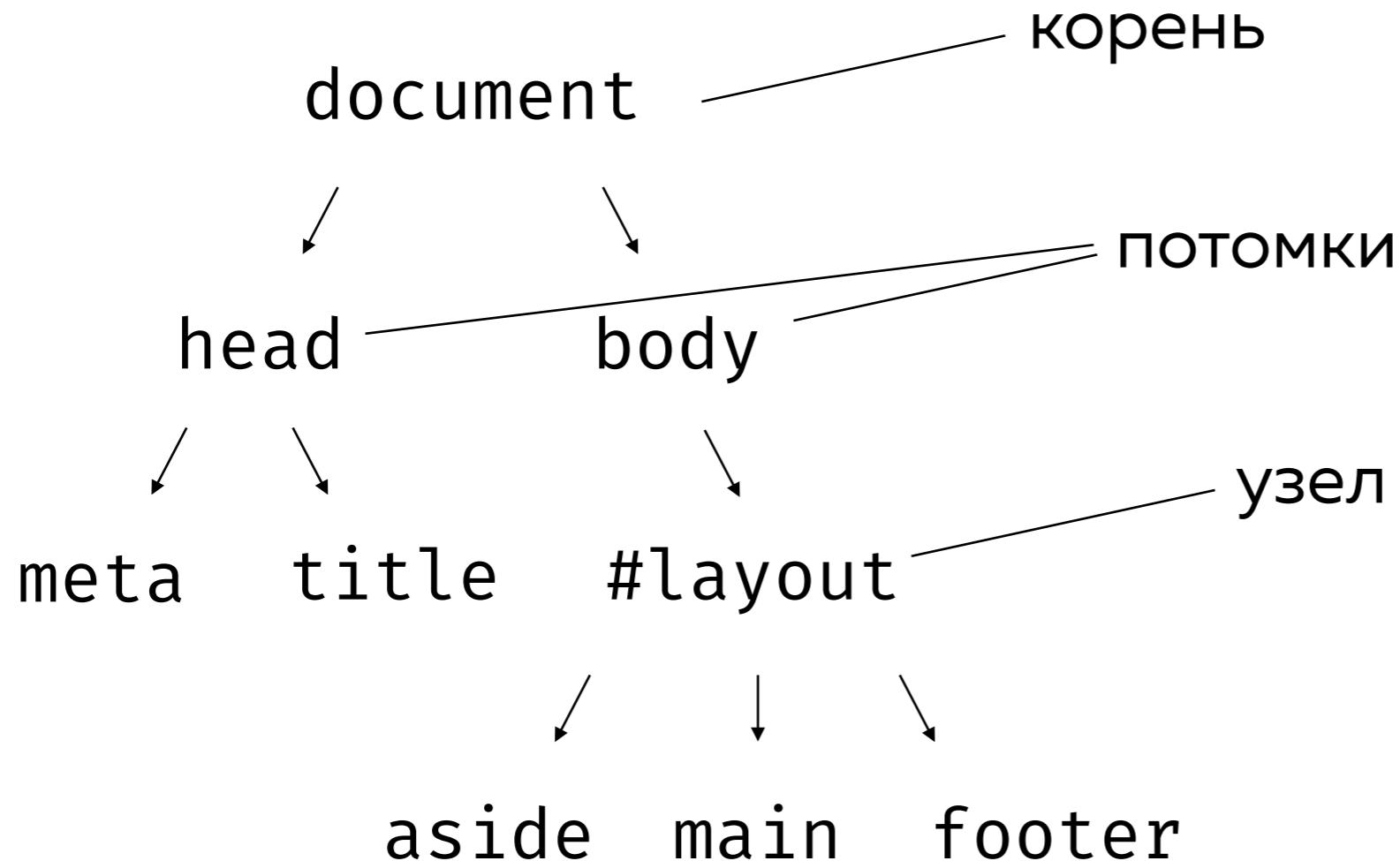
Связанная структура данных, состоящая из вложенных друг в друга повторяющихся узлов. Самый наглядный пример, для фронтенд-разработчика – DOM-дерево





Деревья

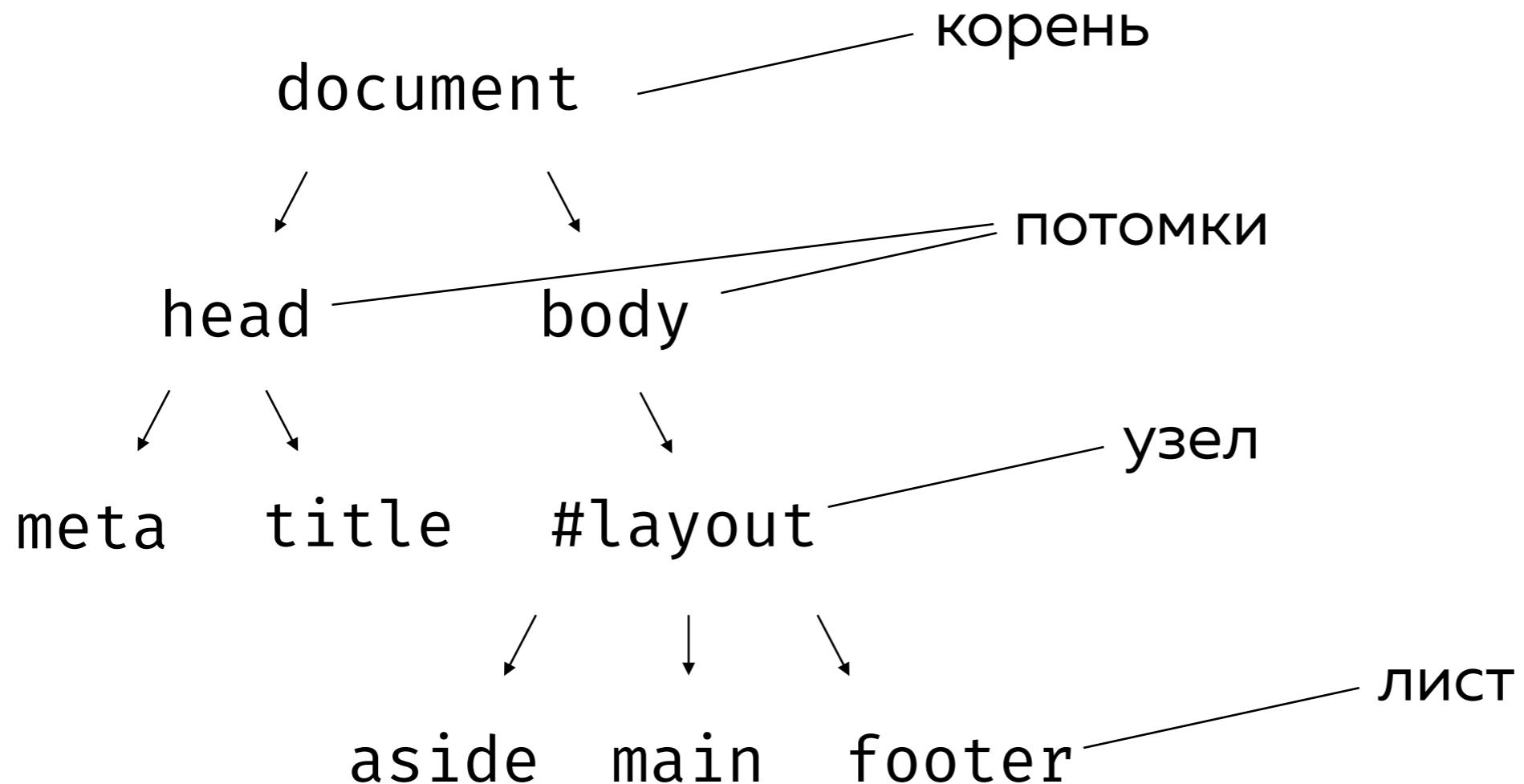
Связанная структура данных, состоящая из вложенных друг в друга повторяющихся узлов. Самый наглядный пример, для фронтенд-разработчика – DOM-дерево





Деревья

Связанная структура данных, состоящая из вложенных друг в друга повторяющихся узлов. Самый наглядный пример, для фронтенд-разработчика – DOM-дерево





Деревья

Интерактивная демонстрация. Использование деревьев
для описания зависимостей в форме



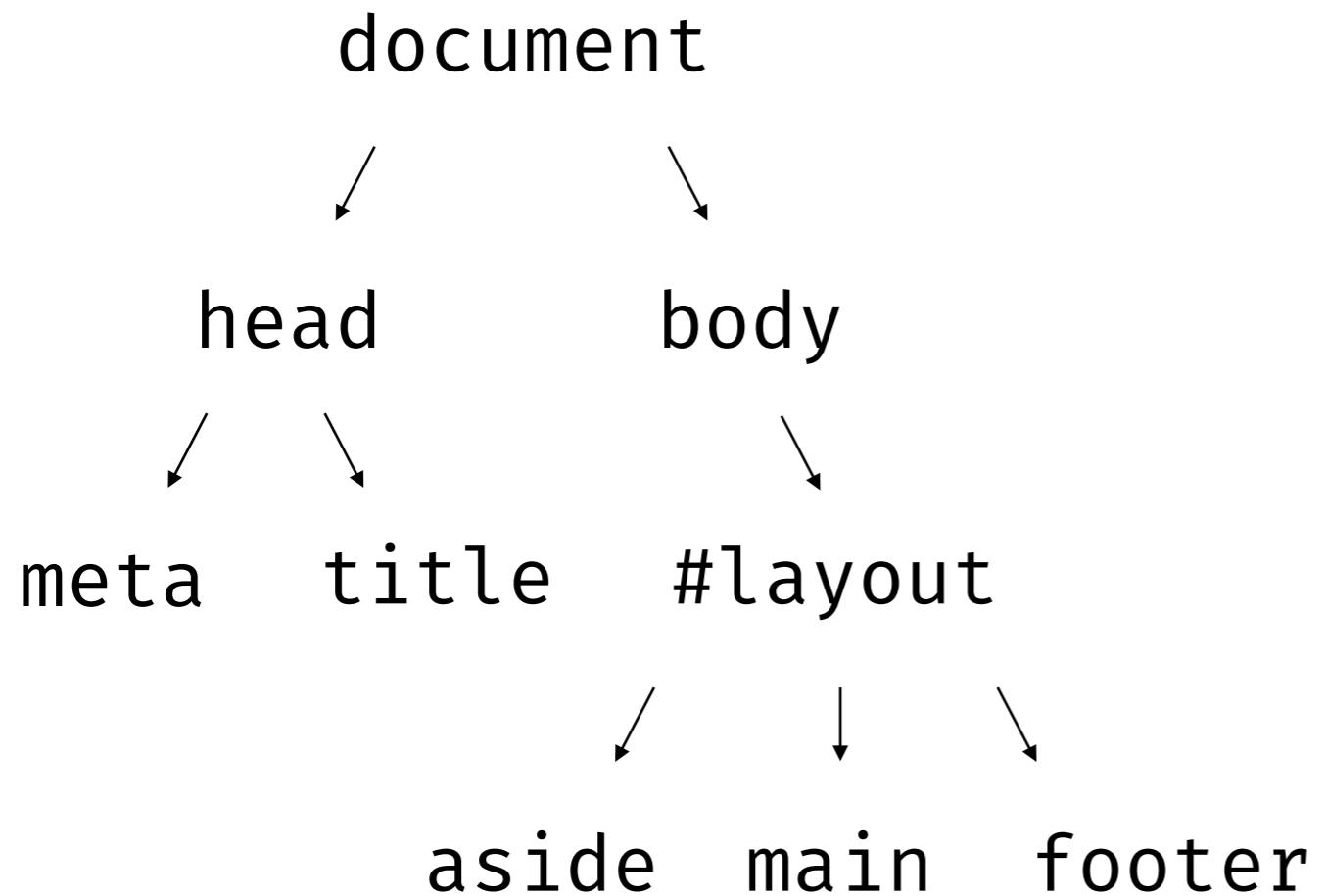


Обход дерева

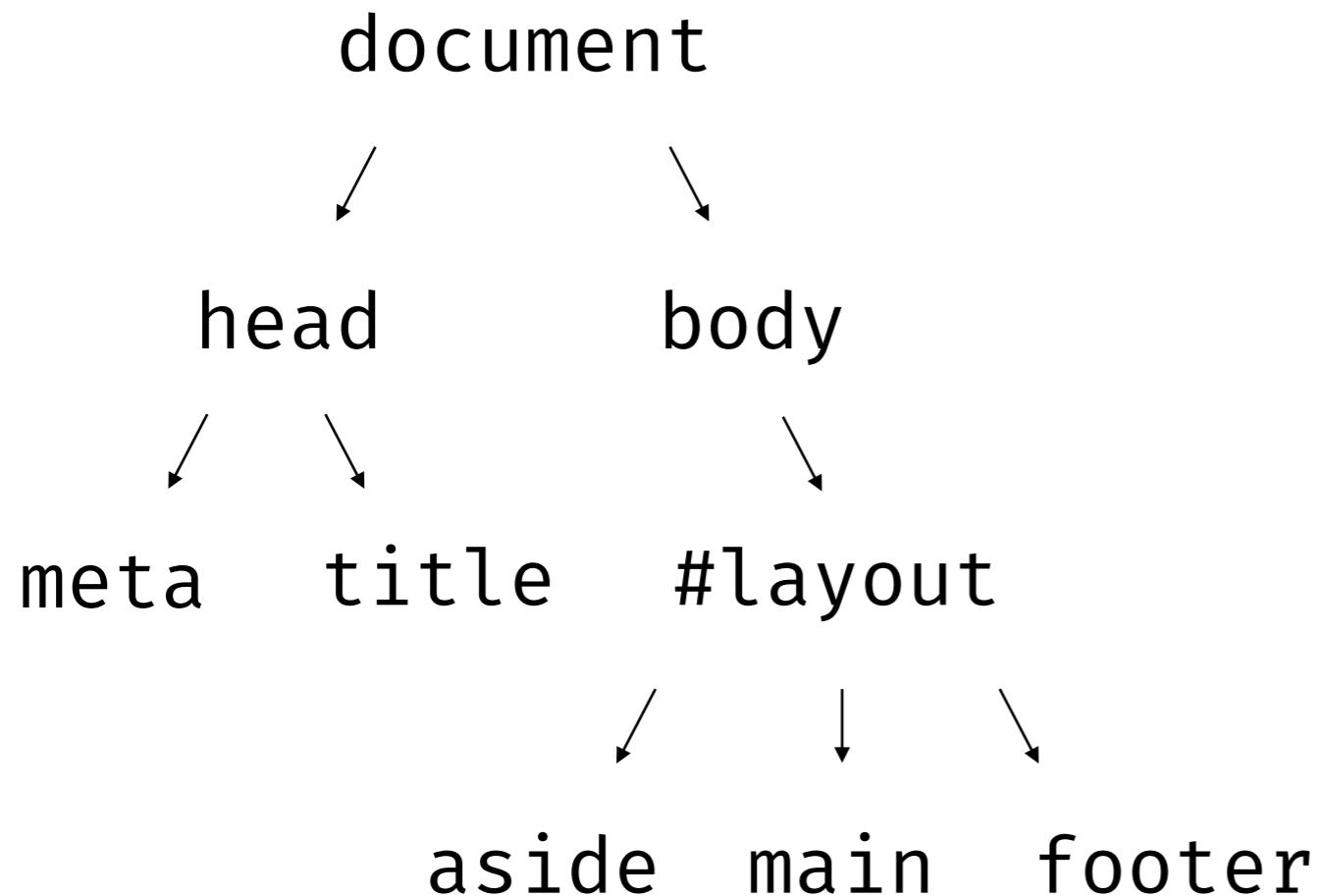
Какими способами можно обходить дерево и на что может повлиять изменение способа обхода



Деревья



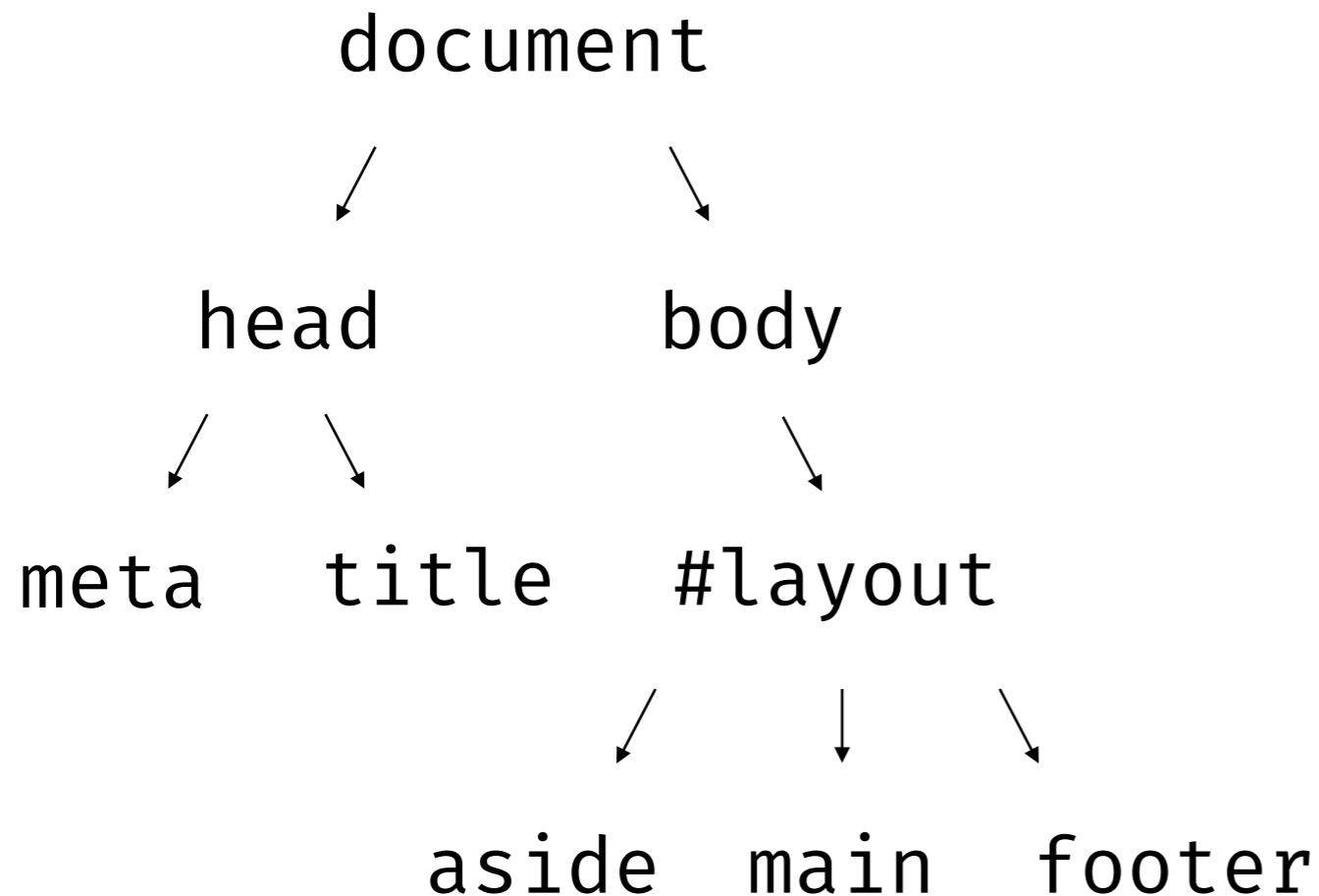
Деревья



Существует несколько способов перебора элементов дерева:



Деревья

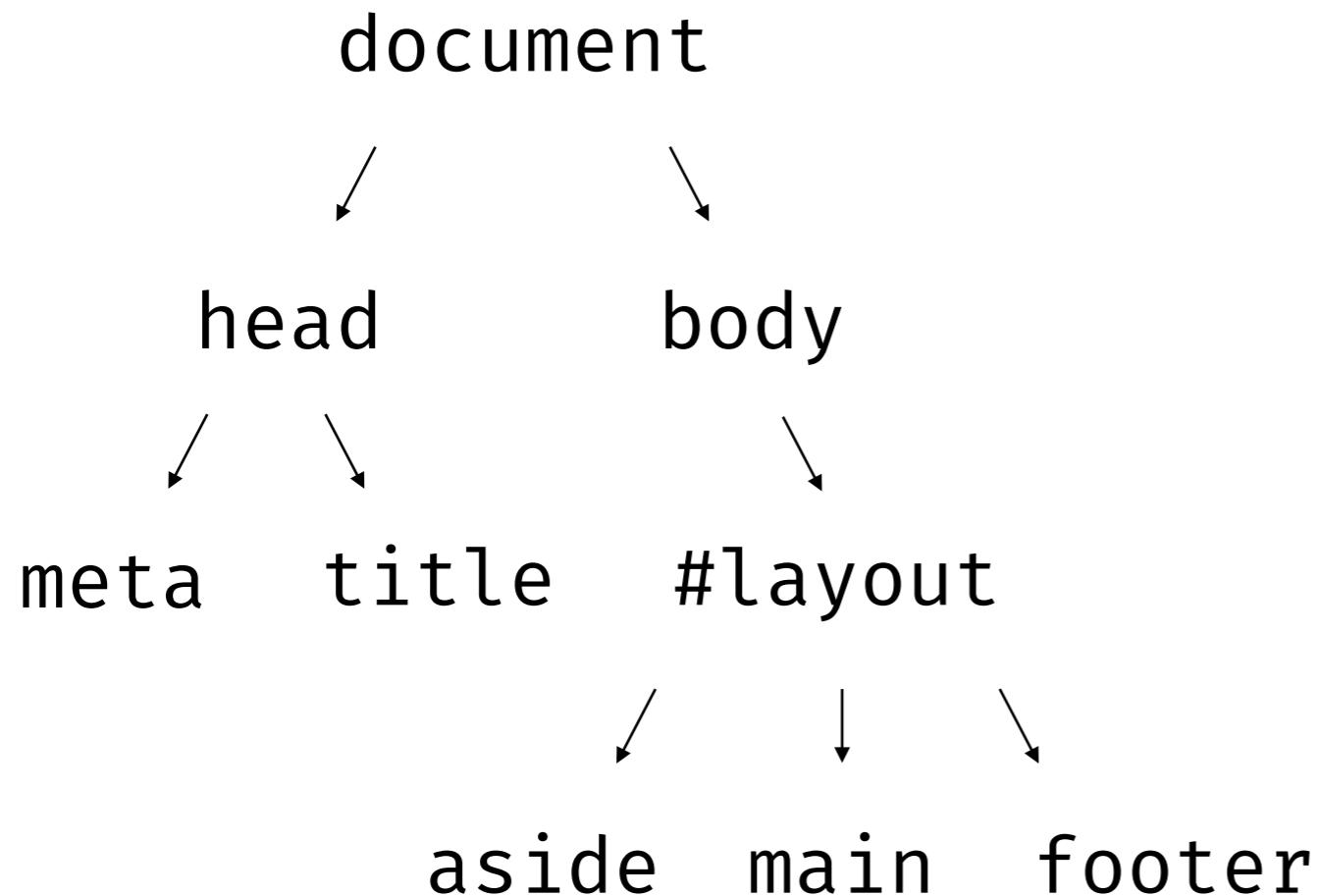


Существует несколько способов перебора элементов дерева:

- DFS (поиск в глубину)



Деревья

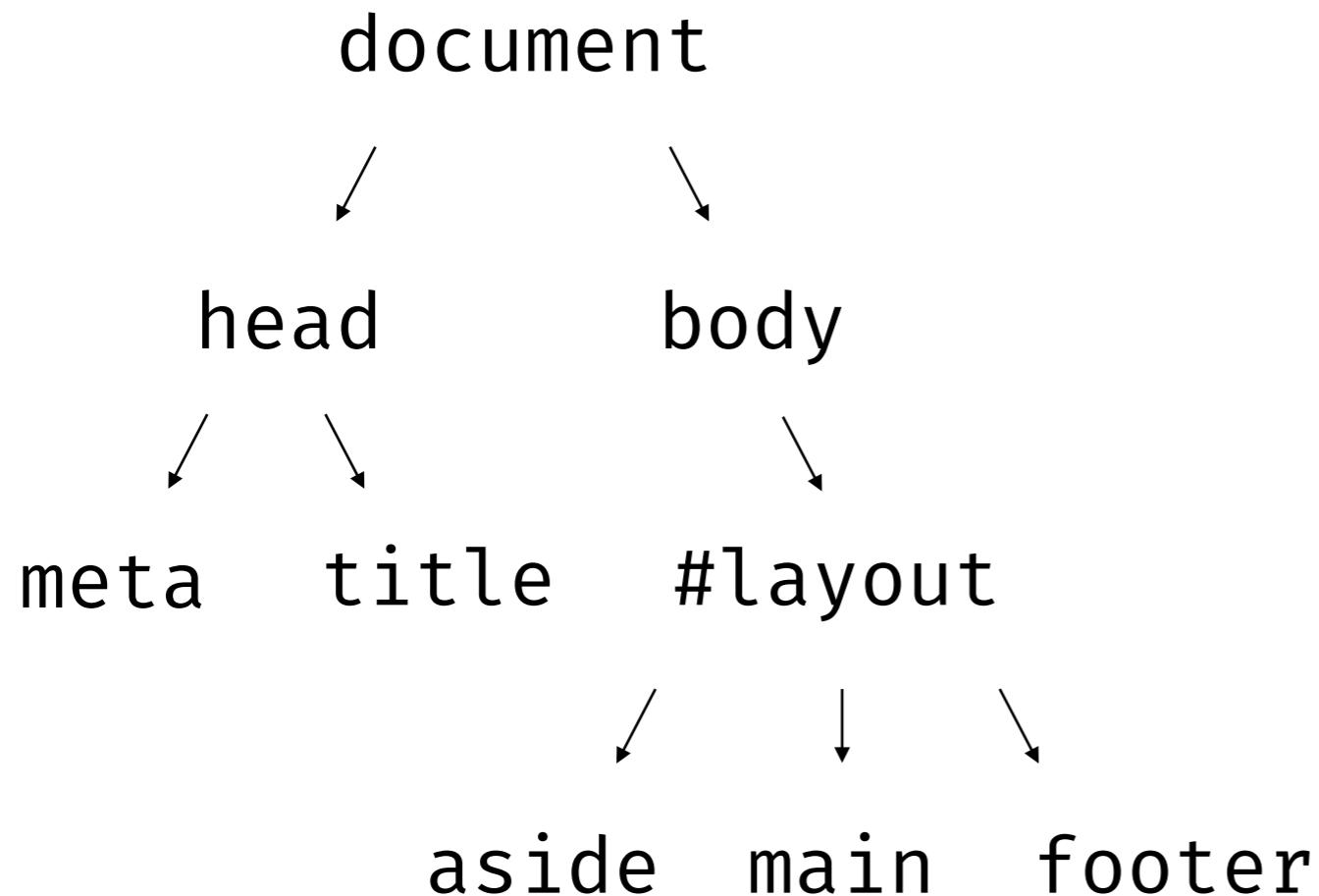


Существует несколько способов перебора элементов дерева:

- DFS (поиск в глубину)
- прямой



Деревья

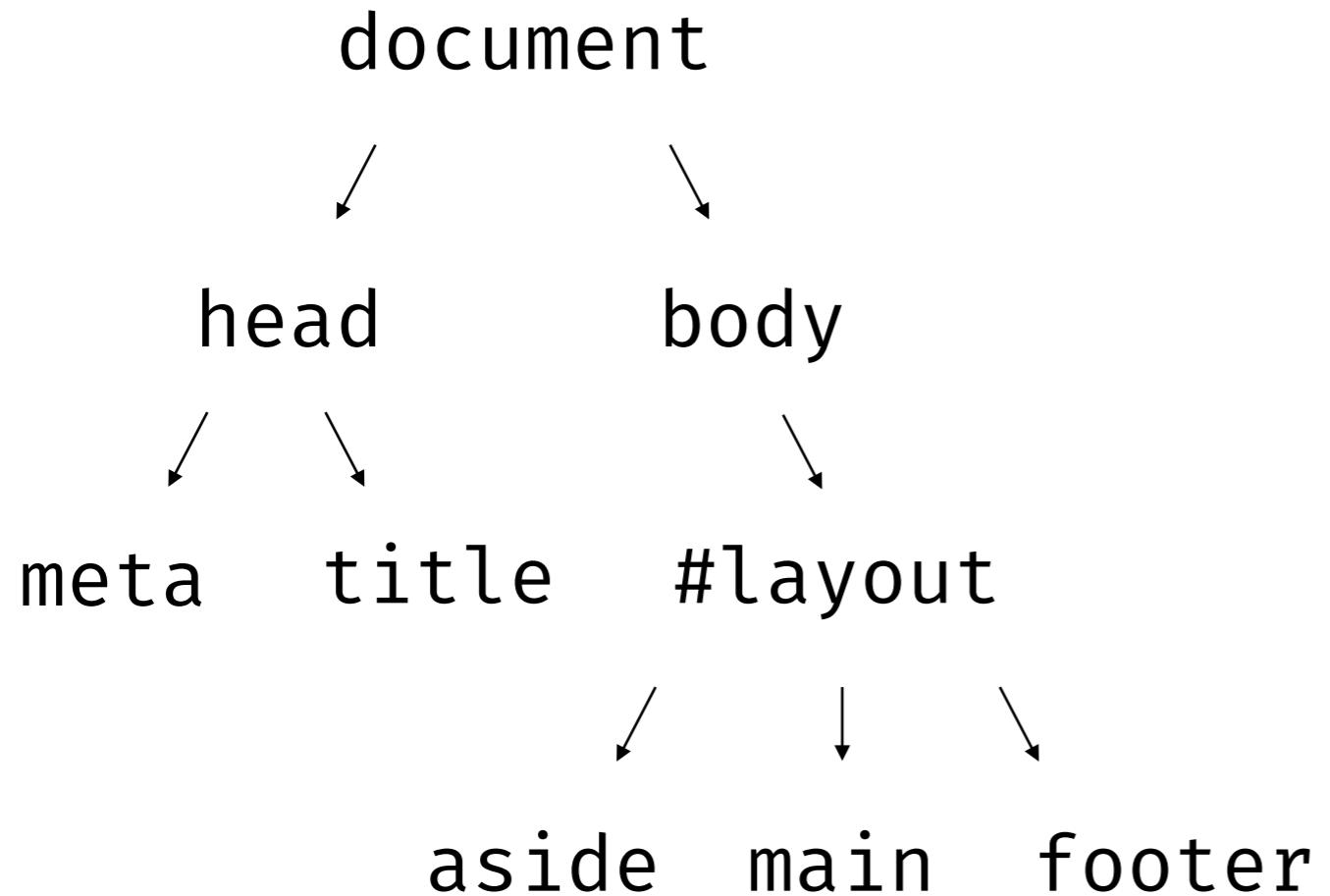


Существует несколько способов перебора элементов дерева:

- DFS (поиск в глубину)
 - прямой
 - обратный



Деревья

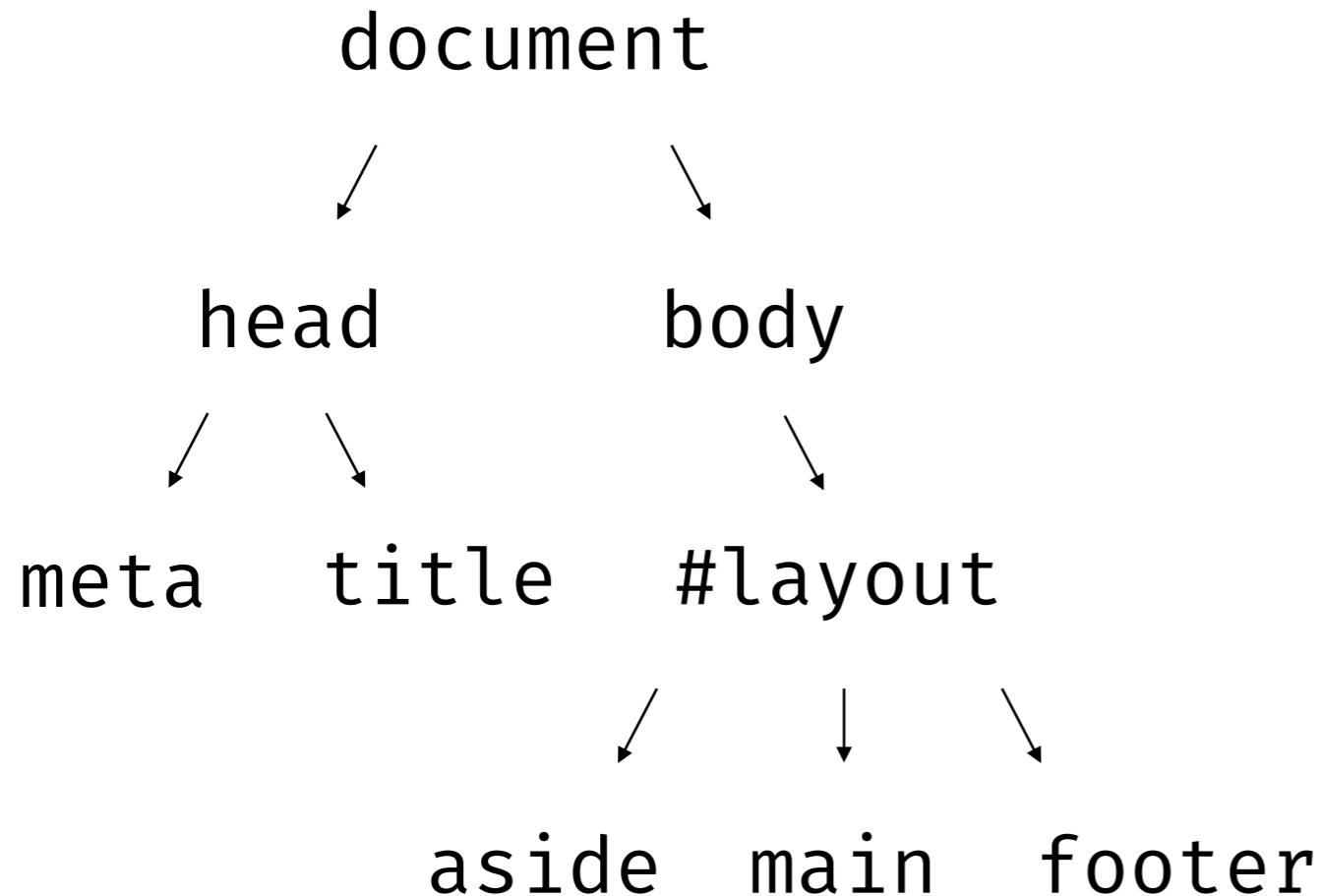


Существует несколько способов перебора элементов дерева:

- DFS (поиск в глубину)
 - прямой
 - обратный
 - симметричный



Деревья



Существует несколько способов перебора элементов дерева:

- DFS (поиск в глубину)
 - прямой
 - обратный
 - симметричный
- BFS (поиск в ширину)



Demo

localhost:8080

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

+ x1 x0.5 x0.75 x1.5 400px 800px 1000px 1500px 1

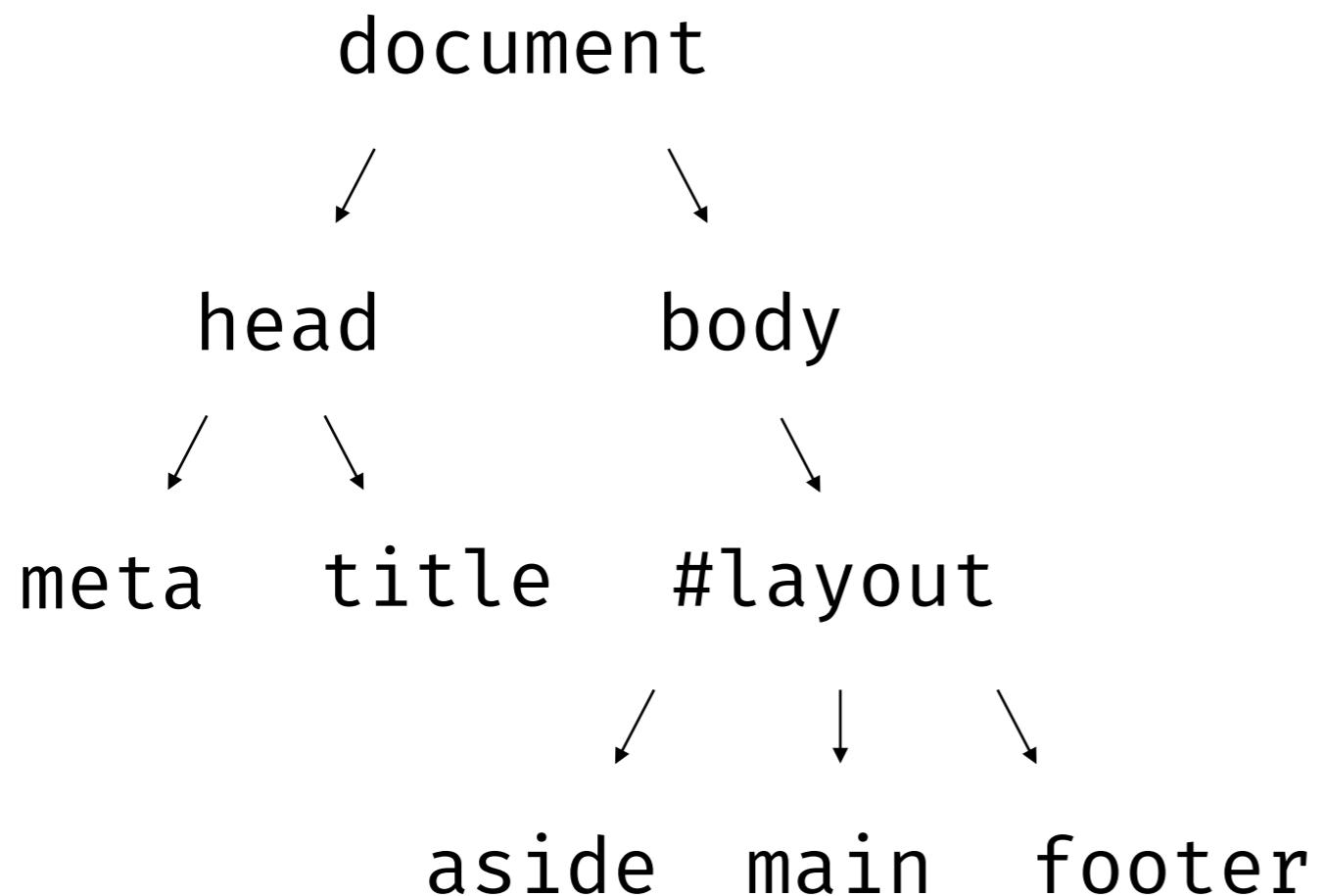
Прямой поиск в глубину

(сербохорв. – *Depth-first search, DFS*) если у узла есть потомки, посещаются сначала они, а только потом соседний узел (всегда идём до самого конца)



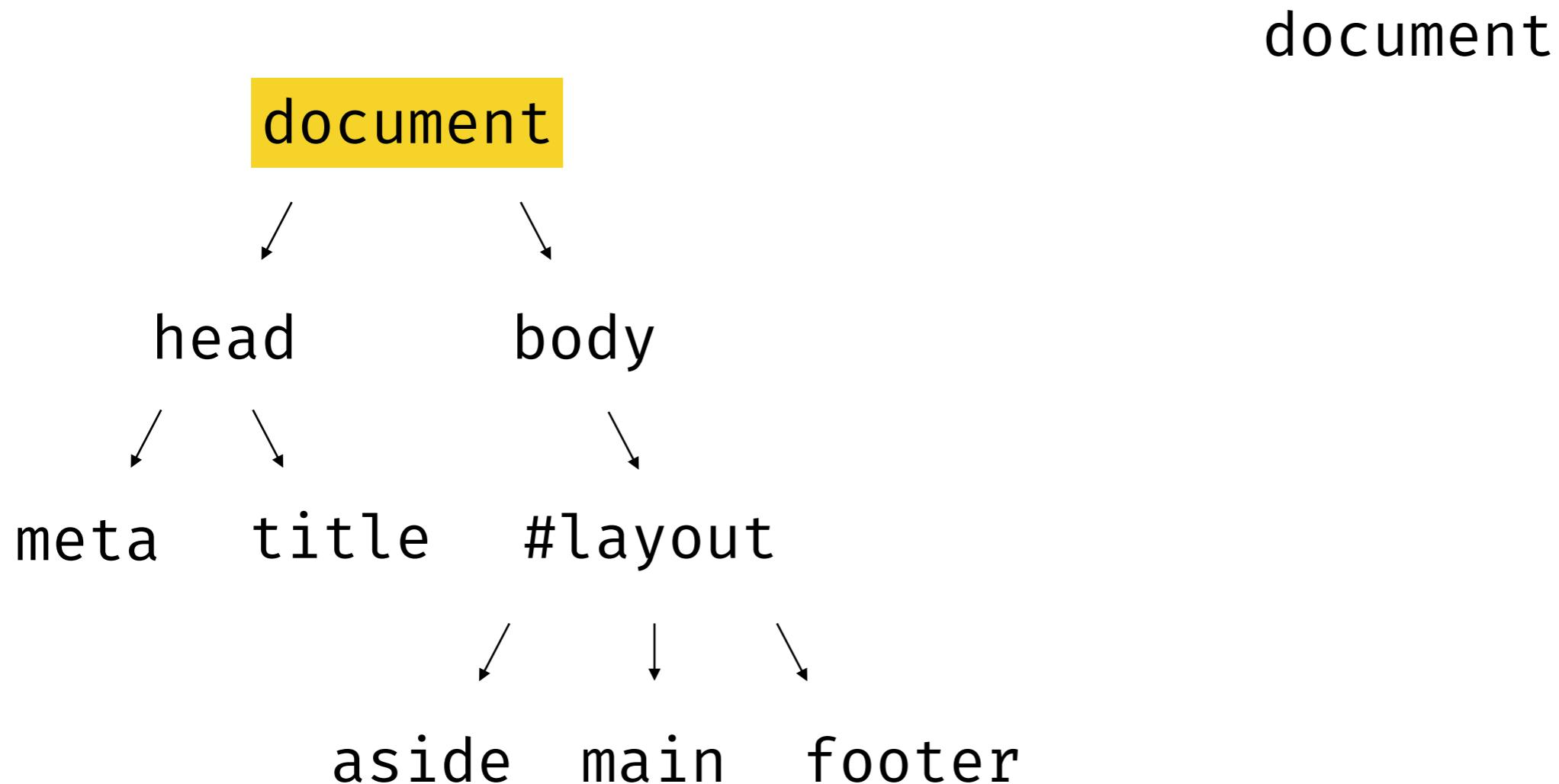
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



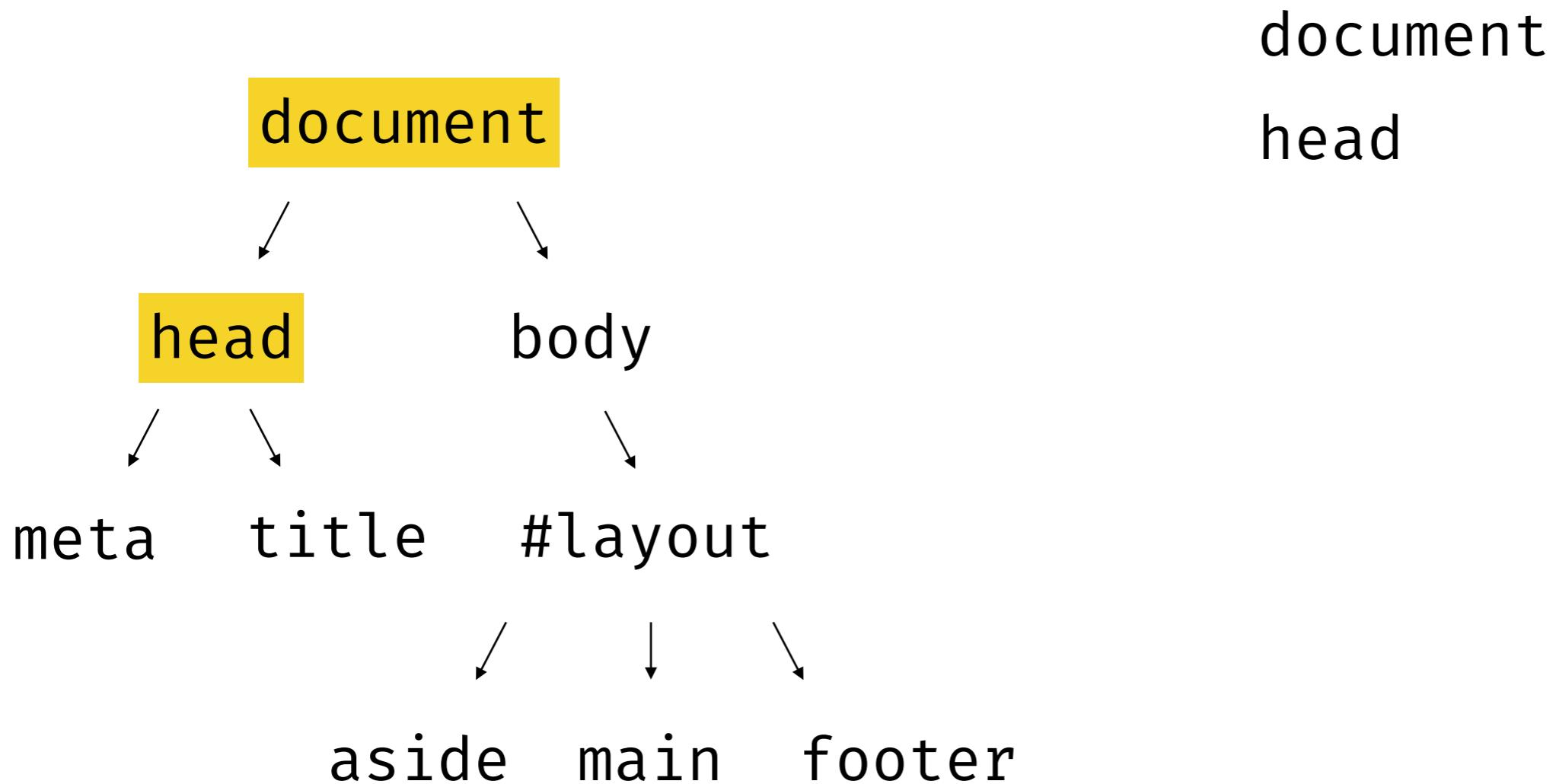
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



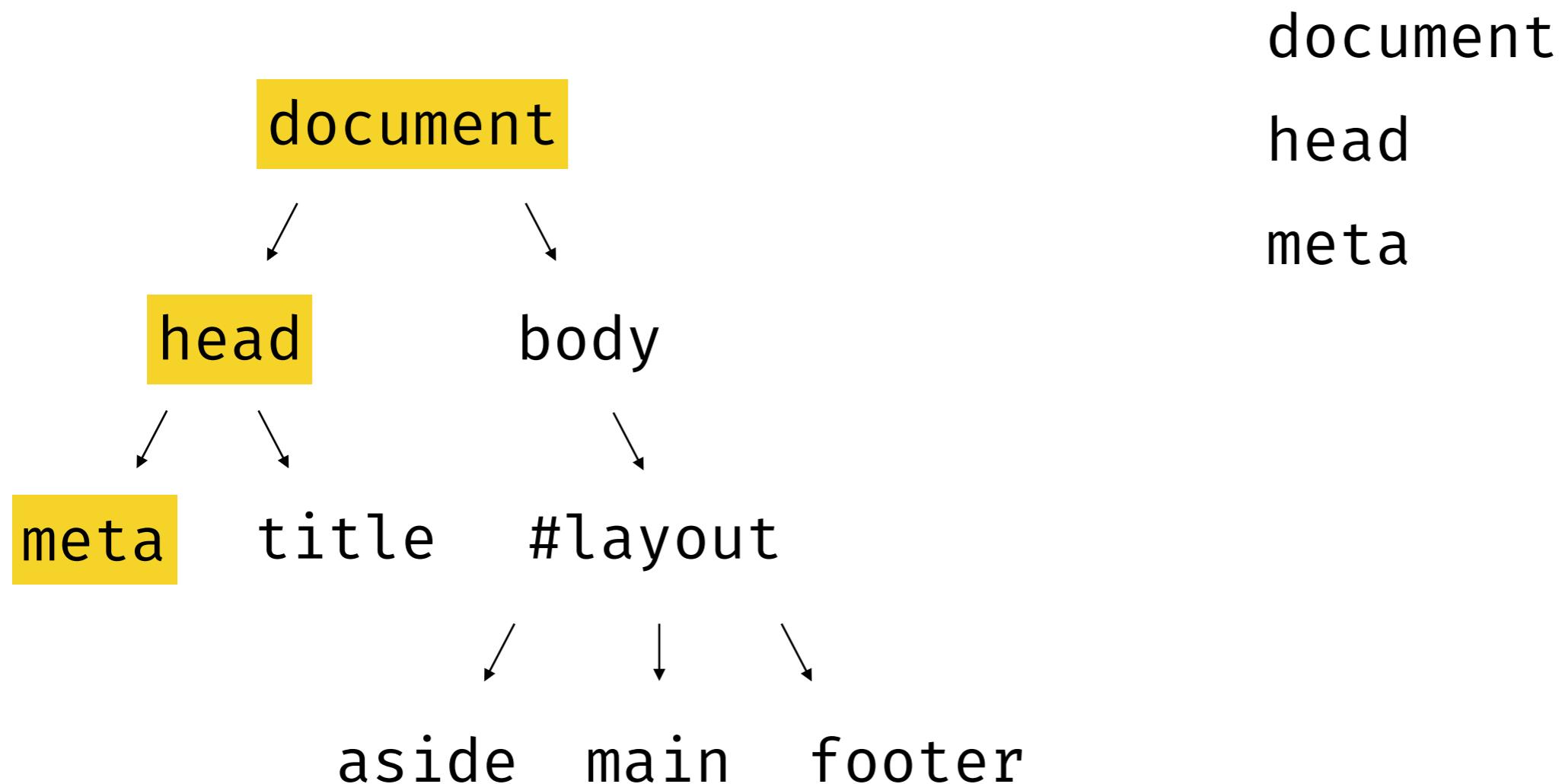
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



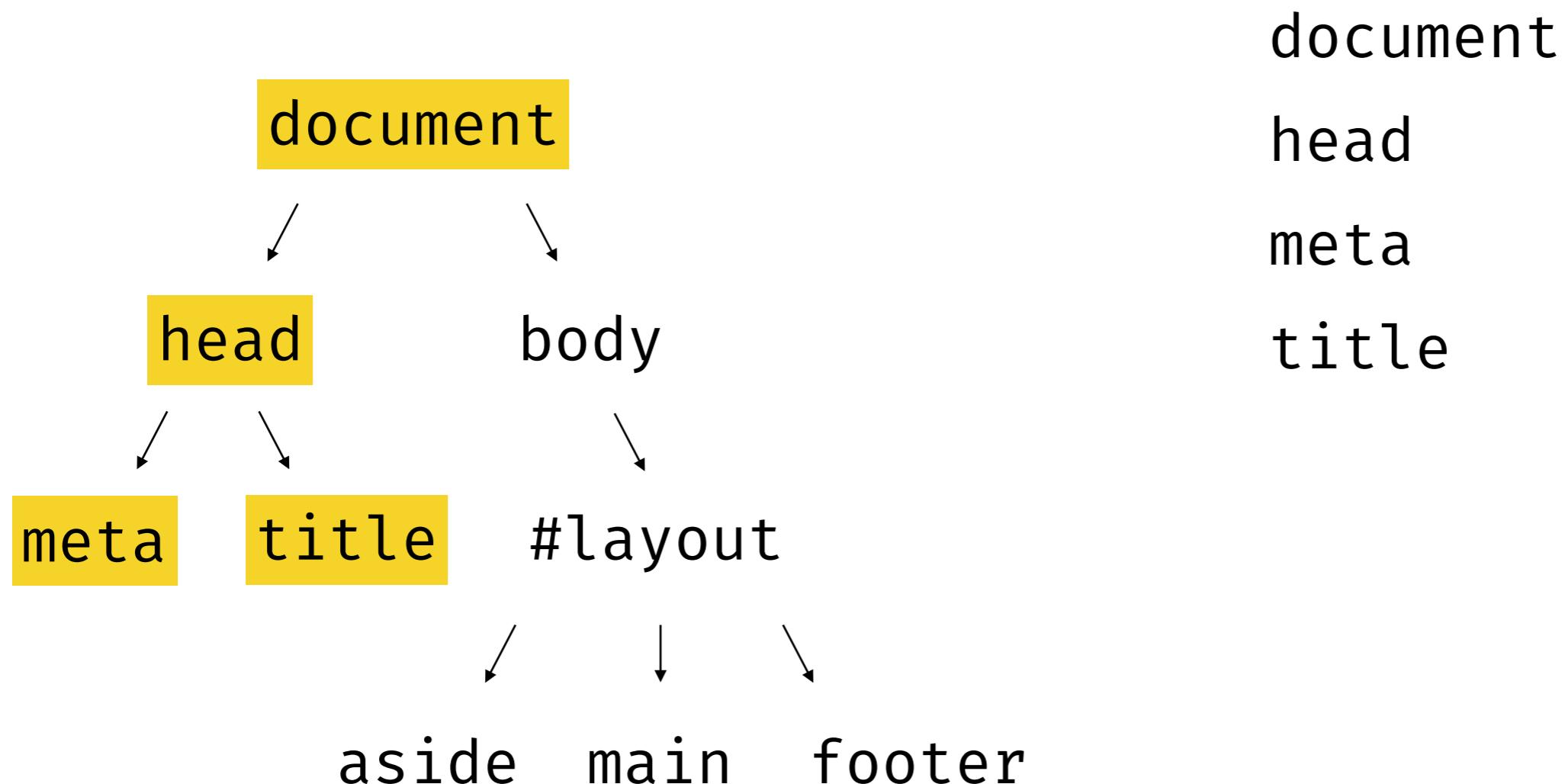
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



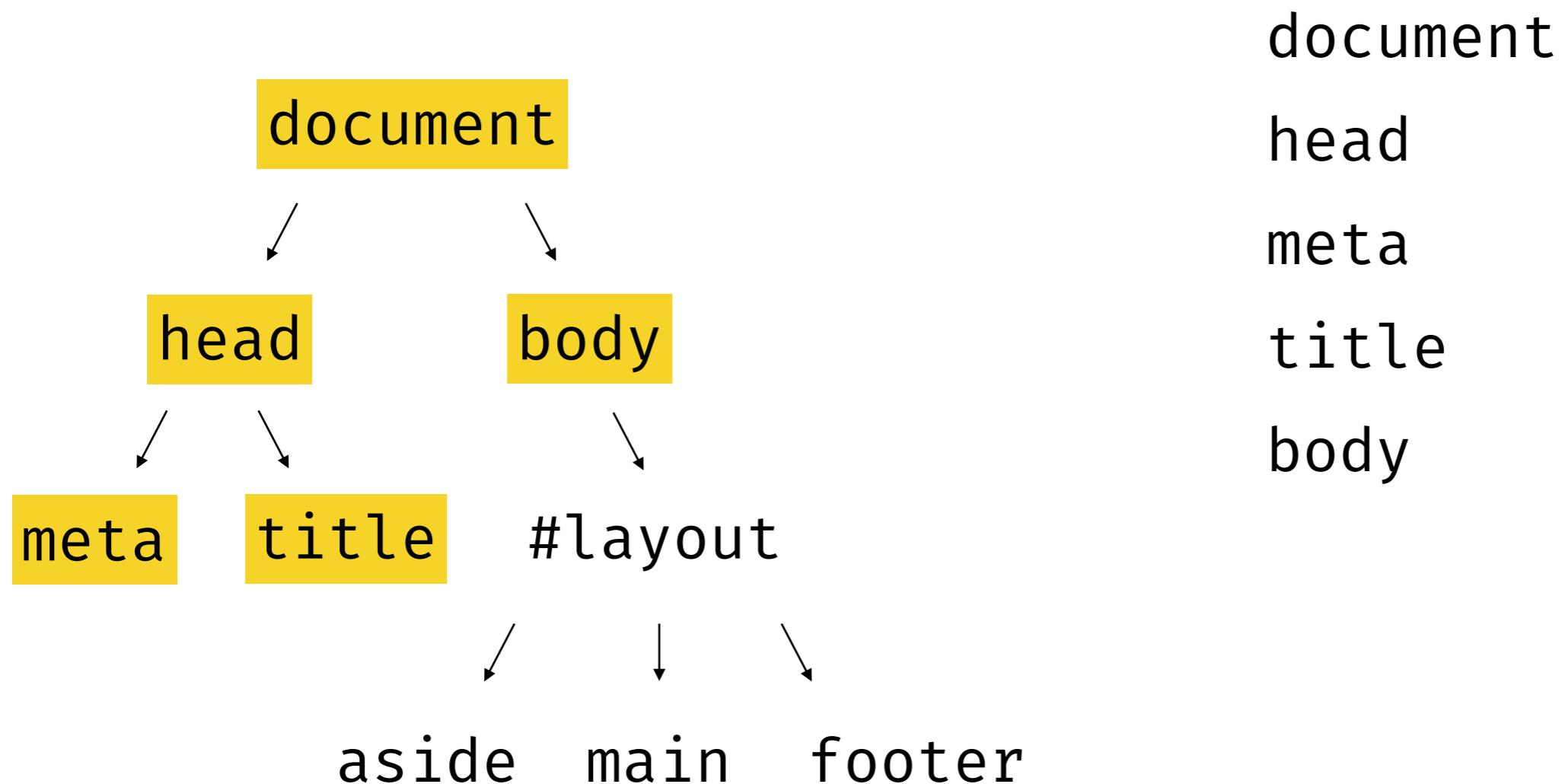
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



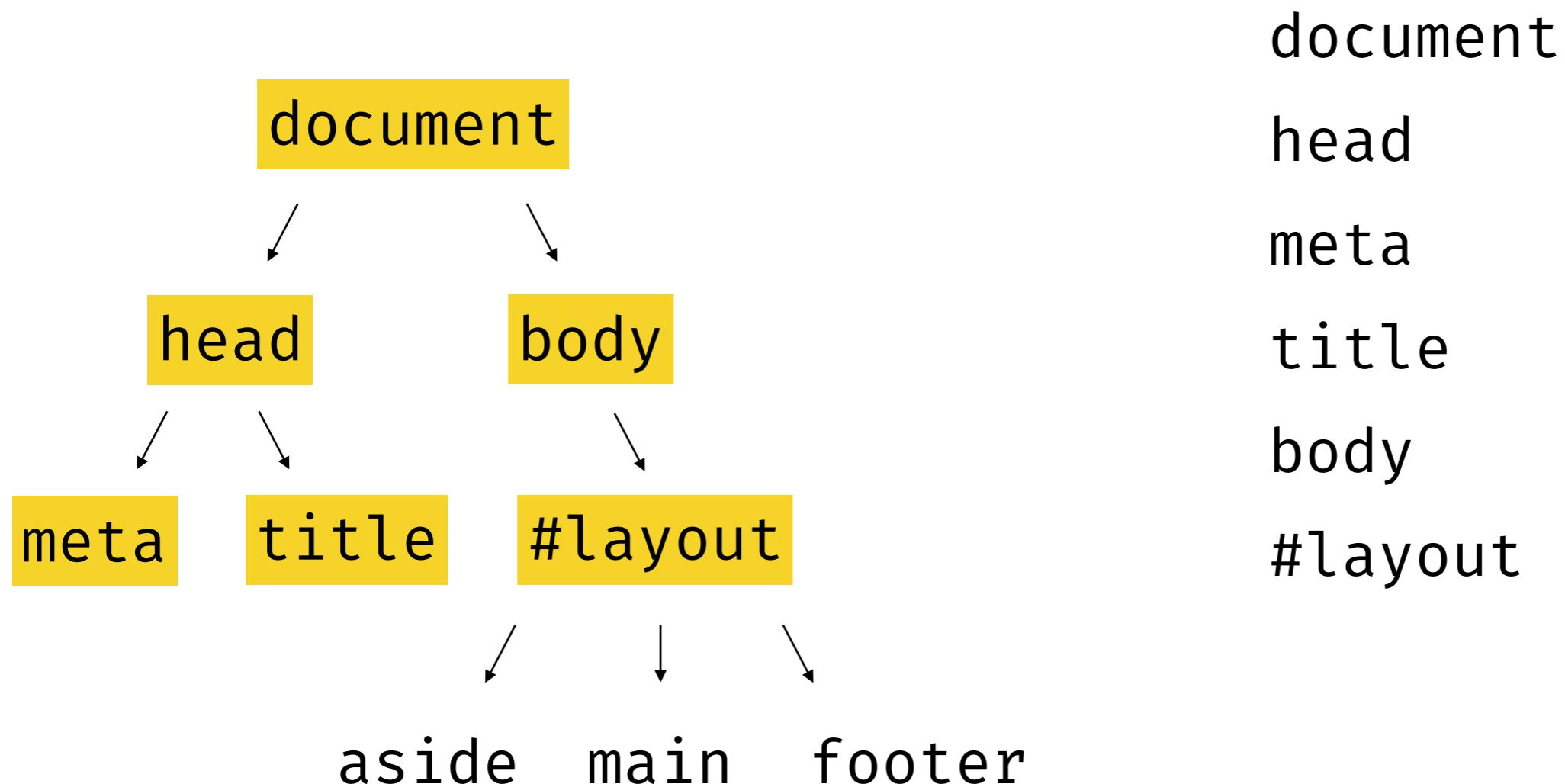
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



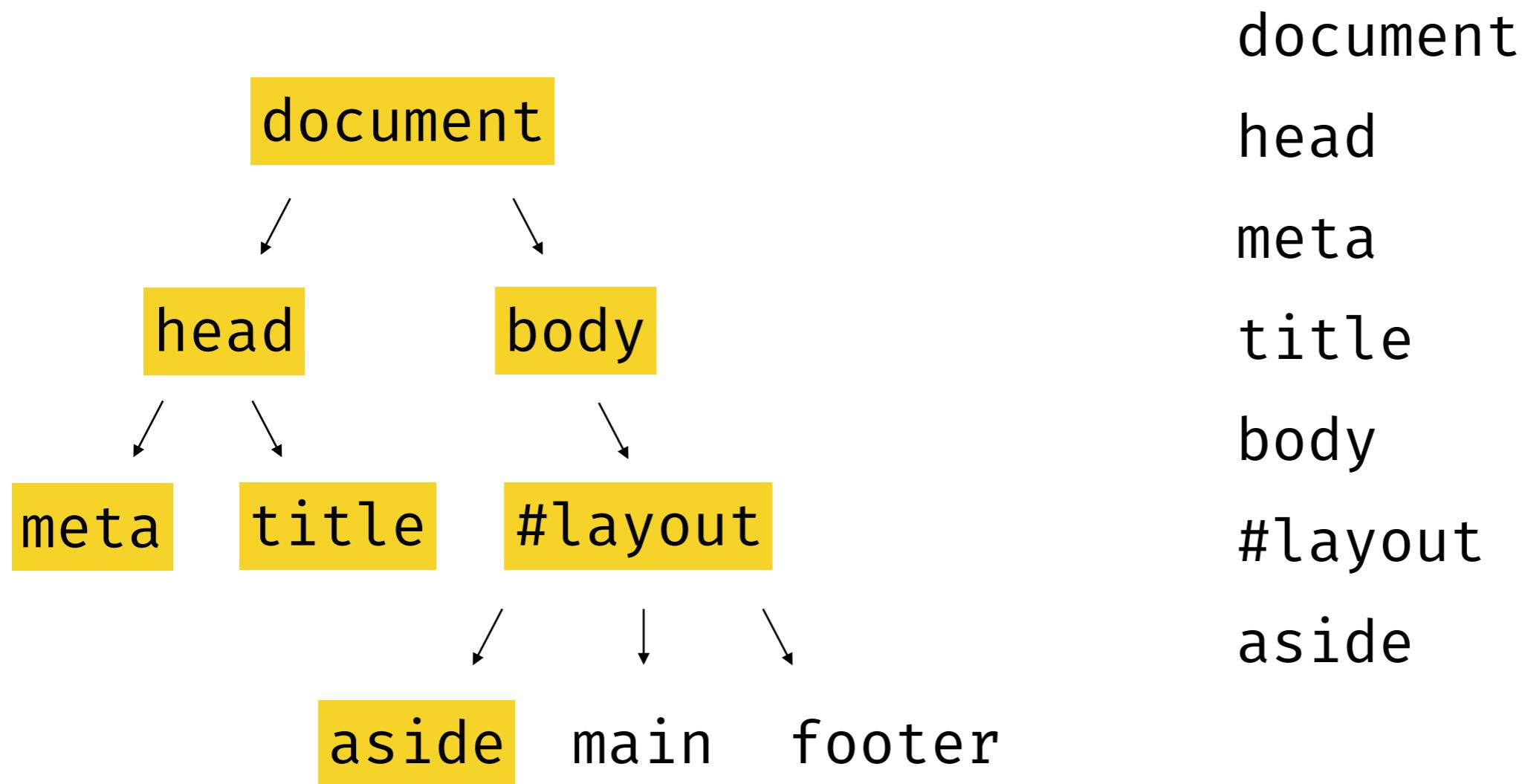
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



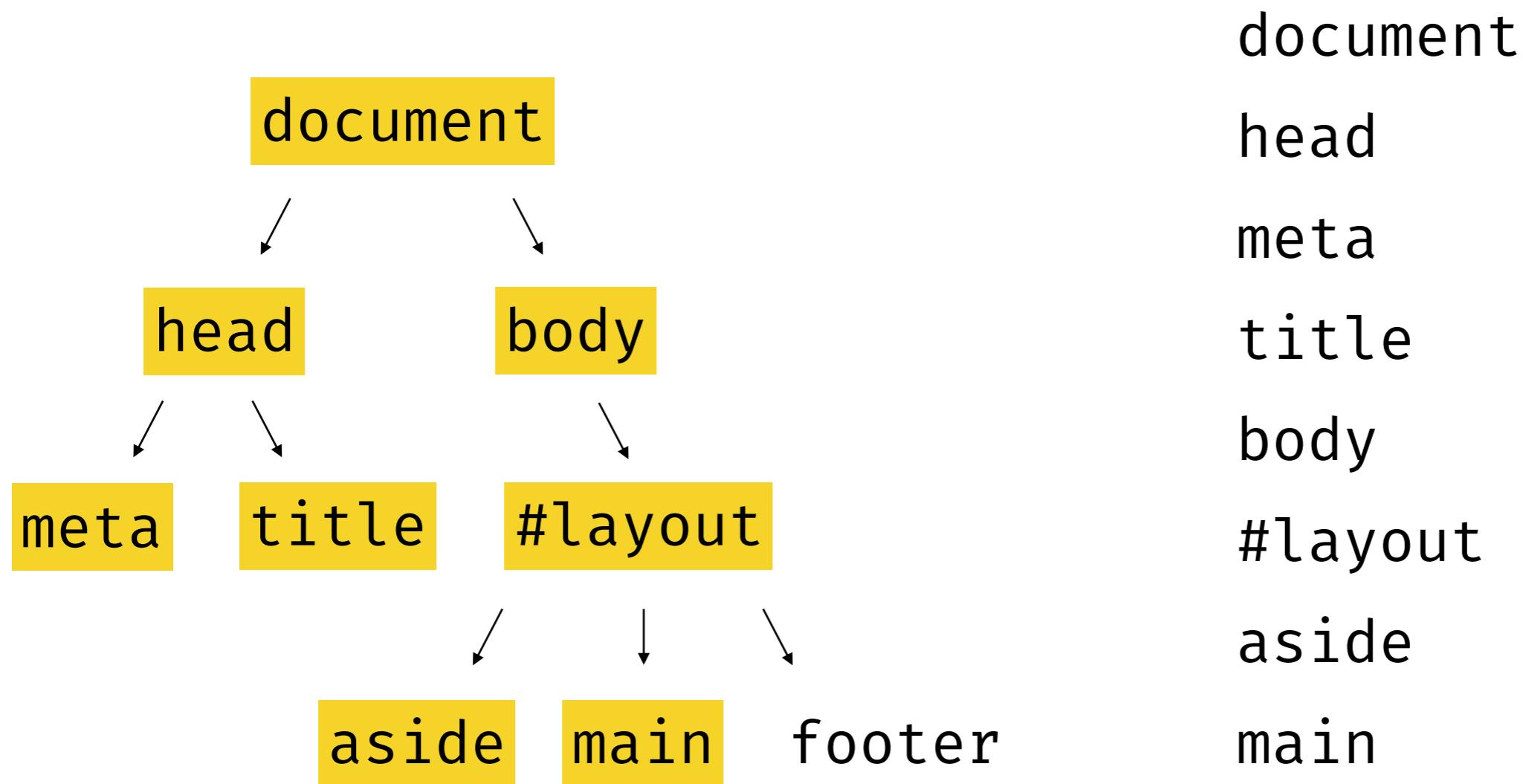
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



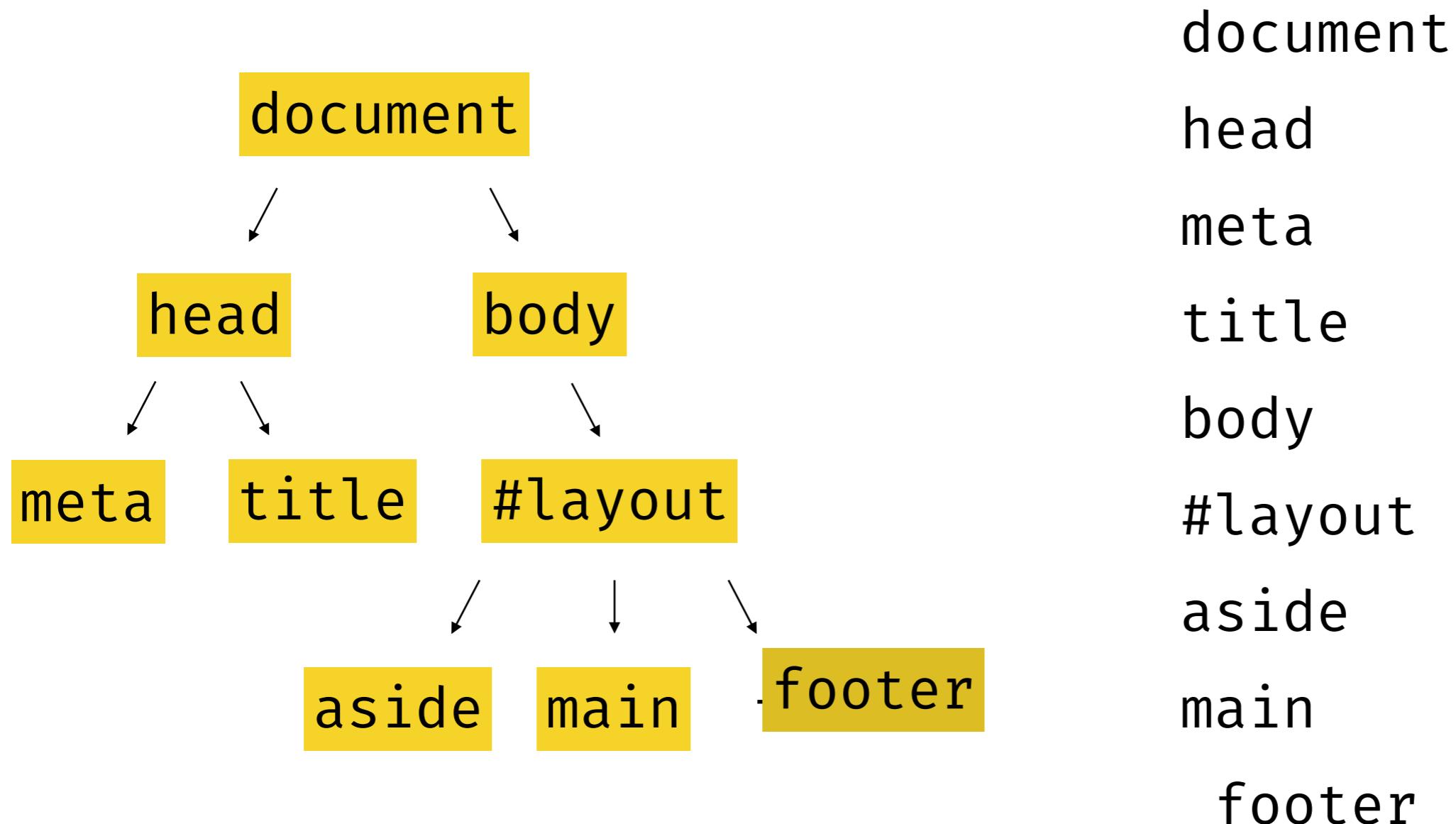
Прямой обход в глубину

Сначала посещается узел, а потом его потомки



Прямой обход в глубину

Сначала посещается узел, а потом его потомки



Demo × Игорь

localhost:8080 ☆ ☑ ⋮

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS LESS +

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript +

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```



The screenshot shows a browser-based code editor interface with the following components:

- Header:** Shows tabs for "Demo" and "Игорь" (Igor).
- Address Bar:** Displays "localhost:8080".
- HTML Editor:** Contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```
- CSS Editor:** Contains the following LESS code:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```
- JavaScript Editor:** Contains the following code:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```
- Toolbar:** Includes buttons for zoom levels (x1, x0.5, x0.75, x1.5) and sizes (400px, 800px, 1000px, 1500px).

The screenshot shows a web-based development interface with three main code editors on the left and a preview area on the right.

HTML:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The preview area on the right displays a single red square element with a bounding box of approximately [180, 180, 220, 220]. Above the preview are several controls: a zoom slider set to "x1", a grid size selector with options "400px", "800px", "1000px", and "1500px", and a "1" button. The top bar includes tabs for "Demo" and "Игорь".

The screenshot shows a web-based development environment with a dark-themed interface. At the top, there's a header bar with three circular icons, a 'Demo' button, and a user name 'Игорь'. Below the header is a browser-style toolbar with back/forward buttons, a refresh icon, a URL field containing 'localhost:8080', a star icon, a React logo, a hexagon icon, and a more options menu.

The main area is divided into three vertical tabs:

- HTML**: Contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```
- CSS**: Contains the following LESS code:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```
- JavaScript**: Contains the following code:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

On the right side of the interface, there's a zoom control with a plus sign and a scale dropdown menu showing options: x1, x0.5, x0.75, x1.5, 400px, 800px, 1000px, and 1500px. A large blue vertical bar is positioned between the HTML and CSS sections. The bottom right corner features a small decorative hexagonal icon.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The screenshot shows a web-based code editor interface with three main sections: HTML, CSS, and JavaScript. The title bar indicates the project is named "Demo" and the user is "Игорь". The browser address bar shows "localhost:8080".

HTML:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The interface includes a zoom control at the top right (x1, x0.5, x0.75, x1.5) and a width selector (400px, 800px, 1000px, 1500px). A "LESS" button is located between the CSS and JavaScript sections.

The screenshot shows a web-based code editor interface with three main sections: HTML, CSS, and JavaScript. The title bar indicates the project is named "Demo" and the user is "Игорь". The browser address bar shows "localhost:8080".

HTML:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The interface includes a zoom control at the top right (x1, x0.5, x0.75, x1.5) and a width selector (400px, 800px, 1000px, 1500px). A "LESS" button is located between the CSS and JavaScript sections.

The screenshot shows a browser-based code editor interface with three tabs: HTML, CSS, and JavaScript. The title bar indicates the window is titled "Demo" and the user is "Игорь". The address bar shows the URL "localhost:8080".

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The interface includes a zoom control at the top right (x1, x0.5, x0.75, x1.5) and a width selector (400px, 800px, 1000px, 1500px). A "LESS" button is visible between the CSS and JavaScript sections.

The screenshot shows a browser-based code editor interface titled "Demo" with a user profile "Игорь". The URL "localhost:8080" is displayed in the address bar. The interface is divided into three main sections: HTML, CSS, and JavaScript.

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The CSS section includes a LESS icon and a plus sign (+) button. The JavaScript section includes a plus sign (+) button.

The screenshot shows a browser-based code editor interface with three tabs: HTML, CSS, and JavaScript. The title bar indicates the window is titled "Demo" and the user is "Игорь". The address bar shows the URL "localhost:8080".

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The interface includes a zoom control at the top right (x1, x0.5, x0.75, x1.5) and a width selector (400px, 800px, 1000px, 1500px). A LESS icon is visible between the CSS and JavaScript panes.

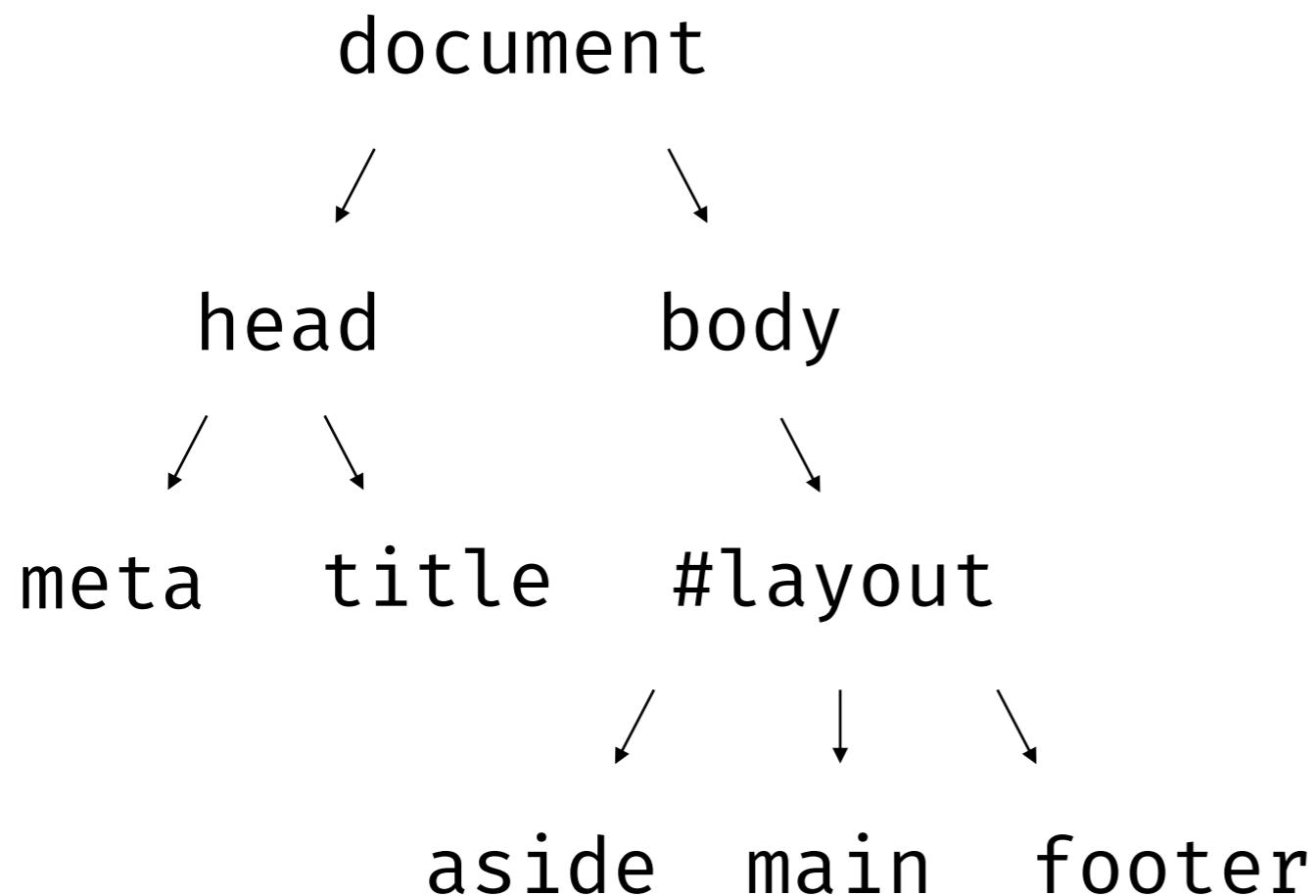
Поиск в ширину

(древнеарамейск. *Breadth-first search, BFS*) узлы посещаются по уровням, будто срезаются слои с торта (смотрим по верхам)



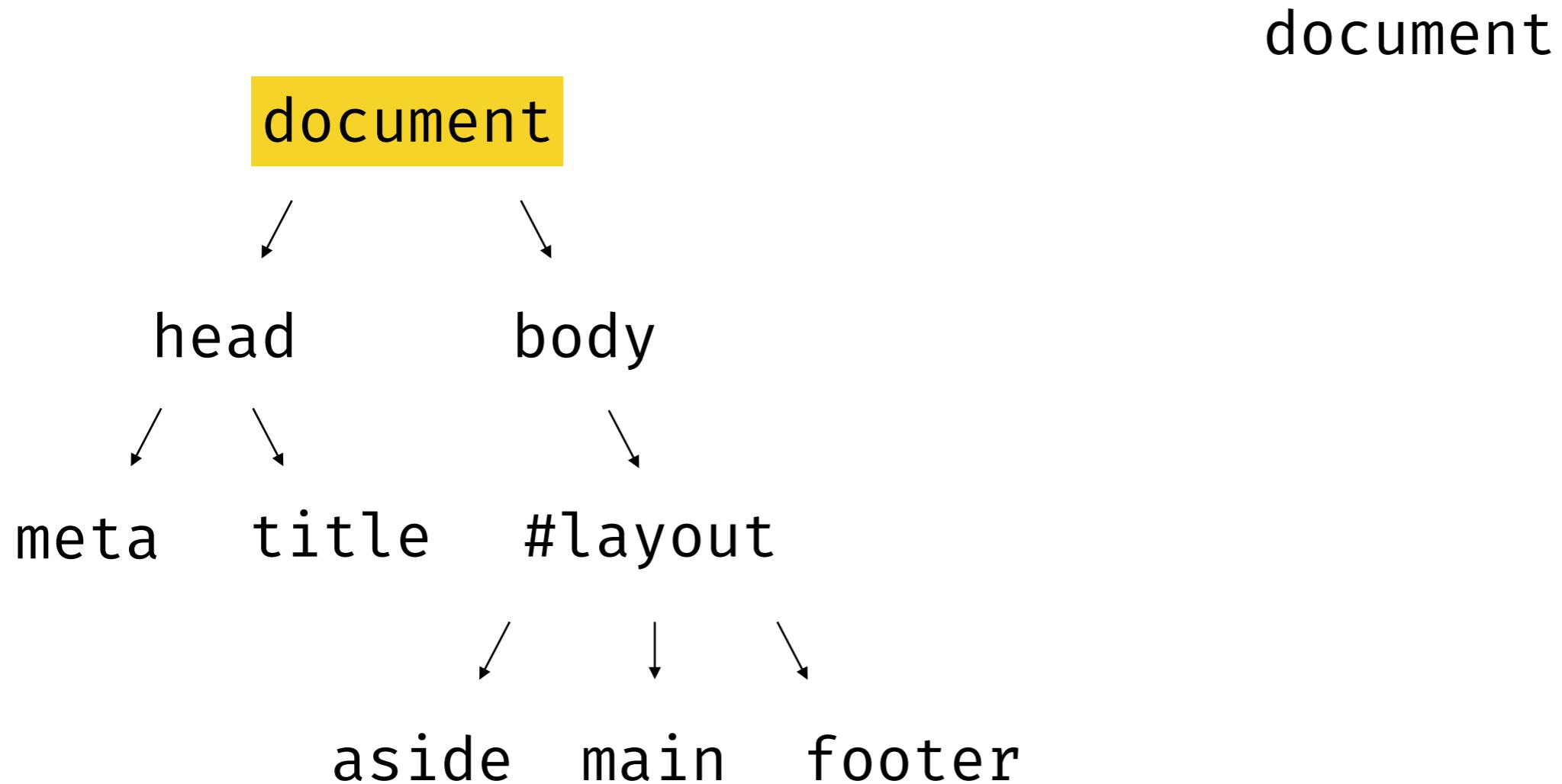
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



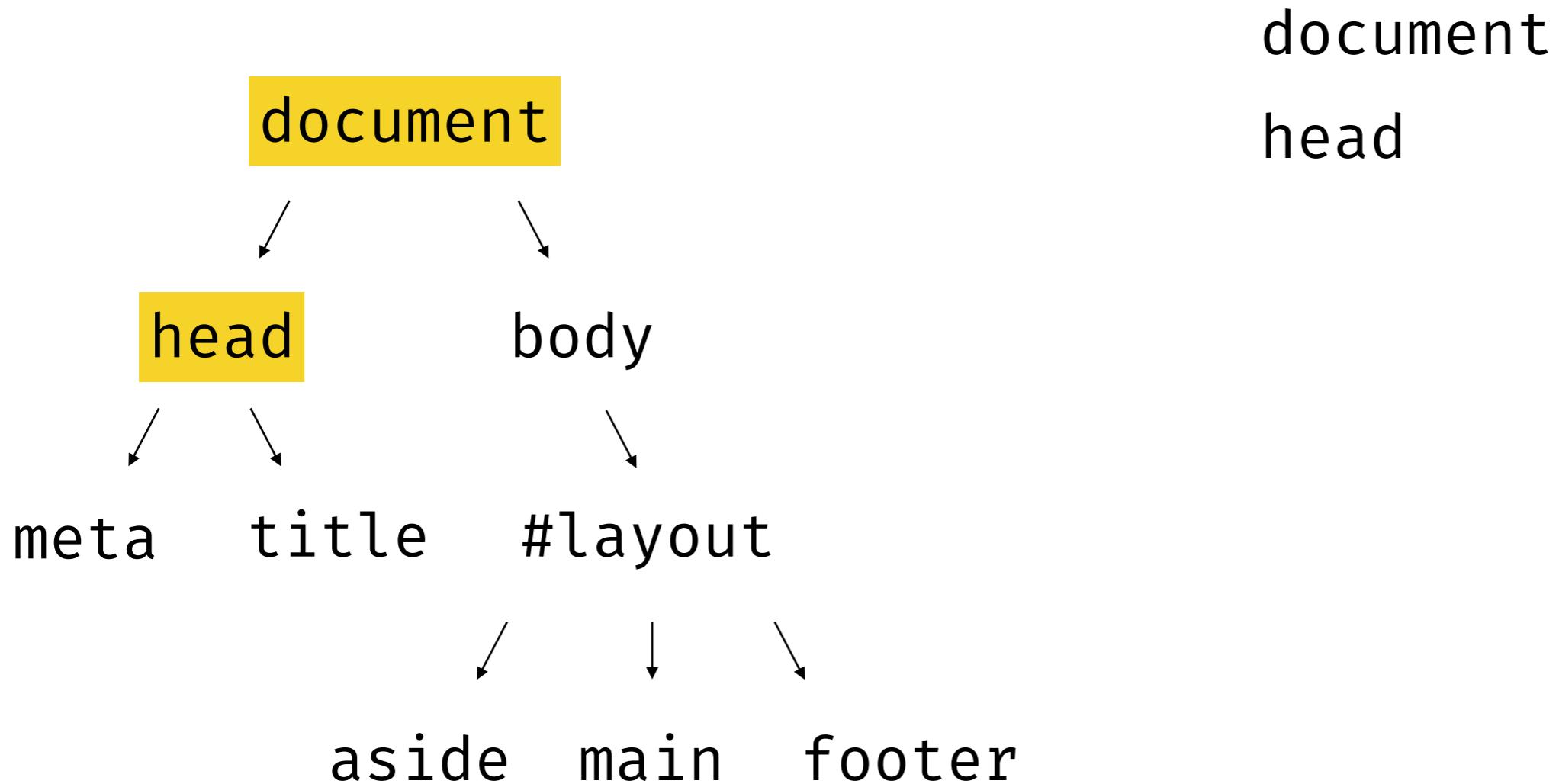
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



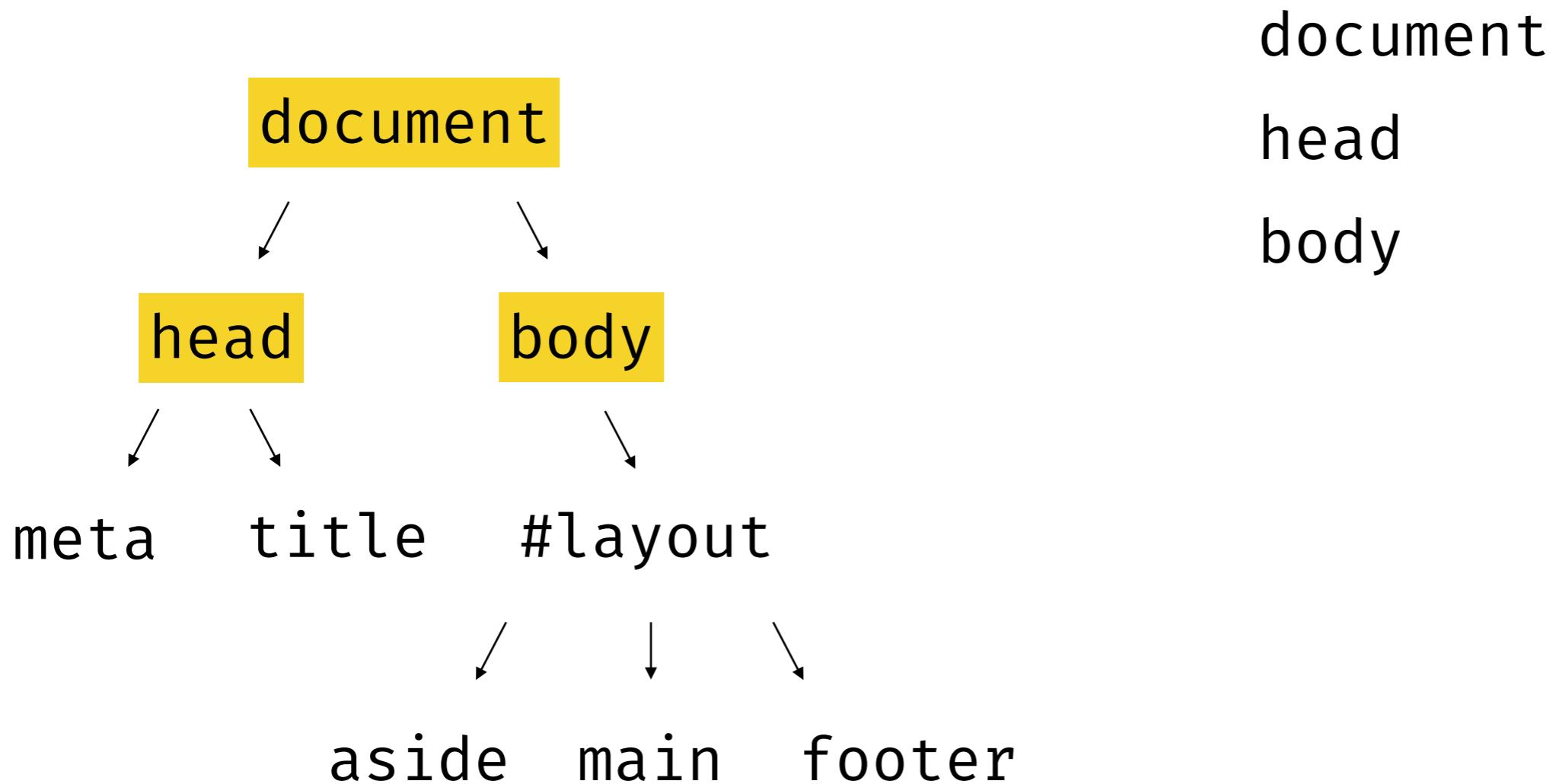
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



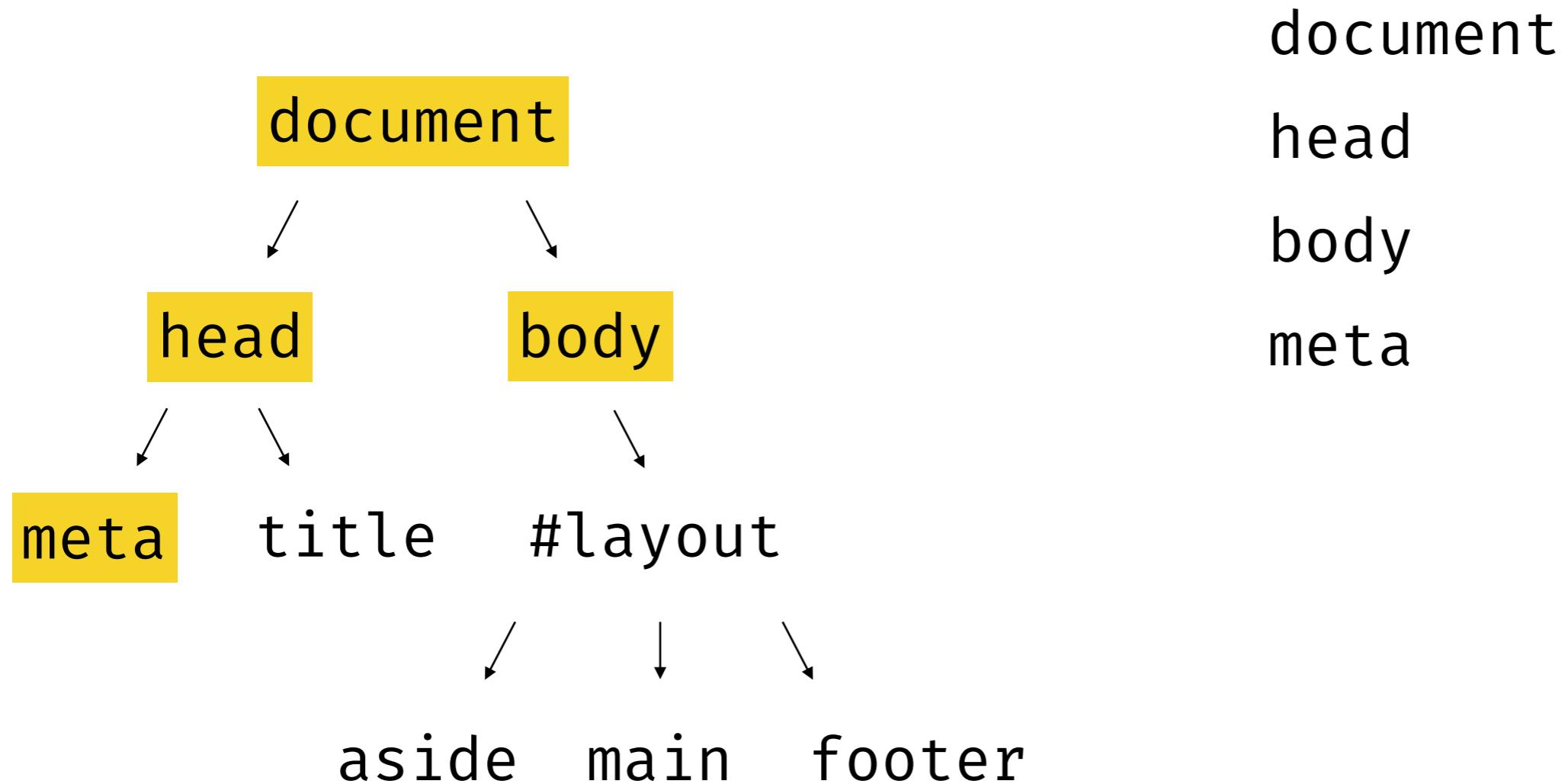
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



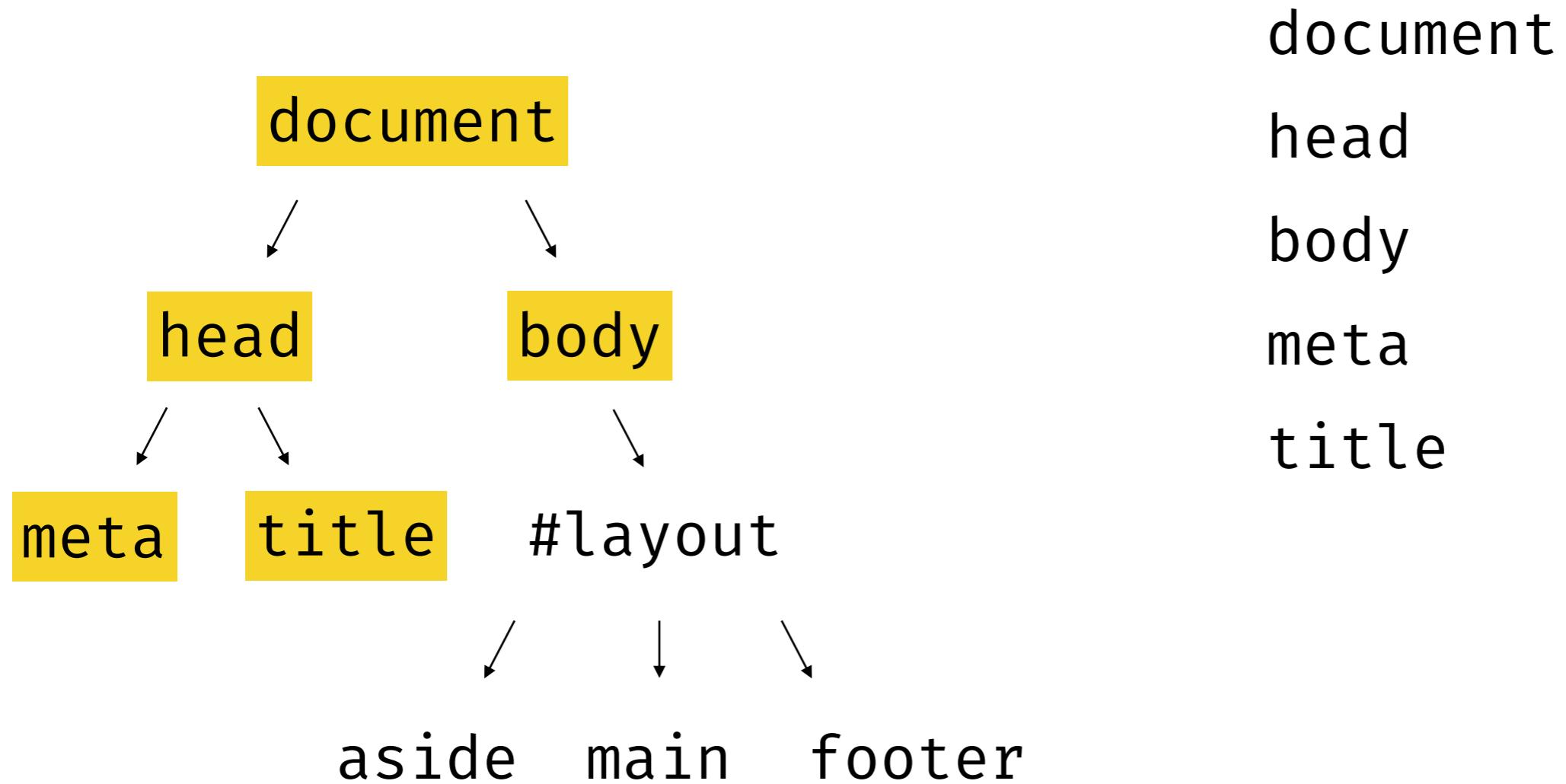
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



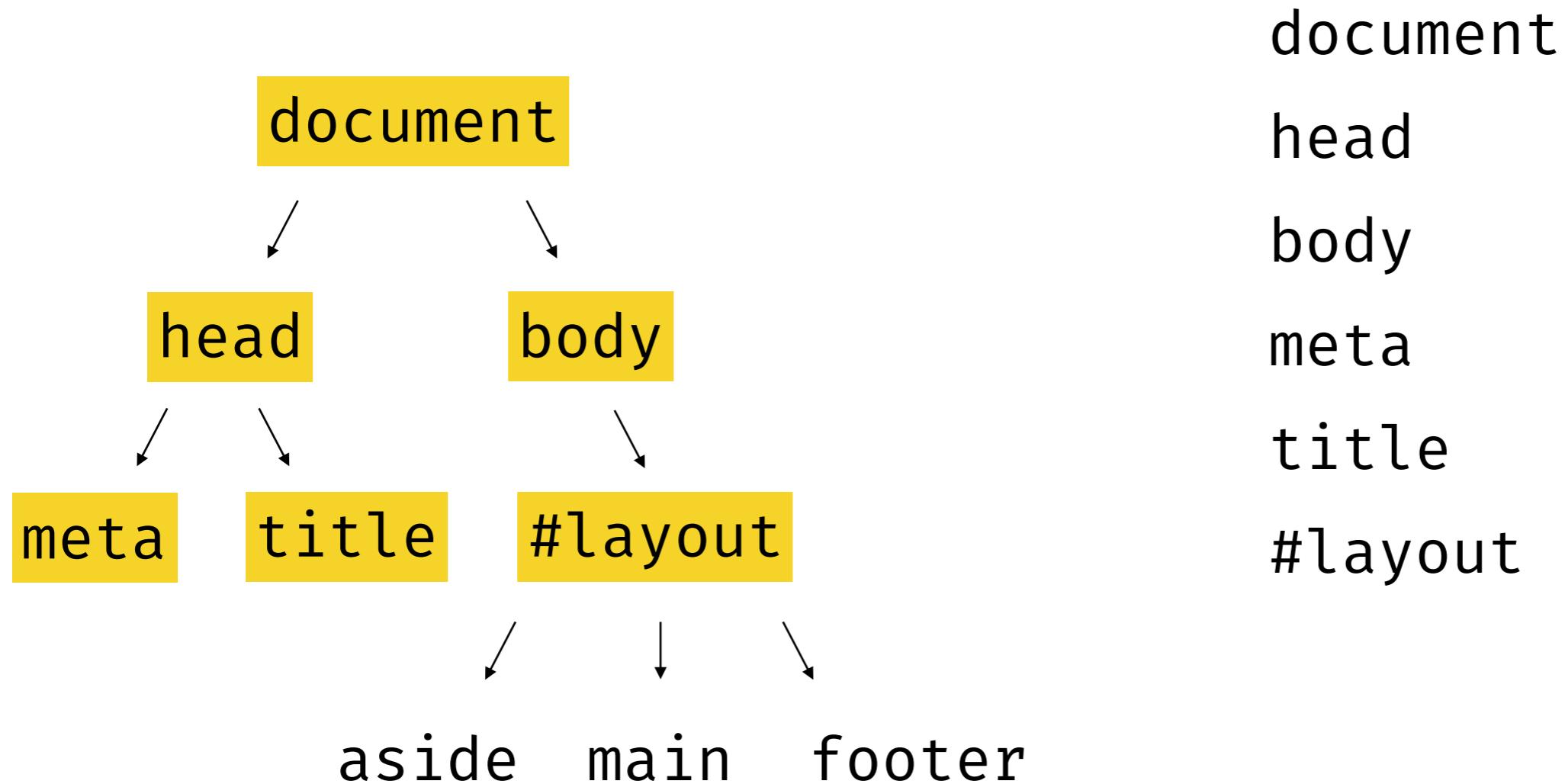
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



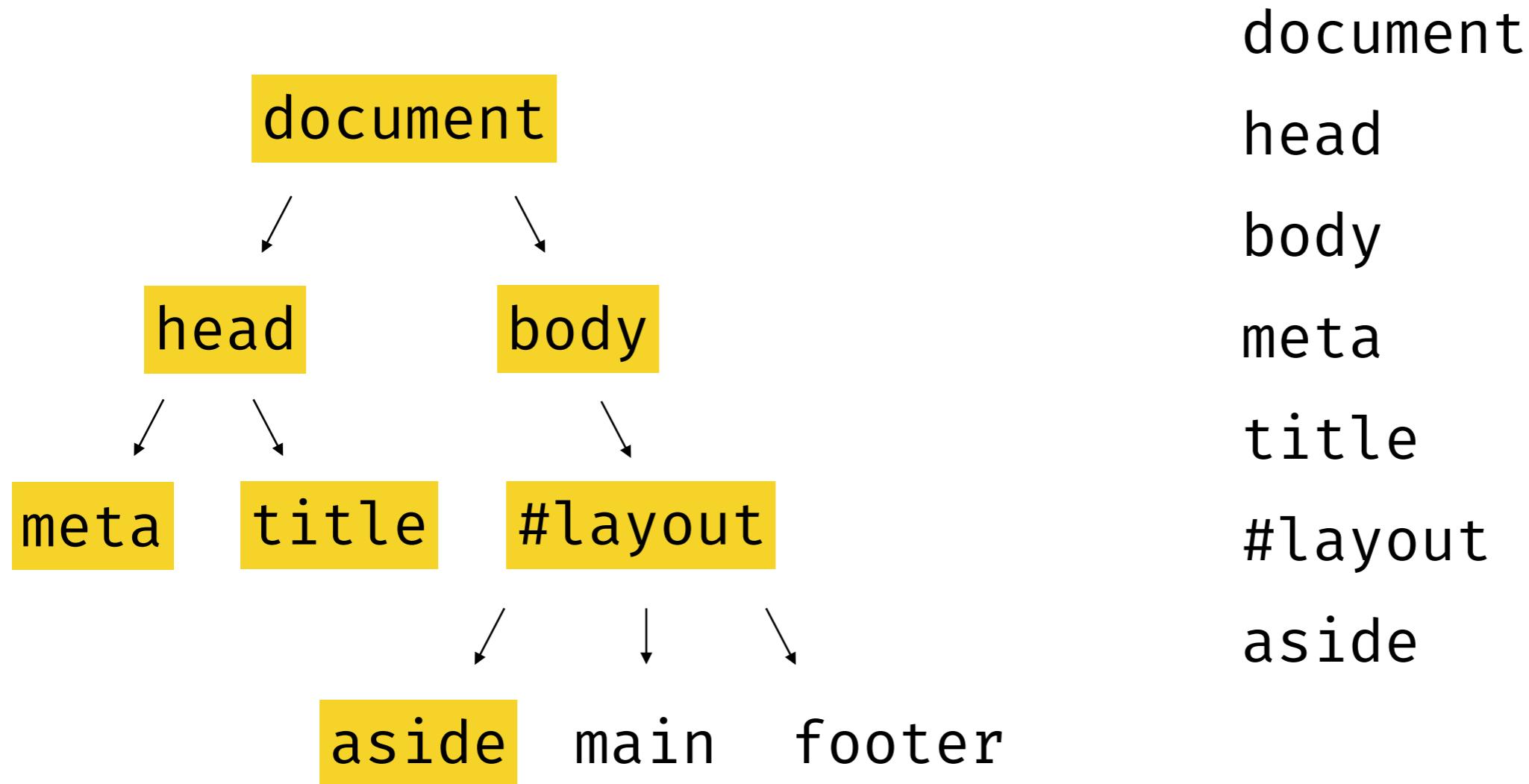
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



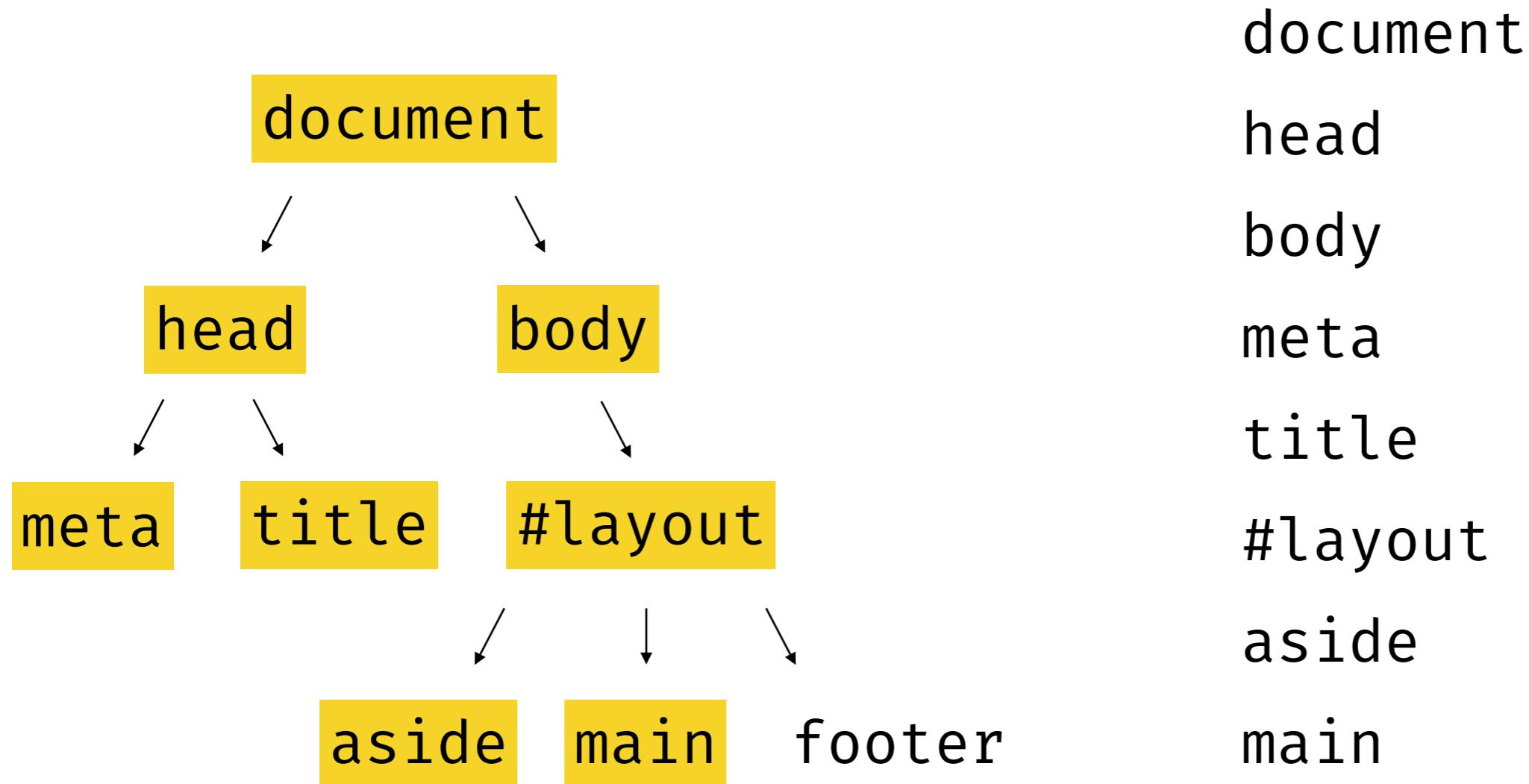
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



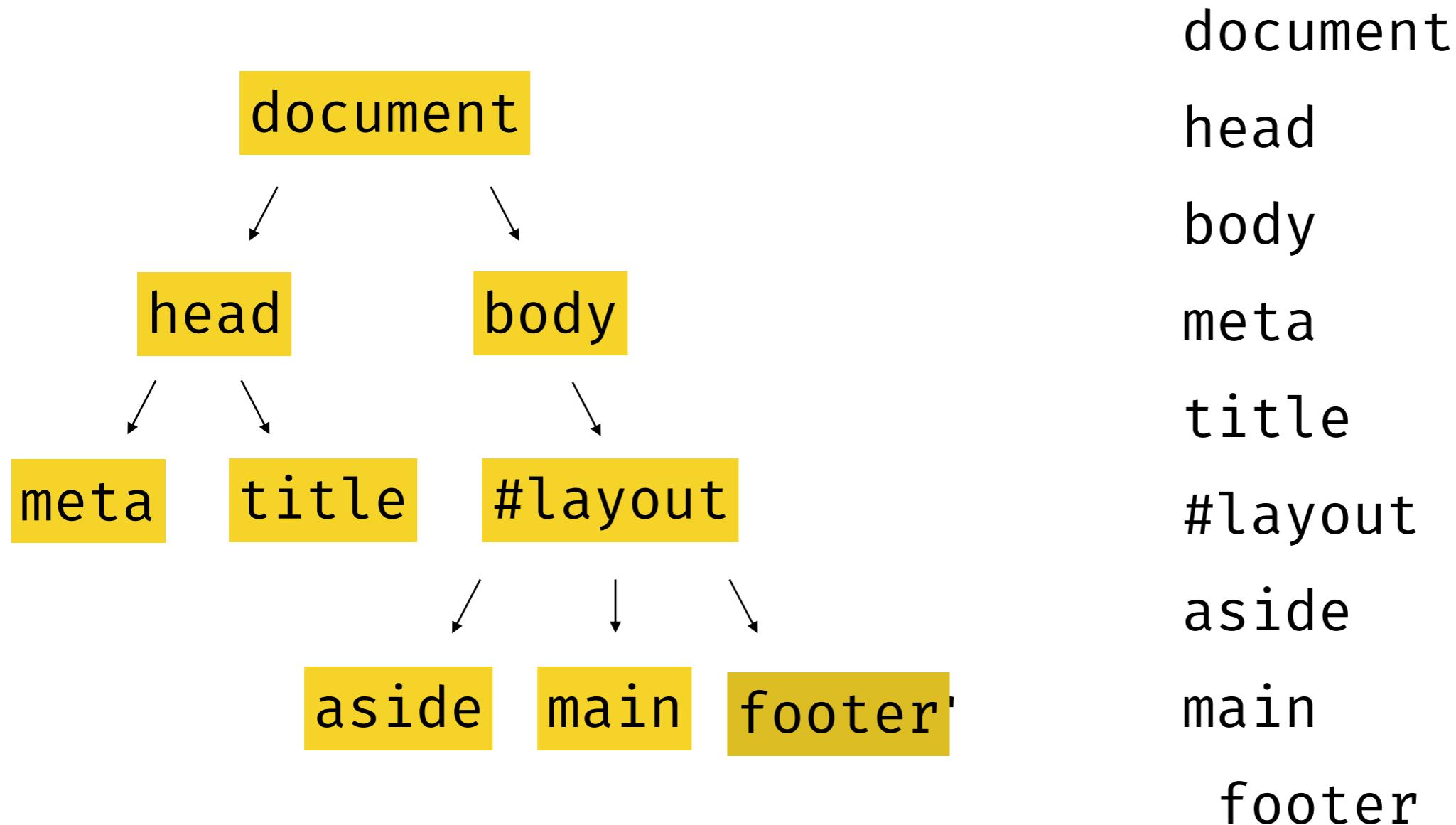
Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



Обход в ширину

Сначала посещаются все узлы на уровне, потом их потомки и так до самого низа



Demo x Игорь

localhost:8080 ☆ ☑ ⋮

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS LESS +

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript +

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```



The screenshot shows a browser-based code editor interface with the following components:

- Header:** Shows tabs for "Demo" and "Игорь" (Igor).
- Address Bar:** Displays "localhost:8080".
- HTML Editor:** Contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```
- CSS Editor:** Contains the following LESS code:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```
- JavaScript Editor:** Contains the following code:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```
- Toolbar:** Includes buttons for zoom levels (x1, x0.5, x0.75, x1.5) and sizes (400px, 800px, 1000px, 1500px).
- Bottom Right Corner:** Features a small hexagonal icon.

The screenshot shows a web-based development interface with three main code editors on the left and a preview area on the right.

HTML:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The preview area on the right displays a single red square element with a bounding box of approximately 400px by 400px. The interface includes a toolbar at the top with various icons and a zoom slider.

The screenshot shows a browser-based code editor interface with the following components:

- Header:** Shows tabs for "Demo" and "localhost:8080".
- Toolbar:** Includes icons for back, forward, refresh, and search, along with a star icon.
- Code Editor:** Contains three tabs: "HTML", "CSS", and "JavaScript".
- HTML Tab:** Displays the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```
- CSS Tab:** Displays the following LESS code:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```
- JavaScript Tab:** Displays the following code:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```
- Layout Tools:** A horizontal toolbar at the top right includes a "+" button, scale factors (x1, x0.5, x0.75, x1.5), and width options (400px, 800px, 1000px, 1500px).

The screenshot shows a browser-based code editor interface with the following components:

- Header:** Shows tabs for "Demo" and "localhost:8080".
- Toolbar:** Includes icons for back, forward, search, and other navigation.
- HTML Editor:** Contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```
- CSS Editor:** Contains the following LESS code:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```
- JavaScript Editor:** Contains the following code:

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```
- Layout Tools:** A floating toolbar with a plus sign (+) and a scale factor dropdown set to "x1". Below it are buttons for "400px", "800px", "1000px", and "1500px".
- Bottom Right Corner:** A small black hexagonal icon with a white geometric pattern.

The screenshot shows a web-based code editor interface with three tabs: HTML, CSS, and JavaScript. The title bar indicates the window is titled "Demo" and is owned by "Игорь". The address bar shows the URL "localhost:8080".

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The screenshot shows a web-based code editor interface with three tabs: HTML, CSS, and JavaScript. The title bar indicates the project is named "Demo" and the user is "Игорь". The browser address bar shows "localhost:8080".

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The interface includes a toolbar at the top with zoom controls (x1, x0.5, x0.75, x1.5) and size options (400px, 800px, 1000px, 1500px). A "LESS" button is located between the CSS and JavaScript tabs.

The screenshot shows a web-based code editor interface with three tabs: HTML, CSS, and JavaScript. The title bar indicates the project is named "Demo" and the user is "Игорь". The browser address bar shows "localhost:8080".

HTML

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```

CSS

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```

JavaScript

```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

The interface includes a zoom control at the top right (x1, x0.5, x0.75, x1.5) and a width selector (400px, 800px, 1000px, 1500px). A "LESS" button is visible between the CSS and JavaScript sections.

The screenshot shows a browser window titled "Demo" with the URL "localhost:8080". The window has a dark blue header bar with three circular icons on the left, a user icon on the right, and the name "Игорь" (Igor) in the top right corner. Below the header is a toolbar with icons for back, forward, search, and other browser functions. The main content area is divided into three vertical sections:

- HTML**: Contains the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <div id="layout"></div>
8 </body>
9 </html>
```
- CSS**: Contains the following LESS code:

```
1 @color: white;
2 @height: 100px;
3 @min-height: calc(@height / 2);
```
- JavaScript**: Contains the following code:

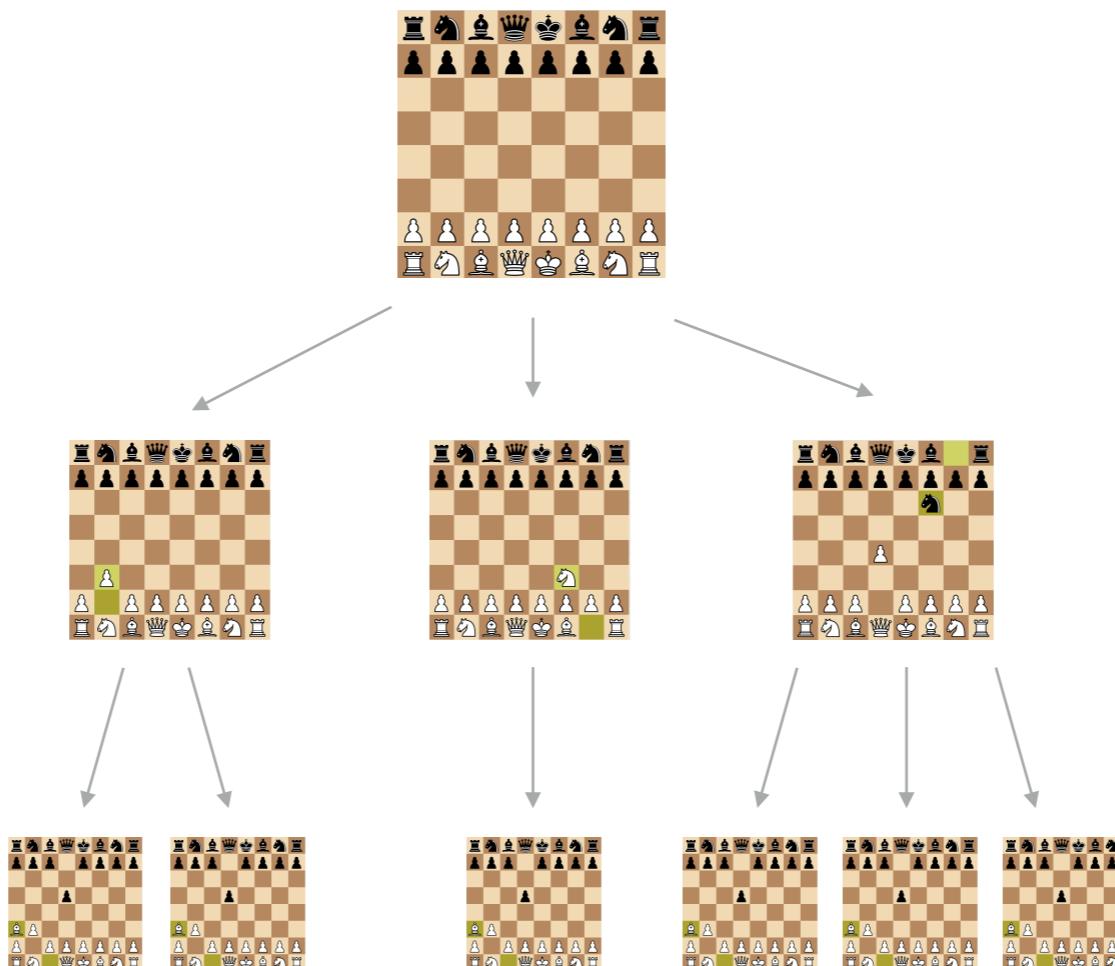
```
1 body.onload = function() {
2   console.log('jopa');
3 }
```

At the top of the code editor, there is a zoom control bar with a plus sign (+), a scale factor "x1" (which is highlighted in blue), and other options like "x0.5", "x0.75", "x1.5", and "x2.0". To the right of the zoom controls are buttons for "400px", "800px", "1000px", and "1500px". A small number "1" is also visible at the far right end of the zoom bar.

A small "LESS" icon is located between the CSS and JavaScript sections. The bottom right corner of the window features a black hexagonal logo with a white geometric pattern.

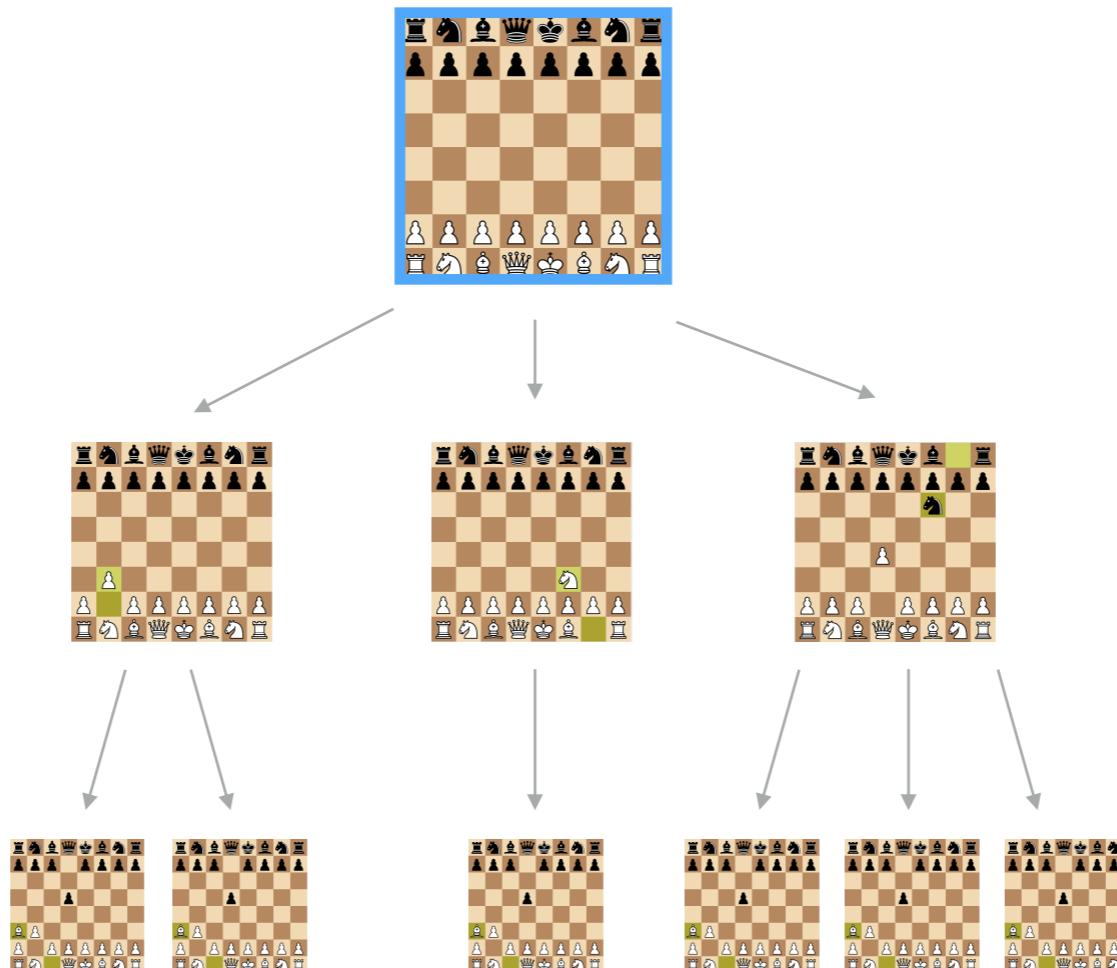
«Лёгкий компьютер» в играх

Обходит дерево сверху вниз и выбирает варианты, в которых не проиграет в ближайшие несколько шагов (*BFS*)



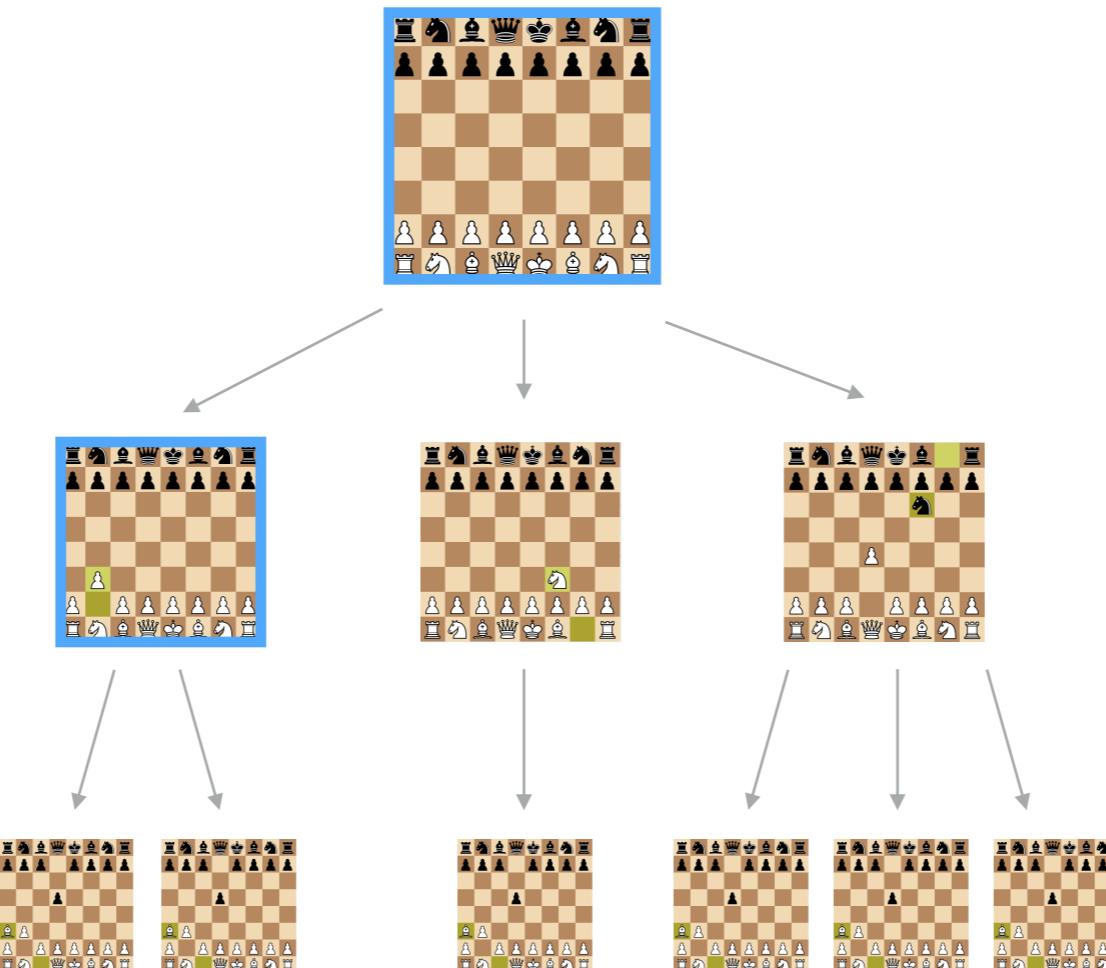
«Лёгкий компьютер» в играх

Обходит дерево сверху вниз и выбирает варианты, в которых не проиграет в ближайшие несколько шагов (BFS)



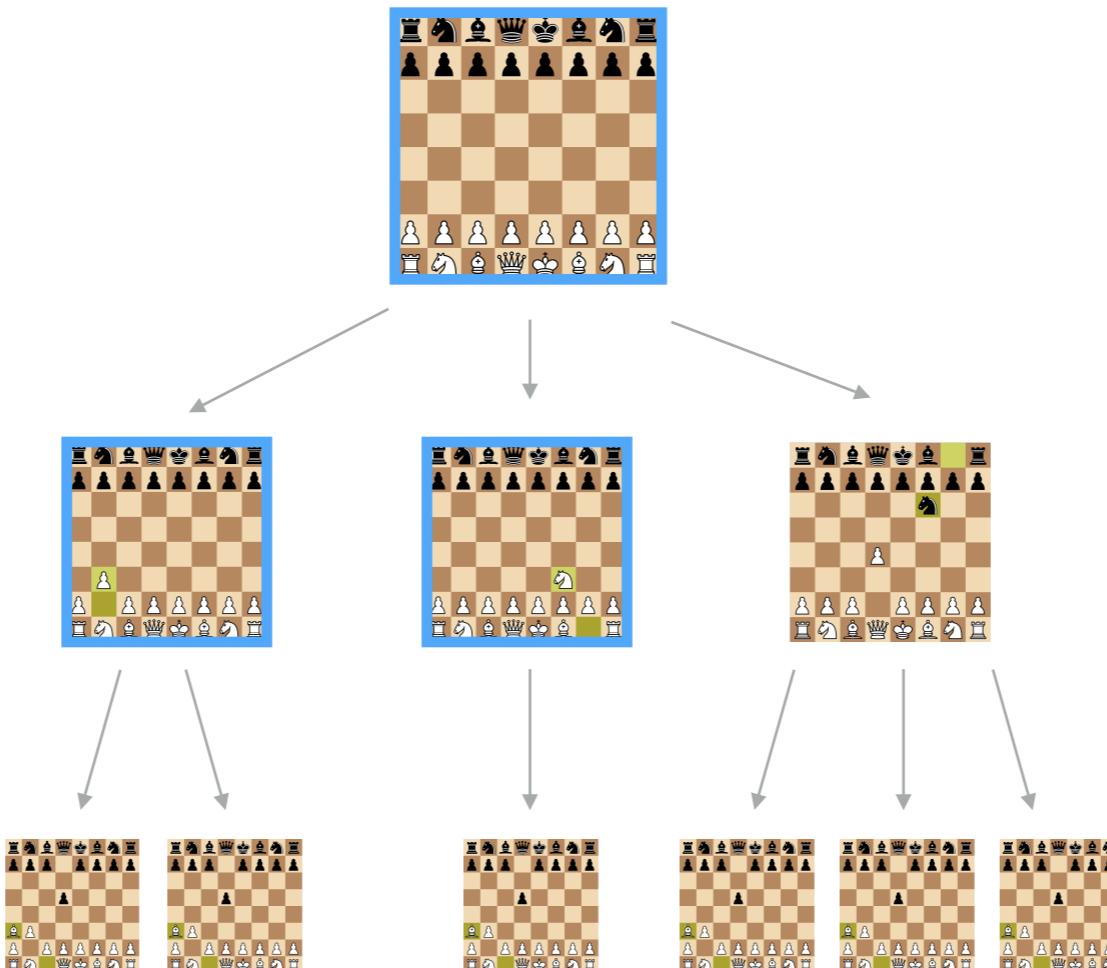
«Лёгкий компьютер» в играх

Обходит дерево сверху вниз и выбирает варианты, в которых не проиграет в ближайшие несколько шагов (BFS)



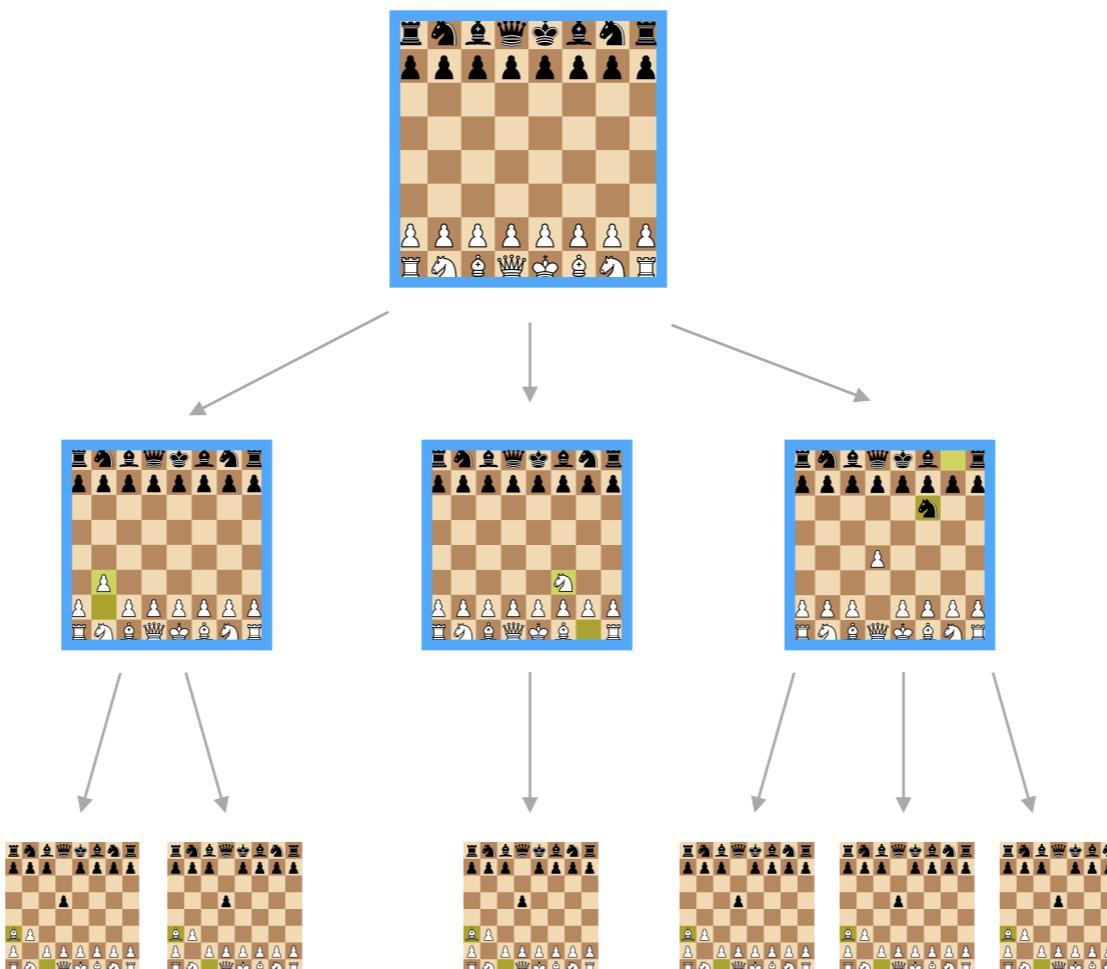
«Лёгкий компьютер» в играх

Обходит дерево сверху вниз и выбирает варианты, в которых не проиграет в ближайшие несколько шагов (BFS)



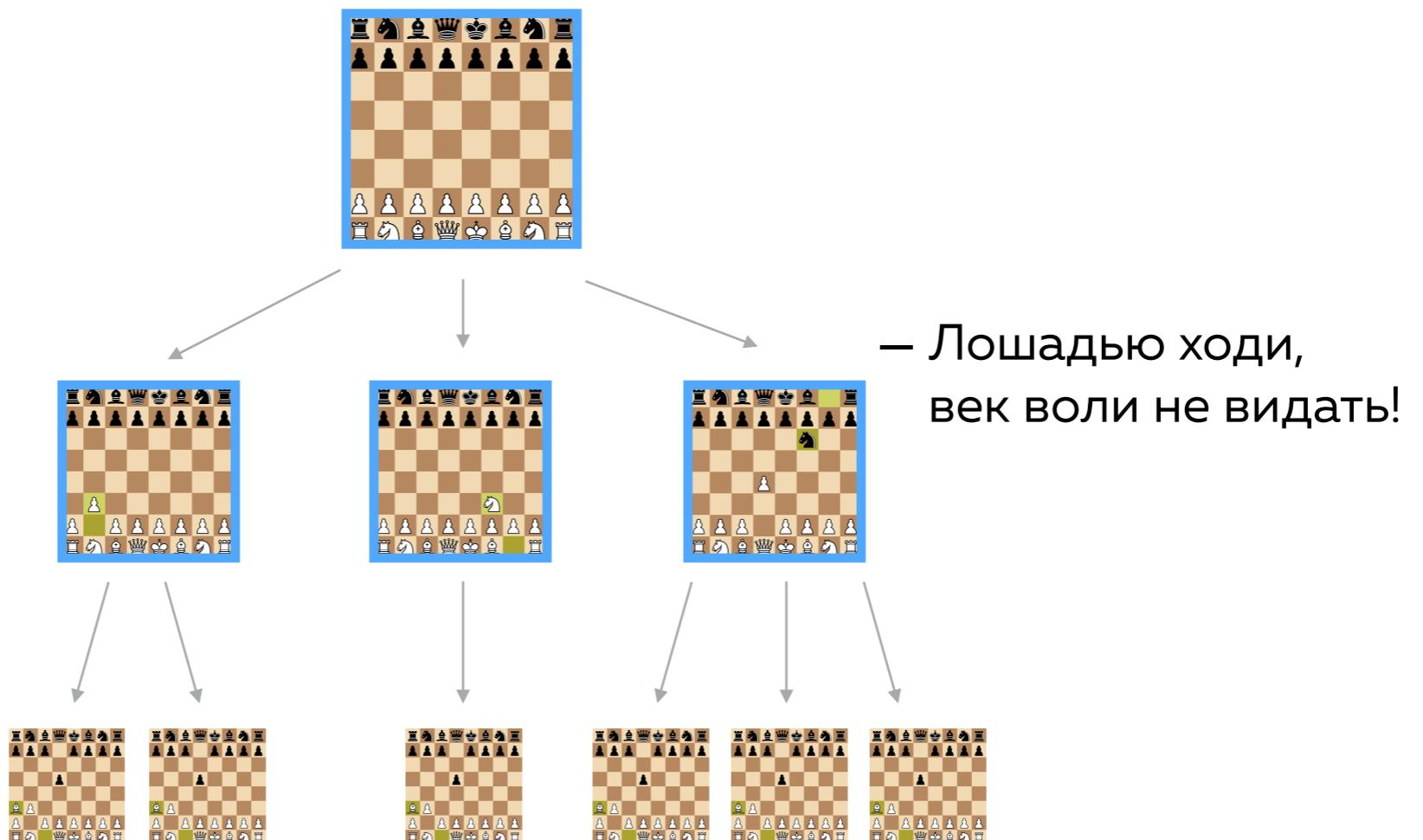
«Лёгкий компьютер» в играх

Обходит дерево сверху вниз и выбирает варианты, в которых не проиграет в ближайшие несколько шагов (BFS)



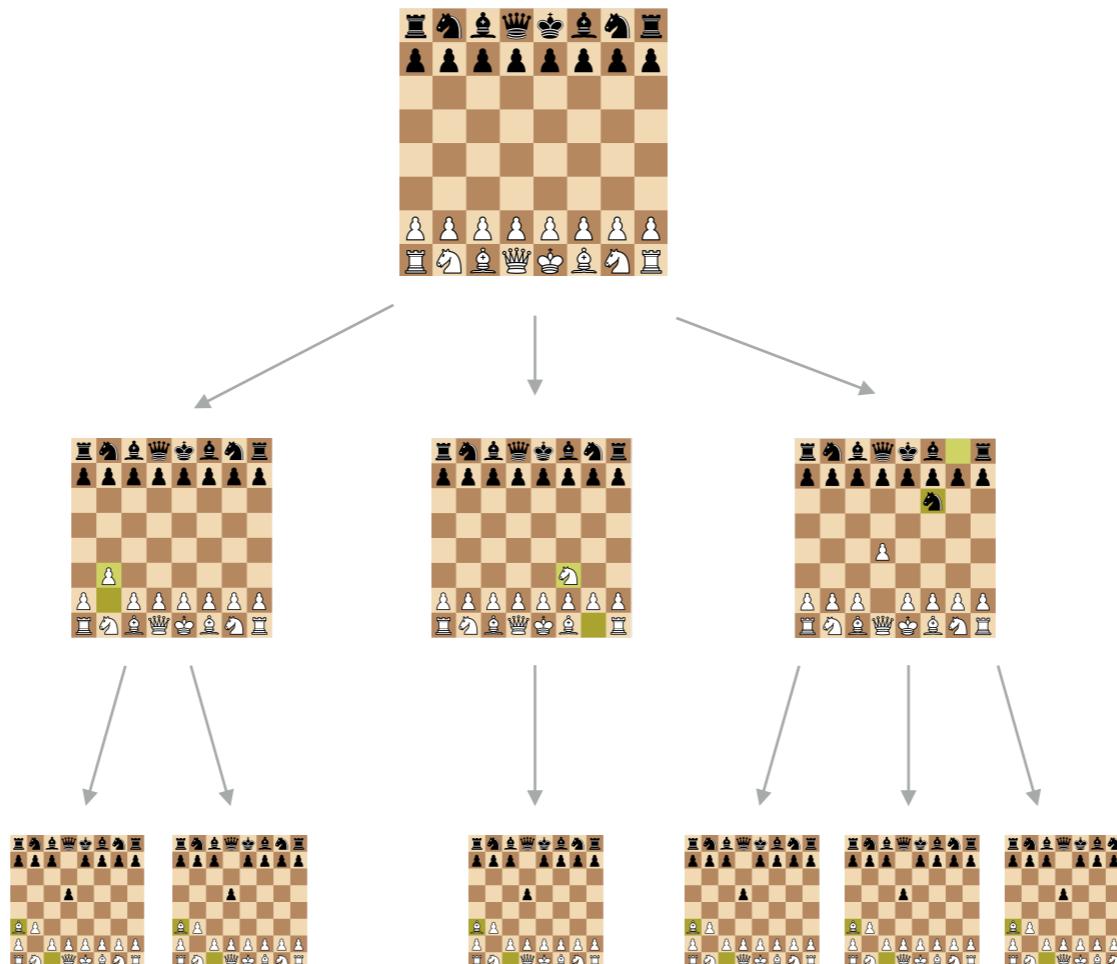
«Лёгкий компьютер» в играх

Обходит дерево сверху вниз и выбирает варианты, в которых не проиграет в ближайшие несколько шагов (BFS)



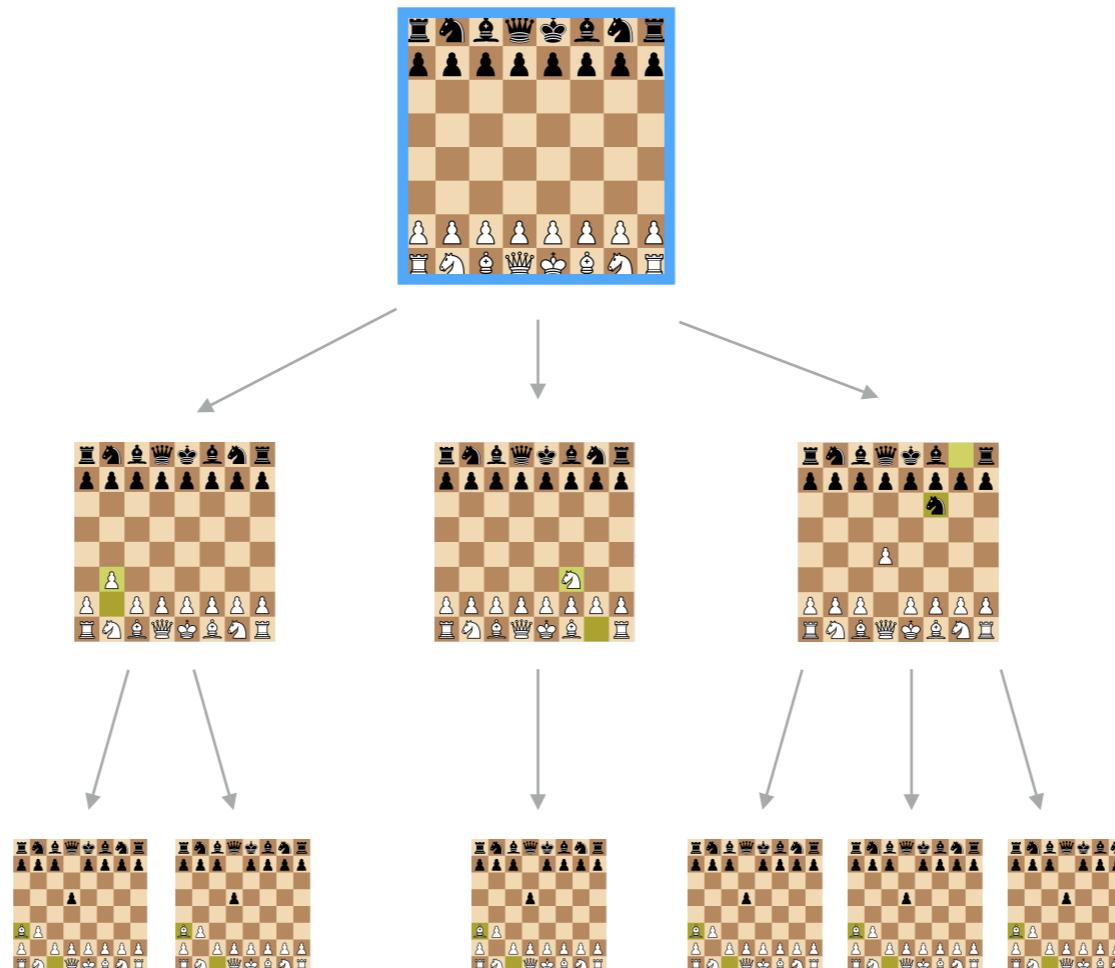
«Сложный компьютер» в играх

Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)



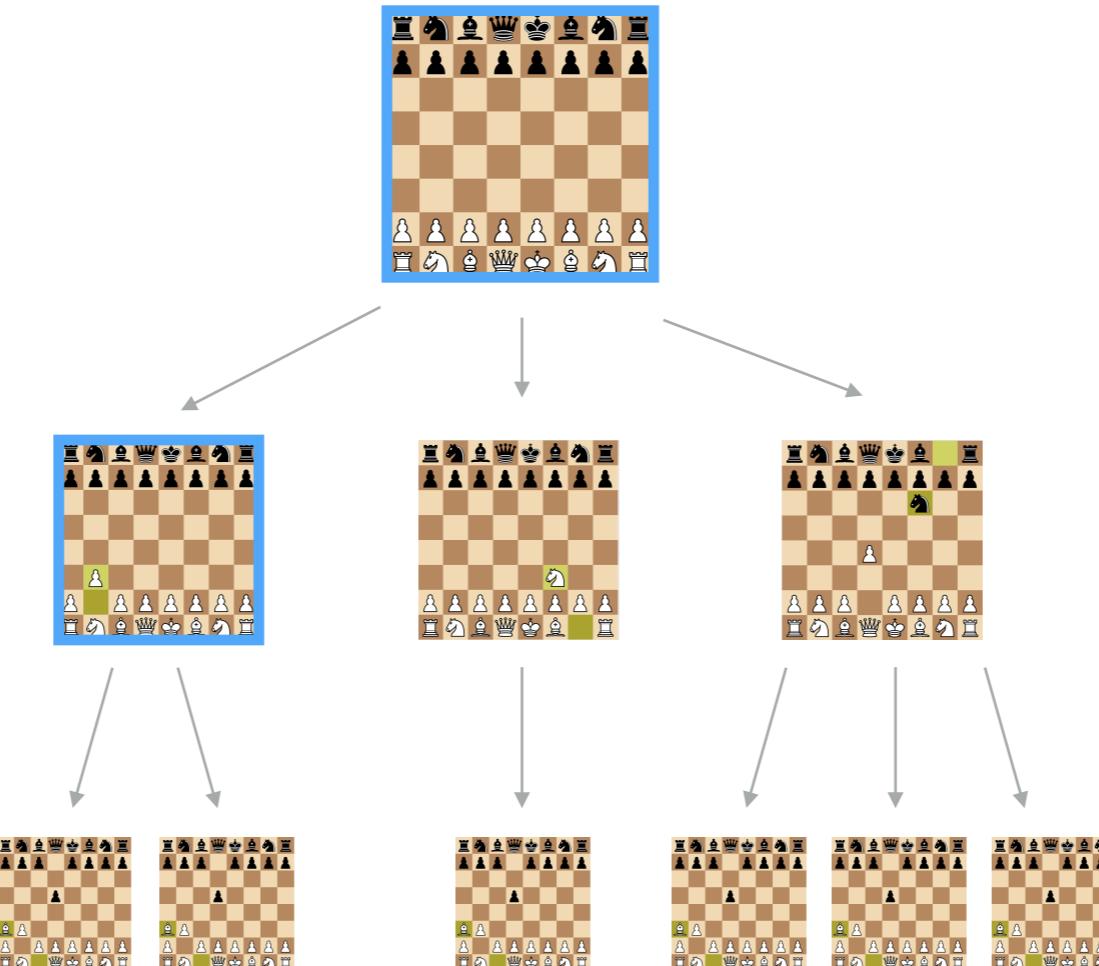
«Сложный компьютер» в играх

Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)



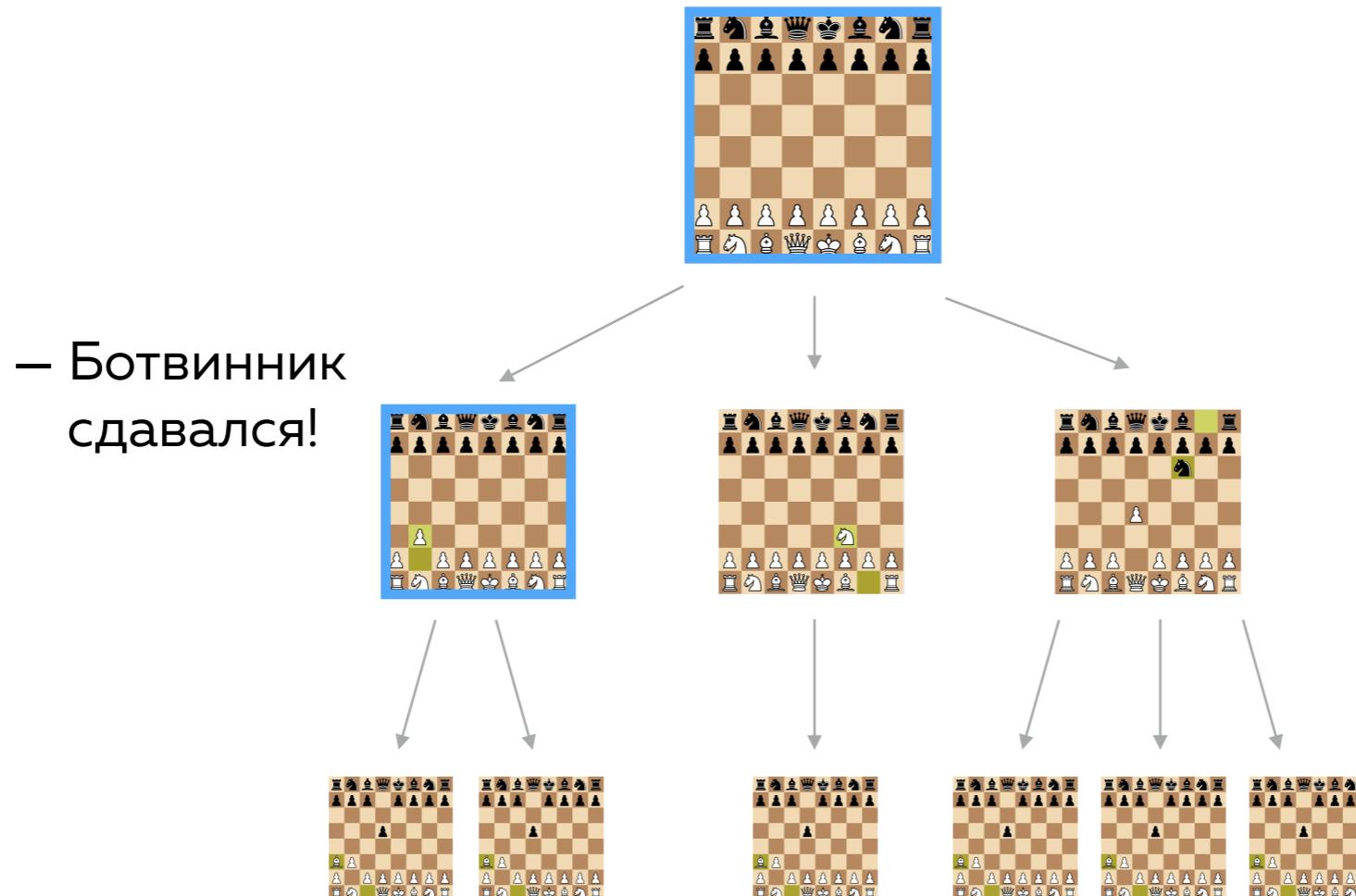
«Сложный компьютер» в играх

Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)



«Сложный компьютер» в играх

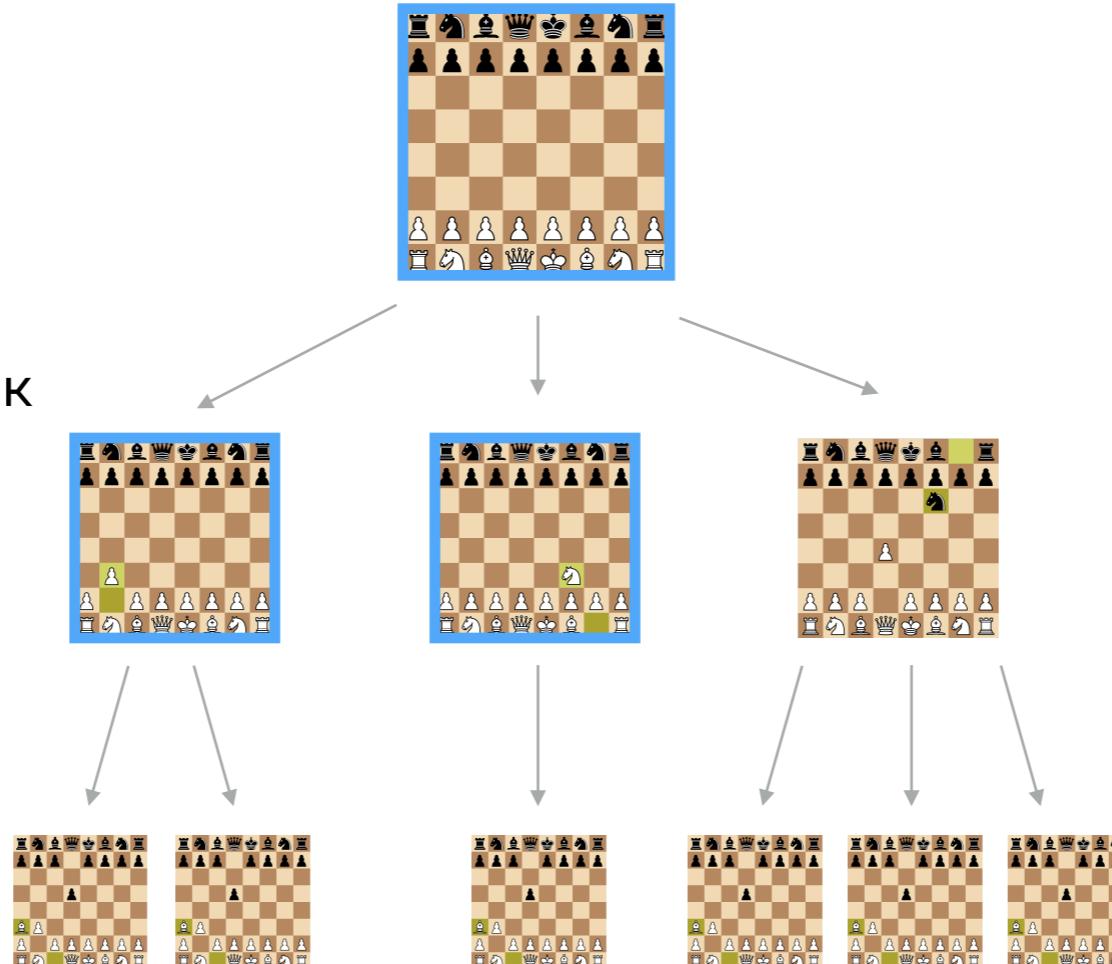
Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)



«Сложный компьютер» в играх

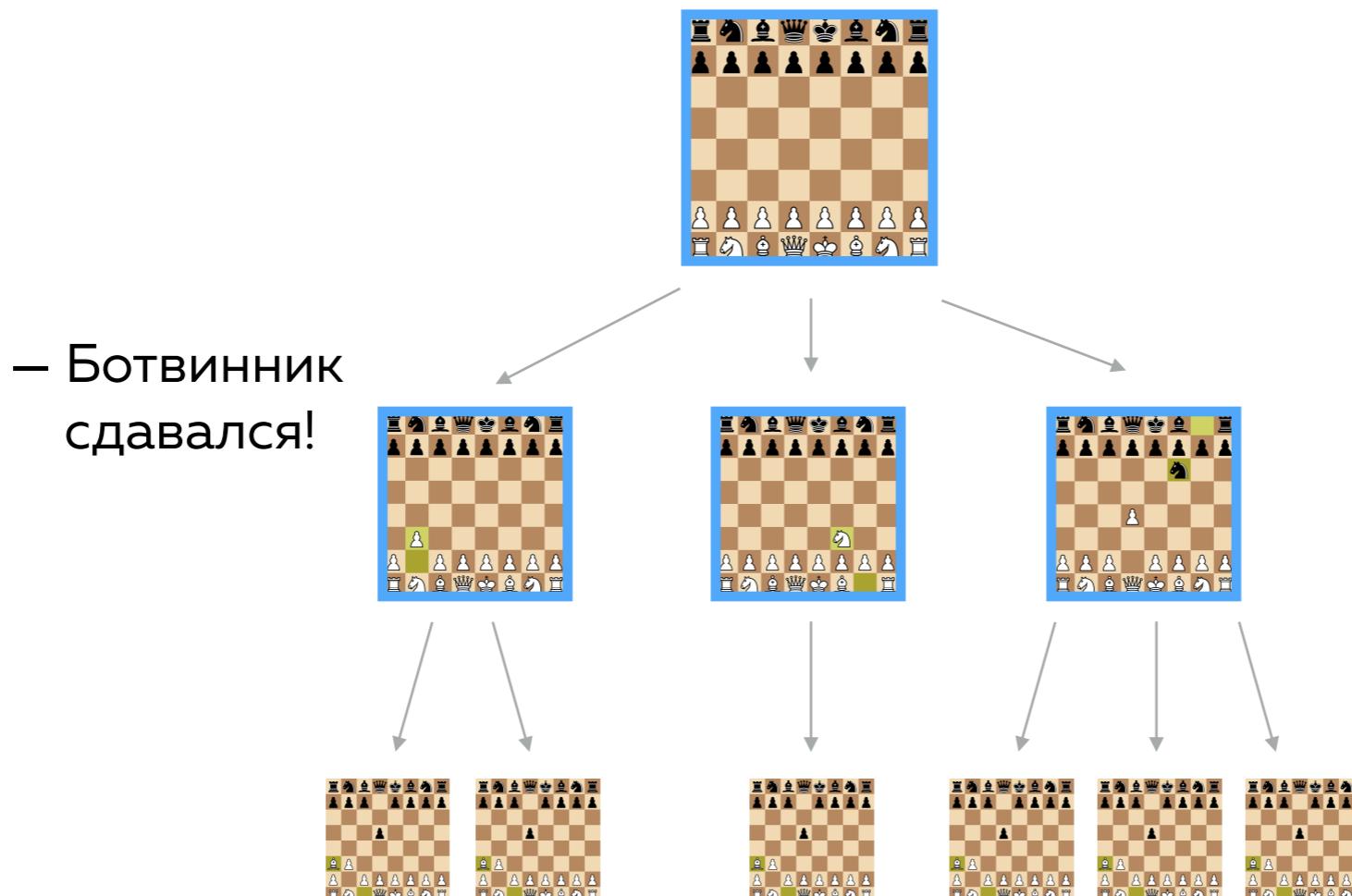
Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)

– Ботвинник
сдавался!



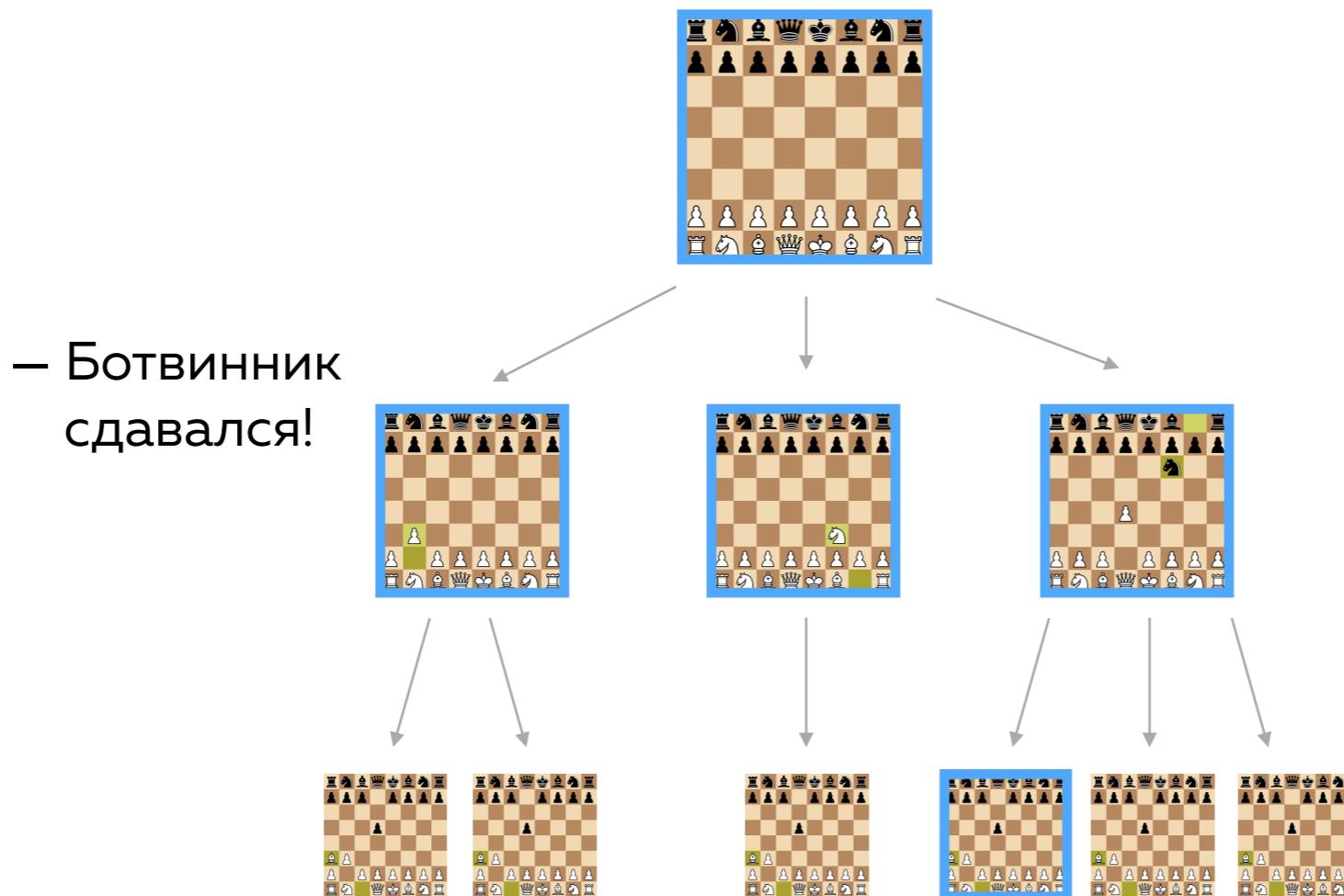
«Сложный компьютер» в играх

Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)



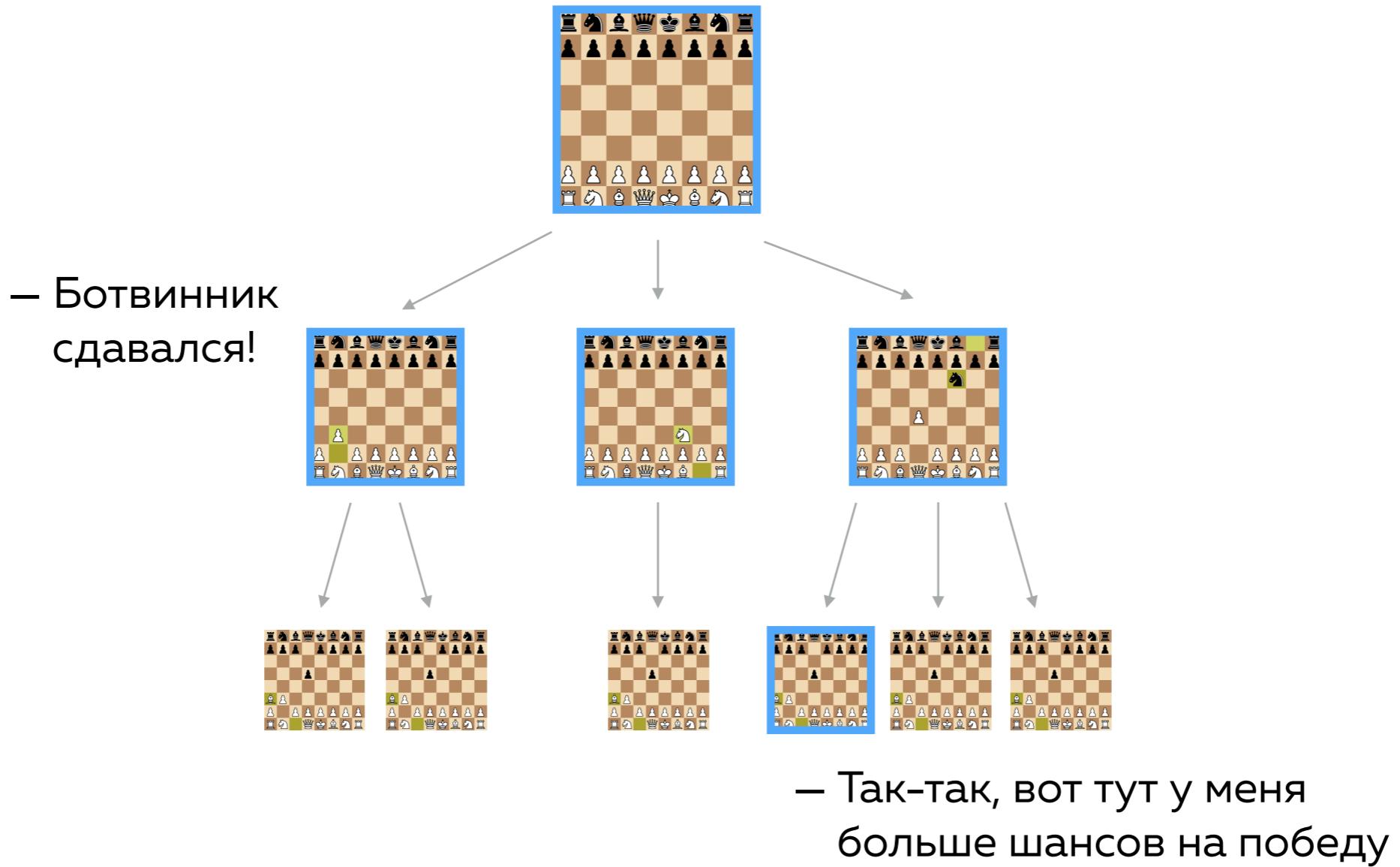
«Сложный компьютер» в играх

Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)



«Сложный компьютер» в играх

Обходит дерево как можно глубже и выбирает выигрышные варианты (DFS)





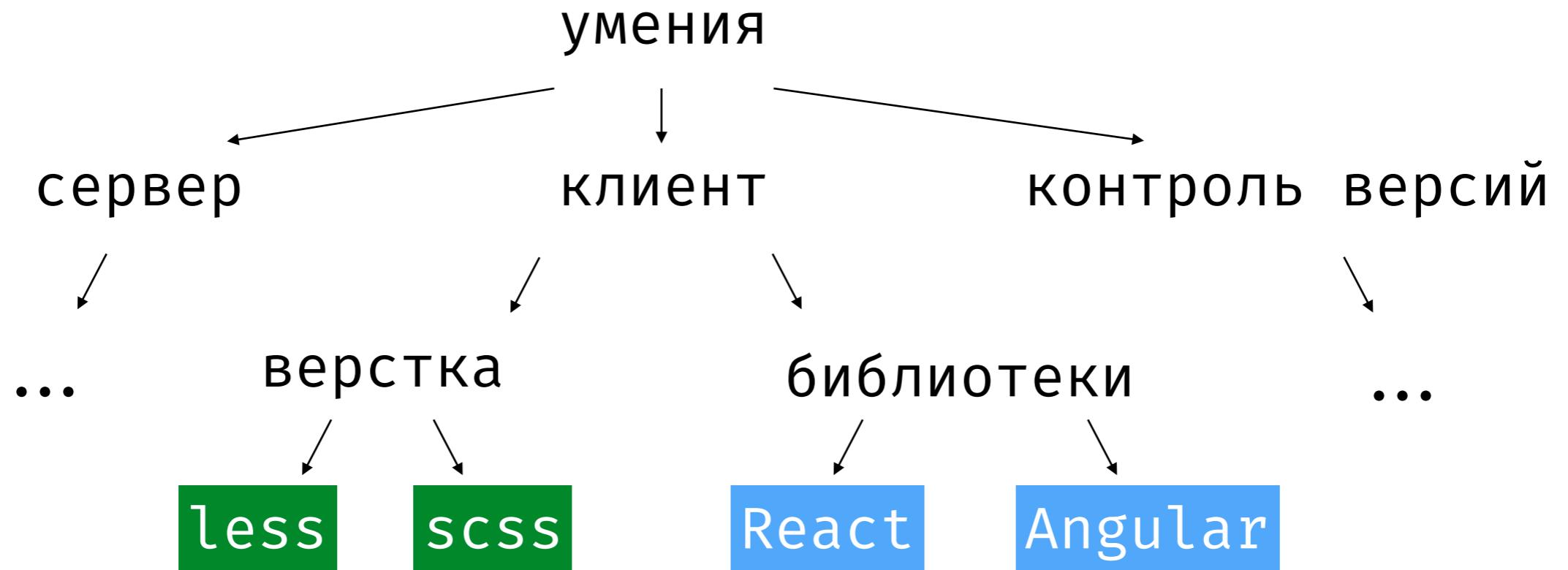
Анализ рынка разработки

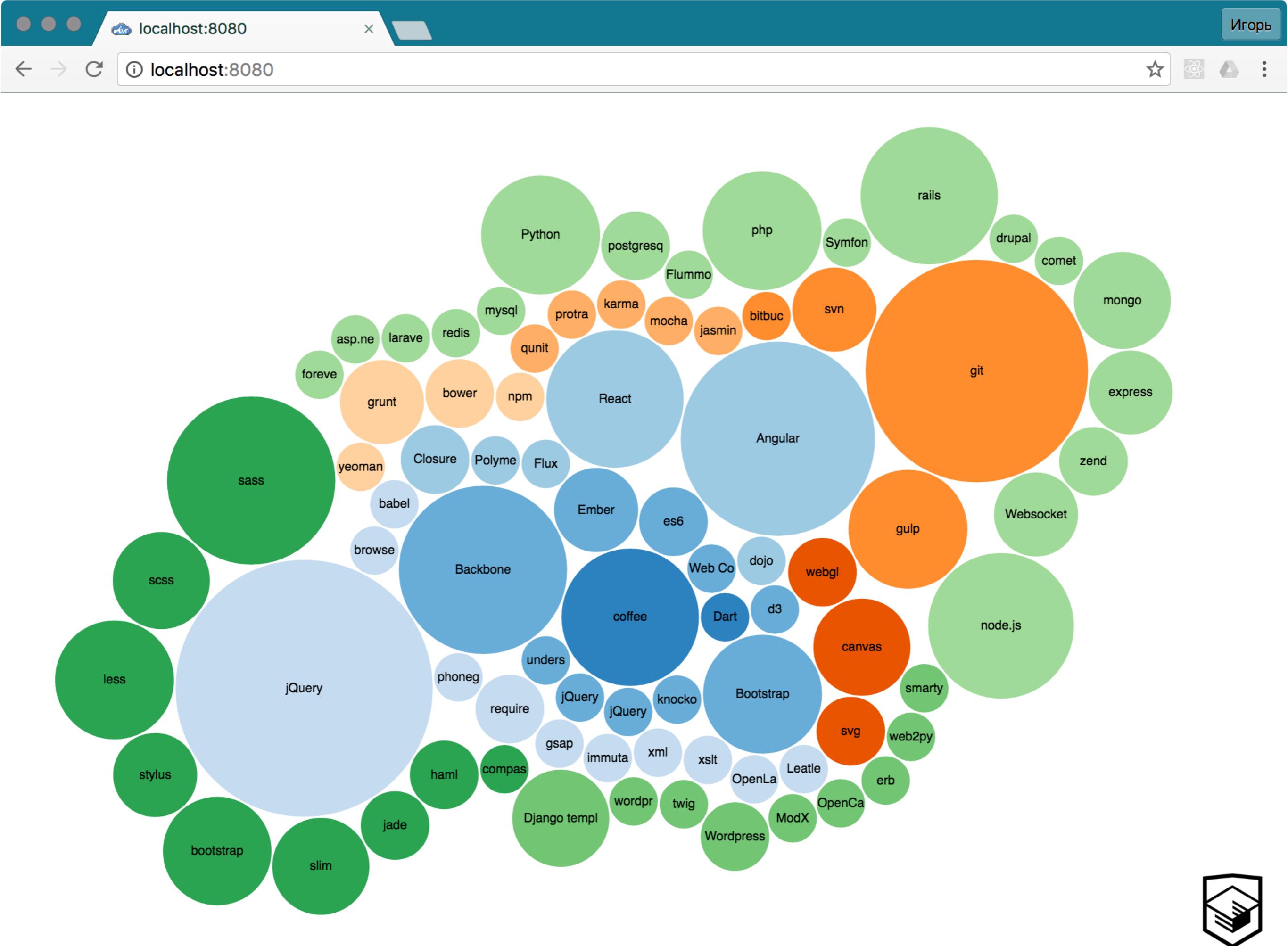
Статистика требований к разработчикам на основе
вакансий «Моего Круга» с помощью библиотеки d3.js



Дерево требований к разработчикам

Листьями являются конкретные технологии, а узлами – группы







Преимущества деревьев





Преимущества деревьев

- сохраняются и передаются в виде строковых данных





Преимущества деревьев

- сохраняются и передаются в виде строковых данных
- легко читаются





Преимущества деревьев

- сохраняются и передаются в виде строковых данных
- легко читаются
- быстро выполняются (*нативный объект JS*)





Преимущества деревьев

- сохраняются и передаются в виде строковых данных
- легко читаются
- быстро выполняются (*нативный объект JS*)
- не имеют чёткого формата описания, поэтому идеально подстраиваются под задачу



Ищите решения, а не инструменты

Через 20 лет не будет ни Бэйбеля, ни Реакта, ни Вебпака, ни Джаваскрипта. А деревья и битовые карты – будут



Что поизучать



Что поизучать

- github.com/htmlacademy/bitset.js
библиотека от HTML Academy для работы с битовыми массивами в JS – обёртки для добавления, удаления, проверки и получения объектов с состоянием



Что поизучать

- github.com/htmlacademy/bitset.js
библиотека от HTML Academy для работы с битовыми массивами в JS – обёртки для добавления, удаления, проверки и получения объектов с состоянием
- o0.github.io/trees
интерактивные демки, примеры для разных областей применения, больше теории, исходники



Что поизучать

- github.com/htmlacademy/bitset.js
библиотека от HTML Academy для работы с битовыми массивами в JS – обёртки для добавления, удаления, проверки и получения объектов с состоянием
- o0.github.io/trees
интерактивные демки, примеры для разных областей применения, больше теории, исходники
- github.com/thejameskyle/itsy-bitsy-data-structures
репозиторий, с интересным и подробным рассказом про алгоритмы и структуры данных на JS





– Спасибо!



tutors@htmlacademy.ru

Наставничество – отличный способ найти себе
в команду новичка и воспитать его так, как считаешь
должным на готовой продуманной инфраструктуре или
просто поделиться опытом

