Arrays, Loops, Functions

ITP 301

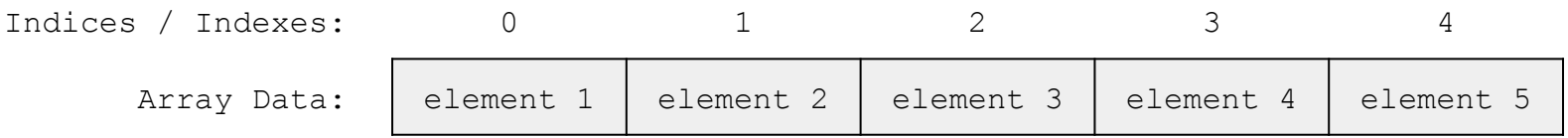
Spring 2021

© 2021 Zune Nguyen

Arrays

Used to store series of data in a single variable.

Often represented as a series of boxes indexed from `0` to `length-1`:



JavaScript array examples:

- `document.querySelectorAll()`
- `document.getElementsByClassName()`
- `document.getElementsByTagName()`
- `.children` Property

© 2021 Zune Nguyen

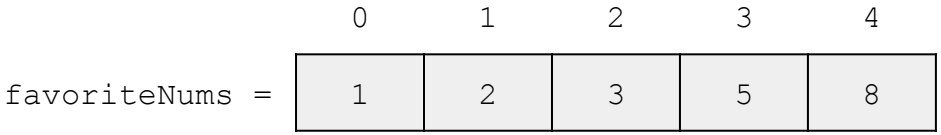
Creating Arrays

Syntax:

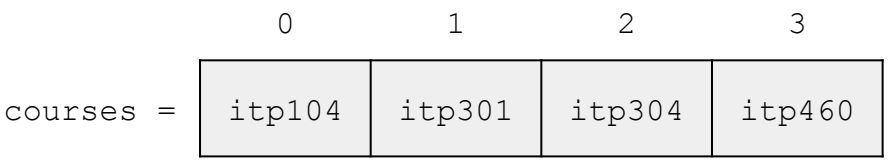
```
var arrayName = [elt1, elt2];
```

emptyArray =

```
var emptyArray = [];
```




```
var favoriteNums = [1, 2, 3, 5, 8];
```



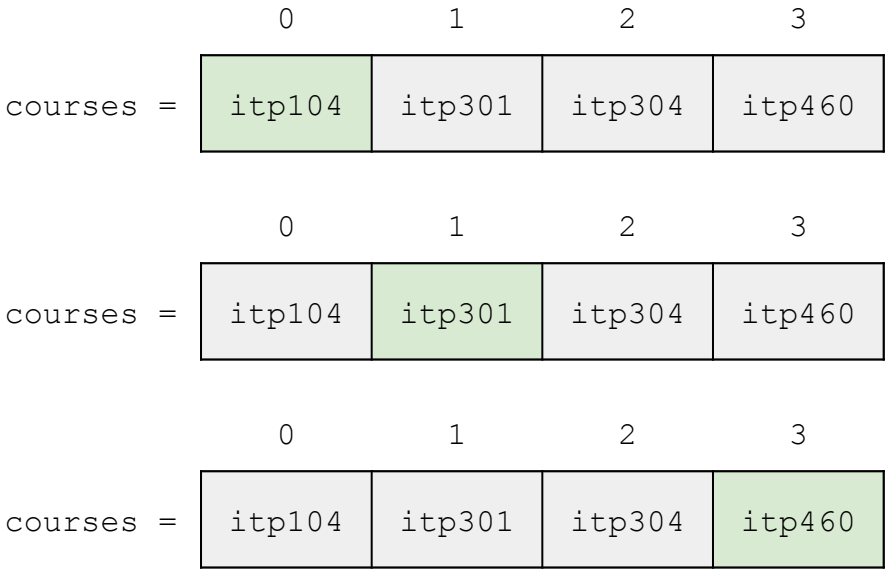

```
var courses = ['itp104', 'itp301', 'itp300', 'itp460'];
```

© 2021 Zune Nguyen

Accessing Arrays

Syntax:

```
arrayName[index]
```




```
courses[0];
```

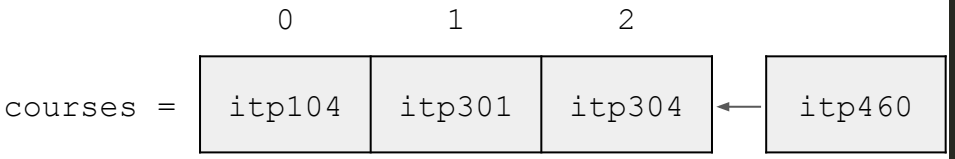
```
courses[1];
```

```
courses[3];
```

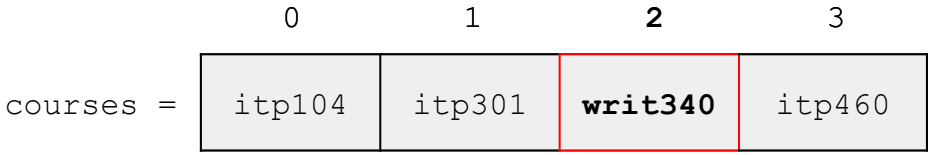
© 2021 Zune Nguyen

Adding Elements

`arrayName.push(elt)`	Add *elt* to the end of *arrayName*
`arrayName[i] = elt`	Add *elt* at index *i* of *arrayName*




```
courses.push ('itp460');
```

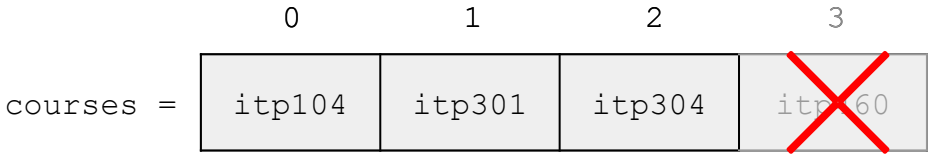


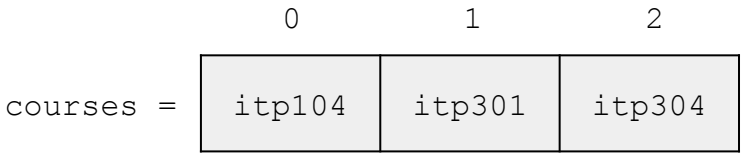

```
courses[2] = 'writ340';
```

© 2021 Zune Nguyen

Removing Elements

`arrayName.pop()`	Remove last element in *arrayName*
`arrayName.splice(i, c)`	Starting at *i*, remove *c* number of elements from *arrayName*



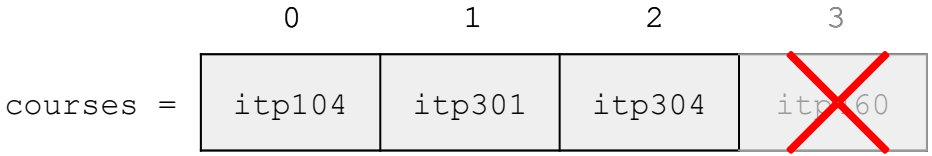


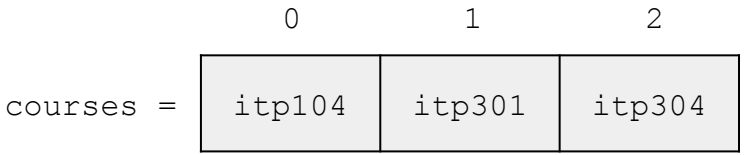

```
courses.pop();
```

© 2021 Zune Nguyen

Removing Elements

`arrayName.pop()`	Remove last element in *arrayName*
`arrayName.splice(i, c)`	Starting at *i*, remove *c* number of elements from *arrayName*



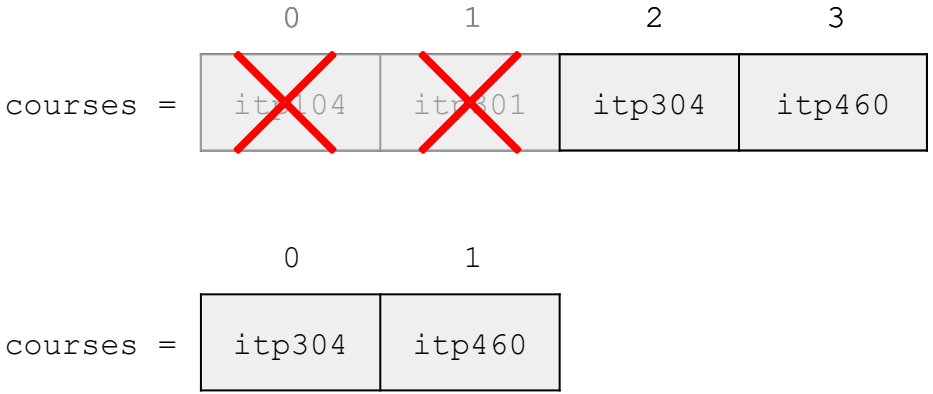



```
courses.splice(3, 1);
```

© 2021 Zune Nguyen

Removing Elements

`arrayName.pop()`	Remove last element in *arrayName*
`arrayName.splice(i, c)`	Starting at *i*, remove *c* number of elements from *arrayName*

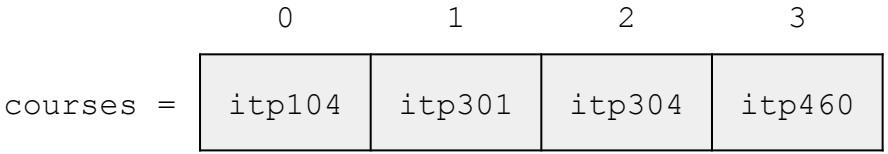



```
courses.splice(0, 2);
```

© 2021 Zune Nguyen

Array Length and Searching

`length`	Sets or returns # of elements in array.
`indexOf(`*`elt`*`)`	Searches & returns position of the *elt* in array. Returns `-1` if *elt* was not found.




```
courses.length // Returns 4

courses.indexOf('itp301') // Returns 1

courses.indexOf('itp460') // Returns 3

courses.indexOf('ITP301') // Returns -1

courses.indexOf('itp 999') // Returns -1
```

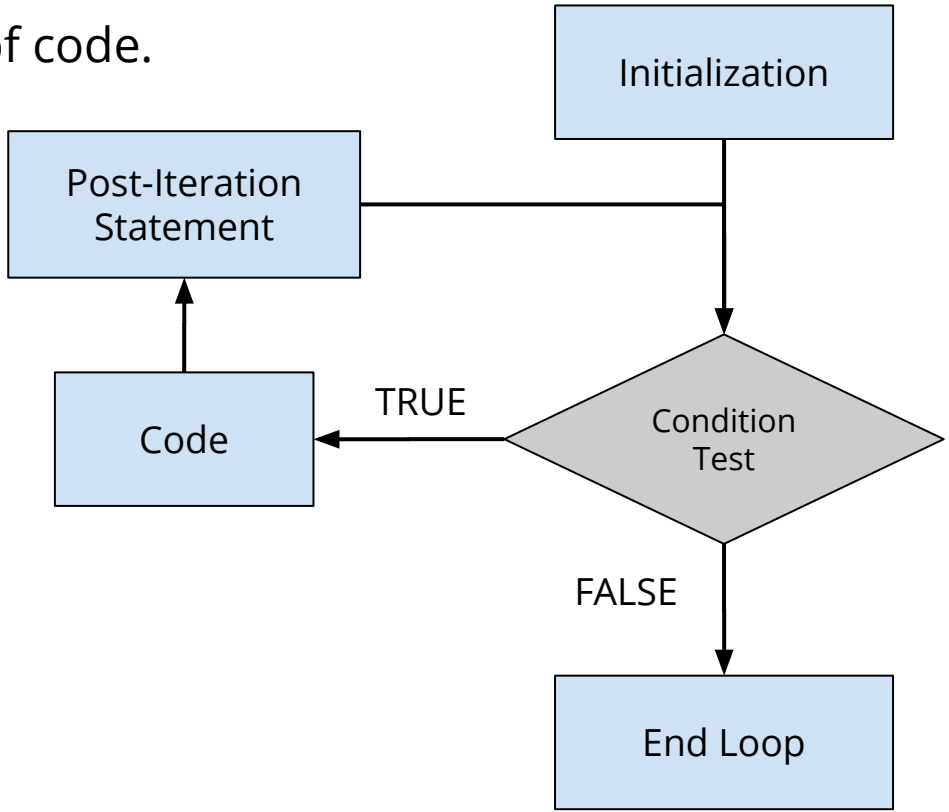
© 2021 Zune Nguyen

Loops

Used to repeatedly execute sections of code.

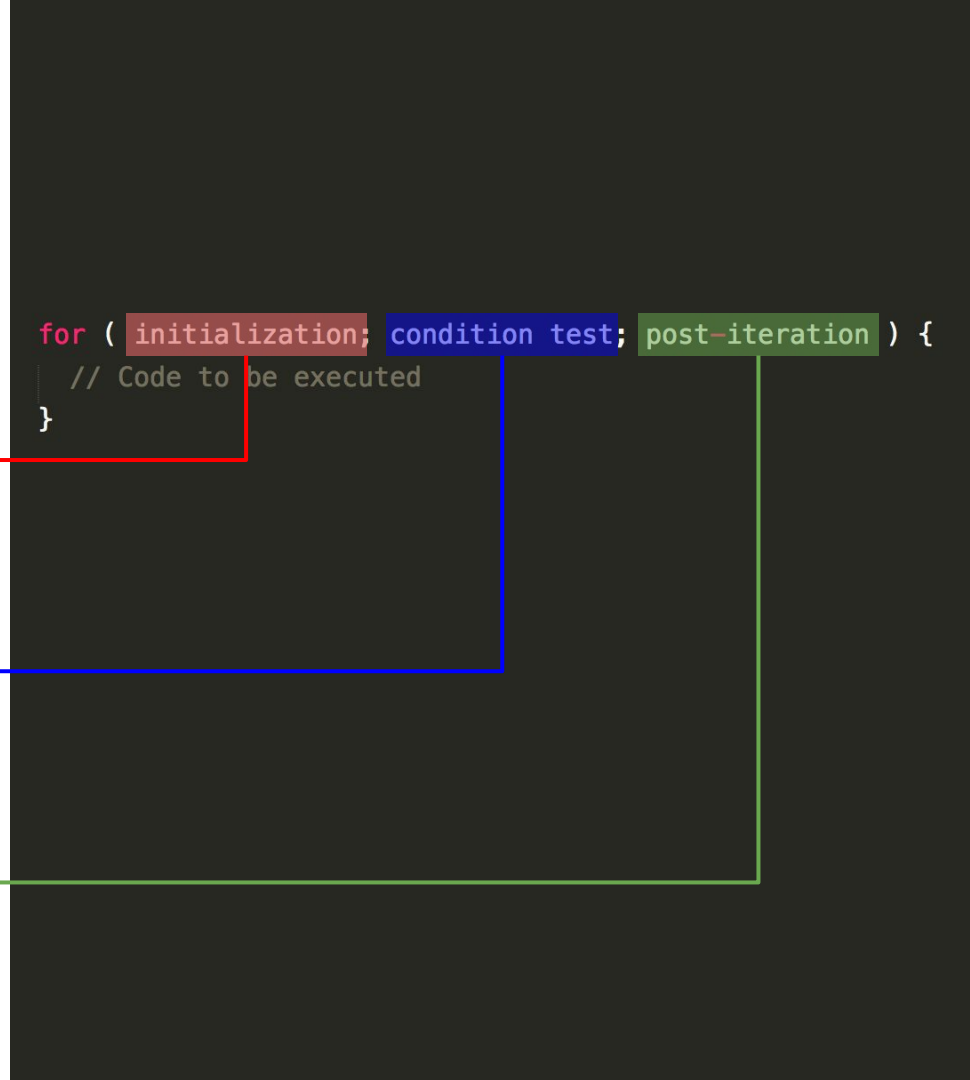
`for` loops consists of 3 parts:

1. Initialization
2. Condition Test
3. Post-Iteration Statement



© 2021 Zune Nguyen

`for` Loop Syntax



Initialization:

Executed once before loop starts.

Condition Test:

Evaluated before each iteration.

Post-Iteration:

Executed after each iteration.

© 2021 Zune Nguyen

`for` Loop Examples

Output numbers 0 through 9.

```
for (var i = 0; i < 10; i++) {
  console.log(i);
}
```

Output contents of array `courses`.

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

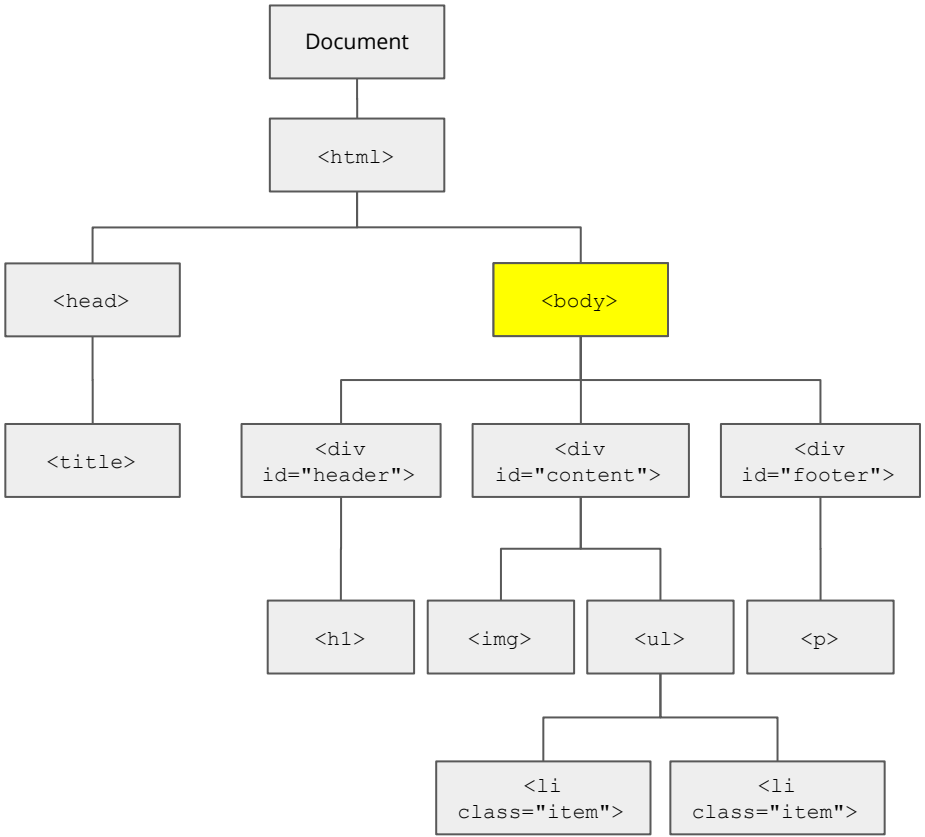
© 2021 Zune Nguyen

DOM Traversal

`parentNode`	Parent element.
`children`	Children elements.
`nextSibling`	Next sibling, including whitespace (text) nodes.
`nextElementSibling`	Next sibling, excluding whitespace (text) nodes.
`previousSibling`	Previous sibling, including whitespace (text) nodes.
`previousElementSibling`	Previous sibling, excluding whitespace (text) nodes.

© 2021 Zune Nguyen

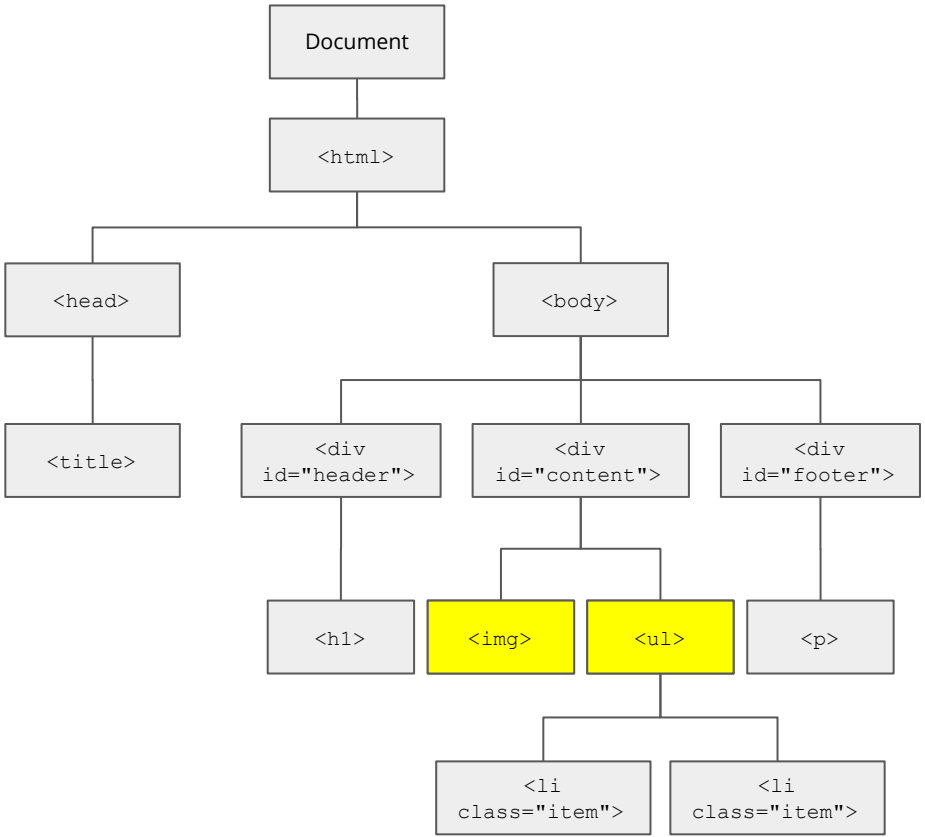
DOM Traversal




```
document.querySelector('#content').parentNode;
```

© 2021 Zune Nguyen

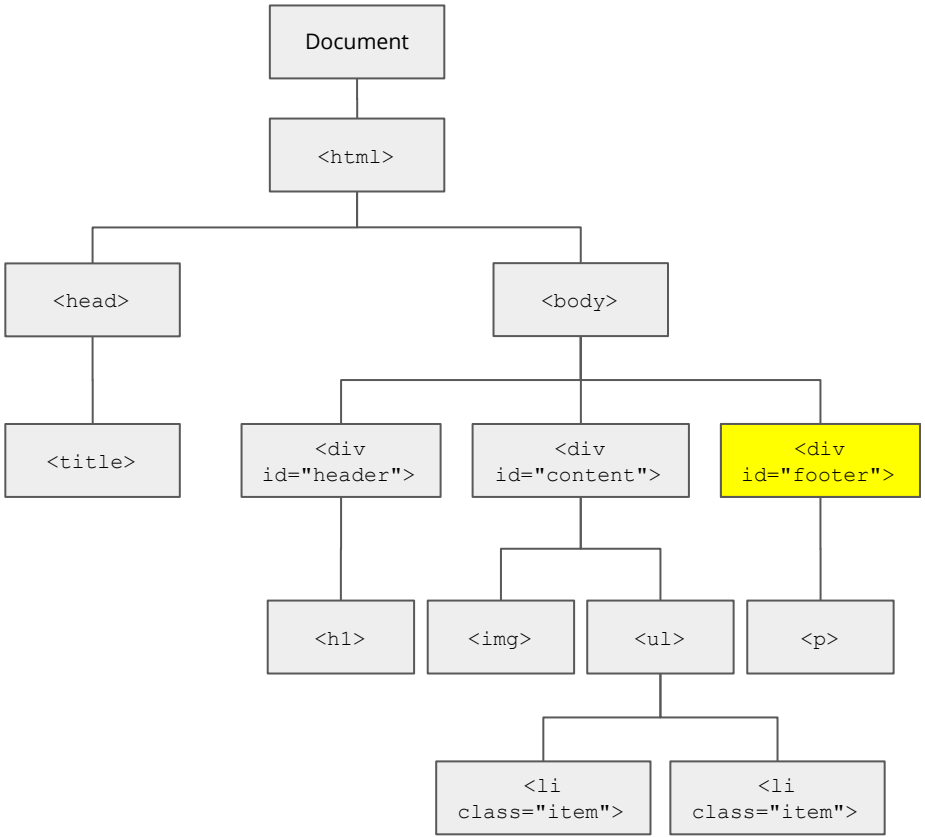
DOM Traversal




```
document.querySelector('#content').parentNode;

document.querySelector('#content').children;
```

© 2021 Zune Nguyen

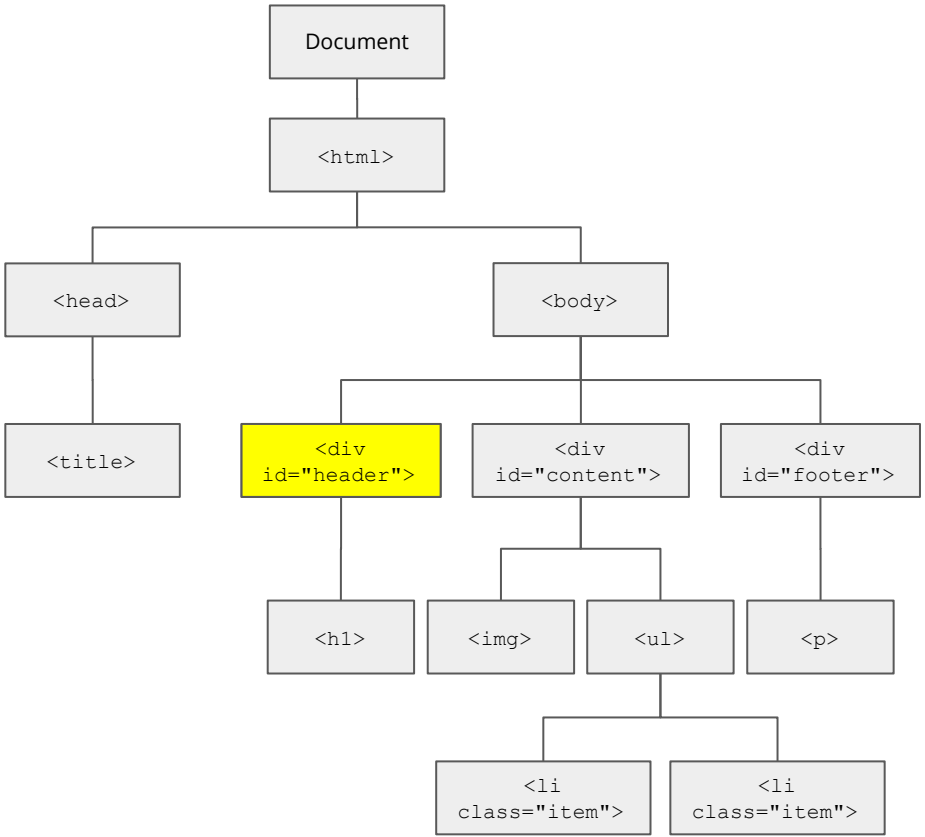
DOM Traversal




```
document.querySelector('#content').parentNode;

document.querySelector('#content').children;

document.querySelector('#content').nextElementSibling;
```

© 2021 Zune Nguyen

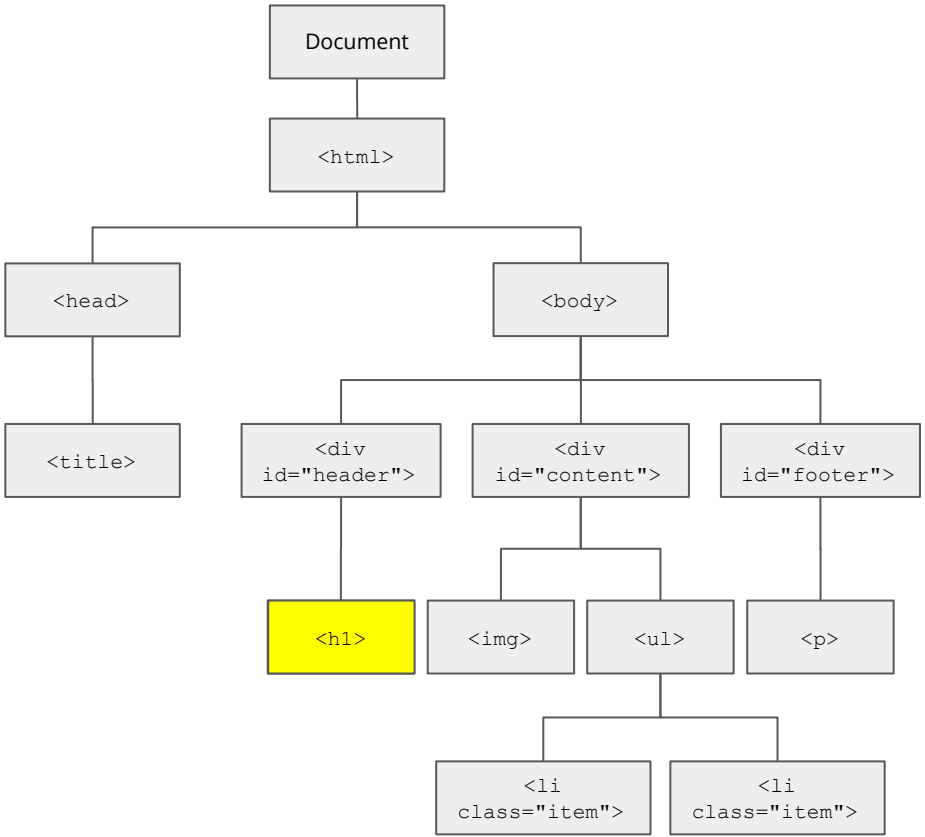
DOM Traversal




```
document.querySelector('#content').parentNode;

document.querySelector('#content').children;

document.querySelector('#content').nextElementSibling;

document.querySelector('#content').previousElementSibling;
```

© 2021 Zune Nguyen

DOM Traversal




```
document.querySelector('#content').parentNode;

document.querySelector('#content').children;

document.querySelector('#content').nextElementSibling;

document.querySelector('#content').previousElementSibling;

document.querySelector('#content').previousElementSibling.children;
```

© 2021 Zune Nguyen

Functions

Section of code that is given a name.

Functions can be invoked, i.e. that section of code can be called or executed.

Some built-in JavaScript functions:

- `alert()`
- `console.log()`
- `document.querySelector()`
- Almost anything that ends with `()`

```
// Function Declaration (creating a function)
function functionName(){
  console.log('Hello World!');
}


// Invoking the function (executing function code)
functionName(); // Prints 'Hello World!'
```

© 2021 Zune Nguyen

Function Parameters

Variables or data given to the function when it is invoked.

```
// Function accepts 2 parameters.
function functionName(parameter_1, parameter_2) {
  console.log(parameter_1);
  alert(parameter_2);
}


// Invoke the function and provide 2 parameters.
functionName('Hello Console!', 'Hello Alert!');
```

© 2021 Zune Nguyen

Function Returns

Functions can return data to the caller.

```
function functionName() {
  return "Hello World!";
}

var data = functionName();
// 'data' is now "Hello World!"
```

```
function add(num1, num2) {
  return (num1 + num2);
}

var result = add(1, 2);
// 'result' is now 3
```

© 2021 Zune Nguyen