IT School
FROM ZERO TO HERO!

JS: Objects&the DOM

Sesiunea 21 - 05/07/2021

/Recap\

JS: current status

- Types: internal, external
- Variables&Constants
- Types&Operators
  - Numbers
  - Strings
  - Conversions
  - Shortcuts&special operators
  - Booleans
  - null&undefined
  - Intro to **objects**
- Program Flow:
  - if
    - if..else
    - ternary
  - switch
  - loops
    - for
    - while
    - do..while
- Functions
- Scope

IT School
FROM ZERO TO HERO!

/Objects&the DOM\

Objects and the DOM: coverage

- Object properties (~variables) and methods (~functions)
- Standard built-in objects
- The DOM (Document Object Model)
- Styling DOM elements
- Event listeners (detecting button clicks)
- Showing/hiding DOM elements (CSS classes come into action)

Object: properties

- Thinking about real-life objects, like a car, some of its props might be:
  - Number of doors
  - Fuel type
  - Engine size
- Consider an object as a group of values/properties
- Object creation in JS: `let person = {};`
- An object's props can have any data type, they do not have to have the same one
- Objects as props to objects is also common

Object: properties

```
let person = {
    name: "John",
};
```

prop's name

prop's value

/Qbjects: more prop data types\

Object: properties

```
// Object creation
let person = {
    name: 'Roxana', //String data type
    age: 32, //Number data type
    partTime: false, //Boolean data type
    skills: {
        html: 5, //Number data type, used as a rating system
        css: 4, //in the current context
        js: 3
    }
};
```

Object: properties

- Accessing the props:
  - Dot notation: `person.name`
  - Square bracket notation: `person['name']`
- Accessing a prop that doesn't exist => **undefined** value
- Assignment: `person.age = 33; person['age'] = 33;`

```
// Object creation
let person = {
    name: 'Roxana', //String data type
    age: 32, //Number data type
    partTime: false, //Boolean data type
    skills: {
        html: 5, //Number data type, used as a rating system
        css: 4, //in the current context
        js: 3
    }
};
```

IT School FROM ZERO TO HERO!

Object: methods

- Methods are functions attached to objects
- Accessing own properties -> using **this** keyword
- **this** will refer to the current object
- We can also pass parameters to these methods

```
let person = {
    name: 'Roxana',
    age: 33,
    partTime: false,
    showInfo: function() {
        replaceMainTitle('Other Title 2');
    }
};

person.showInfo();
```

Standard built-in objects

- JS supports a great number of built-in objects that extend its' flexibility
- Each one of these are used in an unique way
- Unveils specific methods for various data types that we can use
- **new** keyword come into the picture
- Specific typeof: **Object, Number, String**
- built-in objects = global objects
  - Date
  - Math
  - String
  - Number
  - Array
  - Number
  - Object

Standard built-in objects: **Date**

- It's extremely common the necessity of working with dates (and tricky)
- JS Date objects represent a single moment in time
- We have a lot of methods that we can use on dates
- Basically, whenever we have a Date object we can access anything on the prototype
- A few possibilities:
  - Retrieve current date
  - Retrieve minutes from date
  - Retrieve time section from date
  - Retrieve timezone
  - Convert to string
- **moment.js**: friendly js library for working with dates

Methods

```
Date.UTC()
Date.now()
Date.parse()
Date.prototype.getDate()
Date.prototype.getDay()
Date.prototype.getFullYear()
Date.prototype.getHours()
Date.prototype.getMilliseconds()
Date.prototype.getMinutes()
Date.prototype.getMonth()
Date.prototype.getSeconds()
Date.prototype.getTime()
Date.prototype.getTimezoneOffset()
Date.prototype.getUTCDate()
Date.prototype.getUTCDay()
Date.prototype.getUTCFullYear()
Date.prototype.getUTCHours()
Date.prototype.getUTCMilliseconds(
)
```

Standard built-in objects: **Math**

- Built-in math functions:
  - random: creating random numbers
  - round: rounds a number to the nearest integer value
  - abs: absolute value
  - floor
  - ceiling
  - min, max
  - pow
  - sqrt: square root formula
  - toFixed:
- Methods usually expect an input parameter
- Built-in math constants (properties): PI, E, LN10
- Output **typeof**: Number

Properties

```
Math.E
Math.LN10
Math.LN2
Math.LOG10E
Math.LOG2E
Math.PI
Math.SQRT1_2
Math.SQRT2
```

Methods

```
Math.abs()
Math.acos()
Math.acosh()
Math.asin()
Math.asinh()
Math.atan()
Math.atan2()
```

Standard built-in objects: **String**

- Already used the **length** property of this object
- Positioning starts at 0 in a string, NOT at 1
- Useful methods:
  - charAt: the character at a specific index
  - indexOf: returns the index of a substring occurence inside a another string (or -1 if lacking)
  - replace
  - trim: removes whitespaces from both ends
  - toUpperCase
  - toLowerCase

```
String.prototype.toString()
String.prototype.toUpperCase()
String.prototype.trim()
String.prototype.trimEnd()
String.prototype.trimStart()
String.prototype.valueOf()
String.prototype[@@iterator]()
String.raw()
```

Standard built-in objects: **Number**

- Properties: NaN
- Methods
  - parseFloat, parseInt
  - toString
  - toFixed: round the number to a fixed number and decimal places and convert it to string

Properties

```
Number.EPSILON
Number.MAX_SAFE_INTEGER
Number.MAX_VALUE
Number.MIN_SAFE_INTEGER
Number.MIN_VALUE
Number.NEGATIVE_INFINITY
Number.NaN
Number.POSITIVE_INFINITY
```

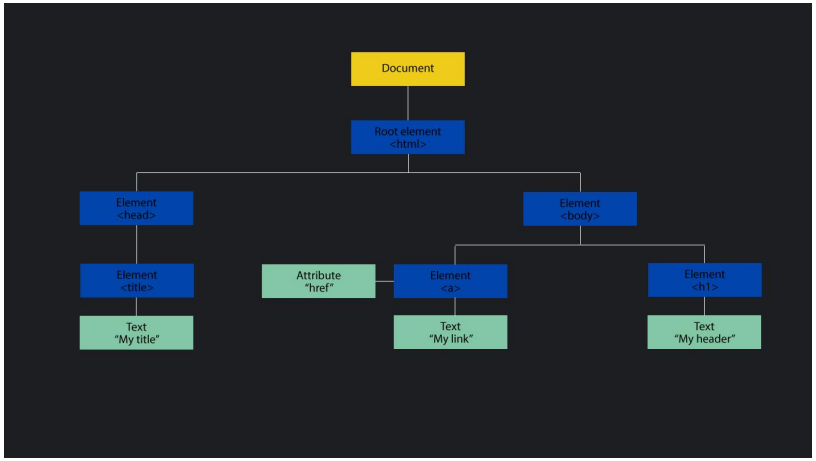
Methods

```
Number.isFinite()
Number.isInteger()
Number.isNaN()
Number.isSafeInteger()
Number.parseFloat()
Number.parseInt()
Number.prototype.toExponential()
Number.prototype.toFixed()
Number.prototype.toLocaleString()
Number.prototype.toPrecision()
```

/DOM -> object\

DOM: an abstraction and in-memory representation of a structured text

```html
<!DOCTYPE html>
    <html>
        <head>
            <title>My title</title>
        </head>
        <body>
            <a href="https://www.google.com/">My link</a>
            <h1>My header</h1>
        </body>
    </html>
</html>
```



DOM: dev tools

Elements Console »

```
<!DOCTYPE html>
<html>
 ▶<head>…</head>
 ▼<body>
   ▼<div class="container">
     ▼<div class="main-info-wrapper">
       ▶<section class="main-bg-wrapper">…</section>
       ▼<section class="info-wrapper">
         ▶<div class="inner-wrapper">…</div>
         </section>
       </div>
     ▶<section class="review-wrapper hidden">…</section>
     </div>
     <script src="resources/utils.js"></script>
     <script src="resources/main4.js"></script>
   </body>
…</html> == $0
```

Objects and the DOM: overview

- An object can contain multiple properties or multiple values that are related
- When we want to modify the webpage in JS the key **object** we work with is called **document**
- The DOM refers to that **document object** along with all of its supporting objects and other features
- Gateway to programming web pages through the DOM

/replaceMainTitle() - dive in\

Objects and the DOM: overview

- **document** -> built-in object
- getElementById('idName') -> method of the document object (more to come)
  - NO **#** symbol here as in CSS, but we need to have that id assigned in HTML in order to have a not null selection
- textContent -> property of the object returned by getElementById()

Objects and the DOM: common operations

- Style DOM elements: occasionally we might want control over this in JS and not via CSS as we've seen until now
  - **style** property (**textContent** is also a property)
  - What's actually happening is inline styling is being set (DevTools) -> highest priority! (almost)
  - CSS properties turn to camelCase notation in JS: font-weight -> fontWeight
  - Measure units also need to be included here: fontSize = '20px'
- Detecting button clicks
- Show/hide DOM elements

Objects and the DOM: intro to interaction

- Common interaction with a website: CLICK (we will see more)
- Good practice to save selectors in const, since we are never planning on re-assigning values to them, just use the, apply props and/or methods to them
- Common debugging practices
  - Check the selected element
- **addEventListener(event, callback function)** method
  - We will be using an anonymous function
  - This func is to be called ONLY when/after the click event happens
  - This event can trigger multiple times

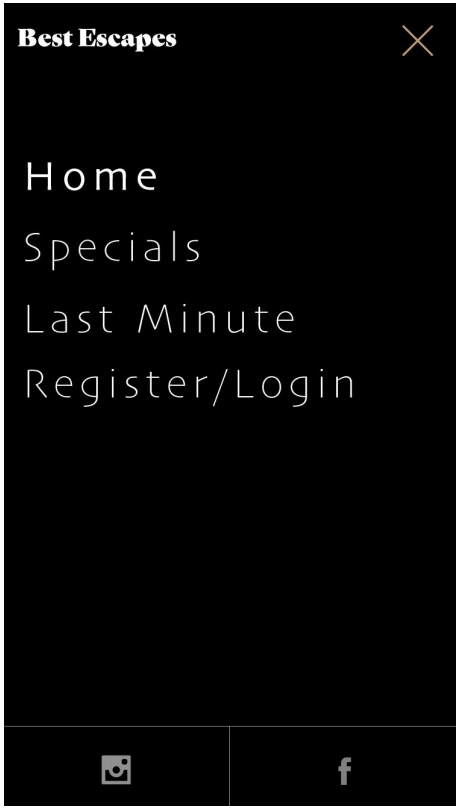
Objects and the DOM: content switch

- Toggle operation: show/hide DOM content
- The content already exists in our HTML
- Class manipulation is going to come into play
  - classList property: add, remove, contains
- Toggle behaviour implementation can be achieved using conditionals
- Don’t forget about UX: at any time the content must adjust to the user’s possible intent (the button should be a standalone invitation)

IT School
FROM ZERO TO HERO!

/PRACTICE\

Assignment: UI - layout#1



Assignment: UI - layout#2

Assignment: requirements

- Mobile menu ICON should be a clickable element at this point
- Add a click event listener to it
- Create a menu element: fixed positioning, priority layer
- Show this element when the menu icon is clicked
- Add a click event listener on the X icon belonging to the menu UI element in order to close it
- Mobile only
- External script as prefered solution, but you can experiment also with the other one

IT School
FROM ZERO TO HERO!

/Q&A\

Resources

- JS: Primitive Values and Object References:

  https://medium.com/@junshengpierre/javascript-primitive-values-object-references-361cfc1cbfb0
- Standard built-in objects: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects
- Moment.js library: https://momentjs.com/
- Date object: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date
- Math object: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math
- String object: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String
- Number object: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number
- HTML Element style prop: https://developer.mozilla.org/en-US/docs/Web/API/ElementCSSInlineStyle/style

IT School
FROM ZERO TO HERO!

Thank you

Next: JS - Arrays