NUS National University of Singapore

School of Computing

Application Systems Development for Business Analytics

BT3103 - Week 3
2019/2020 Semester 2

Thangamani R

Recap from Week 2

- HW Problems
  - UI Links / HTML File Link
  - Revisit HW problems

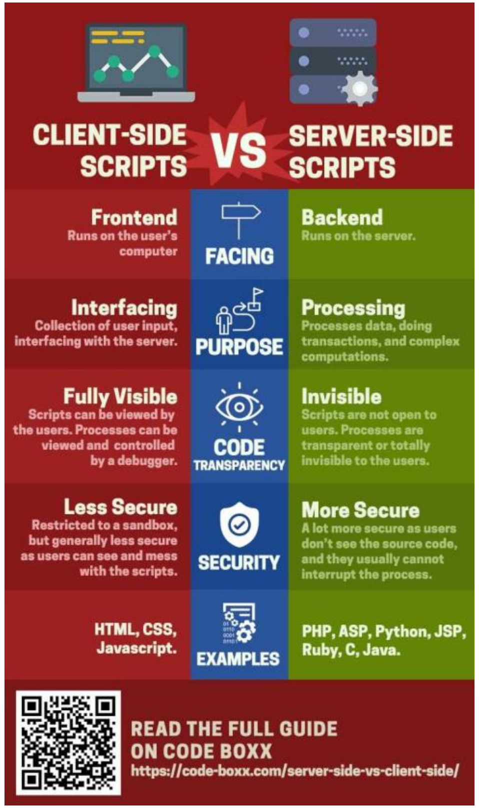
Week 3

- Javascript introduction
- Project Methodology - Waterfall
- Vue JS introduction
- Hands on exercises using Discovery tool

Javascript

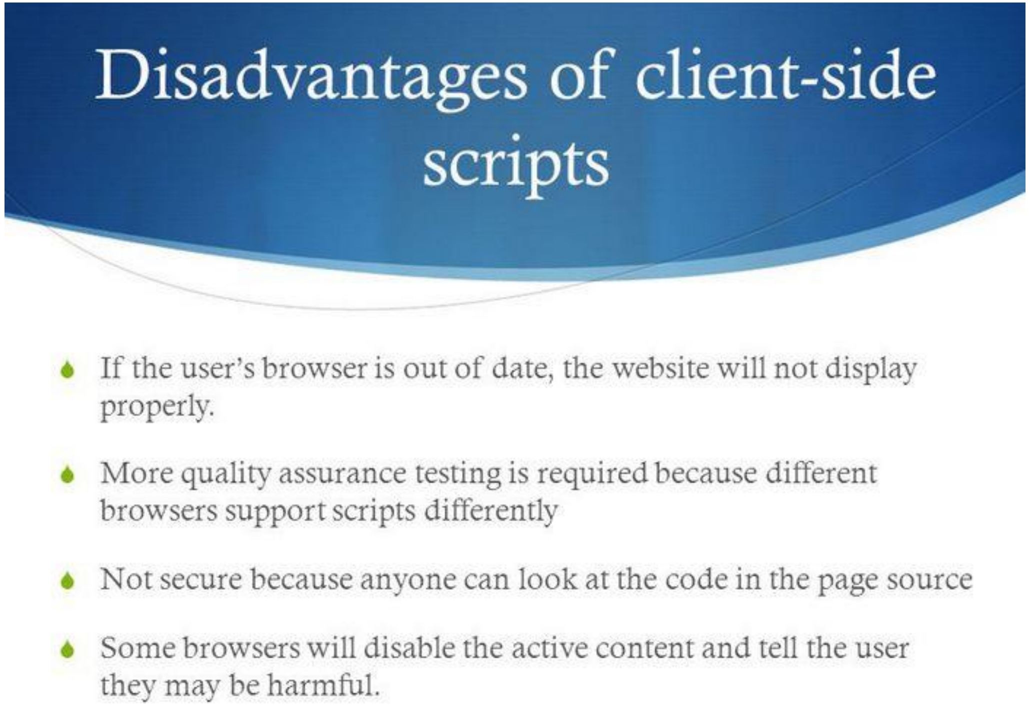
- Client side scripting language
- Runs on the client machine
- Interacts with a copy of the webpage loaded into a web browser in client’s machine.
- Adds interactivity to the HTML pages
- Embedded directly into HTML
- Case sensitive

Javascript



CLIENT-SIDE SCRIPTS	VS	SERVER-SIDE SCRIPTS
Frontend Runs on the user's computer	FACING	**Backend** Runs on the server.
Interfacing Collection of user input, interfacing with the server.	PURPOSE	**Processing** Processes data, doing transactions, and complex computations.
Fully Visible Scripts can be viewed by the users. Processes can be viewed and controlled by a debugger.	CODE TRANSPARENCY	**Invisible** Scripts are not open to users. Processes are transparent or totally invisible to the users.
Less Secure Restricted to a sandbox, but generally less secure as users can see and mess with the scripts.	SECURITY	**More Secure** A lot more secure as users don't see the source code, and they usually cannot interrupt the process.
HTML, CSS, Javascript.	EXAMPLES	PHP, ASP, Python, JSP, Ruby, C, Java.

READ THE FULL GUIDE ON CODE BOXX
https://code-boxx.com/server-side-vs-client-side/



Disadvantages of client-side scripts

- If the user's browser is out of date, the website will not display properly.
- More quality assurance testing is required because different browsers support scripts differently
- Not secure because anyone can look at the code in the page source
- Some browsers will disable the active content and tell the user they may be harmful.

Javascript

- Embedded with the head of the HTML between the
  - *<script></script>* tags.
- Similar to css can be embedded within HTML

  or
- Written as separate javascript file with a .js extension and then linked to HTML.

Javascript

- Embedded with the head of the HTML between the
  - *<script type="text/javascript"></script>* tags.
- Similar to css can be embedded within HTML

  or
- Written as separate javascript file with a .js extension and then linked to HTML.

  <script type="text/javascript" src="myscripts.js"></script>

Javascript

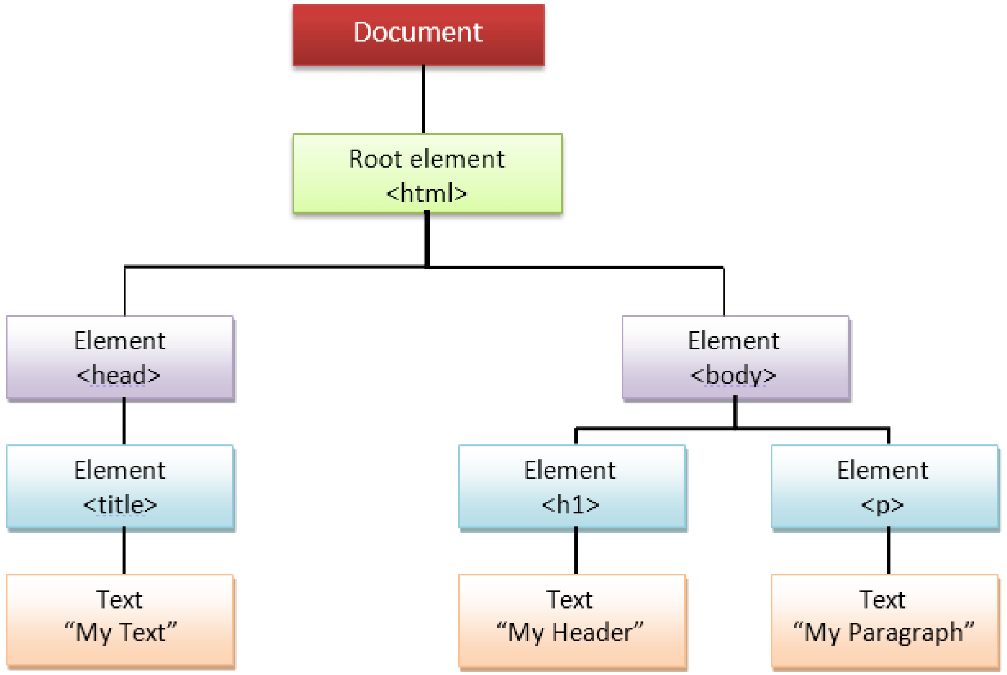
- Dynamic HTML.
  - Web pages are changed after they are loaded and rendered in the browser
  - Such changes are immediately shown on the screen.

Javascript

- Document Object Model (DOM)
  - Web browser creates a DOM of web page when it is loaded.
  - DOM model is created as a tree of objects.
  - Using DOM, javascript can add new elements to the page, update existing elements, react to events etc.

Javascript

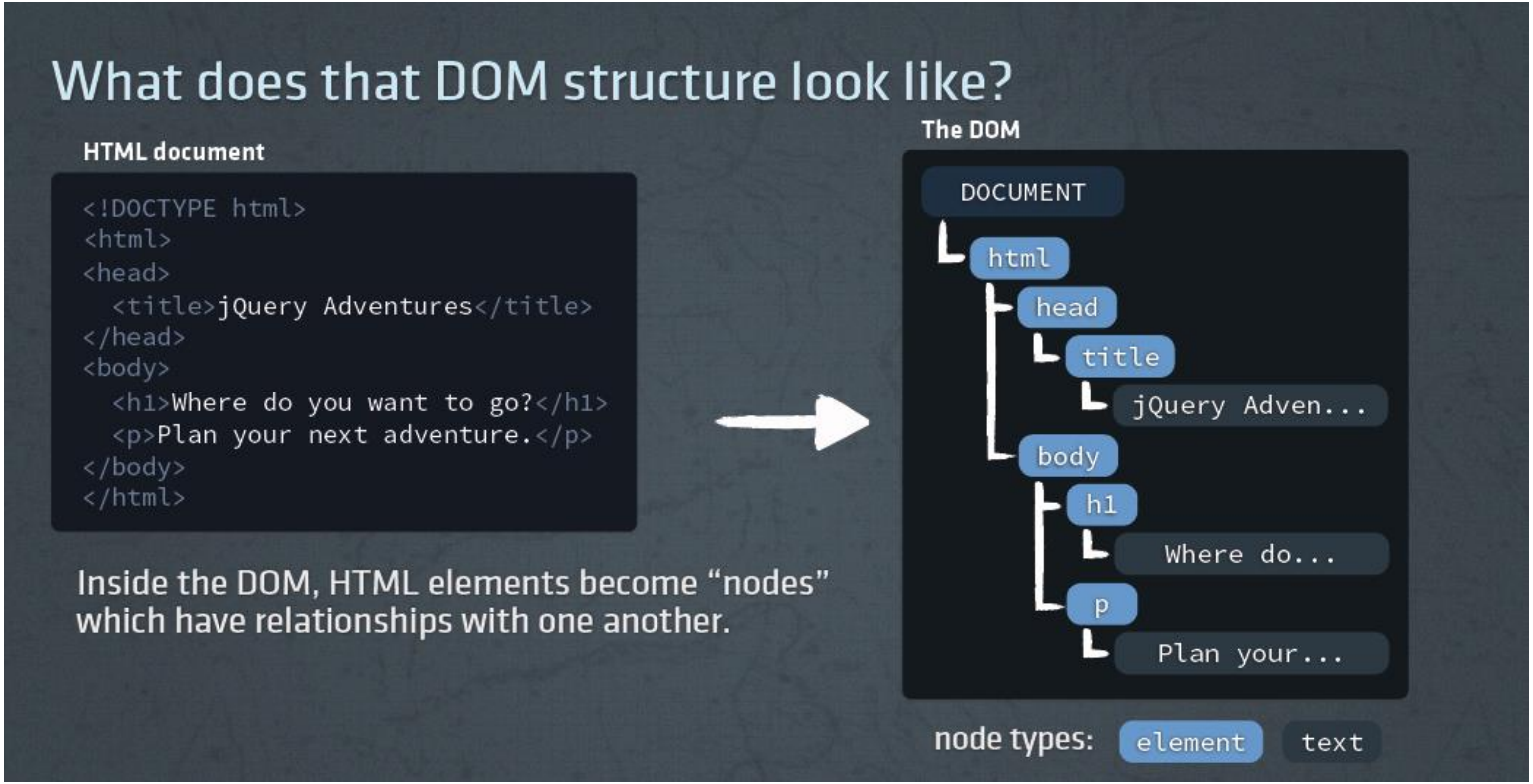
- DOM (Document Object Model)



Javascript

- Document Object Model (DOM)
  - DOM classes HTMLDocument and HTMLElement provide numerous methods for manipulating a loaded page.
  - Retrieve an existing element using DOM or create a new element and set its properties.

Javascript - DOM



What does that DOM structure look like?

HTML document

```
<!DOCTYPE html>
<html>
<head>
  <title>jQuery Adventures</title>
</head>
<body>
  <h1>Where do you want to go?</h1>
  <p>Plan your next adventure.</p>
</body>
</html>
```

Inside the DOM, HTML elements become "nodes" which have relationships with one another.

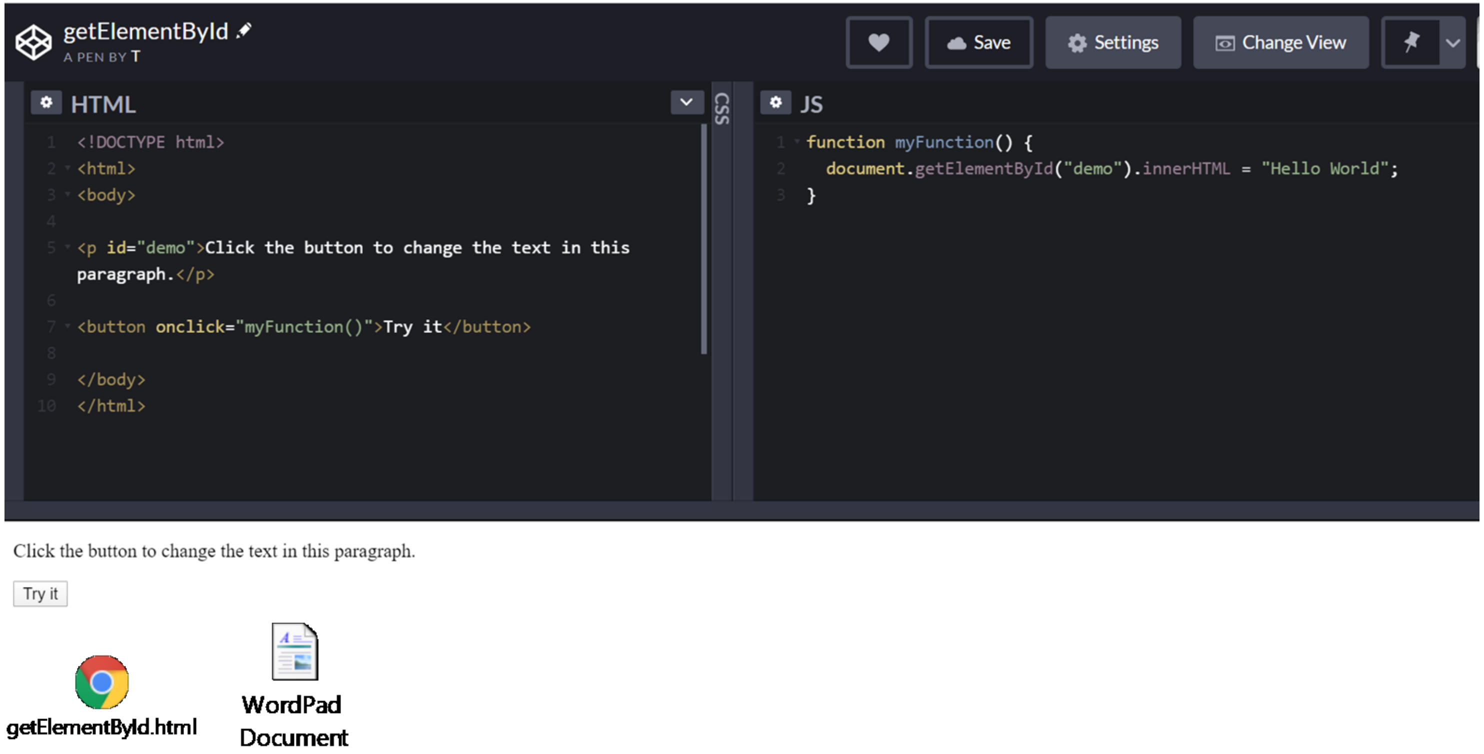
Javascript

- Document Object Model (DOM)
  - change HTML elements in the page
  - change HTML attributes in the page
  - change the CSS styles in the page
  - react to the events in the page

Javascript

- Document Object Model (DOM)
- Methods to manipulate HTML Elements
  - finding HTML elements by id *getElementById()*
  - finding HTML elements by tag name *getElementsByTagName()*
  - finding HTML elements by class *getElementsByClassName()*

Javascript



```
<!DOCTYPE html>
<html>
<body>

<p id="demo">Click the button to change the text in this
paragraph.</p>

<button onclick="myFunction()">Try it</button>

</body>
</html>
```

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Hello World";
}
```

Click the button to change the text in this paragraph.

Javascript –Problem 1

- Use https://codepen.io for the exercise below
- Using the code provided in the previous slide:
  - Define a new *<div>* element with the id as demo1
  - Set the default text in the *<div>* element as “BT3103 - Week 3 ”
  - On button click event, update the text to “Week3 - Javascript Basics” using *getElementById()*

Javascript

- *var* keyword is used to declare variables

```
    var a=5;
 var b,c;
 b=a;
     alert(a);
 alert(b);
 alert(c);
```

Javascript

- *alert()* is used to debug the javascript code.
- Prompts alert() message in the UI.

- *document.write()* is used to output text to page .Can include HTML elements in this.

Javascript – Problem 2

- Use https://codepen.io for the exercise below
  - Define 6 variables var1,var2,var3,var4,var5 and var6
  - Assign the value 5 to var1, 7 to var2, “Hello” to var3 and “World” to var4
  - Assign var1+var3 to var5 and var1+var2 to var6
  - Print the results

Arrays

- List of values
- Two ways to create Array
  - *var myArray=[value1,value2]*
  - Square brackets [ ] mark the start and end of the Array.Elements are separated by commas and can be of any data type.
  - *var myArray = new Array();*

Arrays

- **Length of an Array:**
  - Number of elements in an array is referred to its length.
  - Length of an array is obtained by
    - *myArray.length*

Arrays

- **Index of an Array:**
  - Refers to the position of the element within the Array.
  - First element has an index of 0
  - If we need to retrieve the third element
    - *myArray[2]*

Arrays

- **Adding elements to an Array:**
    - Assign values using square brackets
        - *myArray[3]="Monday"*
    - If the element already exists , it is overwritten.
    - Else new element is added.

Arrays

- **Appending elements to an Array:**
  - *myArray.push()*
  - Method to add elements to the end of the Array.
- **Removing Elements from an Array**
  - *myArray.shift()* and *myArray.pop()*
  - Methods to remove first and last elements of the array respectively.

Arrays – Problem 3

- Use https://codepen.io for the exercise below
  - Define myArray1 with the days of the week from Sunday till Friday
  - Add Saturday to myArray1.
  - Remove Sunday from myArray1
  - Find the length of the myArray1 after each step.

Arrays – Problem 4

WordPad Document

Chrome HTML Document

TRY IT!

- Use https://codepen.io for the exercise below
  - Define myArray2 with the numbers from 4 to 10.
  - Replace the 3rd element in the array with number 11.
  - Get the 5th element of the array
  - Add another element to the array using push()
  - Remove the first and the last elements of the Array.
  - Find the length of the array.

Functions

- **Creating Functions:**
  - **Function Definition:**

```
function functionName(param1,param2){
statement1;
statement2;
}
function displayMsg(userName){
            alert("Hello "+userName);
}
```

Functions

- Javascript has a number of built in functions
- alert() is one of the functions
- Invoke a function by using function name followed by parenthesis()
- Arguments are passed within the parenthesis. Multiple arguments are separated by commas.

Functions

- Function Invocation(Calling Functions)
  - *functionName();*
  - *functionName(arg1);*
  - *functionName(arg1,arg2);*
  - *displayMsg("Pokemon")*

Functions

- Return Statements
  - Used to return the result of function processing
  - *return* keyword is used to return values
  - *return (functionResult);*

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

TRY IT!

Functions – Problem 5

- Use https://codepen.io for the exercise below.
- Create a function greetUser that accepts userName as the parameter
- Create buttons for different users
- On click of the button function code should write a welcome message on the screen with the user name.
- Invoke the function with different userNames

Functions – Problem 6

- Use https://codepen.io for the exercise below.
- Similar to the previous example
  - Create a function multiplyNumbers that takes 3 parameters.
  - multiplyNumbers should return the product of 3 numbers.
  - Invoke the function with different sets of numbers and print the values.

Events

- Web pages issue events in response to various conditions and user actions such as a page having loaded successfully or the user clicking on a button etc.
- Event is a message issued by the browser
- An event always has a type. The type indicates the nature of the event.

Events

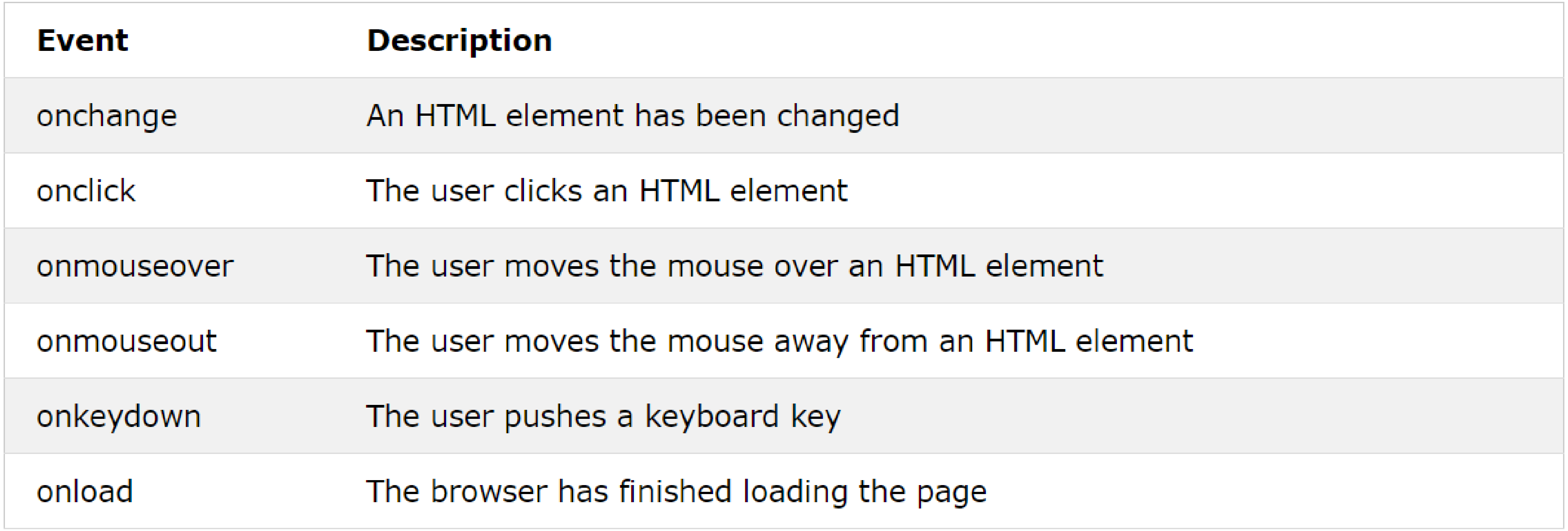
- **Listening to an event:**
  - In order to respond to an event we need to write code that listens for the event occurring.
  - This code is called the event listener.
  - When event listener detects the event it is listening for,it evaluates the expression that performs some action in response to the event.

Events

- **Event Handlers**:
    - When an event listener is triggered the expression it executes is a function or a method call.
    - This function or a method call is referred to as event handler.
    - Event handler is always passed an object containing details about the event.

Events

- **Common Events**



Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

Events

- **Event Handlers**:
    - `<button onclick="myFunction()">Click me</button>`
    - Onclick - Event
    - myFunction() → method to be called when the event occurs.

Events

- **Event Handlers**:

```
document.getElementById("myBtn").addEventListener("click",
    displayDate);
```

- *addEventListener* method is used to attach the event handler to the specified element.
- In this case it attaches the displayDate method to the click event on the myBtn object.
- I.e. It triggers displayDate() method on the button click.

Events

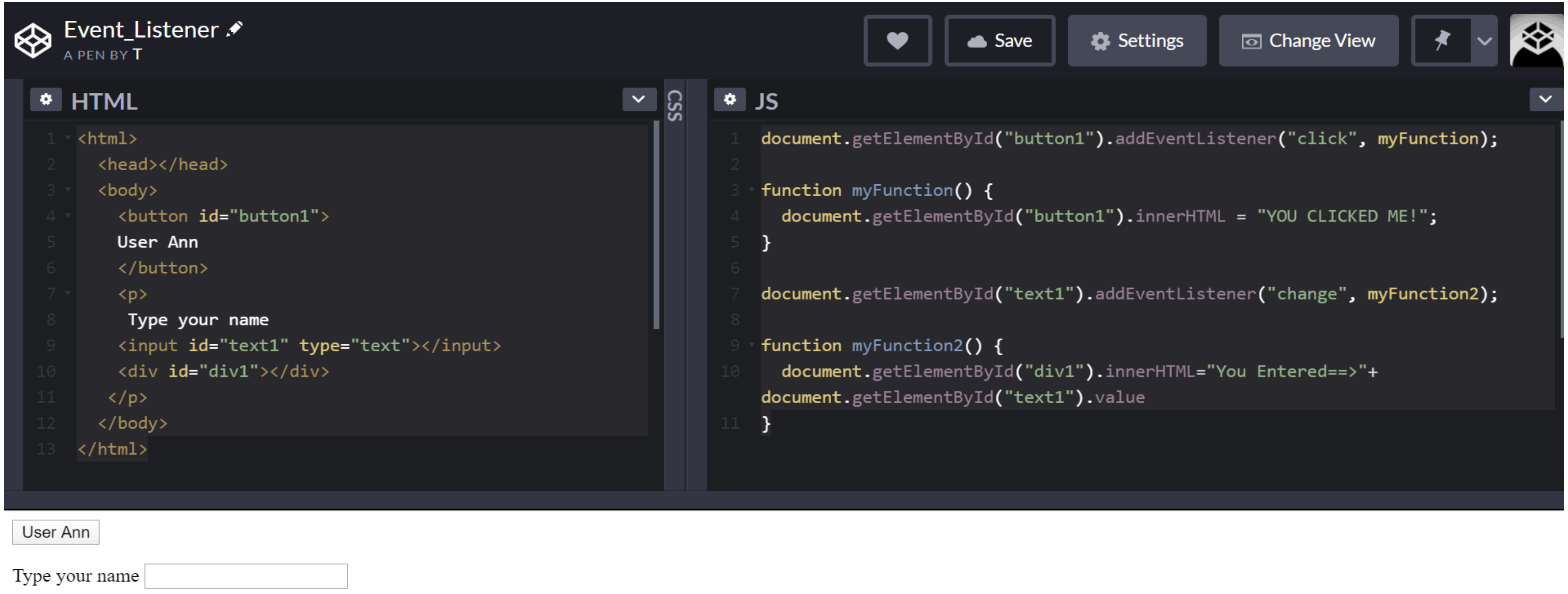
element.addEventListener(event, function, useCapture)

- Event- Type of the event eg. click, change
- Function - Function to be called when the event occurs
- useCapture - optional boolean parameter . Default value is false.

Events

WordPad Document

eventListner.html



```html
<html>
  <head></head>
  <body>
    <button id="button1">
    User Ann
    </button>
    <p>
     Type your name
    <input id="text1" type="text"></input>
    <div id="div1"></div>
   </p>
  </body>
</html>
```

```javascript
document.getElementById("button1").addEventListener("click", myFunction);

function myFunction() {
  document.getElementById("button1").innerHTML = "YOU CLICKED ME!";
}

document.getElementById("text1").addEventListener("change", myFunction2);

function myFunction2() {
  document.getElementById("div1").innerHTML="You Entered==>"+
document.getElementById("text1").value
}
```

User Ann

Type your name

Events – Problem 7

- Use https://codepen.io for the exercise below.
- Define a counter and initialize to 0
- Add two buttons, one to increment and one to decrement a counter
- Capture the button click events and increment and decrement counters accordingly.

Javascript

- Try out the Javascript activities in the Discovery path.

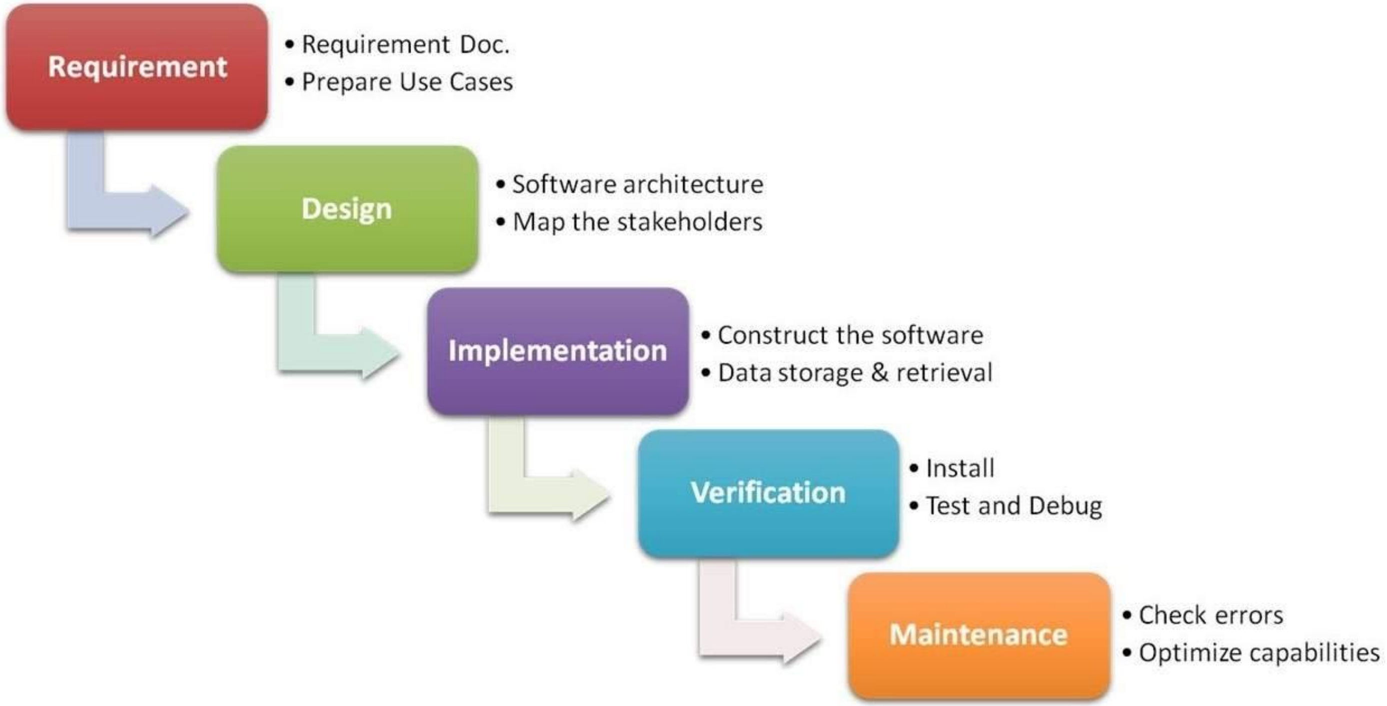
Project Methodologies

- Blue print for how tasks and projects are planned , managed and executed from start to finish.
- Depends on the type of project being executed.
- Organizational approach
- To optimize resources and time
- Size of the project

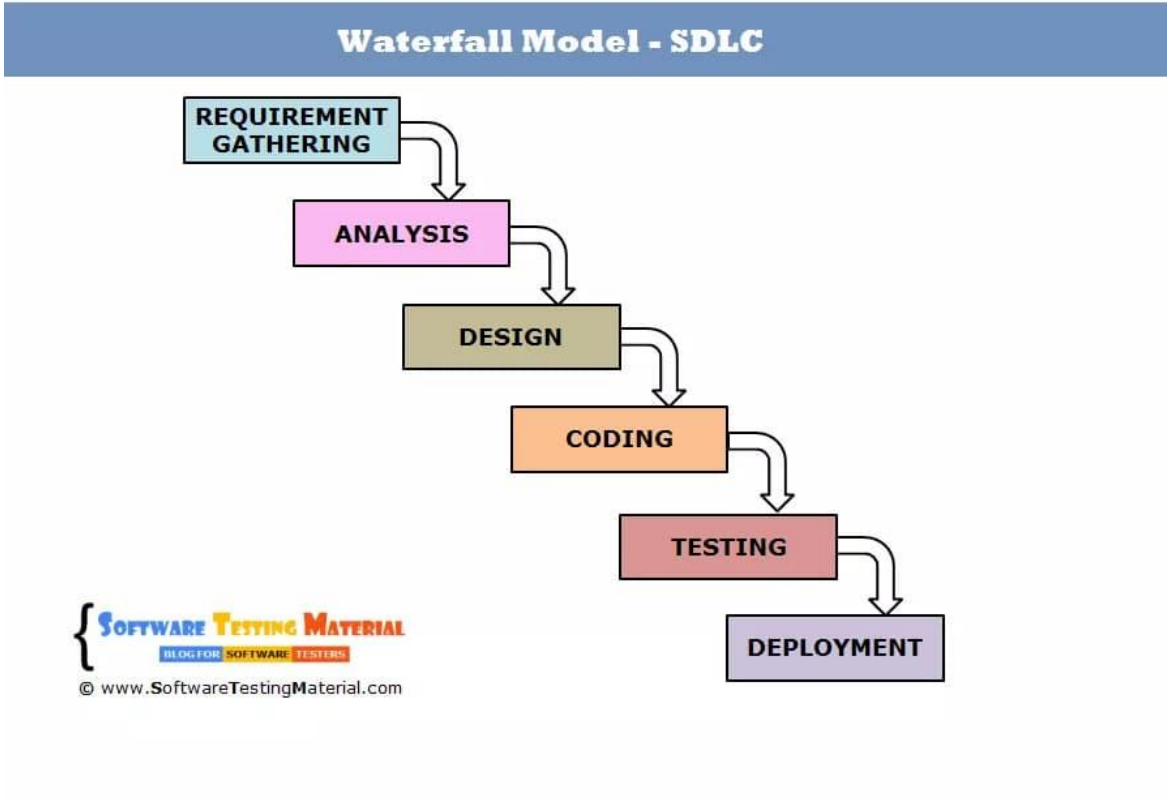
Waterfall

- One of the traditional methodologies
- linear, sequential design approach where progress flows downwards in one direction, like a waterfall.
- Able to move to the next phase only when the current phase is successfully completed.
- Stress is more on documentation.
- Phases are not repeated

Waterfall


Requirement
• Requirement Doc.
• Prepare Use Cases
Design
• Software architecture
• Map the stakeholders
Implementation
• Construct the software
• Data storage & retrieval
Verification
• Install
• Test and Debug
Maintenance
• Check errors
• Optimize capabilities

Waterfall


Waterfall Model - SDLC
REQUIREMENT GATHERING
ANALYSIS
DESIGN
CODING
TESTING
DEPLOYMENT
Software Testing Material
BLOG FOR SOFTWARE TESTERS
© www.SoftwareTestingMaterial.com

Waterfall

- Pros:
- Because project requirements are agreed upon in the first phase, planning and scheduling is simple and clear.
- With a fully laid out project schedule, you can give an accurate estimate for your project cost, resources and deadlines.
- It’s easy to measure progress as you move through the phases and hit milestones.
- Customers aren’t perpetually adding new requirements to the project, delaying production.

Waterfall

- Cons:
    - Difficult for customers to articulate all of their needs at the beginning of the project
    - If the customer is dissatisfied with the product in the verification phase, it can be very costly to go back and design the code again.
    - Lack of customer feedback in the intermediate phases
    - Delayed start in testing

Waterfall

- Best Suited For:
    - Requirements are clearly stated and documented.
    - Chances of surprises are low.
    - Requirements do not change
    - No additional value from going Agile.

Vue JS

- Front end Framework
- Progressive Framework
- Allows you to build applications with minimal effort
- Core Vue.js library focuses only on the view layer
- Additional libraries can be added as needed
- Easy to integrate HTML and javascript
- Reference : https://vuejs.org/v2/guide/

Vue JS

- Easy to use
- Include the below tag to use Vue JS in your code.
- `<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>`
- Production and Dev version , but we are using the DEV version.

Vue JS

- **Create Vue Instance**

  Create a new Vue Instance using the code in the javascript file

```
var app = new Vue({

  })
```

Vue JS

- **El property**

  El property allows to specify where the Vue instance will mount on the page.

```
var app = new Vue({
  el: '#app'
})
```

Vue JS

- **HTML changes**

  Include a <div> tag in the HTML , with the id as app. This is where the Vue instance will get mounted

```
<body>
  <div id="app">
    Hello World!!
  </div>
</body>
```

Vue JS

- **Data property**
  - To bind data to the Vue instance. Vue reactive system monitors the data object for changes and updates the view for those changes

```
data:{
  greeting:"Hello World!"
}
```

Vue JS

- **Data Binding**
  - Mustache syntax
  - Two curly braces surrounding your property
  - {{propertyName}}
  - Execute javascript function.

```
<body>
  <div id="app">
    {{greeting}}
  </div>
</body>
```

Vue JS

- **Methods**
  - Javascript methods
  - Avoid using arrow functions

```
methods:{
  displayMsg:function (){
    alert("Hello There");
  }
}
```

Vue JS

- **Conditional Rendering**
  - Two directives to conditionally show content
  - *v-if* and *v-show*
  - *V-if* - Element is removed from the DOM . Can be used with *v-else* and *v-else-if* directives

```
<div id="example">
    <h1 v-if="a">The condition is true</h1>
   <h1 v-else-if="b">In the else if block, b is true</h1>
   <h1 v-else>In the else block ,neither a or b is true</h1>
</div>
```

Vue JS

- **Conditional Rendering**
  - *V-show*- Hide and show content using the CSS display property.

```
<div id="example">
    <h1 v-show="!a">This is hidden</h1>
    <h1 v-show="a">This is shown</h1>
</div>
```

Vue JS

- **Conditional Rendering**
  - How to choose between *v-show* and *v-if*
  - If the directive is going to change often, use *v-show*. If it is intended to change only occasionally or never after the first render, it’s better to use *v-if*.

Vue JS

- **Lists**
  - With *v-for*, we can iterate (go over each item) through the items of an array and use each object to display content
  - *v-for* can be used to display each item in an array

```
<ul>
<li v-for="item in items">
{{item}}
</li>
</ul>
```

Vue JS

- Try out the Vue JS activities in the Discovery path.

Mid-Sem Project:

Team requirements:

- Each team comprises of 5 to 6 members
- Diverse group
- At least 2 girls / 2 boys in the group
- Choose a team name
- Finalize team and team name before next class.
- Individual and group submission involved for mid-sem
- Mid-sem project details will be revealed in the next class

Wrap up:

What was covered:

a. Javascript

b. Waterfall methodology

c. Vue js

Quiz 2

- Quiz 2 next week (5 marks)
- Topics from week 2 & week 3