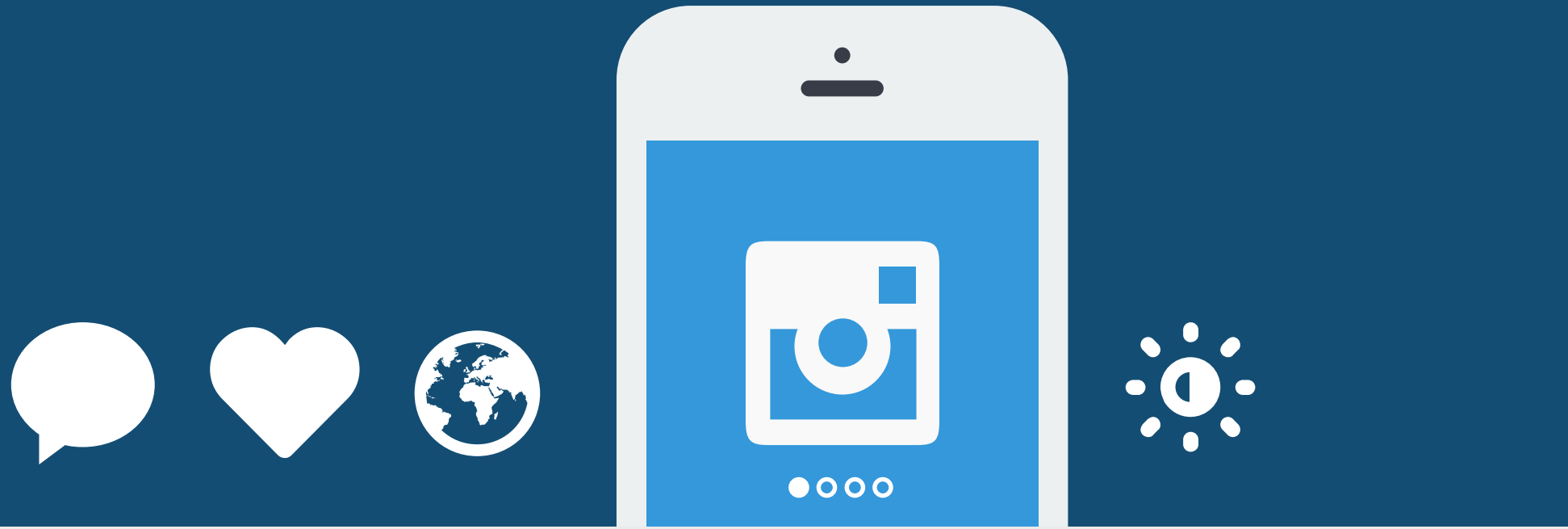
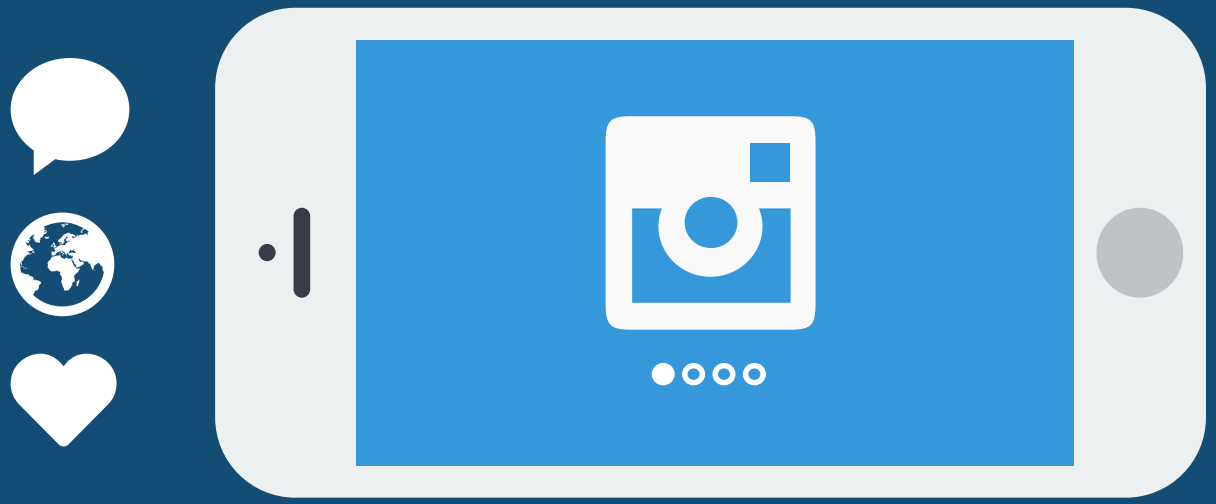
targettrust

treinamento e tecnologia



FUNÇÕES COMO OBJETOS



Funções com Objetos

- O **Javascript** é uma linguagem script dinâmica, fracamente tipada, baseada em protótipos e que possui funções de primeira classe.

Linguagem Dinâmica

- Linguagem de alto nível que executa em tempo de execução várias tarefas que outras linguagens executariam durante a compilação.

Fracamente Tipada

- Suporta conversão de tipos implícita, e não se declara o tipo na inicialização da variável.

Baseada em Protótipos

- Estilo de orientação a objetos onde classes não existem, dessa forma o reuso (herança para linguagens baseadas em classe) é feito clonando objetos existentes que servem assim de protótipos.

Funções de Primeira Classe

- Habilidade da linguagem de passar functions como argumentos para outras functions, retornar functions como valor em outras funções, atribuir functions a variáveis e armazenar functions em estruturas de dados.

Funções como Objetos

- Funções no **Javascript** fazem mais que separar lógica dentro de unidades de execução.

- Possibilitam também a injeção de escopo e a habilidade de criar objetos.

Funções como Objetos

- Ter funções com tantas possibilidades e responsabilidades pode ser considerado uma bênção ou uma maldição. **Bênção** porque isso torna a linguagem versátil e rápida, e **maldição** porque isso faz com que você facilmente dê tiros no seu próprio pé se você não souber o que está fazendo.

- No **Javascript**, funções são objetos (com atributos, métodos e tudo o mais).

- Funções são linkadas a Function.prototype que por sua vez são linkadas a Object.prototype.

Funções como Objetos

- Toda função possui dois parâmetros implícitos. **this**, representando o contexto da função, e **arguments**, representando os argumentos passados para a função.

- Toda função possui também seu próprio membro **prototype**. Seu valor é um objeto com um atributo **constructor** cujo valor é a própria função em si.

Funções como Objetos

Uma função em **Javascript** é composta pelas seguintes partes:

- Palavra reservada **function**.
- Nome da função (opcional). Quando o nome não é declarado, a função é chamada de anônima.
- Parâmetros separados por vírgula. Serão atribuídos a eles variáveis de mesmo nome dentro da função. Ao invés de serem inicializados com **undefined**, são inicializados diretamente pelo **arguments**.
- Comportamento.

O que é um closure?

- Um **closure** é uma função interior que tem acesso a variáveis de uma função exterior – cadeia de escopo. O closure tem três cadeias de escopo: ele tem acesso ao seu próprio escopo (variáveis definidas entre suas chaves), ele tem acesso as variáveis da função exterior e tem acesso as variáveis globais.

- A função interior tem acesso não somente as variáveis da função exterior, mas também aos parâmetros dela. Note que a função interior não pode chamar o objeto arguments da função exterior, entretanto, pode chamar parâmetros da função externa diretamente.

- Você cria um closure adicionando uma função dentro de outra função.

O que é um closure?

```
function showName (firstName, lastName) {
    var nameIntro = "Your name is ";
    //esta função interior tem acesso as variáveis da função exterior, incluindo os parâmetros
    function makeFullName () {
        return nameIntro + firstName + " " + lastName;
    }
    return makeFullName ();
}
showName ("Michael", "Jackson"); //Your name is Michael Jackson
```

O que é um closure?

```
$(function () {
    var selections = [];
    $(".niners").click(function () { //este closure tem acesso as variáveis de selections
        selections.push(this.prop ("name")); //atualiza a variável selection no escopo da função exterior
    });
});
```

Invocando uma função em JavaScript

Funções em **Javascript** podem ser invocadas de quatro formas diferentes, e em cada uma dessas formas this será inicializado com um comportamento diferente.

- Invocação de métodos
- Invocação de função
- Invocação de construtor
- Invocação via apply

Patterns e Técnicas para o uso de funções

- Módulos
- Namespaces
- Cascata
- Memoization