Introduction to JavaScript

23 February 2019

hackerschool
by NUS Hackers

Tools Required

- Google Chrome
  - http://chrome.google.com/
- Sublime Text
  - http://www.sublimetext.com/3
- Visual Studio Code
  - https://code.visualstudio.com/
- Materials
  - https://drive.google.com/open?id=1-QDj-DICDJ8AFr9Aip9SNtmX5l8Ha9JA

Goals

To learn how we can use JavaScript to create a simple webpage, in conjunction with HTML and CSS

What is JavaScript?

- Also known as ECMAScript
- A high-level, interpreted programming language, weakly typed and prototyped based (from Wikipedia)
- A great resource for learning JavaScript:
  - https://developer.mozilla.org/bm/docs/Web/JavaScript
- Files end with **.js** extension

What is JavaScript?

- Enables us to write complex logic to make our websites interactive

Chrome Developer Tools

- Mac: Command + Option + J (Console) or Command + Option + C (Elements)
- Windows: Control + Shift + J (Console) or Control + Shift + C (Elements)

Chrome Developer Tools

Details

First Speaker: Dr Alexandre Gouaillard (CoSMo)

Details: Dr Alexandre Gouaillard is the CEO of CoSMo.

Topic: WebRTC: Real Time Media Revolution

Details: At the beginning of the decade, Google introduced WebRTC to the world, in an effort to fill a gap between desktop and web. Nowadays WebRTC is everywhere, in your phone, your apps, your fridge. Facebook is using it at the core of Messenger and claims a whooping 17 billion calls, for example. More and more complex apps, with programmatic APIs, AI, ... are augmenting the original tech for truly amazing products. Now that WebRTC Next Version is being worked on, it's a good time to position the start-up ecosystem to be ready for the next disruptive wave.

Second Speaker: Louay Mohamed and Vladimir Dimitrov (TransferWise)

Details: Louay and Vladimir are Senior Software Engineers at TransferWise.

Topic: Microservices Monitoring

Details: In the world of Microservices, observability is one of the first challenge you need to think of. We will start with an introduction of the

Copy Meetup

Organizer tools

Friday, February 15, 2019
6:30 PM to 8:30 PM
Add to calendar

Icube Building
21 Heng Mui Keng Terrace · Singapore

How to find us
We're inside the Hangar inside the ICube building

Elements Console Sources Network 2

top Filter Default levels

```
▸GET https://connect.facebook.net/en_US/fbevents.js (index):8
net::ERR_BLOCKED_BY_CLIENT
▸Uncaught (in promise) TypeError: MicrotargetContainer.jsx:2
Cannot read property 'name' of undefined
    at MicrotargetContainer.jsx:2
    at nrWrapper ((index):8)
>
```

Console What's New ×

Highlights from the Chrome 72 update

Visualize performance metrics
Performance metrics like DOMContentLoaded and First Meaningful Paint are now marked in the Timings section of the Performance panel.

Highlight text nodes
Hover over a text node in the DOM Tree to highlight it in the viewport.

Building Blocks - Variables

1. **Number**
2. **String**
3. **Boolean**
4. Undefined

Building Blocks - Variables

```
// Declares a variable with a number value
let numberOfDogs = 5;
let moneyLeft = 1.76;

// Declares a variable with a boolean value
let isStrong = false;

// Declares a variable with a string value
let name = "Jane Doe";

// prints the values
console.log(numberOfDogs);  // 5
console.log(moneyLeft);  // 1.76
console.log(isStrong);  // false
console.log(name);  // Jane Doe
```

Building Blocks - Variables

```
// Declares a constant variable with a number value
const numberOfDogs = 5;

// Will not work
numberOfDogs = 6;
```

Building Blocks - Functions

Think of a function as something that takes an input and then outputs a result

Building Blocks - Functions

```
// This function takes in two values and then outputs the sum
function sum(x , y) {
    return x + y;
}

console.log(sum(1,2));  // returns 3

// Functions can also take in other functions as arguments

function doSomething(fn, x , y) {
    return fn(x,  y);
}

// What does this return?
doSomething(sum, 1, 2);
// Let's try to compose a plus_two function from two plus_one functions
```

Building Blocks - Functions (Scoping)

```
// This function takes in two values and then outputs the sum
let counter = 10;
function sum(x , y) {
    return x + y;
}
sum(1, 2);
console.log(x);  // This will not give 1 but will tell you that x is not
defined
console.log(counter);  // This will give 10


// But, what if we put a console.log(counter) inside the sum function?
```

Building Blocks - Operators

- `!` - Logical NOT
- `&&` - Logical AND
- `||` - Logical OR
- `===` - Strict Equality
- `!==` - Inequality
- `>=, >, <=, <` - other mathematical inequalities

Building Blocks - Arrays & Objects

```
// This function takes in two values and then outputs the sum
const arr = [1, 2, 3, 4];
arr.push("help");   // [1,2,3,4,"help"]
arr[1];  // 2


// Objects are like key value stores
const food = {italian: "pizza", thai: "pad thai"};
// italian and thai are keys, pizza and thai are their stored values
food["italian"];  // pizza
food.thai;   // pad thai
```

Building Blocks - Conditionals

```
const animal = “dog”;
if (animal === “dog”) {
    console.log(“woof”);
} else if (animal === “cat”) {
    console.log(“meow”);
} else {
    console.log(“moo”);
}
```

Building Blocks - Loops

```
// This is a for loop
const arr = [];
for (let i = 0; i < 10; < i++) {
    arr.push(i);
}
console.log(arr)  // [0,1,2,3,4,5,6,7,8,9]

// We can also iterate over arrays with a for … of loop
const arr = [1,2,3,4,5]
for (let element of arr) {
    console.log(element);
}
// prints 1 2 3 4 5 in that order
```

Building Blocks - Loops

```
// This is a while loop. For loops and while loops are actually similar
let exponent = 5;
const base = 2;
let value = 1;
while (exponent > 0) {
    exponent -= 1;   // equivalent to exponent = exponent - 1
    value *= base;
}
console.log(value); // 32
```

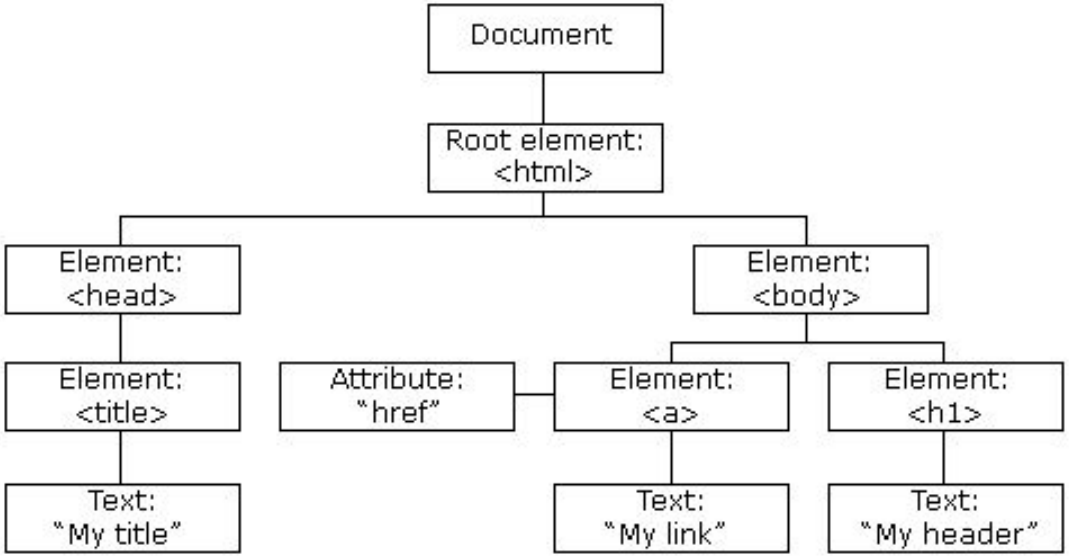
Any questions so far?

Recap - HTML and CSS

- HTML - Structures our webpage
- CSS - Styles our webpage

Recap - HTML

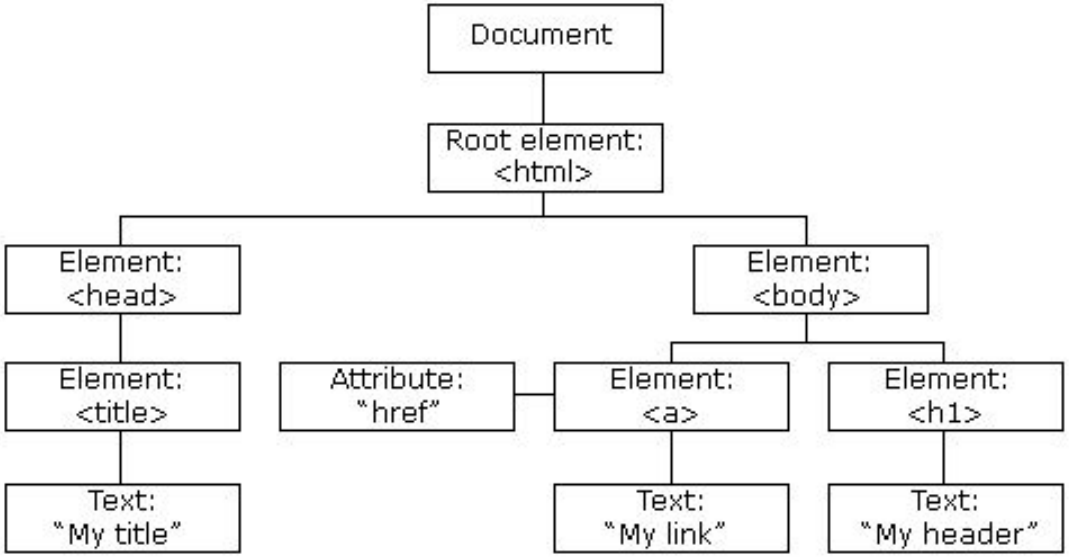
HTML can be represented by the DOM (Document Object Model) tree



Recap - CSS

CSS styles our elements:

```
<h1 style="color: red> My Header</h1>
```



Demo

Welcome | impurehtml.html

```html
<html>
<head></head>
<body>
  <div>
  </div>
</body>
<script>
  const divEl = document.querySelector("div");
  const pEl = document.createElement("p");
  pEl.innerHTML = "I love JavaScript!!!";
  divEl.appendChild(pEl);
</script>
</html>
```

purehtml.html

```html
<html>
<head></head>
<body>
  <div>
    <p>I love JavaScript</p>
  </div>
</body>
</html>
```

Demo Explained

- The `<script>` tag enabled us to write JavaScript within the HTML file
- First, what we did was to select the `<div>` element using `document.querySelector`. Remember that HTML is represented via the DOM tree

Demo Explained

- Then we created the `<p>` element using `document.createElement`.
- We then changed its contents by accessing its `innerHTML` property.
- Lastly, we used `divEl.appendChild` to make the `<p>` element a child of the former. In the tree, the `<p>` element will be a leaf of the `<div>` element.

More on document.querySelector

- document.querySelector provides us with an easy way to get HTML elements. We can do it in three basic ways:
  - Select by id: document.querySelector(“#first-div”) . This will give us the element with the id “first-div”
  - Select by class name: document.querySelector(“.blue-class”) . This will give us the element with the class “blue-class”
  - Select by element: document.querySelector(“div”) . This will give us the div element
  - There are also more complex selector strings that you can use
- As usual, more on MDN: https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector

Listeners

- Another reason why we need to use JavaScript is because of the listeners.
- With pure HTML and CSS, the page is static. Listeners help us to handle some behaviours to make our webpage less boring.
- Listeners are tied to events. When an event is detected, listeners can act accordingly and call a function

Listeners

Here are some examples of events. More can be found on MDN.

Event	Purpose
click	Listens for a click event
submit	When a submit button (for a `<form>` element) is pressed
online	When the browser goes online
resize	When the document view has been resized

Listeners

```
// example of using a clickListener
function clickHandler() {
    console.log("Look at me, I'm Mr Meseeks!");
}

const someElement = document.querySelector("#some-id");

// We use addEventListener to add
someElement.addEventListener("click", clickHandler);
// In this case, clickHandler is a callback function. You could also define
a function in the argument using arrow functions
```

Let’s try to build a simple stopwatch to practice what we have learnt

Stopwatch

HH: MM: SS: MSS

00:00:00:000

Start Stop Reset

Important Functions

```
// Set Interval
const identifier = setInterval(callback function, time in ms);

// Remove Interval
clearInterval(identifier);

// Returns current Date in milliseconds since Jan 1, 1970
let ms = Date.now();
```

Important Functions

```
function formatTime(time) {
  // Define many variables in one line!
  let h = m = s = ms = 0;
  let newTime = '';

  h = Math.floor( time / (60 * 60 * 1000) );
  time = time % (60 * 60 * 1000);
  m = Math.floor( time / (60 * 1000) );
  time = time % (60 * 1000);
  s = Math.floor( time / 1000 );
  ms = time % 1000;

  // could teach string interpolation here as well
  newTime = pad(h, 2) + ':' + pad(m, 2) + ':' + pad(s, 2) + ':' + pad(ms, 3);
  return newTime;
}
```

Important Functions

```
function pad(value, size) {
  // Implement for/while loops here!
  let resultStr = String(value);
  let counter = size - resultStr.length;
  while (counter > 0) {
    counter -= 1;
    resultStr = '0' + resultStr;
  }
  return resultStr;
}
```

Planning

- A start function, for when the user clicks on the start button
- A stop function, when the user clicks on the stop button
- A reset function, when the user clicks on reset button
- Some way to calculate the time elapsed (Hint: Date.now() - startTime in milliseconds)
- Some way to update the clock element every millisecond (Hint: setInterval)

Extra Practice

- Suppose I want a lap button
- I want the user to be able to click on it. After clicking, it should reset the clock to 0, and save the previous timing in list (Hint: Use the `<ol>` element for this)

Hackerschool

Friday **Hacks**

Hack&Roll

NUS Hackerspace

Thanks!

Feedback Form

Upcoming Hackerschool: Advanced JavaScript

Follow us!

www.nushackers.org

www.meetup.com/NUSHackers

www.facebook.com/NUSHackers

coreteam@nushackers.org