Intro the DOM

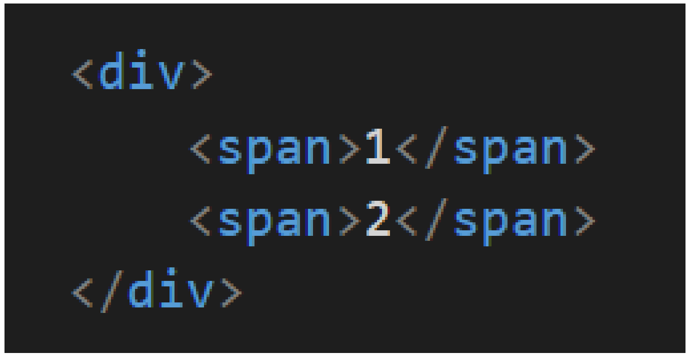
What is the DOM?

DOM stands for Document Object Model.

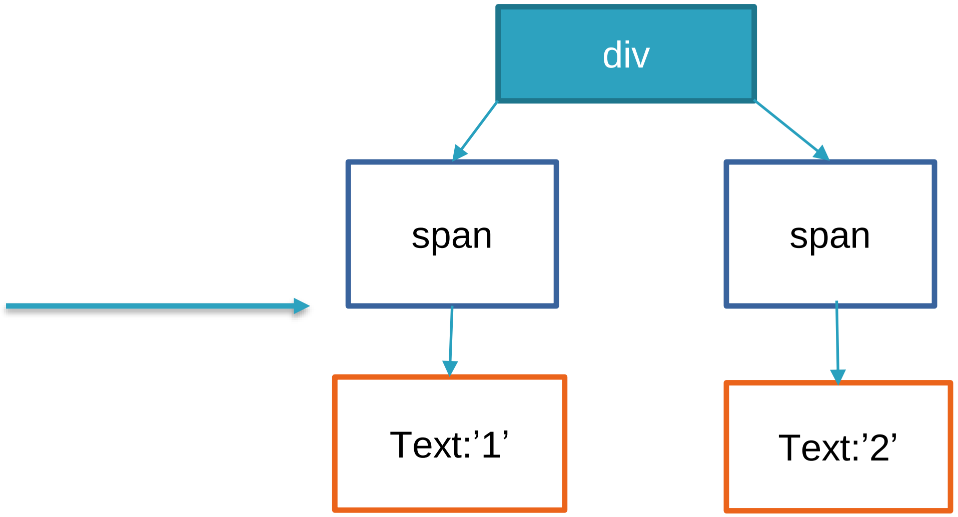
It is created by the browser, when it renders the HTML page.

The DOM is a JS representation of our HTML.

All the HTML elements are represented as objects.

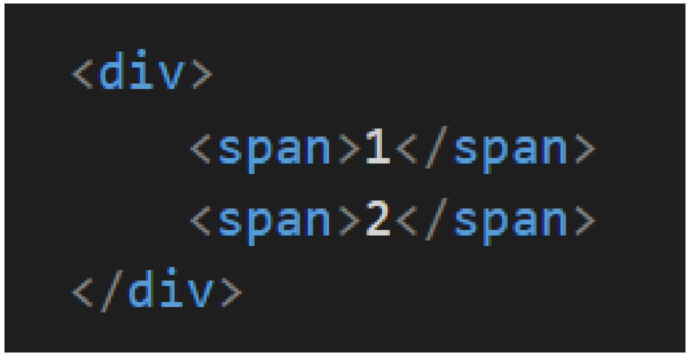


```
<div>
    <span>1</span>
    <span>2</span>
</div>
```

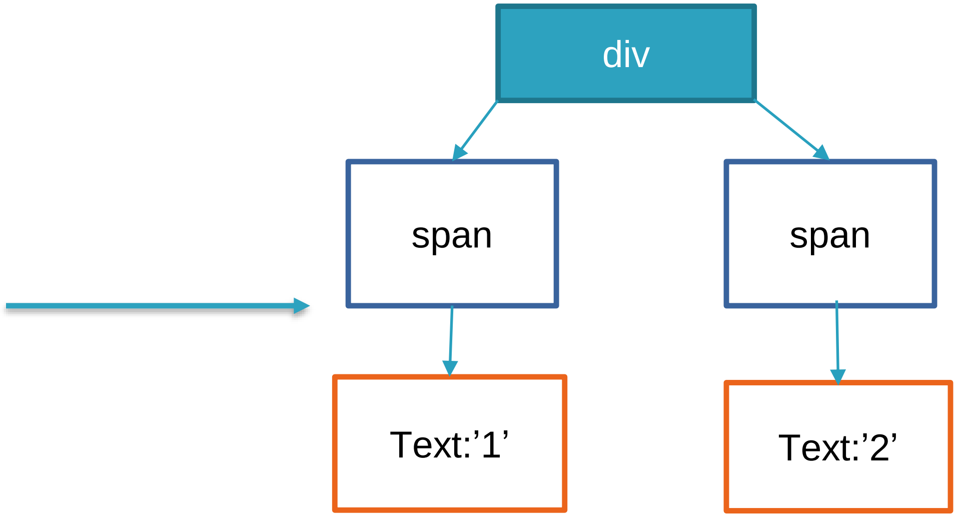


What is the DOM?

Even the text inside the an HTML tag is a node in the DOM tree.

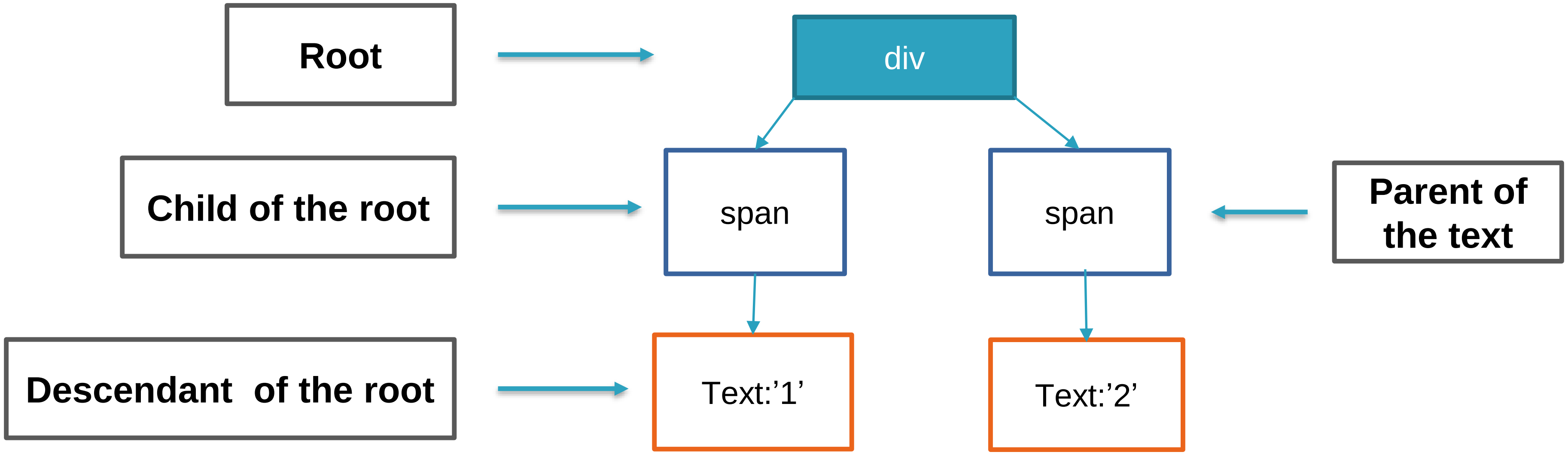


```
<div>
    <span>1</span>
    <span>2</span>
</div>
```



What is the DOM?

Semantics- Hierarchy and Relations



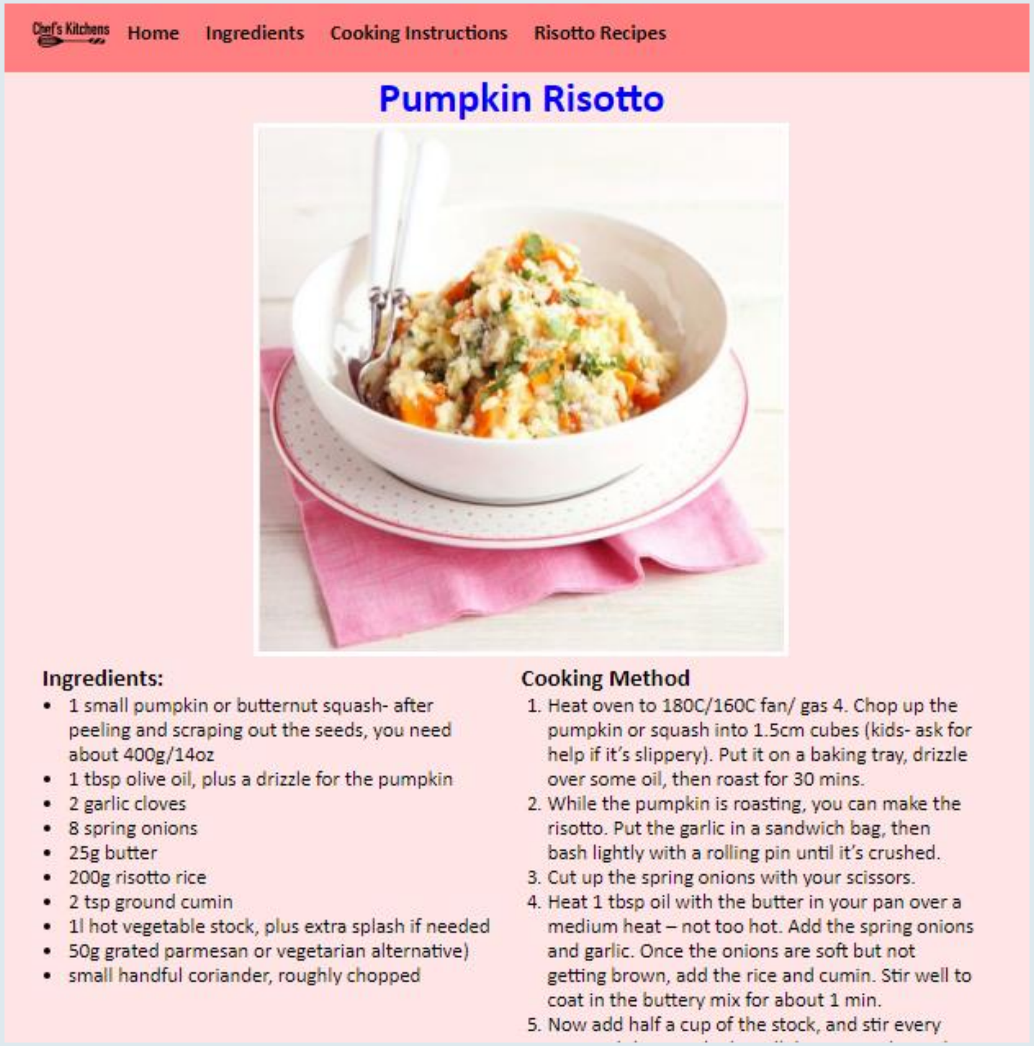
Questions?

```
console.log("Questions?");
```

practice

Your Task:

Draw the DOM of a site.



Pumpkin Risotto

Ingredients:

- 1 small pumpkin or butternut squash- after peeling and scraping out the seeds, you need about 400g/14oz
- 1 tbsp olive oil, plus a drizzle for the pumpkin
- 2 garlic cloves
- 8 spring onions
- 25g butter
- 200g risotto rice
- 2 tsp ground cumin
- 1l hot vegetable stock, plus extra splash if needed
- 50g grated parmesan or vegetarian alternative)
- small handful coriander, roughly chopped

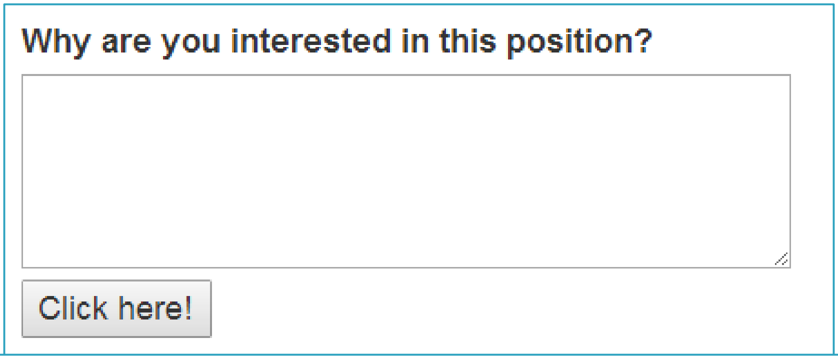
Cooking Method

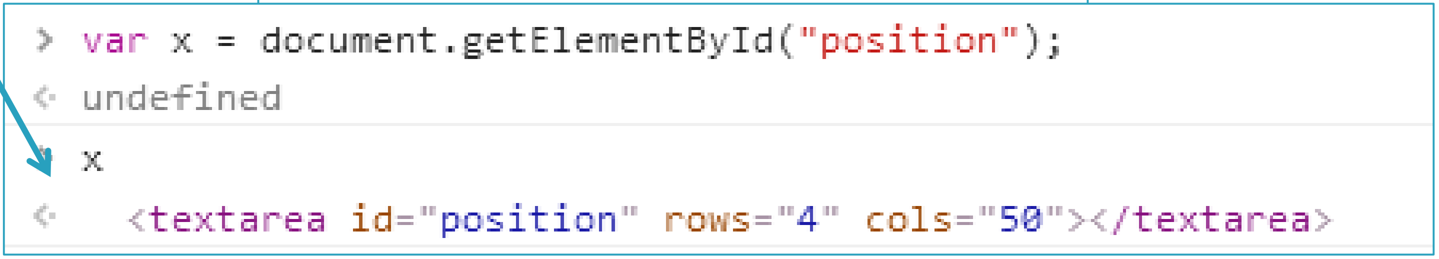
1. Heat oven to 180C/160C fan/ gas 4. Chop up the pumpkin or squash into 1.5cm cubes (kids- ask for help if it's slippery). Put it on a baking tray, drizzle over some oil, then roast for 30 mins.
2. While the pumpkin is roasting, you can make the risotto. Put the garlic in a sandwich bag, then bash lightly with a rolling pin until it's crushed.
3. Cut up the spring onions with your scissors.
4. Heat 1 tbsp oil with the butter in your pan over a medium heat – not too hot. Add the spring onions and garlic. Once the onions are soft but not getting brown, add the rice and cumin. Stir well to coat in the buttery mix for about 1 min.
5. Now add half a cup of the stock, and stir every

getElementByID

When querying for an element, we are, in fact – getting an HTML element from the DOM.

X is a DOM node





```
> var x = document.getElementById("position");
< undefined
> x
<   <textarea id="position" rows="4" cols="50"></textarea>
```

If no such element exists, we will get null in return.

DOM Node Object

Each HTML element is represented in the DOM as a JS object.

It has a lot of properties.

For example, all the html attributes are properties

This a tag

```
<a id="google-link" href="http://google.com" target="_blank" >Google</a>
```

Will have id, href and target as properties:

```
var domNode = document.getElementById("google-link");

domNode.id; // google-link
domNode.href; // http://google.com
domNode.target; // _blank
```

DOM Node Object

It also has a lot of functions.

For example, the same a tag (and any other DOM node)

```
<a id="google-link" href="http://google.com" target="_blank" >Google</a>
```

Will have :

```
domNode.cloneNode();
domNode.hasChildNodes();
```

Look up the rest of the functions and properties in the documentation.

DOM Node Object

We can think about how this tag (html tag)

```
<a id="google-link" href="http://google.com" target="_blank" >Google</a>
```

will be created as an object

```
var domNode = {
  id: "google-link",
  href: "http://google.com",
  target: "_blank",
  cloneNode: function(){
    // clone the node
  },
  hasChildNodes: function(){
    return true;
  }
  // and more...
};
```

DOM Node Object

We said that all the html attributes are represented as properties on the DOM node.

Assume we have this html tag:

```html
<div class="box"></div>
```

Will class be an attribute on the DOM node?

ClassName

The class attribute is a special case.

It will be an attribute on the DOM node but the key will be className.

That is to avoid conflicts with the class keyword. (we will learn about it later).

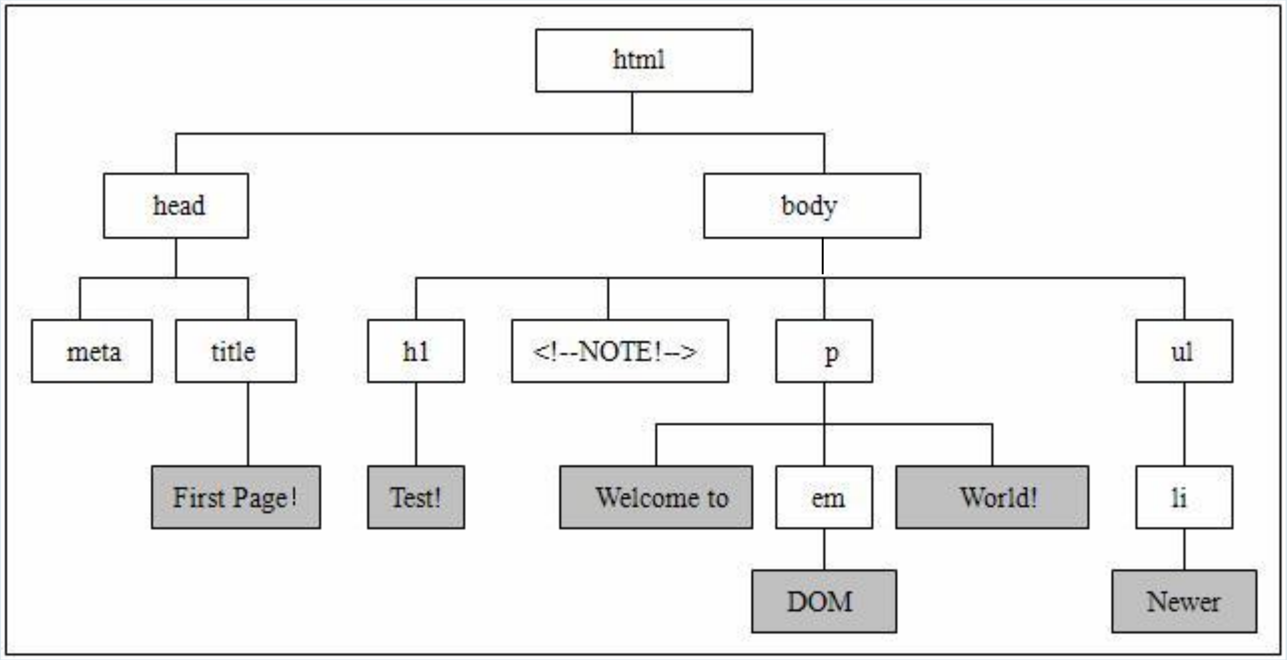
```
For this html element:
<div class="box"></div>

If we retrieve the element from the DOM:
var boxNode = document.getElementsByTagName("div")[0];

We can get the class attribute like this:
boxNode.className; // box
```

Questions?

```
console.log("Questions?");
```

practice

Your Task: Create the html for a site that its DOM is represented by the following tree.



How do we work with it?

We already know how to access the DOM nodes!

```
document.getElementsByClassName("box");
document.getElementById("menu");
document.getElementsByTagName("img");
```

Once we have a DOM node object every change we make to it will change our web page, meaning what the user sees!

Document

Remember `document.getElementById("test");` ?

What is the `document`?

Document

The document is an object that serves as an entry point to the web page's content.

We can directly access the HTML node

```
document.documentElement;
```

```
→ <html>
     <head></head>
     <body>
        <div></div>
     </body>
  </html>
```

And the body node

```
document.body;
```

```
  <html>
     <head></head>
  →  <body>
        <div></div>
     </body>
  </html>
```

Or any other tag:

```
document.getElementById("box");
document.getElementsByTagName("div");
```

```
  <html>
     <head></head>
     <body>
  →     <div id="box"></div>
     </body>
  </html>
```

Document

The document contains other methods and properties.

For Example:

- bgColor
- cookie
- hasFocus

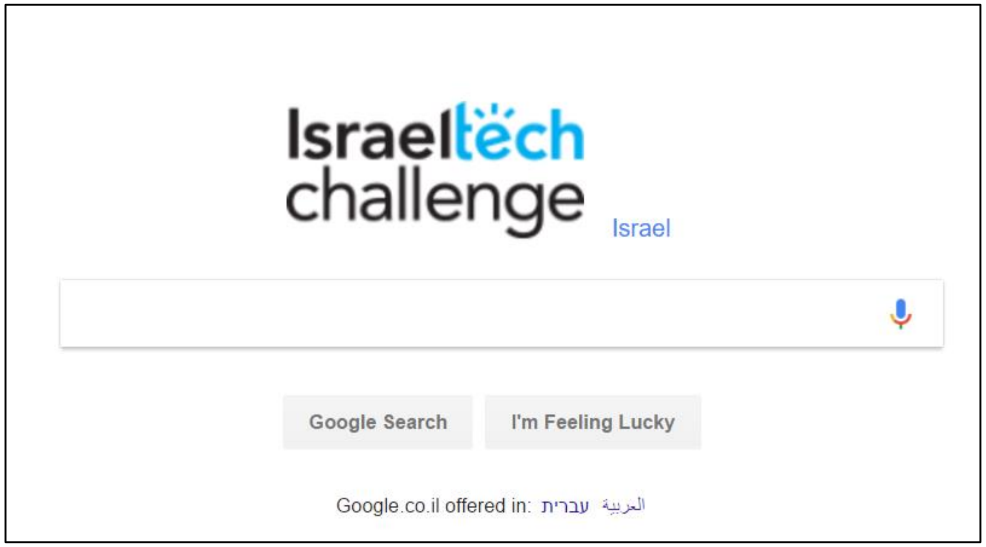
And many more.

DOM vs Static HTML

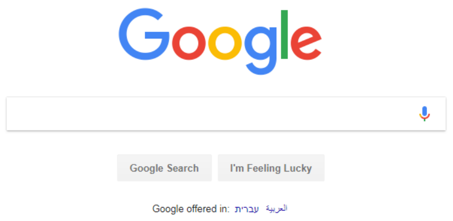
The DOM is a live version of our website, not like the static HTML file.

We can make all sorts of changes to it, that will disappear once the page is reloaded.

Practice – change the google logo on google website using the console!



And refresh



Quick reminder: JS and HTML

HTML + CSS === static content

- everything that is static about our webpage
- Can’t be changed after loading

JS === dynamic content

JavaScript gives us the ability to dynamically modify our pages and interact better with the user.

Changes by the user

Now we can change

Everything!

Get, Add, Change or remove:

- attributes (style, onclick, src…)
- HTML inside an object
- text inside an object
- child elements (an element inside an element)

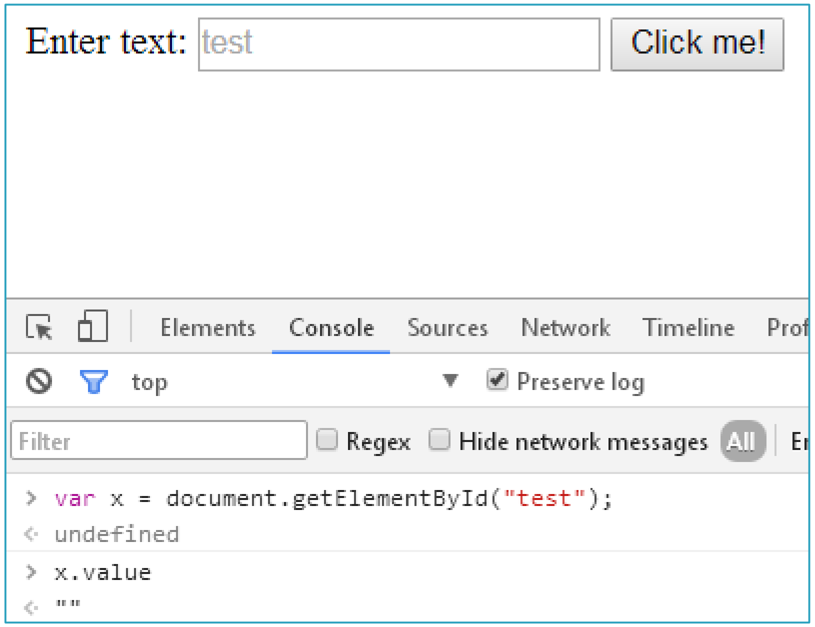
This is called DOM Manipulation

Changes by the user

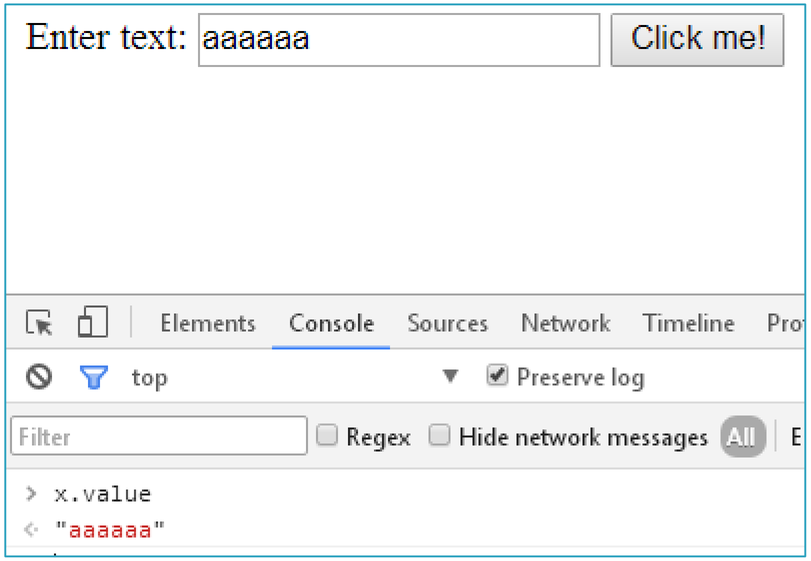
One way for the DOM to change is user interaction

When the user interacts with the DOM => it causes the DOM to change

...If the user typed in text in an input field – the DOM has changed!



```
> var x = document.getElementById("test");
< undefined
> x.value
< ""
```



```
> x.value
< "aaaaaa"
```

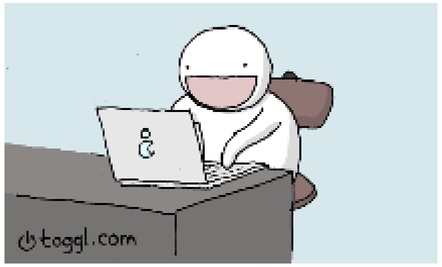
Questions?

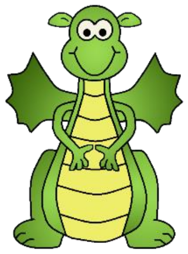
```
console.log("Questions?");
```

Your Story

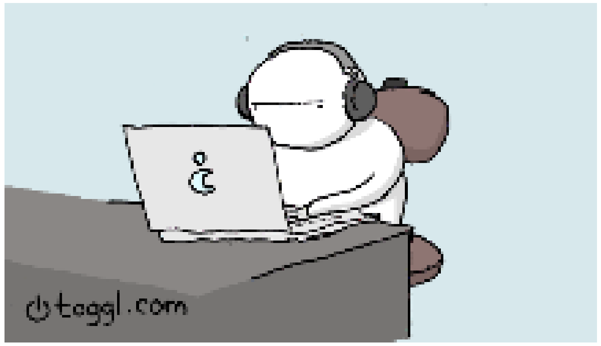
A long time ago, in a magical kingdom far far away…

There were some brave programmers who wanted to create interactive web pages





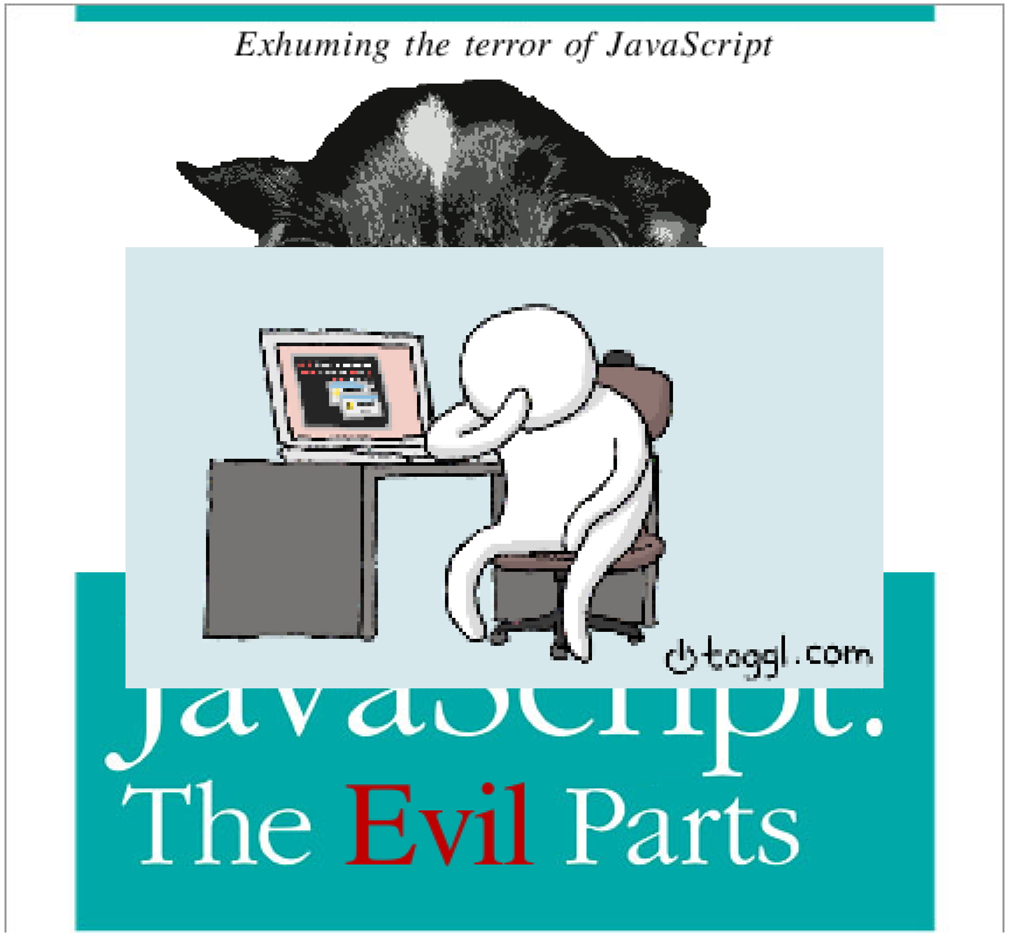
Your Story



The programmers learned to
create HTML pages
and CSS files
He learned to associate classes to style
He learned how to make the page responsive using media queries
and how to use Twitter bootstrap

Your Story

And then they met JavaScript



Taming the beast

Our brave programmers has faced different kinds of beasts before and won, so why should this be any different?

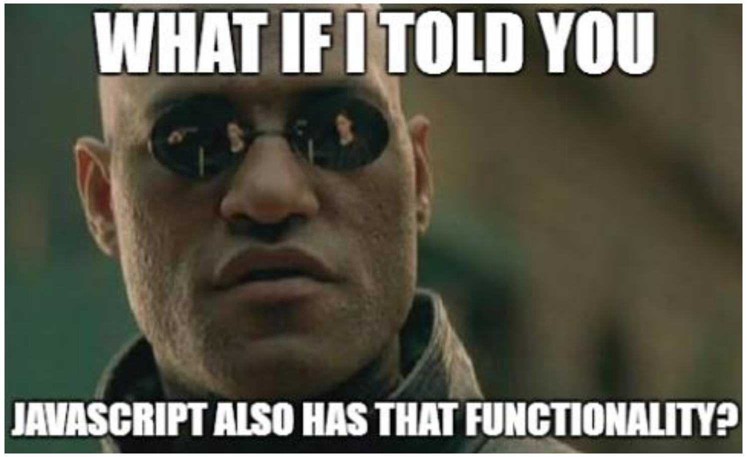
But CSS was so much nicer

CSS had a nice way to apply settings to multiple objects with complicated selectors

```
.menu-links a{
  ...
}
```

But Javascript only has the following
getElementById
getElementsByTagName
getElementsByClassName
getElementsByName

Right…?



< itc >

Query Selector

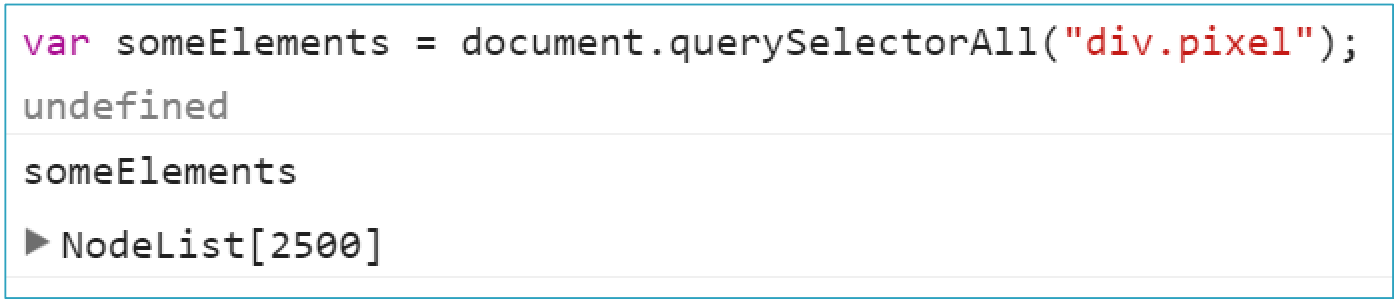
JS too has more general methods for getting elements:

```
var elementList = document.querySelectorAll(selectors);
```

- Returns a list of the elements within the document that match the selectors pattern
  (using depth-first pre-order traversal of the document's nodes)

Query Selector

```
var elementList = document.querySelectorAll(selectors);
```



```
var someElements = document.querySelectorAll("div.pixel");
undefined
someElements
▶ NodeList[2500]
```

In other words

- This function uses the CSS selectors to get the relevant elements from the DOM
- We can use the same selectors that we used in CSS, including pseudo classes!
- The result is a collection of HTML elements

Query Selector

```
var element = document.querySelector("selector");
```

Example: `document.querySelector("div>.dark");`

- Returns the first Element within the document that matches the specified selector, or null if no matches are found.
- also can be achieved with:

```
var firstElement = document.querySelectorAll("div.pixel")[0];
```

Questions?

```
console.log("Questions?");
```

Changing Elements Review

- We can access HTML elements as objects that have many properties and functions for us to use
- Anything we can specify in HTML **can be done via JS**: Add, remove or change any attribute, text or HTML code
- This is called **DOM manipulation**:

```
var testMyName = function () {
    var fname = document.getElementById("firstName");

    if(fname.value==="Dana"){
        fname.value = "Nice name";
        fname.style.color = "green";
    } else {
        fname.value = "too common";
        fname.style.color = "red";
    }
};
```

Attributes in HTML elements

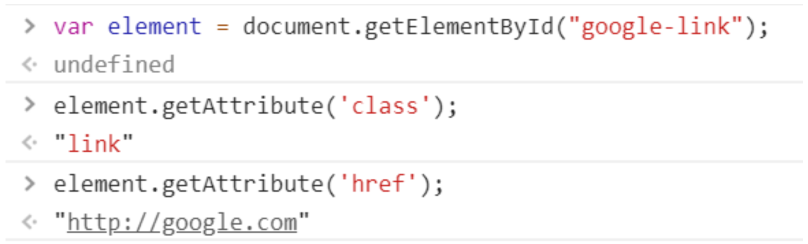
Get

HTML
```
<a id="google-link" href="http://google.com" class="link" >Google</a>
```
Js
```
var element = document.getElementById("google-link");
```

We now obtained easy access to every element we want in the DOM we can read every attribute using :

```
element.getAttribute('attributeName');
```

```
element.getAttribute('class');
element.getAttribute('href');
```



```
> var element = document.getElementById("google-link");
<· undefined
> element.getAttribute('class');
<· "link"
> element.getAttribute('href');
<· "http://google.com"
```

- If the attribute doesn’t exist, we’ll get **Null**

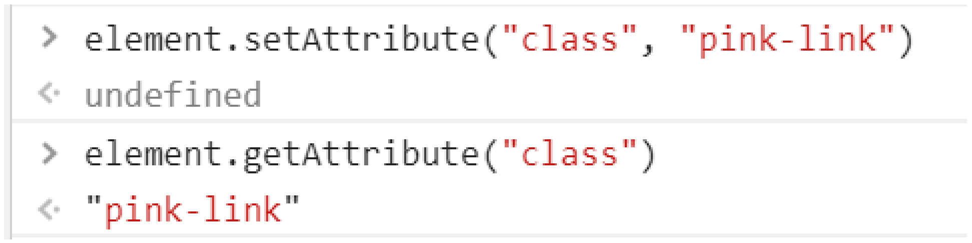
Attributes in HTML elements

Set

We can also set the value to every attribute, using

```
element.setAttribute('attributeName','value');
```

- They don't even have to be standard HTML attributes!



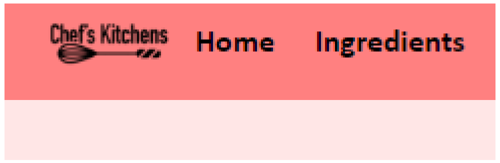
```
> element.setAttribute("class", "pink-link")
< undefined
> element.getAttribute("class")
< "pink-link"
```

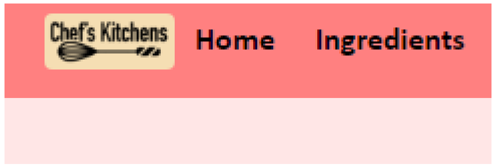
Questions?

```
console.log("Questions?");
```

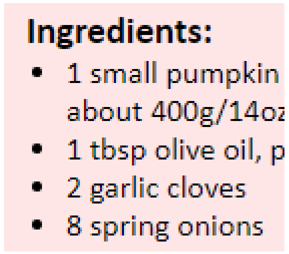
Practice

Use the document.querySelector function to change the logo background color to wheat.
Note: You cannot use the logo id.



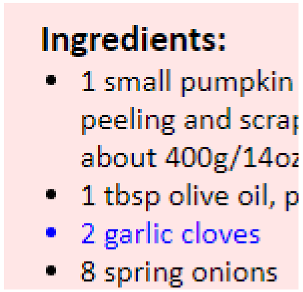


- Use the document.querySelector function and setAttribute to change the logo background color to wheat.
- Use document.querySelectorAll function to change the color of the garlic ingredient to blue.



Ingredients:

- 1 small pumpkin
  about 400g/14oz
- 1 tbsp olive oil, p
- 2 garlic cloves
- 8 spring onions



Ingredients:

- 1 small pumpkin
  peeling and scrap
  about 400g/14oz
- 1 tbsp olive oil, p
- 2 garlic cloves
- 8 spring onions

Break;

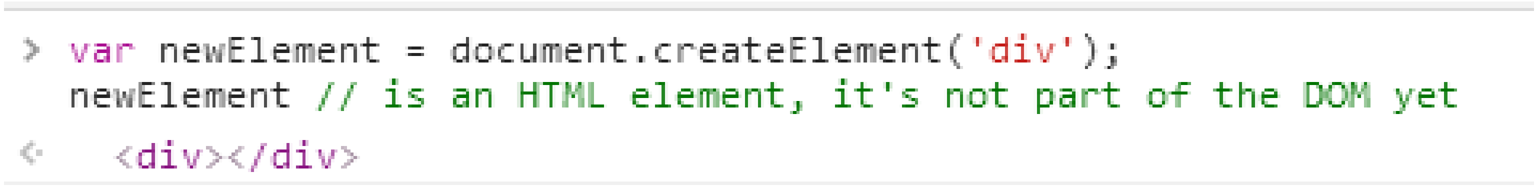
But wait

What else
can we change?

Creating HTML elements

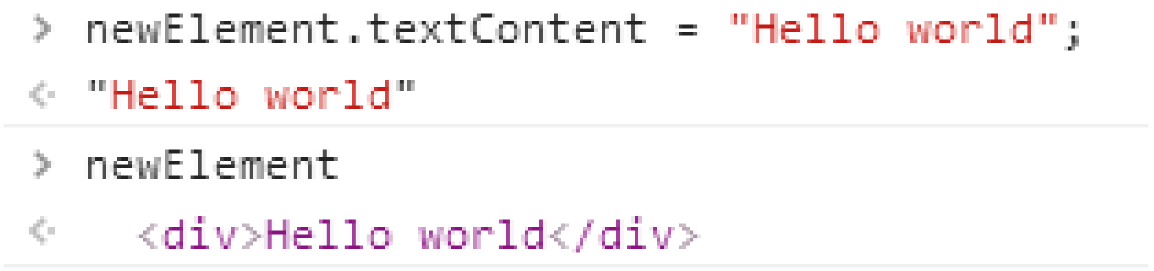
- We can create a new element by asking the document to do it for us.

```
var newElement = document.createElement('div');
newElement // is an HTML element, it's not part of the DOM yet
```



```
> var newElement = document.createElement('div');
  newElement // is an HTML element, it's not part of the DOM yet
<   <div></div>
```

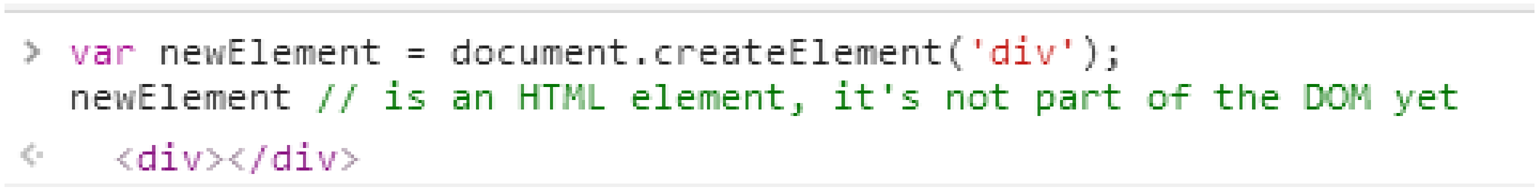
- **Change the element**
  The new element can be changed via JS just like any other element we have in the document.



```
> newElement.textContent = "Hello world";
< "Hello world"
> newElement
<   <div>Hello world</div>
```

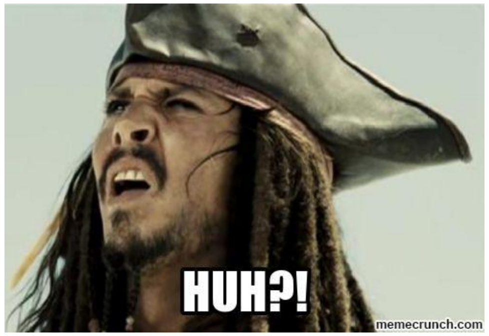
Creating HTML elements

```
var newElement = document.createElement('div');
newElement // is an HTML element, it's not part of the DOM yet
```



```
> var newElement = document.createElement('div');
  newElement // is an HTML element, it's not part of the DOM yet
<    <div></div>
```

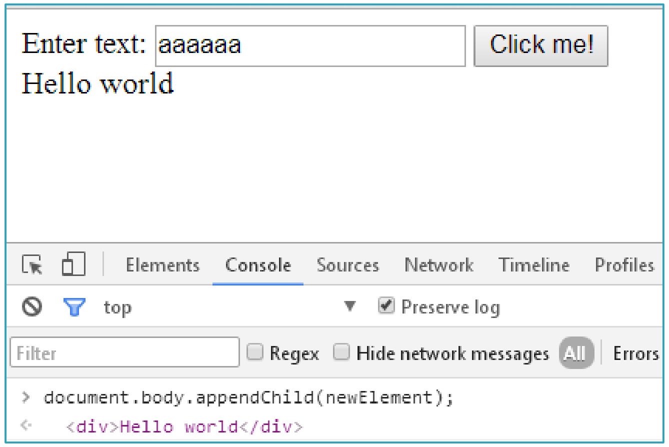
- However, this element is not a part of the DOM.
- It has no parent, so you will never see it on the page. But it does exists!



Appending

- To make an element a part of the page, we need to append it to another element that is already in the DOM.
- We can append it to document.body or to any other element that we know exists, and have a reference to.

```
document.body.appendChild(newElement);
//or
var elem = document.getElementById('someId');
elem.appendChild(newElement);
```



Hello world

```
> document.body.appendChild(newElement);
<   <div>Hello world</div>
```

appendChild function will add it as the **last child** of the parent

Rule #1

Can we append a new element more than once?

- When creating a new element, you can append it many times, to different parent elements.
- This will not create new elements, but only move the one you have created.

Remove an element

- Remove function
  If we have a reference to an element that we want to remove, all we have to do is call the element's remove function.

```
newElement.remove();
```

- Does the element exist?

- Just like before: the element will still exist, but it's just not a part of the DOM anymore.

Q & A

- What would happen if we create two elements, and append one to the other?

  The two elements are now connected, but you will not see any of them.

- What would happen if we then append the parent element?

  you will see both.

- What would happen if instead we'll append the child element?

  you will only see the child.

Rule #2

- js script at the **bottom of the body**: Always include the script used to manipulate the DOM at the bottom of your body.
- Why? Because we can't change elements that don't exist yet

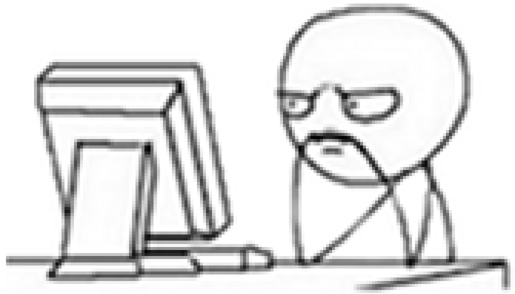
Questions?

```
console.log("Questions?");
```

Responding to events – the next step

We have learned that we can react to events by defining a specific reaction to a specific event in HTML (onclick, onfocus etc…)

But surely,
there must be another way
of handling events,
without having to write
JS inside the HTML



Introducing: event listeners

Now we can do this with JS!

We can create functions that will listen to events like click, and… handle them!

Create a listener function

The first step is to create a listener function, that will act according to the specific event. So far, nothing different.

```
function changeColorToGreen() {
    var buttonElment = document.getElementById("change-color");
    buttonElment.style.color = 'green';
}
```

Add an event listener

We now have a function that reacts to clicking, but we need to connect the clicking to the relevant HTML elements.

In HTML we did

```
<button id="change-color" onclick="change-color()">Change Color</button>
```

In JS, first we need a reference to the element:

```
var buttonElment = document.getElementById("change-color");
```

Then we use a function from the DOM api to connect the listener function:

```
buttonElment.addEventListener('click', changeColorToGreen);
```

Here we've added a click event listener to an element with id “change-color”

When that element is clicked, the `changeColorToGreen` function will run.

Add an event listener

Note: we have different syntax for the name of the event:

```
buttonElment.addEventListener('click', changeColorToGreen);
```

we do not use the html attribute name!!

```
buttonElment.addEventListener('onclick', changeColorToGreen);
```

Here is a list of the event types.

Questions?

```
console.log("Questions?");
```

Add an event listener

Note: we pass the function object!!

```
buttonElment.addEventListener('click', changeColorToGreen);
```

we do not invoke the function!!

```
buttonElment.addEventListener('click', changeColorToGreen());
```

Another Example Breakdown

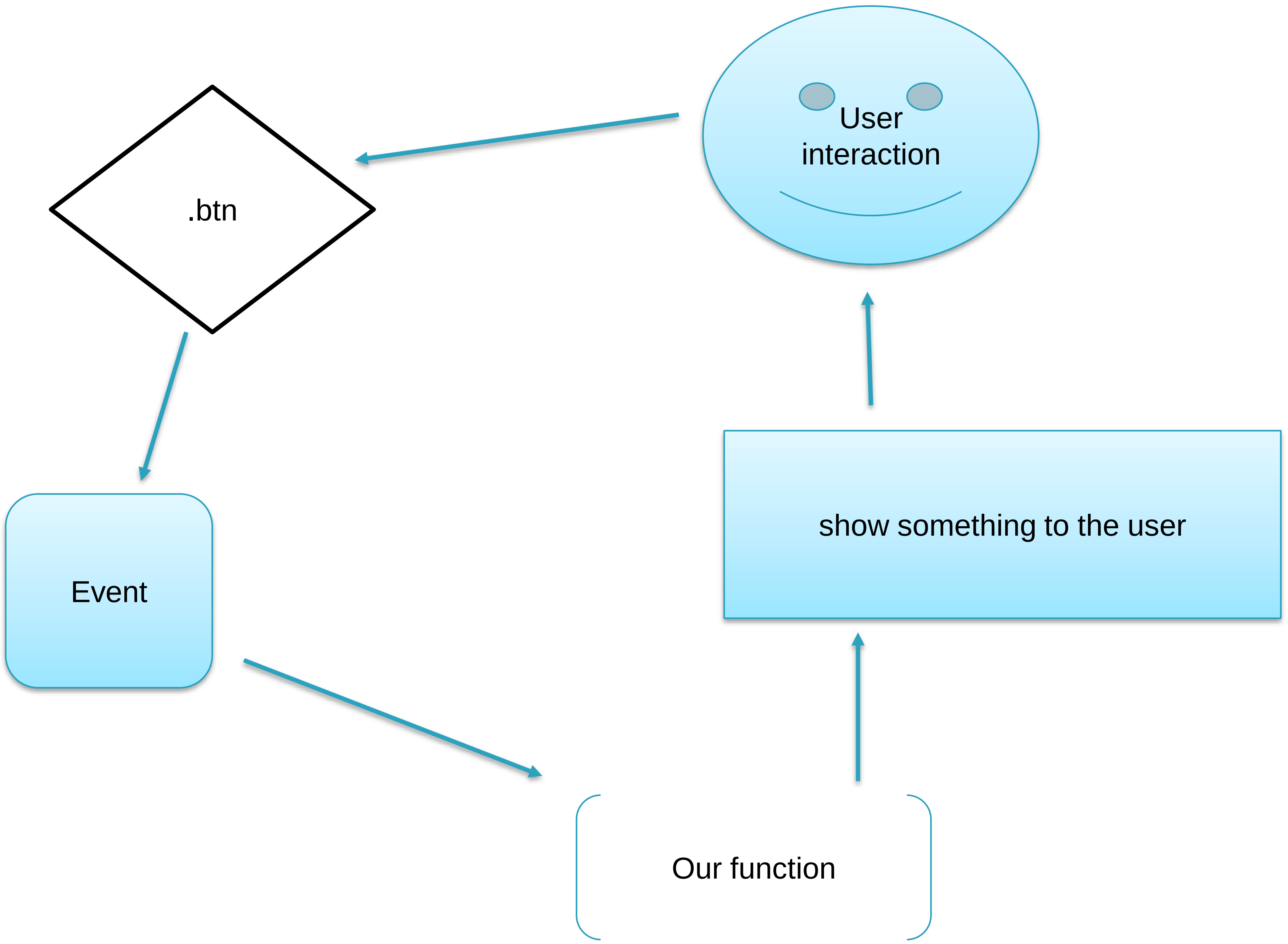
```
var buttonElements = document.getElementsByClassName('btn');
var firstButtonElement = buttonElements[0];
firstButtonElement.addEventListener('click', changeColorToGreen);
```

1. We query for all the elements with the class ‘btn’
2. We only choose the first
3. Assign an eventListener:

   We use one of the elements functions to say ‘whenever the event click happens, call this function’. When the user clicks on the element, the function will be called
4. Click event will trigger the function

   The function code will be executed

Event Handlers Flow



Event Parameter

```
function changeColorToGreen(eventObject) {

}
```

The listener function is expecting a parameter

The parameter is of type object and represents the event

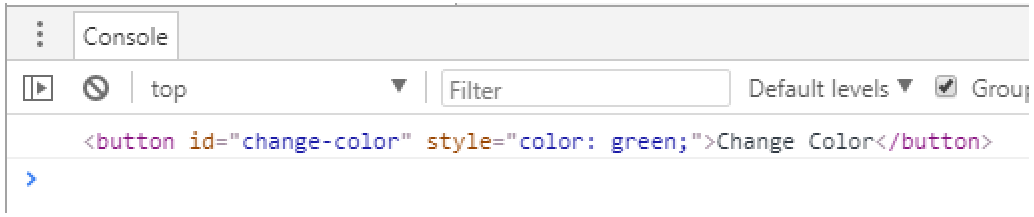
(clickEvent in our example)

target

```
function changeColorToGreen(eventObject) {
    var buttonElment = eventObject.target;
    console.log(buttonElment);
    buttonElment.style.color = 'green';
}
```

One important attribute of the event parameter is the target.

The target is an object representing an HTML element.



```
<button id="change-color" style="color: green;">Change Color</button>
```

In fact, this is the HTML element we clicked on.

Just like if we executed `document.getElementById("change-color");`

We can use the target element to get its value or change it.

Practice

- Add an click eventListener, that will change the color of the title text to gray.

Questions?

```
console.log("Questions?");
```

Other events

List of events we can listen to includes:

- Clicks
- Double clicks
- Keyboard events: user hits a key, user lets go of a key
- User scrolls with the mouse
- User right clicks

We can also listen and respond to events not coming from the user

- The page finishes loading (that is an event!)
  - `window.addEventListener("load", () => {})`
- And more…

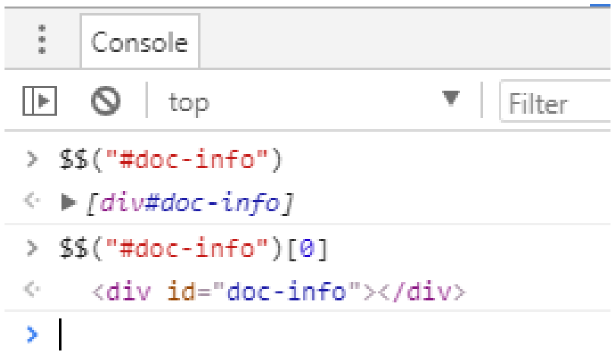
Questions?

```
document.addEventListener("studentQuestion",answerQuestion);
```

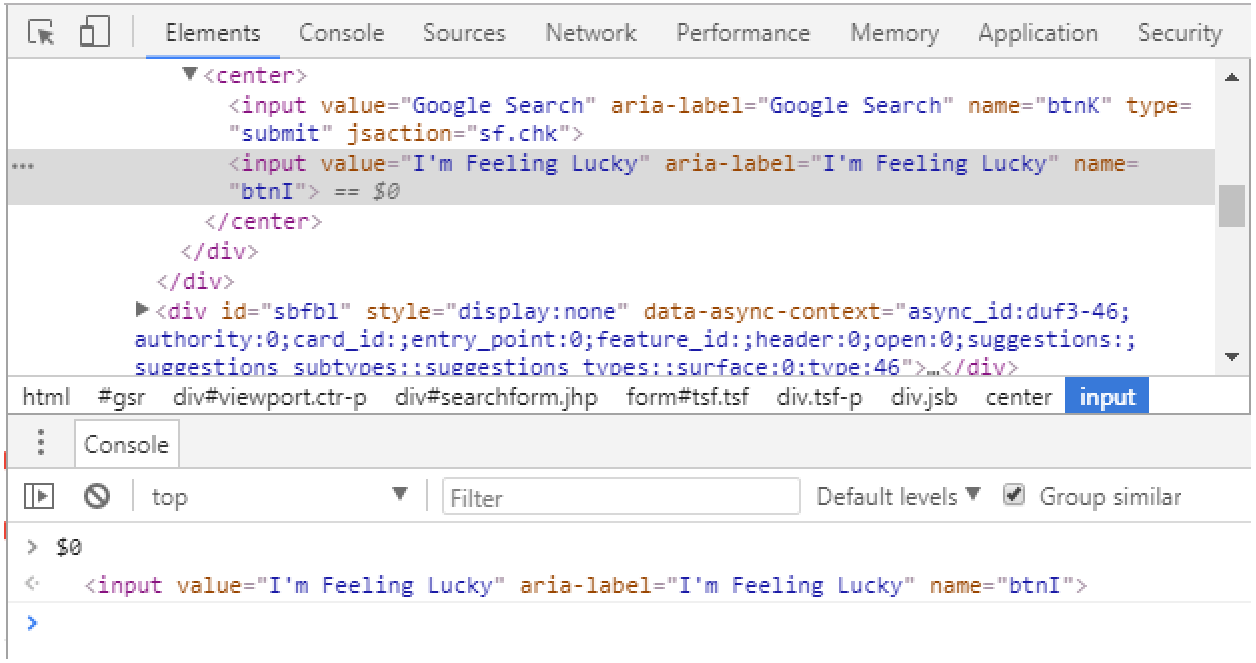
Query the DOM with Devtools

Using the dev tools:

- ➢$$ - equivalent to querySelectorAll
- ➢$0 – the last selected element



```
> $$("#doc-info")
<· ▶ [div#doc-info]
> $$("#doc-info")[0]
<·   <div id="doc-info"></div>
> |
```



```
▼<center>
   <input value="Google Search" aria-label="Google Search" name="btnK" type=
   "submit" jsaction="sf.chk">
   <input value="I'm Feeling Lucky" aria-label="I'm Feeling Lucky" name=
   "btnI"> == $0
 </center>
</div>
</div>
▶<div id="sbfbl" style="display:none" data-async-context="async_id:duf3-46;
authority:0;card_id:;entry_point:0;feature_id:;header:0;open:0;suggestions:;
suggestions_subtypes::suggestions_types::surface:0;type:46">…</div>
```

html #gsr div#viewport.ctr-p div#searchform.jhp form#tsf.tsf div.tsf-p div.jsb center input

```
> $0
<·  <input value="I'm Feeling Lucky" aria-label="I'm Feeling Lucky" name="btnI">
>
```

Change properties

How can we change the text color of an element?

```
var buttonElment = document.getElementById("change-color");
buttonElment.style.color = "green";
```

What if we also want to change the background color?

```
buttonElment.style.background = "red";
```

What happens if we have 10 properties we want to change?

We need to remember that any change that we do to a property is done to the inline style.

Change properties

How can we use the static CSS?

We can create css rules:

```css
.btn {
   position: fixed;
   left: 5px;
   top: 5px;
}
```

And change the class dynamically (add, edit, remove):

```javascript
var buttonElment = document.getElementById("change-color");
buttonElment.className = "btn";
```

Quick reminder: Global variables

- They exist outside the scope of a function.
- They are accessible throughout our JS code.
- We can change them in any point in time.

```
var test = "This is saved in a global var";
var funcName = function(){
    console.log(test);
    return "this is a function";
};
```

Document

Remember `alert("Hello!")` ?

What is it? Where it is coming from?

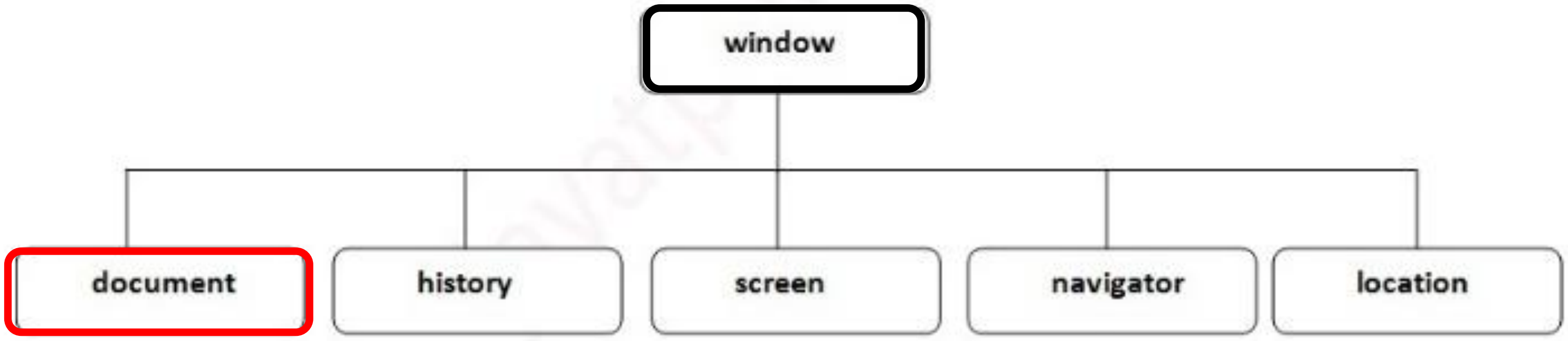
Browser Object Model

The browser provides us a set of properties and functions, which we can use to interact with the browser.

For example: `alert, prompt, confirm.`

Other properties that it has are:
They are all located under a global variable called **window**.

We already know the document

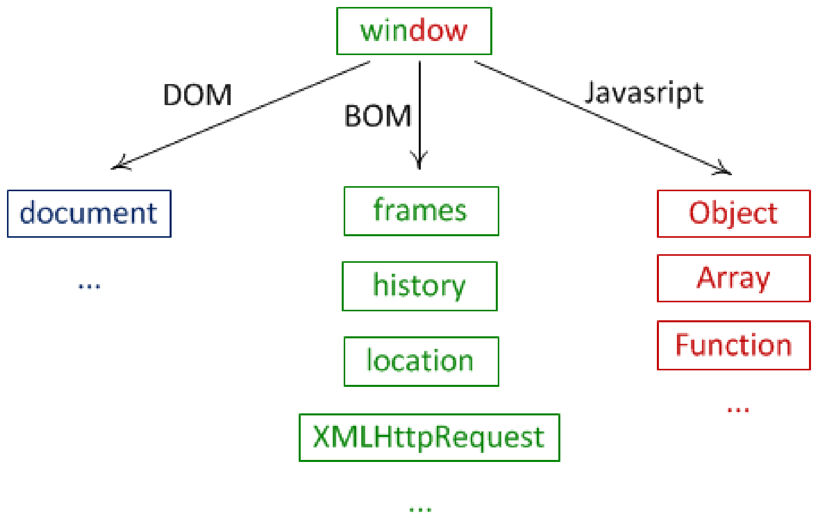


Intro the DOM

Browser Environment

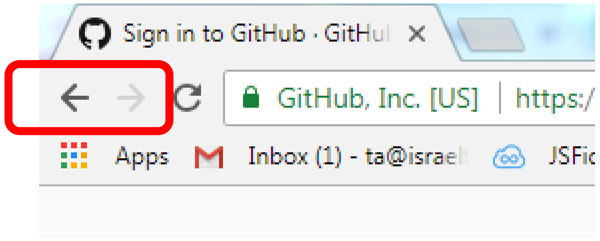
The window is a global object

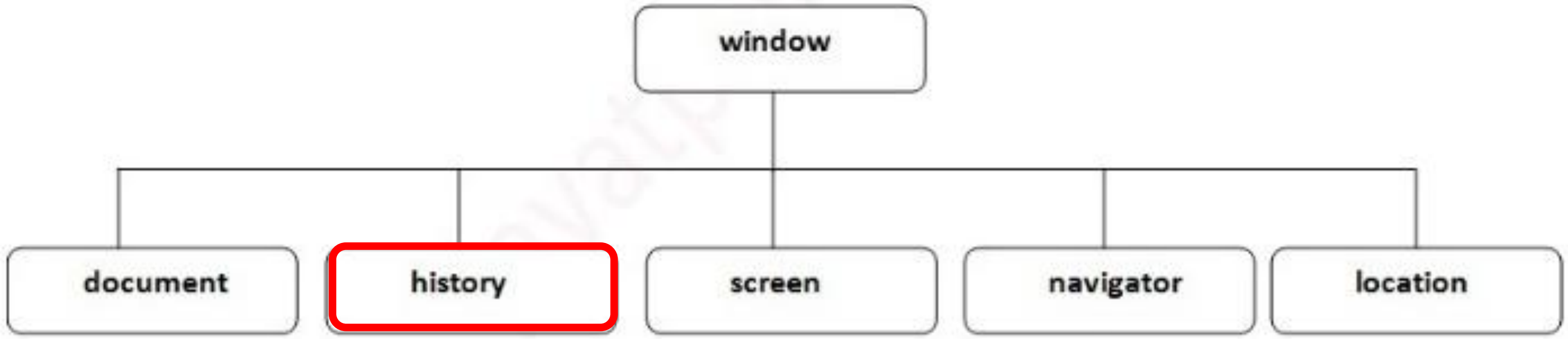
It has browser related functionalities
Functions such as:
focus(),
open() etc.



Browser Object Model - Examples

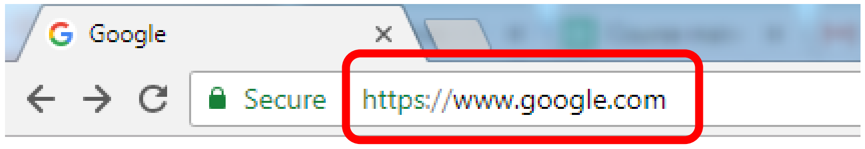
History is an object that handles the browsing history

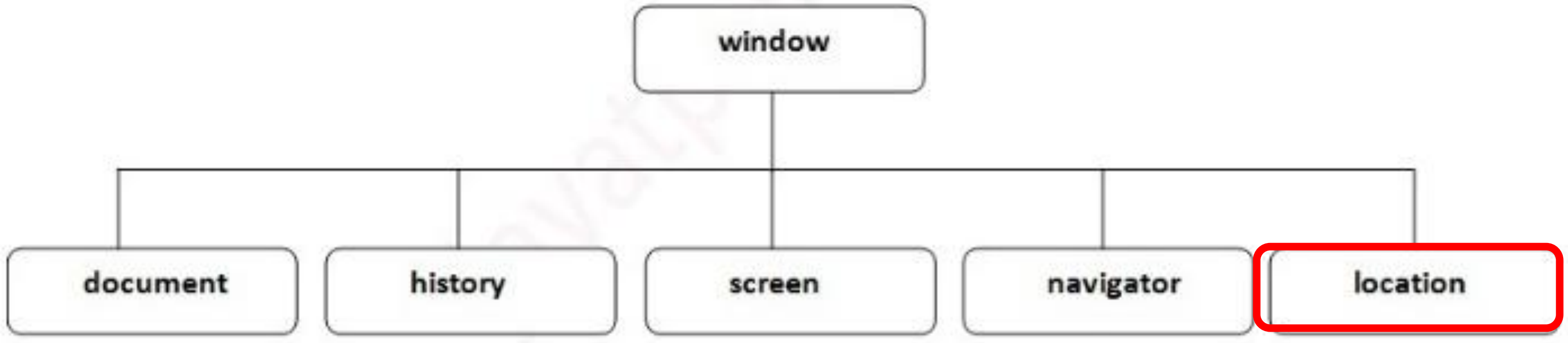




Browser Object Model

Location is an object that handles the url





Questions?

```
console.log("Questions?");
```

Questions?

```
console.log("Questions?");
```

Cheat Sheet

Get + set attribute

```
someElements.getAttribute('attributeName');
someElements.getAttribute('class');
someElements.setAttribute('attrName','value');
someElements.setAttribute('class', 'big-button');
```

Create new element

```
var newElement = document.createElement('div');
```

Append child

```
containerElement.appendChild(newElement);
```

Event Listeners

```
element.addEventListener('event', listenerFunc);
element.addEventListener('click', changeColor);

function changeColorToGreen(eventObject){
  //target = the element that received the event
  var targetElement = eventObject.target;
  targetElement.style.color = 'green';
}
```

Query Functions

```
document.querySelectorAll(".col");
document.querySelector("div");
```

Devtools

```
$$ - querySelectorAll

$0 - last selected element
```

Page load event

```
window.addEventListener("load", ()
=> {});
```

Inner HTML

```
document.getElementById("first").innerHTML = "<span>one</span>“;
```