Week 4 Day 3

Javascript and DOM

Javascript: OOP

REVATURE

Inheritance:

- Prototypical
- Set the __proto__ property of the object to refence it's parent
- Use "newly" implemented classes

Encapsulation:

- Closures
  - Nested function that can access the variables and arguments of its outer function, but can't change them

Polymorphism:

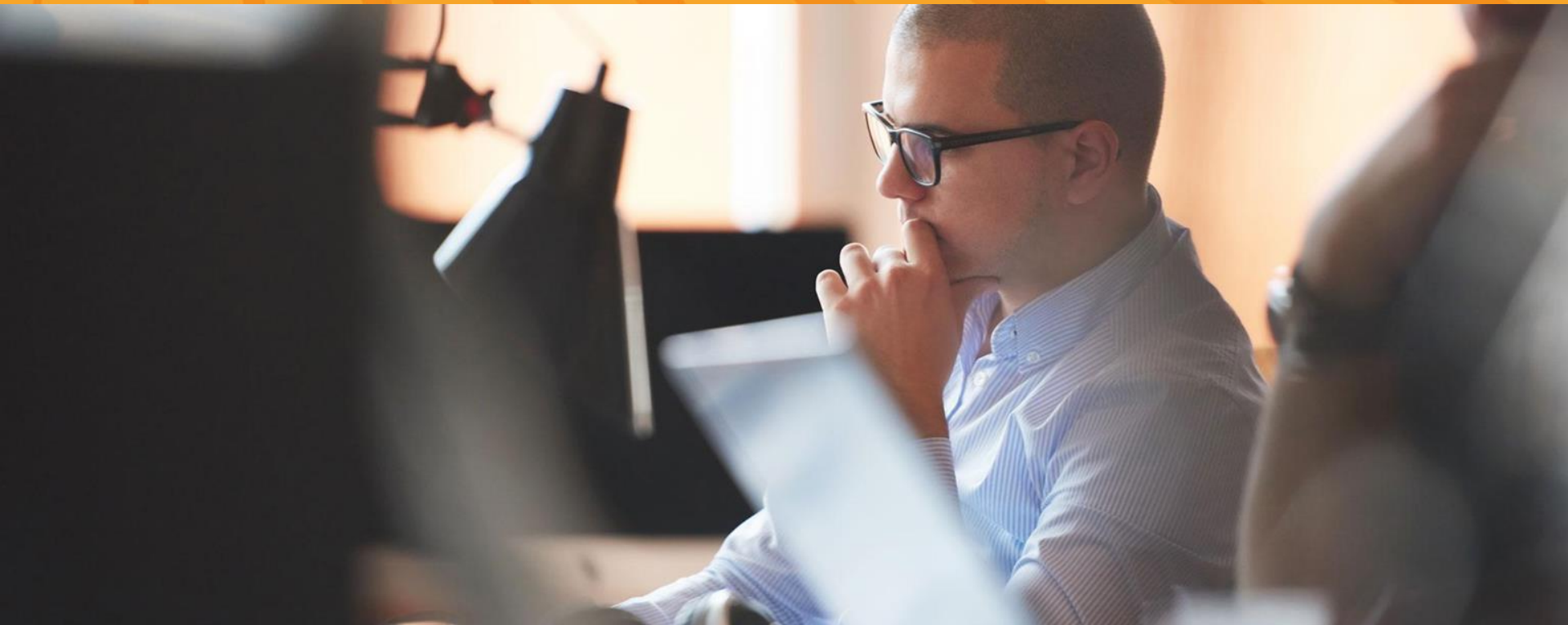
- Type coercion

Abstraction:

- Classes from ES6
- Creates object templates
- Even includes the static keyword and constructors

REVATURE

Javascript OOP DEMO



Javascript: Iterators

REVATURE

- Object which allows you to traverse values in a sequence
  - Use the .next() method to traverse
  - When the iterator is complete, .next() returns done=true
- To get the built-in iterator for an array you use the Arrays symbol object

```
let a = [1,2,3];
let iter = a[Symbol.iterator]()
console.log(iter.next());
console.log(iter.next());
console.log(iter.next());
console.log(iter.next());
```

Javascript: Generator

REVATURE

- Special iterator in JS which generates a new value
  - Use the next() method to generate the next value
  - Use the yield keyword to stop the generator
- Create a generator with *function syntax

```
let fib = function(num) {
    if(num == 0 || num == 1) {
        return 1;
    }
    return fib(num-1) + fib(num-2);
}
function* makeFibGenerator() {
    for (let i = 0; i < 10; i++) {
        yield fib(i);
    }
    return;
}
let fibIter = makeFibGenerator();
let result = fibIter.next();
while(result.value < 50) {
    console.log(result.value);
    result = fibIter.next();
}
```

Javascript: Iterable

REVATURE

- Any Object which implements the @@iterator method
- Marks the object as usable in constructs such as for of loops
- Work in a similar manner to generator functions

```
let fib = function(num) {
    if(num == 0 || num == 1) {
        return 1;
    }
    return fib(num-1) + fib(num-2);
}
const fibIterable = {
    *[Symbol.iterator]() {
        for (let i = 0; i < 9; i++) {
            yield fib(i);
        }
        return;
    }
}
for (let value of fibIterable) {
    console.log(value);
}
```

Javascript DOM

Document Object Model

Javascript: Document Object Model

The DOM is a tree like structure representing the HTML document

- Root is also the html tag
- Access every html element of the page with JS
- Use the document keyword to gain access

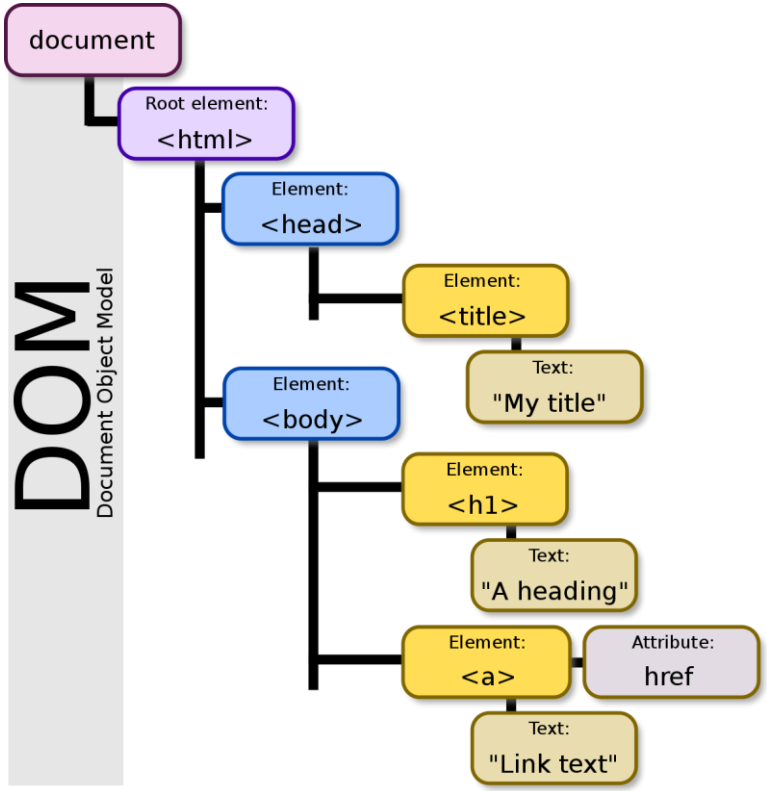


Image source: https://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/DOM-model.svg/1200px-DOM-model.svg.png

Javascript: Select DOM Elements

REVATURE

All methods/properties are preceded by the document keyword

Selecting Specific Elements

- .getElementById(“id”)
- .getElementsByClassName(“class”)
- .getElementsByTagName(“tag”
- .querySelector(“selector)
- .querySelectorAll(“selector)

Gain access to top level nodes

- .documentElement
- .head
- .body

Gain access to parent nodes

- .parentNode
- .parentElement

Gain access to child and sibling nodes

- .childNode
- .firstChild
- .lastChild
- .previousElementSibling
- .nextElementSibling

Javascript: DOM Manipulation

REVATURE

Create a new DOM element : document.createElement()

To modify elements, precede the methods/properties below with the element keyword

- .replaceChild()
- .removeChild()
- .insertBefore()
- .innerText
- .textContent
- .innerHtml
- .cloneNode()

Modify element attributes with the following methods

- .getAttribute(“attribute)
- .setAttribute(“attribute)
- .removeAttribute(“attribute)
- .has Attribute(“attribute)

REVATURE

DOM Manipulation DEMO

Javascript: Events and Listeners

REVATURE

Event:

- When a user interacts with your webpage
- Handle these with Event Listeners/Handlers

Many Types of Events:

- onclick
- onload
- onmouseover
- onkeydown
- onchange
- onsubmit
- Many more

Register a handler:

1. Inline, set the on… attribute of the html element
2. Set the event property of the html element to a JS function
3. Use the element method addEventListener(event, function, useCapture)

Javascript: The Event Object

REVATURE

All Events are represented by an Event object with these properties:

- bubbles
- currentTarget
- preventDefault()
- stopPropagation()
- target
- type

Javascript: Types of Events

REVATURE

There are 16 different subclasses of the Event object, two you will see are:

- MouseEvent
  - onmouseenter, onmouseleave
  - Special properties
    - clientX, clientY
    - movementX, movementY
    - offsetX, offsetY
    - screenX, screenY
    - altKey, ctrlKey, shiftKey
- KeyboardEvent
  - Onkeydown, onkeyup, onkeypress
  - Special properties
    - altKey, ctrlKey, shiftKey
    - key, keycode, which
    - repeat

Javascript: Bubbling and Capturing

REVATURE

Event propagation:

- How the event flows through the components on the page

Bubbling

- Default strategy
- Bottom up approach
- Works with all handlers

Capturing

- Must be declared when creating the handler
- Top down approach
- Only works with handlers registered with .addEventListener()

Javascript: this keyword

REVATURE

- this keyword has multiple meaning depending on where it is used
  - this alone refers to the global window object
  - this in event handlers refers to the HTML element which received the event
  - this in object binding refers to the object

REVATURE

Events DEMO